```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ansi/ext_fmt.h
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

struct mantissa {
        unsigned long h_32;
        unsigned long l_32;
};

struct EXTEND {
        short   sign;
        short   exp;
        struct mantissa mantissa;
#define m1 mantissa.h_32
#define m2 mantissa.l_32
};

```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ansi/loc_time.h
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * loc_time.h - some local definitions
 */
/* $Header: loc_time.h,v 1.1 91/04/22 13:19:51 ceriel Exp $ */

#define YEAR0           1900                    /* the first year */
#define EPOCH_YR        1970            /* EPOCH = Jan 1 1970 00:00:00 */
#define SECS_DAY        (24L * 60L * 60L)
#define LEAPYEAR(year)  (!((year) % 4) && (((year) % 100) || !((year) % 400)))
#define YEARSIZE(year)  (LEAPYEAR(year) ? 366 : 365)
#define FIRSTSUNDAY(timp)       (((timp)->tm_yday - (timp)->tm_wday + 420) % 7)
#define FIRSTDAYOF(timp)        (((timp)->tm_wday - (timp)->tm_yday + 420) % 7)
#define TIME_MAX        ULONG_MAX
#define ABB_LEN         3

extern const int _ytab[2][12];
extern const char *_days[];
extern const char *_months[];

void _tzset(void);
unsigned _dstget(struct tm *timep);

extern long _timezone;
extern long _dst_off;
extern int _daylight;
extern char *_tzname[2];
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/abort.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: abort.c,v 1.3 90/11/22 13:59:37 eck Exp $ */

#if     defined(_POSIX_SOURCE)
#include        <sys/types.h>
#endif
#include        <signal.h>
#include        <stdlib.h>

extern void (*_clean)(void);

void
abort(void)
{
        if (_clean) _clean();           /* flush all output files */
        raise(SIGABRT);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/abs.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: abs.c,v 1.1 89/05/16 13:06:59 eck Exp $ */

#include        <stdlib.h>

int
abs(register int i)
{
        return i >= 0 ? i : -i;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/asctime.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * asctime - print a date
 */
/* $Header: asctime.c,v 1.3 91/04/22 13:20:15 ceriel Exp $ */

#include        <string.h>
#include        <time.h>
#include        "loc_time.h"

#define DATE_STR        "??? ??? ?? ??:??:?? ????\n"

static char *
two_digits(register char *pb, int i, int nospace)
{
        *pb = (i / 10) % 10 + '0';
        if (!nospace && *pb == '0') *pb = ' ';
        pb++;
        *pb++ = (i % 10) + '0';
        return ++pb;
}

static char *
four_digits(register char *pb, int i)
{
        i %= 10000;
        *pb++ = (i / 1000) + '0';
        i %= 1000;
        *pb++ = (i / 100) + '0';
        i %= 100;
        *pb++ = (i / 10) + '0';
        *pb++ = (i % 10) + '0';
        return ++pb;
}

char *asctime(const struct tm *timeptr)
{
        static char buf[26];
        register char *pb = buf;
        register const char *ps;
        register int n;

        strcpy(pb, DATE_STR);
        ps = _days[timeptr->tm_wday];
        n = ABB_LEN;
        while(--n >= 0) *pb++ = *ps++;
        pb++;
        ps = _months[timeptr->tm_mon];
        n = ABB_LEN;
        while(--n >= 0) *pb++ = *ps++;
        pb++;
        pb = two_digits(
                    two_digits(
                            two_digits(two_digits(pb, timeptr->tm_mday, 0)
                                        , timeptr->tm_hour, 1)
                            , timeptr->tm_min, 1)
```

```
                        , timeptr->tm_sec, 1);

        four_digits(pb, timeptr->tm_year + 1900);
        return buf;
}
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/assert.c
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * assert.c - diagnostics
 */
/* $Header: assert.c,v 1.3 90/04/03 15:01:58 eck Exp $ */

#include        <assert.h>
#include        <stdio.h>
#include        <stdlib.h>

void __bad_assertion(const char *mess) {

        fputs(mess, stderr);
        abort();
}
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/atexit.c
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/* $Header: atexit.c,v 1.1 89/12/18 15:11:09 eck Exp $ */

#include        <stdlib.h>

#define NEXITS  32

extern void (*__functab[NEXITS])(void);
extern int __funccnt;

int
atexit(void (*func)(void))
{
        if (__funccnt >= NEXITS)
                return 1;
        __functab[__funccnt++] = func;
        return 0;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/atof.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: atof.c,v 1.2 89/12/18 15:11:50 eck Exp $ */

#include        <stdlib.h>
#include        <errno.h>

double
atof(const char *nptr)
{
        double d;
        int e = errno;

        d = strtod(nptr, (char **) NULL);
        errno = e;
        return d;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/atoi.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: atoi.c,v 1.4 90/05/22 12:22:25 ceriel Exp $ */

#include        <ctype.h>
#include        <stdlib.h>

/* We do not use strtol here for backwards compatibility in behaviour on
   overflow.
*/
int
atoi(register const char *nptr)
{
        int total = 0;
        int minus = 0;

        while (isspace(*nptr)) nptr++;
        if (*nptr == '+') nptr++;
        else if (*nptr == '-') {
                minus = 1;
                nptr++;
        }
        while (isdigit(*nptr)) {
                total *= 10;
                total += (*nptr++ - '0');
        }
        return minus ? -total : total;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ansi/atol.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: atol.c,v 1.3 90/05/22 10:48:12 ceriel Exp $ */

#include        <ctype.h>
#include        <stdlib.h>

/* We do not use strtol here for backwards compatibility in behaviour on
   overflow.
*/
long
atol(register const char *nptr)
{
        long total = 0;
        int minus = 0;

        while (isspace(*nptr)) nptr++;
        if (*nptr == '+') nptr++;
        else if (*nptr == '-') {
                minus = 1;
                nptr++;
        }
        while (isdigit(*nptr)) {
                total *= 10;
                total += (*nptr++ - '0');
        }
        return minus ? -total : total;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ansi/bsearch.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: bsearch.c,v 1.2 89/12/18 15:12:21 eck Exp $ */

#include        <stdlib.h>

void *
bsearch(register const void *key, register const void *base,
        register size_t nmemb, register size_t size,
        int (*compar)(const void *, const void *))
{
        register const void *mid_point;
        register int  cmp;
```

```
        while (nmemb > 0) {
                mid_point = (char *)base + size * (nmemb >> 1);
                if ((cmp = (*compar)(key, mid_point)) == 0)
                        return (void *)mid_point;
                if (cmp >= 0) {
                        base  = (char *)mid_point + size;
                        nmemb = (nmemb - 1) >> 1;
                } else
                        nmemb >>= 1;
        }
        return (void *)NULL;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ansi/calloc.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/* $Header$ */
#include        <stdlib.h>

#define ALIGN(x)        (((x) + (sizeof(size_t) - 1)) & ~(sizeof(size_t) - 1))

void *
calloc(size_t nelem, size_t elsize)
{
        register char *p;
        register size_t *q;
        size_t size = ALIGN(nelem * elsize);

        p = malloc(size);
        if (p == NULL) return NULL;
        q = (size_t *) (p + size);
        while ((char *) q > p) *--q = 0;
        return p;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ansi/chartab.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include        <ctype.h>

char __ctype[] = {
0,
_C,
_C,
_C,
_C,
_C,
_C,
_C,
_C,
_C,
_C|_S,
_C|_S,
```

```
	_C|_S,
	_C|_S,
	_C|_S,
	_C,
	_C,
	_C,
	_C,
	_C,
	_C,
	_C,
	_C,
	_C,
	_C,
	_C,
	_C,
	_C,
	_C,
	_C,
	_C,
	_C,
	_C,
	_S,
	_P,
	_P,
	_P,
	_P,
	_P,
	_P,
	_P,
	_P,
	_P,
	_P,
	_P,
	_P,
	_P,
	_P,
	_P,
	_N,
	_N,
	_N,
	_N,
	_N,
	_N,
	_N,
	_N,
	_N,
	_N,
	_P,
	_P,
	_P,
	_P,
	_P,
	_P,
	_P,
	_U|_X,
	_U|_X,
	_U|_X,
	_U|_X,
	_U|_X,
	_U|_X,
```

```
_U,
_U,
_U,
_U,
_U,
_U,
_U,
_U,
_U,
_U,
_U,
_U,
_U,
_U,
_U,
_U,
_U,
_U,
_U,
_U,
_P,
_P,
_P,
_P,
_P,
_P,
_L|_X,
_L|_X,
_L|_X,
_L|_X,
_L|_X,
_L|_X,
_L,
_L,
_L,
_L,
_L,
_L,
_L,
_L,
_L,
_L,
_L,
_L,
_L,
_L,
_L,
_L,
_L,
_L,
_L,
_L,
_P,
_P,
_P,
_P,
_C,
0,
0,
0,
```

```
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
```

```
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
0,
```

```
0,
0,
0,
0,
0,
};
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/clock.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * clock - determine the processor time used
 */
/* $Header: clock.c,v 1.3 90/01/22 13:08:11 eck Exp $ */

#include        <time.h>

#if     defined(__BSD4_2)

struct timeval {
        long    tv_sec;                 /* seconds */
        long    tv_usec;                /* and microseconds */
};

#define RUSAGE_SELF     0
#define RUSAGE_CHILDREN -1

struct rusage {
        struct timeval ru_utime;        /* user time used */
        struct timeval ru_stime;        /* system time used */
        long    ru_maxrss;
        long    ru_ixrss;               /* integral shared memory size */
        long    ru_idrss;               /* integral unshared data size */
        long    ru_isrss;               /* integral unshared stack size */
        long    ru_minflt;              /* page reclaims */
        long    ru_majflt;              /* page faults */
        long    ru_nswap;               /* swaps */
        long    ru_inblock;             /* block input operations */
        long    ru_oublock;             /* block output operations */
        long    ru_msgsnd;              /* messages sent */
        long    ru_msgrcv;              /* messages received */
        long    ru_nsignals;            /* signals received */
        long    ru_nvcsw;               /* voluntary context switches */
        long    ru_nivcsw;              /* involuntary context switches */
};

void _getrusage(int who, struct rusage *rusage);

#elif   defined(_POSIX_SOURCE) || defined(__USG)

struct tms {
        time_t  tms_utime;              /* user time */
        time_t  tms_stime;              /* system time */
        time_t  tms_cutime;             /* user time, children */
        time_t  tms_cstime;             /* system time, children */
```

```
};

clock_t _times(struct tms *buffer);

#else                                   /* Version 7 UNIX */

struct tbuffer {
        long proc_user_time;
        long proc_system_time;
        long child_user_time;
        long child_system_time;
};

clock_t _times(struct tbuffer *buffer);

#endif

clock_t
clock(void)
{
#if     defined(__BSD4_2)
        struct rusage rusage;

        _getrusage(RUSAGE_SELF, &rusage);

        return (((unsigned long)rusage.ru_utime.tv_sec * CLOCKS_PER_SEC)
                + rusage.ru_utime.tv_usec);
#elif   defined(_POSIX_SOURCE) || defined(__USG)
        struct tms tms;

        _times(&tms);
        /* Assume that time_t can be converted to clock_t for Sys5 */
        return tms.tms_utime;
#else
        struct tbuffer tbuffer;

        _times(&tbuffer);
        return tbuffer.proc_user_time;
#endif
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/ansi/ctime.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * ctime - convers the calendar time to a string
 */
/* $Header: ctime.c,v 1.1 89/06/12 15:21:16 eck Exp $ */

#include        <time.h>

char *
ctime(const time_t *timer)
{
```

```
        return asctime(localtime(timer));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/difftime.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * difftime - compute the difference between two calendar times
 */
/* $Header: difftime.c,v 1.4 90/09/11 10:18:44 eck Exp $ */

#include        <time.h>

double
difftime(time_t time1, time_t time0)
{
        /* be careful: time_t may be unsigned */
        if ((time_t)-1 > 0 && time0 > time1) {
                return - (double) (time0 - time1);
        } else {
                return (double)(time1 - time0);
        }
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/div.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: div.c,v 1.3 90/03/05 13:47:53 eck Exp $ */

#include        <stdlib.h>

static int tmp = -1;

div_t
div(register int numer, register int denom)
{
        div_t r;

        /* The assignment of tmp should not be optimized !! */
        if (tmp == -1) {
                tmp = (tmp / 2 == 0);
        }
        if (numer == 0) {
                r.quot = numer / denom;         /* might trap if denom == 0 */
                r.rem = numer % denom;
        } else if ( !tmp && ((numer < 0) != (denom < 0))) {
                r.quot = (numer / denom) + 1;
                r.rem = numer - (numer / denom + 1) * denom;
```

```
        } else {
                r.quot = numer / denom;
                r.rem = numer % denom;
        }
        return r;
}
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/errlist.c
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: errlist.c,v 1.1 89/05/10 16:22:20 eck Exp $ */

#include        <errno.h>

static const char unknown[] = "Unknown error";

const char *_sys_errlist[] = {
        "Error 0",                      /* EGENERIC */
        "Not owner",                    /* EPERM */
        "No such file or directory",    /* ENOENT */
        "No such process",              /* ESRCH */
        "Interrupted system call",      /* EINTR */
        "I/O error",                    /* EIO */
        "No such device or address",    /* ENXIO */
        "Arg list too long",            /* E2BIG */
        "Exec format error",            /* ENOEXEC */
        "Bad file number",              /* EBADF */
        "No children",                  /* ECHILD */
        "No more processes",            /* EAGAIN */
        "Not enough core",              /* ENOMEM */
        "Permission denied",            /* EACCES */
        "Bad address",                  /* EFAULT */
        "Block device required",        /* ENOTBLK */
        "Resource busy",                /* EBUSY */
        "File exists",                  /* EEXIST */
        "Cross-device link",            /* EXDEV */
        "No such device",               /* ENODEV */
        "Not a directory",              /* ENOTDIR */
        "Is a directory",               /* EISDIR */
        "Invalid argument",             /* EINVAL */
        "File table overflow",          /* ENFILE */
        "Too many open files",          /* EMFILE */
        "Not a typewriter",             /* ENOTTY */
        "Text file busy",               /* ETXTBSY */
        "File too large",               /* EFBIG */
        "No space left on device",      /* ENOSPC */
        "Illegal seek",                 /* ESPIPE */
        "Read-only file system",        /* EROFS */
        "Too many links",               /* EMLINK */
        "Broken pipe",                  /* EPIPE */
        "Math argument",                /* EDOM */
```

```
        "Result too large",             /* ERANGE */
        "Resource deadlock avoided",    /* EDEADLK */
        "File name too long",           /* ENAMETOOLONG */
        "No locks available",           /* ENOLCK */
        "Function not implemented",     /* ENOSYS */
        "Directory not empty",          /* ENOTEMPTY */
        unknown,                        /* 40 */
        unknown,                        /* 41 */
        unknown,                        /* 42 */
        unknown,                        /* 43 */
        unknown,                        /* 44 */
        unknown,                        /* 45 */
        unknown,                        /* 46 */
        unknown,                        /* 47 */
        unknown,                        /* 48 */
        unknown,                        /* 49 */
        "Invalid packet size",          /* EPACKSIZE */
        "Not enough buffers left",      /* EOUTOFBUFS */
        "Illegal ioctl for device",     /* EBADIOCTL */
        "Bad mode for ioctl",           /* EBADMODE */
        "Would block",                  /* EWOULDBLOCK */
        "Bad destination address",      /* EBADDEST */
        "Destination not reachable",    /* EDSTNOTRCH */
        "Already connected",            /* EISCONN */
        "Address in use",               /* EADDRINUSE */
        "Connection refused",           /* ECONNREFUSED */
        "Connection reset",             /* ECONNRESET */
        "Connection timed out",         /* ETIMEDOUT */
        "Urgent data present",          /* EURG */
        "No urgent data present",       /* ENOURG */
        "No connection",                /* ENOTCONN */
        "Already shutdown",             /* ESHUTDOWN */
        "No such connection",           /* ENOCONN */
};

const int _sys_nerr = sizeof(_sys_errlist) / sizeof(_sys_errlist[0]);
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/exit.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: exit.c,v 1.3 90/01/22 13:00:04 eck Exp $ */

#include        <stdio.h>
#include        <stdlib.h>

#define NEXITS  32

void (*__functab[NEXITS])(void);
int __funccnt = 0;

extern void _exit(int);
```

```

/* only flush output buffers when necessary */
int (*_clean)(void) = NULL;

static void
_calls(void)
{
        register int i = __funccnt;

        /* "Called in reversed order of their registration" */
        while (--i >= 0)
                (*__functab[i])();
}

void
exit(int status)
{
        _calls();
        if (_clean) _clean();
        _exit(status) ;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/ext_comp.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1989 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Id: ext_comp.c,v 1.10 1994/06/24 11:53:36 ceriel Exp $ */

/* extended precision arithmetic for the strtod() and cvt() routines */

/* This may require some more work when long doubles get bigger than 8
   bytes. In this case, these routines may become obsolete. ???
*/

#include        "ext_fmt.h"
#include        <float.h>
#include        <errno.h>
#include        <ctype.h>

static int b64_add(struct mantissa *e1, struct mantissa *e2);
static b64_sft(struct mantissa *e1, int n);

static
mul_ext(struct EXTEND *e1, struct EXTEND *e2, struct EXTEND *e3)
{
        /*      Multiply the extended numbers e1 and e2, and put the
                result in e3.
        */
        register int    i,j;            /* loop control */
        unsigned short  mp[4];
        unsigned short  mc[4];
```

```
        unsigned short  result[8];      /* result */

        register unsigned short *pres;

        /* first save the sign (XOR)                    */
        e3->sign = e1->sign ^ e2->sign;

        /* compute new exponent */
        e3->exp = e1->exp + e2->exp + 1;

        /* check for overflow/underflow ??? */

        /* 128 bit multiply of mantissas        */

        /* assign unknown long formats          */
        /* to known unsigned word formats       */
        mp[0] = e1->m1 >> 16;
        mp[1] = (unsigned short) e1->m1;
        mp[2] = e1->m2 >> 16;
        mp[3] = (unsigned short) e1->m2;
        mc[0] = e2->m1 >> 16;
        mc[1] = (unsigned short) e2->m1;
        mc[2] = e2->m2 >> 16;
        mc[3] = (unsigned short) e2->m2;
        for (i = 8; i--;) {
                result[i] = 0;
        }
        /*
         *      fill registers with their components
         */
        for(i=4, pres = &result[4];i--;pres--) if (mp[i]) {
                unsigned short k = 0;
                unsigned long mpi = mp[i];
                for(j=4;j--;) {
                        unsigned long tmp = (unsigned long)pres[j] + k;
                        if (mc[j]) tmp += mpi * mc[j];
                        pres[j] = tmp;
                        k = tmp >> 16;
                }
                pres[-1] = k;
        }

        if (! (result[0] & 0x8000)) {
                e3->exp--;
                for (i = 0; i <= 3; i++) {
                        result[i] <<= 1;
                        if (result[i+1]&0x8000) result[i] |= 1;
                }
                result[4] <<= 1;
        }
        /*
         *      combine the registers to a total
         */
        e3->m1 = ((unsigned long)(result[0]) << 16) + result[1];
        e3->m2 = ((unsigned long)(result[2]) << 16) + result[3];
        if (result[4] & 0x8000) {
                if (++e3->m2 == 0) {
                        if (++e3->m1 == 0) {
                                e3->m1 = 0x80000000;
                                e3->exp++;
```

```
                        }
                }
        }
}

static
add_ext(struct EXTEND *e1, struct EXTEND *e2, struct EXTEND *e3)
{
        /*      Add two extended numbers e1 and e2, and put the result
                in e3
        */
        struct EXTEND ce2;
        int diff;

        if ((e2->m1 | e2->m2) == 0L) {
                *e3 = *e1;
                return;
        }
        if ((e1->m1 | e1->m2) == 0L) {
                *e3 = *e2;
                return;
        }
        ce2 = *e2;
        *e3 = *e1;
        e1 = &ce2;

        /* adjust mantissas to equal power */
        diff = e3->exp - e1->exp;
        if (diff < 0) {
                diff = -diff;
                e3->exp += diff;
                b64_sft(&(e3->mantissa), diff);
        }
        else if (diff > 0) {
                e1->exp += diff;
                b64_sft(&(e1->mantissa), diff);
        }
        if (e1->sign != e3->sign) {
                /* e3 + e1 = e3 - (-e1) */
                if (e1->m1 > e3->m1 ||
                    (e1->m1 == e3->m1 && e1->m2 > e3->m2)) {
                        /*      abs(e1) > abs(e3) */
                        if (e3->m2 > e1->m2) {
                                e1->m1 -= 1;    /* carry in */
                        }
                        e1->m1 -= e3->m1;
                        e1->m2 -= e3->m2;
                        *e3 = *e1;
                }
                else {
                        if (e1->m2 > e3->m2)
                                e3->m1 -= 1;    /* carry in */
                        e3->m1 -= e1->m1;
                        e3->m2 -= e1->m2;
                }
        }
        else {
                if (b64_add(&e3->mantissa,&e1->mantissa)) {/* addition carry */
                        b64_sft(&e3->mantissa,1);/* shift mantissa one bit RIGHT */
                        e3->m1 |= 0x80000000L;  /* set max bit  */
```

```
				e3->exp++;		/* increase the exponent */
			}
		}
		if ((e3->m2 | e3->m1) != 0L) {
			/* normalize */
			if (e3->m1 == 0L) {
				e3->m1 = e3->m2; e3->m2 = 0L; e3->exp -= 32;
			}
			if (!(e3->m1 & 0x80000000)) {
				unsigned long l = 0x40000000;
				int cnt = -1;

				while (! (l & e3->m1)) {
					l >>= 1; cnt--;
				}
				e3->exp += cnt;
				b64_sft(&(e3->mantissa), cnt);
			}
		}
}

static int
cmp_ext(struct EXTEND *e1, struct EXTEND *e2)
{
		struct EXTEND tmp;

		e2->sign = ! e2->sign;
		add_ext(e1, e2, &tmp);
		e2->sign = ! e2->sign;
		if (tmp.m1 == 0 && tmp.m2 == 0) return 0;
		if (tmp.sign) return -1;
		return 1;
}

static
b64_sft(struct mantissa *e1, int n)
{
		if (n > 0) {
			if (n > 63) {
				e1->l_32 = 0;
				e1->h_32 = 0;
				return;
			}
			if (n >= 32) {
				e1->l_32 = e1->h_32;
				e1->h_32 = 0;
				n -= 32;
			}
			if (n > 0) {
				e1->l_32 >>= n;
				if (e1->h_32 != 0) {
					e1->l_32 |= (e1->h_32 << (32 - n));
					e1->h_32 >>= n;
				}
			}
			return;
		}
		n = -n;
		if (n > 0) {
			if (n > 63) {
```

```
                        e1->l_32 = 0;
                        e1->h_32 = 0;
                        return;
                }
                if (n >= 32) {
                        e1->h_32 = e1->l_32;
                        e1->l_32 = 0;
                        n -= 32;
                }
                if (n > 0) {
                        e1->h_32 <<= n;
                        if (e1->l_32 != 0) {
                                e1->h_32 |= (e1->l_32 >> (32 - n));
                                e1->l_32 <<= n;
                        }
                }
        }
}

static int
b64_add(struct mantissa *e1, struct mantissa *e2)
                /*
                 * pointers to 64 bit 'registers'
                 */
{
        register int    overflow;
        int             carry;

                        /* add higher pair of 32 bits */
        overflow = ((unsigned long) 0xFFFFFFFF - e1->h_32 < e2->h_32);
        e1->h_32 += e2->h_32;

                        /* add lower pair of 32 bits */
        carry = ((unsigned long) 0xFFFFFFFF - e1->l_32 < e2->l_32);
        e1->l_32 += e2->l_32;
        if ((carry) && (++e1->h_32 == 0))
                return(1);              /* had a 64 bit overflow */
        else
                return(overflow);       /* return status from higher add */
}

/* The following tables can be computed with the following bc(1)
   program:

obase=16
scale=0
define t(x){
        auto a, b, c
        a=2;b=1;c=2^32;n=1
        while(a<x) {
                b=a;n+=n;a*=a
        }
        n/=2
        a=b
        while(b<x) {
                a=b;b*=c;n+=32
        }
        n-=32
        b=a
        while(a<x) {
```

```
                b=a;a+=a;n+=1
        }
        n-=1
        x*=16^16
        b=x%a
        x/=a
        if(a<=(2*b)) x+=1
        obase=10
        n
        obase=16
        return(x)
}
for (i=1;i<28;i++) {
        t(10^i)
}
0
for (i=1;i<20;i++) {
        t(10^(28*i))
}
0
define r(x){
        auto a, b, c
        a=2;b=1;c=2^32;n=1
        while(a<x) {
                b=a;n+=n;a*=a
        }
        n/=2
        a=b
        while(b<x) {
                a=b;b*=c;n+=32
        }
        n-=32
        b=a
        while(a<x) {
                b=a;a+=a;n+=1
        }
        a=b
        a*=16^16
        b=a%x
        a/=x
        if(x<=(2*b)) a+=1
        obase=10
        -n
        obase=16
        return(a)
}
for (i=1;i<28;i++) {
        r(10^i)
}
0
for (i=1;i<20;i++) {
        r(10^(28*i))
}
0

*/
static struct EXTEND ten_powers[] = {   /* representation of 10 ** i */
        { 0,    0,      0x80000000,     0 },
        { 0,    3,      0xA0000000,     0 },
        { 0,    6,      0xC8000000,     0 },
```

```
        { 0,    9,      0xFA000000,     0 },
        { 0,    13,     0x9C400000,     0 },
        { 0,    16,     0xC3500000,     0 },
        { 0,    19,     0xF4240000,     0 },
        { 0,    23,     0x98968000,     0 },
        { 0,    26,     0xBEBC2000,     0 },
        { 0,    29,     0xEE6B2800,     0 },
        { 0,    33,     0x9502F900,     0 },
        { 0,    36,     0xBA43B740,     0 },
        { 0,    39,     0xE8D4A510,     0 },
        { 0,    43,     0x9184E72A,     0 },
        { 0,    46,     0xB5E620F4,     0x80000000 },
        { 0,    49,     0xE35FA931,     0xA0000000 },
        { 0,    53,     0x8E1BC9BF,     0x04000000 },
        { 0,    56,     0xB1A2BC2E,     0xC5000000 },
        { 0,    59,     0xDE0B6B3A,     0x76400000 },
        { 0,    63,     0x8AC72304,     0x89E80000 },
        { 0,    66,     0xAD78EBC5,     0xAC620000 },
        { 0,    69,     0xD8D726B7,     0x177A8000 },
        { 0,    73,     0x87867832,     0x6EAC9000 },
        { 0,    76,     0xA968163F,     0x0A57B400 },
        { 0,    79,     0xD3C21BCE,     0xCCEDA100 },
        { 0,    83,     0x84595161,     0x401484A0 },
        { 0,    86,     0xA56FA5B9,     0x9019A5C8 },
        { 0,    89,     0xCECB8F27,     0xF4200F3A }
};
static struct EXTEND big_ten_powers[] = {  /* representation of 10 ** (28*i) */
        { 0,    0,      0x80000000,     0 },
        { 0,    93,     0x813F3978,     0xF8940984 },
        { 0,    186,    0x82818F12,     0x81ED44A0 },
        { 0,    279,    0x83C7088E,     0x1AAB65DB },
        { 0,    372,    0x850FADC0,     0x9923329E },
        { 0,    465,    0x865B8692,     0x5B9BC5C2 },
        { 0,    558,    0x87AA9AFF,     0x79042287 },
        { 0,    651,    0x88FCF317,     0xF22241E2 },
        { 0,    744,    0x8A5296FF,     0xE33CC930 },
        { 0,    837,    0x8BAB8EEF,     0xB6409C1A },
        { 0,    930,    0x8D07E334,     0x55637EB3 },
        { 0,    1023,   0x8E679C2F,     0x5E44FF8F },
        { 0,    1116,   0x8FCAC257,     0x558EE4E6 },
        { 0,    1209,   0x91315E37,     0xDB165AA9 },
        { 0,    1302,   0x929B7871,     0xDE7F22B9 },
        { 0,    1395,   0x940919BB,     0xD4620B6D },
        { 0,    1488,   0x957A4AE1,     0xEBF7F3D4 },
        { 0,    1581,   0x96EF14C6,     0x454AA840 },
        { 0,    1674,   0x98678061,     0x27ECE4F5 },
        { 0,    1767,   0x99E396C1,     0x3A3ACFF2 }
};

static struct EXTEND r_ten_powers[] = { /* representation of 10 ** -i */
        { 0,    0,      0x80000000,     0 },
        { 0,    -4,     0xCCCCCCCC,     0xCCCCCCCD },
        { 0,    -7,     0xA3D70A3D,     0x70A3D70A },
        { 0,    -10,    0x83126E97,     0x8D4FDF3B },
        { 0,    -14,    0xD1B71758,     0xE219652C },
        { 0,    -17,    0xA7C5AC47,     0x1B478423 },
        { 0,    -20,    0x8637BD05,     0xAF6C69B6 },
        { 0,    -24,    0xD6BF94D5,     0xE57A42BC },
        { 0,    -27,    0xABCC7711,     0x8461CEFD },
        { 0,    -30,    0x89705F41,     0x36B4A597 },
```

```
        { 0,    -34,    0xDBE6FECE,     0xBDEDD5BF },
        { 0,    -37,    0xAFEBFF0B,     0xCB24AAFF },
        { 0,    -40,    0x8CBCCC09,     0x6F5088CC },
        { 0,    -44,    0xE12E1342,     0x4BB40E13 },
        { 0,    -47,    0xB424DC35,     0x095CD80F },
        { 0,    -50,    0x901D7CF7,     0x3AB0ACD9 },
        { 0,    -54,    0xE69594BE,     0xC44DE15B },
        { 0,    -57,    0xB877AA32,     0x36A4B449 },
        { 0,    -60,    0x9392EE8E,     0x921D5D07 },
        { 0,    -64,    0xEC1E4A7D,     0xB69561A5 },
        { 0,    -67,    0xBCE50864,     0x92111AEB },
        { 0,    -70,    0x971DA050,     0x74DA7BEF },
        { 0,    -74,    0xF1C90080,     0xBAF72CB1 },
        { 0,    -77,    0xC16D9A00,     0x95928A27 },
        { 0,    -80,    0x9ABE14CD,     0x44753B53 },
        { 0,    -84,    0xF79687AE,     0xD3EEC551 },
        { 0,    -87,    0xC6120625,     0x76589DDB },
        { 0,    -90,    0x9E74D1B7,     0x91E07E48 }
};

static struct EXTEND r_big_ten_powers[] = { /* representation of 10 ** -(28*i) */
        { 0,    0,      0x80000000,     0 },
        { 0,    -94,    0xFD87B5F2,     0x8300CA0E },
        { 0,    -187,   0xFB158592,     0xBE068D2F },
        { 0,    -280,   0xF8A95FCF,     0x88747D94 },
        { 0,    -373,   0xF64335BC,     0xF065D37D },
        { 0,    -466,   0xF3E2F893,     0xDEC3F126 },
        { 0,    -559,   0xF18899B1,     0xBC3F8CA2 },
        { 0,    -652,   0xEF340A98,     0x172AACE5 },
        { 0,    -745,   0xECE53CEC,     0x4A314EBE },
        { 0,    -838,   0xEA9C2277,     0x23EE8BCB },
        { 0,    -931,   0xE858AD24,     0x8F5C22CA },
        { 0,    -1024,  0xE61ACF03,     0x3D1A45DF },
        { 0,    -1117,  0xE3E27A44,     0x4D8D98B8 },
        { 0,    -1210,  0xE1AFA13A,     0xFBD14D6E },
        { 0,    -1303,  0xDF82365C,     0x497B5454 },
        { 0,    -1396,  0xDD5A2C3E,     0xAB3097CC },
        { 0,    -1489,  0xDB377599,     0xB6074245 },
        { 0,    -1582,  0xD91A0545,     0xCDB51186 },
        { 0,    -1675,  0xD701CE3B,     0xD387BF48 },
        { 0,    -1768,  0xD4EEC394,     0xD6258BF8 }
};

#define TP      (int)(sizeof(ten_powers)/sizeof(ten_powers[0]))
#define BTP     (int)(sizeof(big_ten_powers)/sizeof(big_ten_powers[0]))
#define MAX_EXP (TP * BTP - 1)

static
add_exponent(struct EXTEND *e, int exp)
{
        int neg = exp < 0;
        int divsz, modsz;
        struct EXTEND x;

        if (neg) exp = -exp;
        divsz = exp / TP;
        modsz = exp % TP;
        if (neg) {
                mul_ext(e, &r_ten_powers[modsz], &x);
                mul_ext(&x, &r_big_ten_powers[divsz], e);
```

```
        }
        else {
                mul_ext(e, &ten_powers[modsz], &x);
                mul_ext(&x, &big_ten_powers[divsz], e);
        }
}

_str_ext_cvt(const char *s, char **ss, struct EXTEND *e)
{
        /*      Like strtod, but for extended precision */
        register int    c;
        int             dotseen = 0;
        int             digitseen = 0;
        int             exp = 0;

        if (ss) *ss = (char *)s;
        while (isspace(*s)) s++;

        e->sign = 0;
        e->exp = 0;
        e->m1 = e->m2 = 0;

        c = *s;
        switch(c) {
        case '-':
                e->sign = 1;
        case '+':
                s++;
        }
        while (c = *s++, isdigit(c) || (c == '.' && ! dotseen++)) {
                if (c == '.') continue;
                digitseen = 1;
                if (e->m1 <= (unsigned long)(0xFFFFFFFF)/10) {
                        struct mantissa a1;

                        a1 = e->mantissa;
                        b64_sft(&(e->mantissa), -3);
                        b64_sft(&a1, -1);
                        b64_add(&(e->mantissa), &a1);
                        a1.h_32 = 0;
                        a1.l_32 = c - '0';
                        b64_add(&(e->mantissa), &a1);
                }
                else exp++;
                if (dotseen) exp--;
        }
        if (! digitseen) return;

        if (ss) *ss = (char *)s - 1;

        if (c == 'E' || c == 'e') {
                int     exp1 = 0;
                int     sign = 1;
                int     exp_overflow = 0;

                switch(*s) {
                case '-':
                        sign = -1;
                case '+':
                        s++;
```

```
                }
                if (c = *s, isdigit(c)) {
                        do {
                                int tmp;

                                exp1 = 10 * exp1 + (c - '0');
                                if ((tmp = sign * exp1 + exp) > MAX_EXP ||
                                     tmp < -MAX_EXP) {
                                        exp_overflow = 1;
                                }
                        } while (c = *++s, isdigit(c));
                        if (ss) *ss = (char *)s;
                }
                exp += sign * exp1;
                if (exp_overflow) {
                        exp = sign * MAX_EXP;
                        if (e->m1 != 0 || e->m2 != 0) errno = ERANGE;
                }
        }
        if (e->m1 == 0 && e->m2 == 0) return;
        e->exp = 63;
        while (! (e->m1 & 0x80000000)) {
                b64_sft(&(e->mantissa),-1);
                e->exp--;
        }
        add_exponent(e, exp);
}

#include        <math.h>

static
ten_mult(struct EXTEND *e)
{
        struct EXTEND e1 = *e;

        e1.exp++;
        e->exp += 3;
        add_ext(e, &e1, e);
}

#define NDIGITS 128
#define NSIGNIFICANT 19

char *
_ext_str_cvt(struct EXTEND *e, int ndigit, int *decpt, int *sign, int ecvtflag)
{
        /*      Like cvt(), but for extended precision */

        static char buf[NDIGITS+1];
        struct EXTEND m;
        register char *p = buf;
        register char *pe;
        int findex = 0;

        if (ndigit < 0) ndigit = 0;
        if (ndigit > NDIGITS) ndigit = NDIGITS;
        pe = &buf[ndigit];
        buf[0] = '\0';

        *sign = 0;
```

```
        if (e->sign) {
                *sign = 1;
                e->sign = 0;
        }

        *decpt = 0;
        if (e->m1 != 0) {
                register struct EXTEND *pp = &big_ten_powers[1];

                while(cmp_ext(e,pp) >= 0) {
                        pp++;
                        findex = pp - big_ten_powers;
                        if (findex >= BTP) break;
                }
                pp--;
                findex = pp - big_ten_powers;
                mul_ext(e,&r_big_ten_powers[findex],e);
                *decpt += findex * TP;
                pp = &ten_powers[1];
                while(pp < &ten_powers[TP] && cmp_ext(e, pp) >= 0) pp++;
                pp--;
                findex = pp - ten_powers;
                *decpt += findex;

                if (cmp_ext(e, &ten_powers[0]) < 0) {
                        pp = &r_big_ten_powers[1];
                        while(cmp_ext(e,pp) < 0) pp++;
                        pp--;
                        findex = pp - r_big_ten_powers;
                        mul_ext(e, &big_ten_powers[findex], e);
                        *decpt -= findex * TP;
                        /* here, value >= 10 ** -28 */
                        ten_mult(e);
                        (*decpt)--;
                        pp = &r_ten_powers[0];
                        while(cmp_ext(e, pp) < 0) pp++;
                        findex = pp - r_ten_powers;
                        mul_ext(e, &ten_powers[findex], e);
                        *decpt -= findex;
                        findex = 0;
                }
                (*decpt)++;     /* because now value in [1.0, 10.0) */
        }
        if (! ecvtflag) {
                /* for fcvt() we need ndigit digits behind the dot */
                pe += *decpt;
                if (pe > &buf[NDIGITS]) pe = &buf[NDIGITS];
        }
        m.exp = -62;
        m.sign = 0;
        m.m1 = 0xA0000000;
        m.m2 = 0;
        while (p <= pe) {
                struct EXTEND oneminm;

                if (p - pe > NSIGNIFICANT) {
                        findex = 0;
                        e->m1 = 0;
                }
                if (findex) {
```

```
			struct EXTEND tc, oldtc;
			int count = 0;

			oldtc.exp = 0;
			oldtc.sign = 0;
			oldtc.m1 = 0;
			oldtc.m2 = 0;
			tc = ten_powers[findex];
			while (cmp_ext(e, &tc) >= 0) {
				oldtc = tc;
				add_ext(&tc, &ten_powers[findex], &tc);
				count++;
			}
			*p++ = count + '0';
			oldtc.sign = 1;
			add_ext(e, &oldtc, e);
			findex--;
			continue;
		}
		if (e->m1) {
			m.sign = 1;
			add_ext(&ten_powers[0], &m, &oneminm);
			m.sign = 0;
			if (e->exp >= 0) {
				struct EXTEND x;

				x.m2 = 0; x.exp = e->exp;
				x.sign = 1;
				x.m1 = e->m1>>(31-e->exp);
				*p++ = (x.m1) + '0';
				x.m1 = x.m1 << (31-e->exp);
				add_ext(e, &x, e);
			}
			else *p++ = '0';
			/* Check that remainder is still significant */
			if (cmp_ext(&m, e) > 0 || cmp_ext(e, &oneminm) > 0) {
				if (e->m1 && e->exp >= -1) *(p-1) += 1;
				e->m1 = 0;
				continue;
			}
			ten_mult(&m);
			ten_mult(e);
		}
		else *p++ = '0';
	}
	if (pe >= buf) {
		p = pe;
		*p += 5;	/* round of at the end */
		while (*p > '9') {
			*p = '0';
			if (p > buf) ++*--p;
			else {
				*p = '1';
				++*decpt;
				if (! ecvtflag) {
					/* maybe add another digit at the end,
					   because the point was shifted right
					*/
					if (pe > buf) *pe = '0';
					pe++;
```

```
                                }
                        }
                }
                *pe = '\0';
        }
        return buf;
}

_dbl_ext_cvt(double value, struct EXTEND *e)
{
        /*      Convert double to extended
        */
        int exponent;

        value = frexp(value, &exponent);
        e->sign = value < 0.0;
        if (e->sign) value = -value;
        e->exp = exponent - 1;
        value *= 4294967296.0;
        e->m1 = value;
        value -= e->m1;
        value *= 4294967296.0;
        e->m2 = value;
}

static struct EXTEND max_d;

double
_ext_dbl_cvt(struct EXTEND *e)
{
        /*      Convert extended to double
        */
        double f;
        int sign = e->sign;

        e->sign = 0;
        if (e->m1 == 0 && e->m2 == 0) {
                return 0.0;
        }
        if (max_d.exp == 0) {
                _dbl_ext_cvt(DBL_MAX, &max_d);
        }
        if (cmp_ext(&max_d, e) < 0) {
                f = HUGE_VAL;
                errno = ERANGE;
        }
        else    f = ldexp((double)e->m1*4294967296.0 + (double)e->m2, e->exp-63);
        if (sign) f = -f;
        if (f == 0.0 && (e->m1 != 0 || e->m2 != 0)) {
                errno = ERANGE;
        }
        return f;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/getenv.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: getenv.c,v 1.4 91/04/24 12:18:03 ceriel Exp $ */

#include        <stdlib.h>

extern const char **_penvp;

char *
getenv(const char *name)
{
        register const char **v = _penvp;
        register const char *p, *q;

        if (v == NULL || name == NULL)
                return (char *)NULL;
        while ((p = *v++) != NULL) {
                q = name;
                while (*q && (*q == *p++))
                        q++;
                if (*q || (*p != '='))
                        continue;
                return (char *)p + 1;
        }
        return (char *)NULL;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/gmtime.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * gmtime - convert the calendar time into broken down time
 */
/* $Header: gmtime.c,v 1.4 91/04/22 13:20:27 ceriel Exp $ */

#include        <time.h>
#include        <limits.h>
#include        "loc_time.h"

struct tm *
gmtime(register const time_t *timer)
{
        static struct tm br_time;
        register struct tm *timep = &br_time;
        time_t time = *timer;
        register unsigned long dayclock, dayno;
        int year = EPOCH_YR;

        dayclock = (unsigned long)time % SECS_DAY;
        dayno = (unsigned long)time / SECS_DAY;
```

```

        timep->tm_sec = dayclock % 60;
        timep->tm_min = (dayclock % 3600) / 60;
        timep->tm_hour = dayclock / 3600;
        timep->tm_wday = (dayno + 4) % 7;       /* day 0 was a thursday */
        while (dayno >= YEARSIZE(year)) {
                dayno -= YEARSIZE(year);
                year++;
        }
        timep->tm_year = year - YEAR0;
        timep->tm_yday = dayno;
        timep->tm_mon = 0;
        while (dayno >= _ytab[LEAPYEAR(year)][timep->tm_mon]) {
                dayno -= _ytab[LEAPYEAR(year)][timep->tm_mon];
                timep->tm_mon++;
        }
        timep->tm_mday = dayno + 1;
        timep->tm_isdst = 0;

        return timep;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/isalnum.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include        <ctype.h>

int (isalnum)(int c) {
        return isalnum(c);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/isalpha.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include        <ctype.h>

int (isalpha)(int c) {
        return isalpha(c);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/ansi/isascii.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include        <ctype.h>

int (isascii)(int c) {
        return isascii(c);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/ansi/iscntrl.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include        <ctype.h>

int (iscntrl)(int c) {
        return iscntrl(c);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/ansi/isdigit.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include        <ctype.h>

int (isdigit)(int c) {
        return isdigit(c);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/ansi/isgraph.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include        <ctype.h>

int (isgraph)(int c) {
        return isgraph(c);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ansi/islower.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include        <ctype.h>

int (islower)(int c) {
        return islower(c);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ansi/isprint.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include        <ctype.h>

int (isprint)(int c) {
        return isprint(c);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ansi/ispunct.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include        <ctype.h>

int (ispunct)(int c) {
        return ispunct(c);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ansi/isspace.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include        <ctype.h>

int (isspace)(int c) {
        return isspace(c);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/isupper.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include        <ctype.h>

int (isupper)(int c) {
        return isupper(c);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/isxdigit.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include        <ctype.h>

int (isxdigit)(int c) {
        return isxdigit(c);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/labs.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: labs.c,v 1.1 89/05/16 13:08:11 eck Exp $ */

#include        <stdlib.h>

long
labs(register long l)
{
        return l >= 0 ? l : -l;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/ldiv.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: ldiv.c,v 1.3 90/03/05 13:48:03 eck Exp $ */

#include        <stdlib.h>

static long tmp = -1;

ldiv_t
ldiv(register long numer, register long denom)
{
        ldiv_t r;

        /* The assignment of tmp should not be optimized !! */
        if (tmp == -1) {
                tmp = (tmp / 2 == 0);
        }
        if (numer == 0) {
                r.quot = numer / denom;         /* might trap if denom == 0 */
                r.rem = numer % denom;
        } else if ( !tmp && ((numer < 0) != (denom < 0))) {
                r.quot = (numer / denom) + 1;
                r.rem = numer - (numer / denom + 1) * denom;
        } else {
                r.quot = numer / denom;
                r.rem = numer % denom;
        }
        return r;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/localeconv.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * localeconv - set components of a struct according to current locale
 */
/* $Header: localeconv.c,v 1.2 89/12/18 15:48:58 eck Exp $ */

#include        <limits.h>
#include        <locale.h>

extern struct lconv _lc;

struct lconv *
localeconv(void)
{
        register struct lconv *lcp = &_lc;

```

```
        lcp->decimal_point = ".";
        lcp->thousands_sep = "";
        lcp->grouping = "";
        lcp->int_curr_symbol = "";
        lcp->currency_symbol = "";
        lcp->mon_decimal_point = "";
        lcp->mon_thousands_sep = "";
        lcp->mon_grouping = "";
        lcp->positive_sign = "";
        lcp->negative_sign = "";
        lcp->int_frac_digits = CHAR_MAX;
        lcp->frac_digits = CHAR_MAX;
        lcp->p_cs_precedes = CHAR_MAX;
        lcp->p_sep_by_space = CHAR_MAX;
        lcp->n_cs_precedes = CHAR_MAX;
        lcp->n_sep_by_space = CHAR_MAX;
        lcp->p_sign_posn = CHAR_MAX;
        lcp->n_sign_posn = CHAR_MAX;

        return lcp;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ansi/localtime.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * localtime - convert a calendar time into broken down time
 */
/* $Header: localtime.c,v 1.3 91/04/22 13:20:36 ceriel Exp $ */

#include        <time.h>
#include        "loc_time.h"

/* We must be careful, since an int can't represent all the seconds in a day.
 * Hence the adjustment of minutes when adding timezone and dst information.
 * This assumes that both must be expressable in multiples of a minute.
 * Furthermore, it is assumed that both fit into an integer when expressed as
 * minutes (this is about 22 days, so this should not cause any problems).
 */
struct tm *
localtime(const time_t *timer)
{
        struct tm *timep;
        unsigned dst;

        _tzset();
        timep = gmtime(timer);                  /* tm->tm_isdst == 0 */
        timep->tm_min -= _timezone / 60;
        timep->tm_sec -= _timezone % 60;
        mktime(timep);

        dst = _dstget(timep);
        if (dst) {
                timep->tm_min += dst / 60;
                timep->tm_sec += dst % 60;
```

```
                mktime(timep);
        }
        return timep;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ansi/malloc.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/* $Header$ */

/* replace undef by define */
#undef   DEBUG          /* check assertions */
#undef   SLOWDEBUG      /* some extra test loops (requires DEBUG) */

#include        <stdlib.h>
#include        <string.h>
#include        <errno.h>

#ifdef DEBUG
#define ASSERT(b)       if (!(b)) assert_failed();
#else
#define ASSERT(b)       /* empty */
#endif

#if _EM_WSIZE == _EM_PSIZE
#define ptrint          int
#else
#define ptrint          long
#endif

#if     _EM_PSIZE == 2
#define BRKSIZE         1024
#else
#define BRKSIZE         4096
#endif
#define PTRSIZE         ((int) sizeof(void *))
#define Align(x,a)      (((x) + (a - 1)) & ~(a - 1))
#define NextSlot(p)     (* (void **) ((p) - PTRSIZE))
#define NextFree(p)     (* (void **) (p))

/*
 * A short explanation of the data structure and algorithms.
 * An area returned by malloc() is called a slot. Each slot
 * contains the number of bytes requested, but preceeded by
 * an extra pointer to the next the slot in memory.
 * '_bottom' and '_top' point to the first/last slot.
 * More memory is asked for using brk() and appended to top.
 * The list of free slots is maintained to keep malloc() fast.
 * '_empty' points the the first free slot. Free slots are
 * linked together by a pointer at the start of the
 * user visable part, so just after the next-slot pointer.
 * Free slots are merged together by free().
 */

extern void *_sbrk(int);
extern int _brk(void *);
static void *_bottom, *_top, *_empty;

```

```
static int grow(size_t len)
{
  register char *p;

  ASSERT(NextSlot((char *)_top) == 0);
  errno = ENOMEM;
  if ((char *) _top + len < (char *) _top
      || (p = (char *)Align((ptrint)_top + len, BRKSIZE)) < (char *) _top
      || _brk(p) != 0)
        return(0);
  NextSlot((char *)_top) = p;
  NextSlot(p) = 0;
  free(_top);
  _top = p;
  return 1;
}

void *
malloc(size_t size)
{
  register char *prev, *p, *next, *new;
  register unsigned len, ntries;

  if (size == 0) return NULL;
  errno = ENOMEM;
  for (ntries = 0; ntries < 2; ntries++) {
        if ((len = Align(size, PTRSIZE) + PTRSIZE) < 2 * PTRSIZE)
                return NULL;
        if (_bottom == 0) {
                if ((p = _sbrk(2 * PTRSIZE)) == (char *) -1)
                        return NULL;
                p = (char *) Align((ptrint)p, PTRSIZE);
                p += PTRSIZE;
                _top = _bottom = p;
                NextSlot(p) = 0;
        }
#ifdef SLOWDEBUG
        for (p = _bottom; (next = NextSlot(p)) != 0; p = next)
                ASSERT(next > p);
        ASSERT(p == _top);
#endif
        for (prev = 0, p = _empty; p != 0; prev = p, p = NextFree(p)) {
                next = NextSlot(p);
                new = p + len;  /* easily overflows!! */
                if (new > next || new <= p)
                        continue;               /* too small */
                if (new + PTRSIZE < next) {     /* too big, so split */
                        /* + PTRSIZE avoids tiny slots on free list */
                        NextSlot(new) = next;
                        NextSlot(p) = new;
                        NextFree(new) = NextFree(p);
                        NextFree(p) = new;
                }
                if (prev)
                        NextFree(prev) = NextFree(p);
                else
                        _empty = NextFree(p);
                return p;
        }
        if (grow(len) == 0)
```

```
                        break;
          }
          ASSERT(ntries != 2);
          return NULL;
        }

        void *
        realloc(void *oldp, size_t size)
        {
          register char *prev, *p, *next, *new;
          char *old = oldp;
          register size_t len, n;

          if (!old) return malloc(size);
          else if (!size) {
                free(oldp);
                return NULL;
          }
          len = Align(size, PTRSIZE) + PTRSIZE;
          next = NextSlot(old);
          n = (int)(next - old);                        /* old length */
          /*
           * extend old if there is any free space just behind it
           */
          for (prev = 0, p = _empty; p != 0; prev = p, p = NextFree(p)) {
                if (p > next)
                        break;
                if (p == next) {        /* 'next' is a free slot: merge */
                        NextSlot(old) = NextSlot(p);
                        if (prev)
                                NextFree(prev) = NextFree(p);
                        else
                                _empty = NextFree(p);
                        next = NextSlot(old);
                        break;
                }
          }
          new = old + len;
          /*
           * Can we use the old, possibly extended slot?
           */
          if (new <= next && new >= old) {              /* it does fit */
                if (new + PTRSIZE < next) {             /* too big, so split */
                        /* + PTRSIZE avoids tiny slots on free list */
                        NextSlot(new) = next;
                        NextSlot(old) = new;
                        free(new);
                }
                return old;
          }
          if ((new = malloc(size)) == NULL)             /* it didn't fit */
                return NULL;
          memcpy(new, old, n);                          /* n < size */
          free(old);
          return new;
        }

        void
        free(void *ptr)
        {
```

```c
  register char *prev, *next;
  char *p = ptr;

  if (!p) return;

  ASSERT(NextSlot(p) > p);
  for (prev = 0, next = _empty; next != 0; prev = next, next = NextFree(next))
        if (p < next)
                break;
  NextFree(p) = next;
  if (prev)
        NextFree(prev) = p;
  else
        _empty = p;
  if (next) {
        ASSERT(NextSlot(p) <= next);
        if (NextSlot(p) == next) {              /* merge p and next */
                NextSlot(p) = NextSlot(next);
                NextFree(p) = NextFree(next);
        }
  }
  if (prev) {
        ASSERT(NextSlot(prev) <= p);
        if (NextSlot(prev) == p) {              /* merge prev and p */
                NextSlot(prev) = NextSlot(p);
                NextFree(prev) = NextFree(p);
        }
  }
}

#ifdef DEBUG
static assert_failed()
{
        write(2, "assert failed in lib/malloc.c\n", 30);
        abort();
}
#endif
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ansi/mblen.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: mblen.c,v 1.2 89/12/18 15:12:50 eck Exp $ */

#include        <stdlib.h>
#include        <limits.h>

#define CHAR_SHIFT      8

int
mblen(const char *s, size_t n)
{
        if (s == (const char *)NULL) return 0;  /* no state dependent codings */
```

```
        if (n <= 0) return 0;
        return (*s != 0);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/mbstowcs.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: mbstowcs.c,v 1.2 89/12/18 15:12:59 eck Exp $ */

#include        <stdlib.h>

size_t
mbstowcs(register wchar_t *pwcs, register const char *s, size_t n)
{
        register int i = n;

        while (--i >= 0) {
                if (!(*pwcs++ = *s++))
                        return n - i - 1;
        }
        return n - i;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/mbtowc.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: mbtowc.c,v 1.3 90/03/28 16:36:45 eck Exp $ */

#include        <stdlib.h>
#include        <limits.h>

int
mbtowc(wchar_t *pwc, register const char *s, size_t n)
{
        if (s == (const char *)NULL) return 0;
        if (n <= 0) return 0;
        if (pwc) *pwc = *s;
        return (*s != 0);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/memchr.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: memchr.c,v 1.3 90/08/28 13:52:11 eck Exp $ */

#include        <string.h>

void *
memchr(const void *s, register int c, register size_t n)
{
        register const unsigned char *s1 = s;

        c = (unsigned char) c;
        if (n) {
                n++;
                while (--n > 0) {
                        if (*s1++ != c) continue;
                        return (void *) --s1;
                }
        }
        return NULL;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/memcmp.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Id: memcmp.c,v 1.6 1994/06/24 11:56:26 ceriel Exp $ */

#include        <string.h>

int
memcmp(const void *s1, const void *s2, size_t n)
{
        register const unsigned char *p1 = s1, *p2 = s2;

        if (n) {
                n++;
                while (--n > 0) {
                        if (*p1++ == *p2++) continue;
                        return *--p1 - *--p2;
                }
        }
        return 0;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/ansi/memcpy.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: memcpy.c,v 1.4 90/08/28 13:52:31 eck Exp $ */

#include        <string.h>

void *
memcpy(void *s1, const void *s2, register size_t n)
{
        register char *p1 = s1;
        register const char *p2 = s2;


        if (n) {
                n++;
                while (--n > 0) {
                        *p1++ = *p2++;
                }
        }
        return s1;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/ansi/memmove.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: memmove.c,v 1.3 90/08/28 13:52:42 eck Exp $ */

#include        <string.h>

void *
memmove(void *s1, const void *s2, register size_t n)
{
        register char *p1 = s1;
        register const char *p2 = s2;

        if (n>0) {
                if (p2 <= p1 && p2 + n > p1) {
                        /* overlap, copy backwards */
                        p1 += n;
                        p2 += n;
                        n++;
                        while (--n > 0) {
                                *--p1 = *--p2;
                        }
                } else {
                        n++;
```

```
			while (--n > 0) {
				*p1++ = *p2++;
			}
		}
	}
	return s1;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
				src/lib/ansi/memset.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: memset.c,v 1.3 90/08/28 13:52:54 eck Exp $ */

#include	<string.h>

void *
memset(void *s, register int c, register size_t n)
{
	register char *s1 = s;

	if (n>0) {
		n++;
		while (--n > 0) {
			*s1++ = c;
		}
	}
	return s;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
				src/lib/ansi/misc.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * misc - data and miscellaneous routines
 */
/* $Header: misc.c,v 1.11 91/04/22 13:20:45 ceriel Exp $ */

#include	<ctype.h>
#include	<time.h>
#include	<stdlib.h>
#include	<string.h>

#if	defined(__BSD4_2)

struct timeval {
	long	tv_sec;		/* seconds */
	long	tv_usec;	/* and microseconds */
```

```
};

struct timezone {
        int     tz_minuteswest; /* minutes west of Greenwich */
        int     tz_dsttime;     /* type of dst correction */
};

int _gettimeofday(struct timeval *tp, struct timezone *tzp);

#elif   !defined(_POSIX_SOURCE) && !defined(__USG)
#if     !defined(_MINIX)                /* MINIX has no ftime() */
struct timeb {
        long    time;
        unsigned short millitm;
        short   timezone;
        short   dstflag;
};
void _ftime(struct timeb *bp);
#endif
#endif

#include        "loc_time.h"

#define RULE_LEN        120
#define TZ_LEN          10

/* Make sure that the strings do not end up in ROM.
 * These strings probably contain the wrong value, and we cannot obtain the
 * right value from the system. TZ is the only help.
 */
static char ntstr[TZ_LEN + 1] = "GMT";  /* string for normal time */
static char dststr[TZ_LEN + 1] = "GDT"; /* string for daylight saving */

long    _timezone = 0;
long    _dst_off = 60 * 60;
int     _daylight = 0;
char    *_tzname[2] = {ntstr, dststr};

#if     defined(__USG) || defined(_POSIX_SOURCE)
char    *tzname[2] = {ntstr, dststr};

#if     defined(__USG)
long    timezone = 0;
int     daylight = 0;
#endif
#endif

static struct dsttype {
        char ds_type;           /* Unknown, Julian, Zero-based or M */
        int ds_date[3];         /* months, weeks, days */
        long ds_sec;            /* usually 02:00:00 */
}       dststart = { 'U', { 0, 0, 0 }, 2 * 60 * 60 }
        , dstend = { 'U', { 0, 0, 0 }, 2 * 60 * 60 };

const char *_days[] = {
                        "Sunday", "Monday", "Tuesday", "Wednesday",
                        "Thursday", "Friday", "Saturday"
                };

const char *_months[] = {
```

```
                                "January", "February", "March",
                                "April", "May", "June",
                                "July", "August", "September",
                                "October", "November", "December"
                        };

const int _ytab[2][12] = {
                { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 },
                { 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 }
        };

#if     !defined(_POSIX_SOURCE) && !defined(__USG)
#define USE_TABLE       1
#endif

#if     USE_TABLE
static int usetable = 1;

typedef struct table {
        const char *tz_name;
        const int daylight;
        const long zoneoffset;
} TABLE;

#define HOUR(x) ((x) * 60*60)

static TABLE TimezoneTable[] = {
        {"GMT", 0,      HOUR(0) },      /* Greenwich Mean */
        {"BST", 60*60,  HOUR(0) },      /* British Summer */
        {"WAT", 0,      HOUR(1) },      /* West Africa */
        {"AT", 0,       HOUR(2) },      /* Azores */
        {"BST", 0,      HOUR(3) },      /* Brazil Standard */
        {"NFT", 0,      HOUR(3.5) },    /* Newfoundland */
        {"NDT", 60*60,  HOUR(3.5) },    /* Newfoundland Daylight */
        {"AST", 0,      HOUR(4) },      /* Atlantic Standard */
        {"ADT", 60*60,  HOUR(4) },      /* Atlantic Daylight */
        {"EST", 0,      HOUR(5) },      /* Eastern Standard */
        {"EDT", 60*60,  HOUR(5) },      /* Eastern Daylight */
        {"CST", 0,      HOUR(6) },      /* Central Standard */
        {"CDT", 60*60,  HOUR(6) },      /* Central Daylight */
        {"MST", 0,      HOUR(7) },      /* Mountain Standard */
        {"MDT", 60*60,  HOUR(7) },      /* Mountain Daylight */
        {"PST", 0,      HOUR(8) },      /* Pacific Standard */
        {"PDT", 60*60,  HOUR(8) },      /* Pacific Daylight */
        {"YST", 0,      HOUR(9) },      /* Yukon Standard */
        {"YDT", 60*60,  HOUR(9) },      /* Yukon Daylight */
        {"HST", 0,      HOUR(10) },     /* Hawaii Standard */
        {"HDT", 60*60,  HOUR(10) },     /* Hawaii Daylight */
        {"NT", 0,       HOUR(11) },     /* Nome */
        {"IDLW", 0,     HOUR(12) },     /* International Date Line West */
        {"MET", 0,      -HOUR(1) },     /* Middle European */
        {"MDT", 60*60,  -HOUR(1) },     /* Middle European Summer */
        {"EET", 0,      -HOUR(2) },     /* Eastern Europe, USSR Zone 1 */
        {"BT", 0,       -HOUR(3) },     /* Baghdad, USSR Zone 2 */
        {"IT", 0,       -HOUR(3.5) },   /* Iran */
        {"ZP4", 0,      -HOUR(4) },     /* USSR Zone 3 */
        {"ZP5", 0,      -HOUR(5) },     /* USSR Zone 4 */
        {"IST", 0,      -HOUR(5.5) },   /* Indian Standard */
        {"ZP6", 0,      -HOUR(6) },     /* USSR Zone 5 */
        {"NST", 0,      -HOUR(6.5) },   /* North Sumatra */
```

```
        {"SST", 0,      -HOUR(7) },     /* South Sumatra, USSR Zone 6 */
        {"WAST", 0,     -HOUR(7) },     /* West Australian Standard */
        {"WADT", 60*60, -HOUR(7) },     /* West Australian Daylight */
        {"JT", 0,       -HOUR(7.5) },   /* Java (3pm in Cronusland!) */
        {"CCT", 0,      -HOUR(8) },     /* China Coast, USSR Zone 7 */
        {"JST", 0,      -HOUR(9) },     /* Japan Standard, USSR Zone 8 */
        {"CAST", 0,     -HOUR(9.5) },   /* Central Australian Standard */
        {"CADT", 60*60, -HOUR(9.5) },   /* Central Australian Daylight */
        {"EAST", 0,     -HOUR(10) },    /* Eastern Australian Standard */
        {"EADT", 60*60, -HOUR(10) },    /* Eastern Australian Daylight */
        {"NZT", 0,      -HOUR(12) },    /* New Zealand */
        {"NZDT", 60*60, -HOUR(12) },    /* New Zealand Daylight */
        {  NULL, 0, 0  }
};

/*
 * The function ZoneFromTable() searches the table for the current
 * timezone.  It saves the last one found in ntstr or dststr, depending on
 * wheter the name is for daylight-saving-time or not.
 * Both ntstr and dststr are TZ_LEN + 1 chars.
 */
static void
ZoneFromTable(long timezone)
{
        register TABLE *tptr = TimezoneTable;

        while (tptr->tz_name != NULL) {
                if (tptr->zoneoffset == timezone) {
                        if (tptr->daylight == 0) {
                                strncpy(ntstr,tptr->tz_name, TZ_LEN);
                                ntstr[TZ_LEN] = '\0';
                        } else {
                                strncpy(dststr,tptr->tz_name, TZ_LEN);
                                dststr[TZ_LEN] = '\0';
                        }
                }
                tptr++;
        }
}
#endif  /* USE_TABLE */

static const char *
parseZoneName(register char *buf, register const char *p)
{
        register int n = 0;

        if (*p == ':') return NULL;
        while (*p && !isdigit(*p) && *p != ',' && *p != '-' && *p != '+') {
                if (n < TZ_LEN)
                        *buf++ = *p;
                p++;
                n++;
        }
        if (n < 3) return NULL;                         /* error */
        *buf = '\0';
        return p;
}

static const char *
parseTime(register long *tm, const char *p, register struct dsttype *dst)
```

```
{
        register int n = 0;
        register const char *q = p;
        char ds_type = (dst ? dst->ds_type : '\0');

        if (dst) dst->ds_type = 'U';

        *tm = 0;
        while(*p >= '0' && *p <= '9') {
                n = 10 * n + (*p++ - '0');
        }
        if (q == p) return NULL;        /* "The hour shall be required" */
        if (n < 0 || n >= 24)   return NULL;
        *tm = n * 60 * 60;
        if (*p == ':') {
                p++;
                n = 0;
                while(*p >= '0' && *p <= '9') {
                        n = 10 * n + (*p++ - '0');
                }
                if (q == p) return NULL;        /* format error */
                if (n < 0 || n >= 60)   return NULL;
                *tm += n * 60;
                if (*p == ':') {
                        p++;
                        n = 0;
                        while(*p >= '0' && *p <= '9') {
                                n = 10 * n + (*p++ - '0');
                        }
                        if (q == p) return NULL;        /* format error */
                        if (n < 0 || n >= 60)   return NULL;
                        *tm += n;
                }
        }
        if (dst) {
                dst->ds_type = ds_type;
                dst->ds_sec = *tm;
        }
        return p;
}

static const char *
parseDate(register char *buf, register const char *p, struct dsttype *dstinfo)
{
        register const char *q;
        register int n = 0;
        int cnt = 0;
        const int bnds[3][2] =  {       { 1, 12 },
                                        { 1, 5 },
                                        { 0, 6}
                                 };
        char ds_type;

        if (*p != 'M') {
                if (*p == 'J') {
                        *buf++ = *p++;
                        ds_type = 'J';
                }
                else    ds_type = 'Z';
                q = p;
```

```
                while(*p >= '0' && *p <= '9') {
                        n = 10 * n + (*p - '0');
                        *buf++ = *p++;
                }
                if (q == p) return NULL;        /* format error */
                if (n < (ds_type == 'J') || n > 365) return NULL;
                dstinfo->ds_type = ds_type;
                dstinfo->ds_date[0] = n;
                return p;
        }
        ds_type = 'M';
        do {
                *buf++ = *p++;
                q = p;
                n = 0;
                while(*p >= '0' && *p <= '9') {
                        n = 10 * n + (*p - '0');
                        *buf++ = *p++;
                }
                if (q == p) return NULL;        /* format error */
                if (n < bnds[cnt][0] || n > bnds[cnt][1]) return NULL;
                dstinfo->ds_date[cnt] = n;
                cnt++;
        } while (cnt < 3 && *p == '.');
        if (cnt != 3) return NULL;
        *buf = '\0';
        dstinfo->ds_type = ds_type;
        return p;
}

static const char *
parseRule(register char *buf, register const char *p)
{
        long time;
        register const char *q;

        if (!(p = parseDate(buf, p, &dststart))) return NULL;
        buf += strlen(buf);
        if (*p == '/') {
                q = ++p;
                if (!(p = parseTime(&time, p, &dststart))) return NULL;
                while( p != q) *buf++ = *q++;
        }
        if (*p != ',') return NULL;
        p++;
        if (!(p = parseDate(buf, p, &dstend))) return NULL;
        buf += strlen(buf);
        if (*p == '/') {
                q = ++p;
                if (!(p = parseTime(&time, p, &dstend))) return NULL;
                while(*buf++ = *q++);
        }
        if (*p) return NULL;
        return p;
}

/* The following routine parses timezone information in POSIX-format. For
 * the requirements, see IEEE Std 1003.1-1988 section 8.1.1.
 * The function returns as soon as it spots an error.
 */
```

```c
static void
parseTZ(const char *p)
{
        long tz, dst = 60 * 60, sign = 1;
        static char lastTZ[2 * RULE_LEN];
        static char buffer[RULE_LEN];

        if (!p) return;

#if     USE_TABLE
        usetable = 0;
#endif
        if (*p == ':') {
                /*
                 * According to POSIX, this is implementation defined.
                 * Since it depends on the particular operating system, we
                 * can do nothing.
                 */
                return;
        }

        if (!strcmp(lastTZ, p)) return;         /* nothing changed */

        *_tzname[0] = '\0';
        *_tzname[1] = '\0';
        dststart.ds_type = 'U';
        dststart.ds_sec = 2 * 60 * 60;
        dstend.ds_type = 'U';
        dstend.ds_sec = 2 * 60 * 60;

        if (strlen(p) > 2 * RULE_LEN) return;
        strcpy(lastTZ, p);

        if (!(p = parseZoneName(buffer, p))) return;

        if (*p == '-') {
                sign = -1;
                p++;
        } else if (*p == '+') p++;

        if (!(p = parseTime(&tz, p, NULL))) return;
        tz *= sign;
        _timezone = tz;
        strncpy(_tzname[0], buffer, TZ_LEN);

        if (!(_daylight = (*p != '\0'))) return;

        buffer[0] = '\0';
        if (!(p = parseZoneName(buffer, p))) return;
        strncpy(_tzname[1], buffer, TZ_LEN);

        buffer[0] = '\0';
        if (*p && (*p != ','))
                if (!(p = parseTime(&dst, p, NULL))) return;
        _dst_off = dst;                 /* dst was initialized to 1 hour */
        if (*p) {
                if (*p != ',') return;
                p++;
                if (strlen(p) > RULE_LEN) return;
                if (!(p = parseRule(buffer, p))) return;
```

```
        }
}

void
_tzset(void)
{
#if     defined(__BSD4_2)

        struct timeval tv;
        struct timezone tz;

        _gettimeofday(&tv, &tz);
        _daylight = tz.tz_dsttime;
        _timezone = tz.tz_minuteswest * 60L;

#elif   !defined(_POSIX_SOURCE) && !defined(__USG)

#if     !defined(_MINIX)                /* MINIX has no ftime() */
        struct timeb time;

        _ftime(&time);
        _timezone = time.timezone * 60L;
        _daylight = time.dstflag;
#endif

#endif  /* !_POSIX_SOURCE && !__USG */

        parseTZ(getenv("TZ"));          /* should go inside #if */

#if     defined(__USG) || defined(_POSIX_SOURCE)
        tzname[0] = _tzname[0];
        tzname[1] = _tzname[1];
#if     defined(__USG)
        timezone = _timezone;
        daylight = _daylight;
#endif
#endif  /* __USG || _POSIX_SOURCE */
}

static int
last_sunday(register int day, register struct tm *timep)
{
        int first = FIRSTSUNDAY(timep);

        if (day >= 58 && LEAPYEAR(YEAR0 + timep->tm_year)) day++;
        if (day < first) return first;
        return day - (day - first) % 7;
}

static int
date_of(register struct dsttype *dst, struct tm *timep)
{
        int leap = LEAPYEAR(YEAR0 + timep->tm_year);
        int firstday, tmpday;
        register int day, month;

        if (dst->ds_type != 'M') {
                return dst->ds_date[0] -
                            (dst->ds_type == 'J'
                                && leap
```

```
                                    && dst->ds_date[0] < 58);
        }
        day = 0;
        month = 1;
        while (month < dst->ds_date[0]) {
                day += _ytab[leap][month - 1];
                month++;
        }
        firstday = (day + FIRSTDAYOF(timep)) % 7;
        tmpday = day;
        day += (dst->ds_date[2] - firstday + 7) % 7
                + 7 * (dst->ds_date[1] - 1);
        if (day >= tmpday + _ytab[leap][month]) day -= 7;
        return day;
}

/*
 * The default dst transitions are those for Western Europe (except Great
 * Britain).
 */
unsigned
_dstget(register struct tm *timep)
{
        int begindst, enddst;
        register struct dsttype *dsts = &dststart, *dste = &dstend;
        int do_dst = 0;

        if (_daylight == -1)
                _tzset();

        timep->tm_isdst = _daylight;
        if (!_daylight) return 0;

        if (dsts->ds_type != 'U')
                begindst = date_of(dsts, timep);
        else begindst = last_sunday(89, timep); /* last Sun before Apr */
        if (dste->ds_type != 'U')
                enddst = date_of(dste, timep);
        else enddst = last_sunday(272, timep);  /* last Sun in Sep */

        /* assume begindst != enddst (otherwise it would be no use) */
        if (begindst < enddst) {                /* northern hemisphere */
                if (timep->tm_yday > begindst && timep->tm_yday < enddst)
                        do_dst = 1;
        } else {                                /* southern hemisphere */
                if (timep->tm_yday > begindst || timep->tm_yday < enddst)
                        do_dst = 1;
        }

        if (!do_dst
            && (timep->tm_yday == begindst || timep->tm_yday == enddst)) {
                long dsttranssec;       /* transition when day is this old */
                long cursec;

                if (timep->tm_yday == begindst)
                        dsttranssec = dsts->ds_sec;
                else    dsttranssec = dste->ds_sec;
                cursec = ((timep->tm_hour * 60) + timep->tm_min) * 60L
                            + timep->tm_sec;

```

```
                if ((timep->tm_yday == begindst && cursec >= dsttranssec)
                    || (timep->tm_yday == enddst && cursec < dsttranssec))
                        do_dst = 1;
        }
#if USE_TABLE
        if (usetable) ZoneFromTable(_timezone);
#endif
        if (do_dst) return _dst_off;
        timep->tm_isdst = 0;
        return 0;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/mktime.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * mktime - convert local time into calendar time
 */
/* $Header: mktime.c,v 1.5 91/04/22 13:20:54 ceriel Exp $ */

/* Michael A. Temari <temari@ix.netcom.com>   03/01/96    */
/*   -  fixed bug is structure fixup code                 */

#include        <time.h>
#include        <limits.h>
#include        "loc_time.h"

/* The code assumes that unsigned long can be converted to time_t.
 * A time_t should not be wider than unsigned long, since this would mean
 * that the check for overflow at the end could fail.
 */
time_t
mktime(register struct tm *timep)
{
        register long day, year;
        register int tm_year;
        int yday, month;
        register unsigned long seconds;
        int overflow;
        unsigned dst;

        timep->tm_min += timep->tm_sec / 60;
        timep->tm_sec %= 60;
        if (timep->tm_sec < 0) {
                timep->tm_sec += 60;
                timep->tm_min--;
        }
        timep->tm_hour += timep->tm_min / 60;
        timep->tm_min = timep->tm_min % 60;
        if (timep->tm_min < 0) {
                timep->tm_min += 60;
                timep->tm_hour--;
        }
        day = timep->tm_hour / 24;
        timep->tm_hour= timep->tm_hour % 24;
```

```
        if (timep->tm_hour < 0) {
                timep->tm_hour += 24;
                day--;
        }
        timep->tm_year += timep->tm_mon / 12;
        timep->tm_mon %= 12;
        if (timep->tm_mon < 0) {
                timep->tm_mon += 12;
                timep->tm_year--;
        }
        day += (timep->tm_mday - 1);
        while (day < 0) {
                if(--timep->tm_mon < 0) {
                        timep->tm_year--;
                        timep->tm_mon = 11;
                }
                day += _ytab[LEAPYEAR(YEAR0 + timep->tm_year)][timep->tm_mon];
        }
        while (day >= _ytab[LEAPYEAR(YEAR0 + timep->tm_year)][timep->tm_mon]) {
                day -= _ytab[LEAPYEAR(YEAR0 + timep->tm_year)][timep->tm_mon];
                if (++(timep->tm_mon) == 12) {
                        timep->tm_mon = 0;
                        timep->tm_year++;
                }
        }
        timep->tm_mday = day + 1;
        _tzset();                       /* set timezone and dst info  */
        year = EPOCH_YR;
        if (timep->tm_year < year - YEAR0) return (time_t)-1;
        seconds = 0;
        day = 0;                        /* means days since day 0 now */
        overflow = 0;

        /* Assume that when day becomes negative, there will certainly
         * be overflow on seconds.
         * The check for overflow needs not to be done for leapyears
         * divisible by 400.
         * The code only works when year (1970) is not a leapyear.
         */
#if     EPOCH_YR != 1970
#error  EPOCH_YR != 1970
#endif
        tm_year = timep->tm_year + YEAR0;

        if (LONG_MAX / 365 < tm_year - year) overflow++;
        day = (tm_year - year) * 365;
        if (LONG_MAX - day < (tm_year - year) / 4 + 1) overflow++;
        day += (tm_year - year) / 4
                + ((tm_year % 4) && tm_year % 4 < year % 4);
        day -= (tm_year - year) / 100
                + ((tm_year % 100) && tm_year % 100 < year % 100);
        day += (tm_year - year) / 400
                + ((tm_year % 400) && tm_year % 400 < year % 400);

        yday = month = 0;
        while (month < timep->tm_mon) {
                yday += _ytab[LEAPYEAR(tm_year)][month];
                month++;
        }
        yday += (timep->tm_mday - 1);
```

```
        if (day + yday < 0) overflow++;
        day += yday;

        timep->tm_yday = yday;
        timep->tm_wday = (day + 4) % 7;         /* day 0 was thursday (4) */

        seconds = ((timep->tm_hour * 60L) + timep->tm_min) * 60L + timep->tm_sec;

        if ((TIME_MAX - seconds) / SECS_DAY < day) overflow++;
        seconds += day * SECS_DAY;

        /* Now adjust according to timezone and daylight saving time */

        if (((_timezone > 0) && (TIME_MAX - _timezone < seconds))
            || ((_timezone < 0) && (seconds < -_timezone)))
                overflow++;
        seconds += _timezone;

        if (timep->tm_isdst < 0)
                dst = _dstget(timep);
        else if (timep->tm_isdst)
                dst = _dst_off;
        else dst = 0;

        if (dst > seconds) overflow++;  /* dst is always non-negative */
        seconds -= dst;

        if (overflow) return (time_t)-1;

        if ((time_t)seconds != seconds) return (time_t)-1;
        return (time_t)seconds;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/ansi/qsort.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: qsort.c,v 1.3 90/08/28 14:03:24 eck Exp $ */

#include        <stdlib.h>

static  void qsort1(char *, char *, size_t);
static  int (*qcompar)(const char *, const char *);
static  void qexchange(char *, char *, size_t);
static  void q3exchange(char *, char *, char *, size_t);

void
qsort(void *base, size_t nel, size_t width,
      int (*compar)(const void *, const void *))
{
        /* when nel is 0, the expression '(nel - 1) * width' is wrong */
        if (!nel) return;
        qcompar = (int (*)(const char *, const char *)) compar;
```

```
        qsort1(base, (char *)base + (nel - 1) * width, width);
}

static void
qsort1(char *a1, char *a2, register size_t width)
{
        register char *left, *right;
        register char *lefteq, *righteq;
        int cmp;

        for (;;) {
                if (a2 <= a1) return;
                left = a1;
                right = a2;
                lefteq = righteq = a1 + width * (((a2-a1)+width)/(2*width));
                /*
                   Pick an element in the middle of the array.
                   We will collect the equals around it.
                   "lefteq" and "righteq" indicate the left and right
                   bounds of the equals respectively.
                   Smaller elements end up left of it, larger elements end
                   up right of it.
                */
again:
                while (left < lefteq && (cmp = (*qcompar)(left, lefteq)) <= 0) {
                        if (cmp < 0) {
                                /* leave it where it is */
                                left += width;
                        }
                        else {
                                /* equal, so exchange with the element to
                                   the left of the "equal"-interval.
                                */
                                lefteq -= width;
                                qexchange(left, lefteq, width);
                        }
                }
                while (right > righteq) {
                        if ((cmp = (*qcompar)(right, righteq)) < 0) {
                                /* smaller, should go to left part
                                */
                                if (left < lefteq) {
                                        /* yes, we had a larger one at the
                                           left, so we can just exchange
                                        */
                                        qexchange(left, right, width);
                                        left += width;
                                        right -= width;
                                        goto again;
                                }
                                /* no more room at the left part, so we
                                   move the "equal-interval" one place to the
                                   right, and the smaller element to the
                                   left of it.
                                   This is best expressed as a three-way
                                   exchange.
                                */
                                righteq += width;
                                q3exchange(left, righteq, right, width);
                                lefteq += width;
```

```
                                left = lefteq;
                        }
                        else if (cmp == 0) {
                                /* equal, so exchange with the element to
                                   the right of the "equal-interval"
                                */
                                righteq += width;
                                qexchange(right, righteq, width);
                        }
                        else    /* just leave it */ right -= width;
                }
                if (left < lefteq) {
                        /* larger element to the left, but no more room,
                           so move the "equal-interval" one place to the
                           left, and the larger element to the right
                           of it.
                        */
                        lefteq -= width;
                        q3exchange(right, lefteq, left, width);
                        righteq -= width;
                        right = righteq;
                        goto again;
                }
                /* now sort the "smaller" part */
                qsort1(a1, lefteq - width, width);
                /* and now the larger, saving a subroutine call
                   because of the for(;;)
                */
                a1 = righteq + width;
        }
        /*NOTREACHED*/
}

static void
qexchange(register char *p, register char *q,
          register size_t n)
{
        register int c;

        while (n-- > 0) {
                c = *p;
                *p++ = *q;
                *q++ = c;
        }
}

static void
q3exchange(register char *p, register char *q, register char *r,
           register size_t n)
{
        register int c;

        while (n-- > 0) {
                c = *p;
                *p++ = *r;
                *r++ = *q;
                *q++ = c;
        }
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/raise.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: raise.c,v 1.3 90/11/22 13:55:50 eck Exp $ */

#if     defined(_POSIX_SOURCE)
#include        <sys/types.h>
#endif
#include        <signal.h>

int _kill(int pid, int sig);
pid_t _getpid(void);

int
raise(int sig)
{
        if (sig < 0 || sig > _NSIG)
                return -1;
        return _kill(_getpid(), sig);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/rand.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: rand.c,v 1.3 90/06/30 20:02:45 ceriel Exp $ */

#include        <stdlib.h>

static unsigned long int next = 1;

int rand(void)
{
        next = next * 1103515245 + 12345;
        return (unsigned int)(next/(2 * (RAND_MAX +1L)) % (RAND_MAX+1L));
}

void srand(unsigned int seed)
{
        next = seed;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/setlocale.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * setlocale - set the programs locale
 */
/* $Header: setlocale.c,v 1.2 89/12/18 15:49:11 eck Exp $ */

#include        <locale.h>
#include        <string.h>

struct lconv _lc;

char *
setlocale(int category, const char *locale)
{
        if (!locale) return "C";
        if (*locale && strcmp(locale, "C")) return (char *)NULL;

        switch(category) {
        case LC_ALL:
        case LC_CTYPE:
        case LC_COLLATE:
        case LC_TIME:
        case LC_NUMERIC:
        case LC_MONETARY:
                return *locale ? (char *)locale : "C";
        default:
                return (char *)NULL;
        }
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/sigmisc.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * sigmisc.c - used to get a signal mask
 */
/* $Header: sigmisc.c,v 1.2 90/10/15 11:23:08 eck Exp $ */

#if     defined(_POSIX_SOURCE)

/* This can't be done in setjmp.e, since SIG_SETMASK is defined in
 * <signal.h>. This is a C-file, which can't be included.
 */

#include        <sys/types.h>
#include        <signal.h>
#include        <stddef.h>

int _sigprocmask(int, sigset_t *, sigset_t *);

static void
__testsigset(void) {
        /* This switch compiles when a sigset_t has the right size. */
```

```
        switch(0) {
        case 0:
        case sizeof(sigset_t) <= sizeof(long): break;
        }
}

void
__newsigset(sigset_t *p)
{
        /* The SIG_SETMASK is not significant */
        _sigprocmask(SIG_SETMASK, NULL, p);
}

void
__oldsigset(sigset_t *p)
{
        _sigprocmask(SIG_SETMASK, p, NULL);
}
#endif  /* _POSIX_SOURCE */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/signal.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/* SYSVR4 and ANSI compatible signal(2). */

#include <lib.h>
#define sigaction       _sigaction
#define sigemptyset     _sigemptyset
#include <signal.h>

PUBLIC sighandler_t signal(sig, disp)
int sig;                        /* signal number */
sighandler_t disp;              /* signal handler, or SIG_DFL, or SIG_IGN */
{
  struct sigaction sa, osa;

  if (sig <= 0 || sig > _NSIG || sig == SIGKILL) {
        errno = EINVAL;
        return(SIG_ERR);
  }
  sigemptyset(&sa.sa_mask);

#ifdef WANT_UNRELIABLE_SIGNALS
  /* Allow the signal being handled to interrupt the signal handler. */
  sa.sa_flags = SA_NODEFER;

  /* When signal is caught, reset signal handler to SIG_DFL for all but
   * SIGILL and SIGTRAP.
   */
  if (sig != SIGILL && sig != SIGTRAP) sa.sa_flags |= SA_RESETHAND;
#else
  sa.sa_flags = 0;
#endif

  sa.sa_handler = disp;
  if (sigaction(sig, &sa, &osa) < 0) return(SIG_ERR);
  return(osa.sa_handler);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/strcat.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: strcat.c,v 1.2 90/05/31 18:33:06 ceriel Exp $ */

#include        <string.h>

char *
strcat(char *ret, register const char *s2)
{
        register char *s1 = ret;

        while (*s1++ != '\0')
                /* EMPTY */ ;
        s1--;
        while (*s1++ = *s2++)
                /* EMPTY */ ;
        return ret;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/strchr.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: strchr.c,v 1.3 90/08/28 13:53:00 eck Exp $ */

#include        <string.h>

char *
strchr(register const char *s, register int c)
{
        c = (char) c;

        while (c != *s) {
                if (*s++ == '\0') return NULL;
        }
        return (char *)s;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/strcmp.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Id: strcmp.c,v 1.4 1994/06/24 11:56:43 ceriel Exp $ */

#include        <string.h>

int
strcmp(register const char *s1, register const char *s2)
{
        while (*s1 == *s2++) {
                if (*s1++ == '\0') {
                        return 0;
                }
        }
        if (*s1 == '\0') return -1;
        if (*--s2 == '\0') return 1;
        return (unsigned char) *s1 - (unsigned char) *s2;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/strcoll.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: strcoll.c,v 1.2 90/08/28 13:53:23 eck Exp $ */

#include        <string.h>
#include        <locale.h>

int
strcoll(register const char *s1, register const char *s2)
{
        while (*s1 == *s2++) {
                if (*s1++ == '\0') {
                        return 0;
                }
        }
        return *s1 - *--s2;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/strcpy.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: strcpy.c,v 1.2 90/05/31 18:33:13 ceriel Exp $ */

#include        <string.h>

char *
strcpy(char *ret, register const char *s2)
{
        register char *s1 = ret;

        while (*s1++ = *s2++)
                /* EMPTY */ ;

        return ret;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/strcspn.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: strcspn.c,v 1.2 89/12/18 16:01:42 eck Exp $ */

#include        <string.h>

size_t
strcspn(const char *string, const char *notin)
{
        register const char *s1, *s2;

        for (s1 = string; *s1; s1++) {
                for(s2 = notin; *s2 != *s1 && *s2; s2++)
                        /* EMPTY */ ;
                if (*s2)
                        break;
        }
        return s1 - string;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/strerror.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: strerror.c,v 1.3 90/08/28 13:53:31 eck Exp $ */

#include        <string.h>

/*
 * I don't know why, but X3J11 says that strerror() should be in declared
 * in <string.h>.  That is why the function is defined here.
 */
char *
strerror(register int errnum)
{
        extern const char *_sys_errlist[];
        extern const int _sys_nerr;

        if (errnum < 0 || errnum >= _sys_nerr)
                return "unknown error";
        return (char *)_sys_errlist[errnum];
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/strftime.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * strftime - convert a structure to a string, controlled by an argument
 */
/* $Header: strftime.c,v 1.3 91/04/22 13:21:03 ceriel Exp $ */

#include        <time.h>
#include        "loc_time.h"

/* The width can be -1 in both s_prnt() as in u_prnt(). This
 * indicates that as many characters as needed should be printed.
 */
static char *
s_prnt(char *s, size_t maxsize, const char *str, int width)
{
        while (width > 0 || (width < 0 && *str)) {
                if (!maxsize) break;
                *s++ = *str++;
                maxsize--;
                width--;
        }
        return s;
}

static char *
u_prnt(char *s, size_t maxsize, unsigned val, int width)
```

```
{
        int c;

        c = val % 10;
        val = val / 10;
        if (--width > 0 || (width < 0 && val != 0))
                s = u_prnt(s, (maxsize ? maxsize - 1 : 0), val, width);
        if (maxsize) *s++ = c + '0';
        return s;
}

size_t
strftime(char *s, size_t maxsize,
                const char *format, const struct tm *timeptr)
{
        size_t n;
        char *firsts, *olds;

        if (!format) return 0;

        _tzset();       /* for %Z conversion */
        firsts = s;
        while (maxsize && *format) {
                while (maxsize && *format && *format != '%') {
                        *s++ = *format++;
                        maxsize--;
                }
                if (!maxsize || !*format) break;
                format++;

                olds = s;
                switch (*format++) {
                case 'a':
                        s = s_prnt(s, maxsize,
                                        _days[timeptr->tm_wday], ABB_LEN);
                        maxsize -= s - olds;
                        break;
                case 'A':
                        s = s_prnt(s, maxsize, _days[timeptr->tm_wday], -1);
                        maxsize -= s - olds;
                        break;
                case 'b':
                        s = s_prnt(s, maxsize,
                                        _months[timeptr->tm_mon], ABB_LEN);
                        maxsize -= s - olds;
                        break;
                case 'B':
                        s = s_prnt(s, maxsize, _months[timeptr->tm_mon], -1);
                        maxsize -= s - olds;
                        break;
                case 'c':
                        n = strftime(s, maxsize,
                                        "%a %b %d %H:%M:%S %Y", timeptr);
                        if (n) maxsize -= n;
                        else maxsize = 0;
                        s += n;
                        break;
                case 'd':
                        s = u_prnt(s, maxsize, timeptr->tm_mday, 2);
                        maxsize -= s - olds;
```

```
                                break;
                        case 'H':
                                s = u_prnt(s, maxsize, timeptr->tm_hour, 2);
                                maxsize -= s - olds;
                                break;
                        case 'I':
                                s = u_prnt(s, maxsize,
                                                (timeptr->tm_hour + 11) % 12 + 1, 2);
                                maxsize -= s - olds;
                                break;
                        case 'j':
                                s = u_prnt(s, maxsize, timeptr->tm_yday + 1, 3);
                                maxsize -= s - olds;
                                break;
                        case 'm':
                                s = u_prnt(s, maxsize, timeptr->tm_mon + 1, 2);
                                maxsize -= s - olds;
                                break;
                        case 'M':
                                s = u_prnt(s, maxsize, timeptr->tm_min, 2);
                                maxsize -= s - olds;
                                break;
                        case 'p':
                                s = s_prnt(s, maxsize,
                                            (timeptr->tm_hour < 12) ? "AM" : "PM", 2);
                                maxsize -= s - olds;
                                break;
                        case 'S':
                                s = u_prnt(s, maxsize, timeptr->tm_sec, 2);
                                maxsize -= s - olds;
                                break;
                        case 'U':
                                s = u_prnt(s, maxsize,          /* ??? */
                                    (timeptr->tm_yday + 7 - timeptr->tm_wday) / 7, 2);
                                maxsize -= s - olds;
                                break;
                        case 'w':
                                s = u_prnt(s, maxsize, timeptr->tm_wday, 1);
                                maxsize -= s - olds;
                                break;
                        case 'W':
                                s = u_prnt(s, maxsize,          /* ??? */
                                    (timeptr->tm_yday+7-(timeptr->tm_wday+6)%7)/7,2);
                                maxsize -= s - olds;
                                break;
                        case 'x':
                                n = strftime(s, maxsize, "%a %b %d %Y", timeptr);
                                if (n) maxsize -= n;
                                else maxsize = 0;
                                s += n;
                                break;
                        case 'X':
                                n = strftime(s, maxsize, "%H:%M:%S", timeptr);
                                if (n) maxsize -= n;
                                else maxsize = 0;
                                s += n;
                                break;
                        case 'y':
                                s = u_prnt(s, maxsize, timeptr->tm_year % 100, 2);
                                maxsize -= s - olds;
```

```
                        break;
                case 'Y':
                        s = u_prnt(s, maxsize, timeptr->tm_year + YEAR0, -1);
                        maxsize -= s - olds;
                        break;
                case 'Z':
                        s = s_prnt(s, maxsize,
                                        _tzname[(timeptr->tm_isdst > 0)], -1);
                        maxsize -= s - olds;
                        break;
                case '%':
                        *s++ = '%';
                        maxsize--;
                        break;
                default:
                        /* A conversion error. Leave the loop. */
                        while (*format) format++;
                        break;
                }

        }
        if (maxsize) {
                *s = '\0';
                return s - firsts;
        }
        return 0;       /* The buffer is full */
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/strlen.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: strlen.c,v 1.4 90/08/28 13:53:42 eck Exp $ */

#include        <string.h>

size_t
strlen(const char *org)
{
        register const char *s = org;

        while (*s++)
                /* EMPTY */ ;

        return --s - org;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/strncat.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: strncat.c,v 1.3 90/08/28 13:53:54 eck Exp $ */

#include        <string.h>

char *
strncat(char *ret, register const char *s2, size_t n)
{
        register char *s1 = ret;

        if (n > 0) {
                while (*s1++)
                        /* EMPTY */ ;
                s1--;
                while (*s1++ = *s2++)  {
                        if (--n > 0) continue;
                        *s1 = '\0';
                        break;
                }
                return ret;
        } else return s1;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/strncmp.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Id: strncmp.c,v 1.4 1994/06/24 11:57:04 ceriel Exp $ */

#include        <string.h>

int
strncmp(register const char *s1, register const char *s2, register size_t n)
{
        if (n) {
                do {
                        if (*s1 != *s2++)
                                break;
                        if (*s1++ == '\0')
                                return 0;
                } while (--n > 0);
                if (n > 0) {
                        if (*s1 == '\0') return -1;
```

```
                        if (*--s2 == '\0') return 1;
                        return (unsigned char) *s1 - (unsigned char) *s2;
                }
        }
        return 0;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ansi/strncpy.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: strncpy.c,v 1.3 90/08/28 13:54:11 eck Exp $ */

#include        <string.h>

char *
strncpy(char *ret, register const char *s2, register size_t n)
{
        register char *s1 = ret;

        if (n>0) {
                while((*s1++ = *s2++) && --n > 0)
                        /* EMPTY */ ;
                if ((*--s2 == '\0') && --n > 0) {
                        do {
                                *s1++ = '\0';
                        } while(--n > 0);
                }
        }
        return ret;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ansi/strpbrk.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: strpbrk.c,v 1.2 89/12/18 16:02:21 eck Exp $ */

#include        <string.h>

char *
strpbrk(register const char *string, register const char *brk)
{
        register const char *s1;

        while (*string) {
                for (s1 = brk; *s1 && *s1 != *string; s1++)
```

```
				/* EMPTY */ ;
			if (*s1)
				return (char *)string;
			string++;
		}
		return (char *)NULL;
	}
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
				src/lib/ansi/strrchr.c
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: strrchr.c,v 1.3 90/08/28 13:54:21 eck Exp $ */

#include	<string.h>

char *
strrchr(register const char *s, int c)
{
	register const char *result = NULL;

	c = (char) c;

	do {
		if (c == *s)
			result = s;
	} while (*s++ != '\0');

	return (char *)result;
}
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
				src/lib/ansi/strspn.c
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: strspn.c,v 1.1 89/05/11 10:09:09 eck Exp $ */

#include	<string.h>

size_t
strspn(const char *string, const char *in)
{
	register const char *s1, *s2;

	for (s1 = string; *s1; s1++) {
		for (s2 = in; *s2 && *s2 != *s1; s2++)
```

```
                        /* EMPTY */ ;
                if (*s2 == '\0')
                        break;
        }
        return s1 - string;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ansi/strstr.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: strstr.c,v 1.3 90/08/28 13:54:28 eck Exp $ */

#include        <string.h>

char *
strstr(register const char *s, register const char *wanted)
{
        register const size_t len = strlen(wanted);

        if (len == 0) return (char *)s;
        while (*s != *wanted || strncmp(s, wanted, len))
                if (*s++ == '\0')
                        return (char *)NULL;
        return (char *)s;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ansi/strtok.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: strtok.c,v 1.2 90/08/28 13:54:38 eck Exp $ */

#include        <string.h>

char *
strtok(register char *string, const char *separators)
{
        register char *s1, *s2;
        static char *savestring;

        if (string == NULL) {
                string = savestring;
                if (string == NULL) return (char *)NULL;
        }

        s1 = string + strspn(string, separators);
```

```
        if (*s1 == '\0') {
                savestring = NULL;
                return (char *)NULL;
        }

        s2 = strpbrk(s1, separators);
        if (s2 != NULL)
                *s2++ = '\0';
        savestring = s2;
        return s1;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/strtol.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: strtol.c,v 1.4 90/05/11 15:22:19 eck Exp $ */

#include        <ctype.h>
#include        <errno.h>
#include        <limits.h>
#include        <stdlib.h>

static unsigned long
string2long(register const char *nptr, char **endptr,
                        int base, int is_signed);

long int
strtol(register const char *nptr, char **endptr, int base)
{
        return (signed long)string2long(nptr, endptr, base, 1);
}

unsigned long int
strtoul(register const char *nptr, char **endptr, int base)
{
        return (unsigned long)string2long(nptr, endptr, base, 0);
}

#define between(a, c, z)  ((unsigned) ((c) - (a)) <= (unsigned) ((z) - (a)))

static unsigned long
string2long(register const char *nptr, char ** const endptr,
                        int base, int is_signed)
{
        register unsigned int v;
        register unsigned long val = 0;
        register int c;
        int ovfl = 0, sign = 1;
        const char *startnptr = nptr, *nrstart;

        if (endptr) *endptr = (char *)nptr;
```

```
        while (isspace(*nptr)) nptr++;
        c = *nptr;

        if (c == '-' || c == '+') {
                if (c == '-') sign = -1;
                nptr++;
        }
        nrstart = nptr;                 /* start of the number */

        /* When base is 0, the syntax determines the actual base */
        if (base == 0)
                if (*nptr == '0')
                        if (*++nptr == 'x' || *nptr == 'X') {
                                base = 16;
                                nptr++;
                        }
                        else    base = 8;
                else    base = 10;
        else if (base==16 && *nptr=='0' && (*++nptr =='x' || *nptr =='X'))
                nptr++;

        for (;;) {
                c = *nptr;
                if (between('0', c, '9')) {
                        v = c - '0';
                } else
                if (between('a', c, 'z')) {
                        v = c - 'a' + 0xa;
                } else
                if (between('A', c, 'Z')) {
                        v = c - 'A' + 0xA;
                } else {
                        break;
                }
                if (v >= base) break;
                if (val > (ULONG_MAX - v) / base) ovfl++;
                val = (val * base) + v;
                nptr++;
        }
        if (endptr) {
                if (nrstart == nptr) *endptr = (char *)startnptr;
                else *endptr = (char *)nptr;
        }

        if (!ovfl) {
                /* Overflow is only possible when converting a signed long. */
                if (is_signed
                    && (   (sign < 0 && val > -(unsigned long)LONG_MIN)
                        || (sign > 0 && val > LONG_MAX)))
                    ovfl++;
        }

        if (ovfl) {
                errno = ERANGE;
                if (is_signed)
                        if (sign < 0) return LONG_MIN;
                        else return LONG_MAX;
                else return ULONG_MAX;
        }
        return (long) sign * val;
```

```
}
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/strxfrm.c
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: strxfrm.c,v 1.4 90/08/28 13:54:46 eck Exp $ */

#include        <string.h>

size_t
strxfrm(register char *s1, register const char *save, register size_t n)
{
        register const char *s2 = save;

        while (*s2) {
                if (n > 1) {
                        n--;
                        *s1++ = *s2++;
                } else
                        s2++;
        }
        if (n > 0)
                *s1++ = '\0';
        return s2 - save;
}
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/system.c
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: system.c,v 1.4 90/11/22 13:59:54 eck Exp $ */

#if     defined(_POSIX_SOURCE)
#include        <sys/types.h>
#endif
#include        <stdlib.h>
#include        <signal.h>

extern pid_t _fork(void);
extern pid_t _wait(int *);
extern void _exit(int);
extern void _execve(const char *path, const char ** argv, const char ** envp);
extern int _close(int);

#define FAIL    127

```

```
extern const char **_penvp;
static const char *exec_tab[] = {
        "sh",                   /* argv[0] */
        "-c",                   /* argument to the shell */
        NULL,                   /* to be filled with user command */
        NULL                    /* terminating NULL */
        };

int
system(const char *str)
{
        int pid, exitstatus, waitval;
        int i;

        if ((pid = _fork()) < 0) return str ? -1 : 0;

        if (pid == 0) {
                for (i = 3; i <= 20; i++)
                        _close(i);
                if (!str) str = "cd .";         /* just testing for a shell */
                exec_tab[2] = str;              /* fill in command */
                _execve("/bin/sh", exec_tab, _penvp);
                /* get here if execve fails ... */
                _exit(FAIL);    /* see manual page */
        }
        while ((waitval = _wait(&exitstatus)) != pid) {
                if (waitval == -1) break;
        }
        if (waitval == -1) {
                /* no child ??? or maybe interrupted ??? */
                exitstatus = -1;
        }
        if (!str) {
                if (exitstatus == FAIL << 8)            /* execve() failed */
                        exitstatus = 0;
                else exitstatus = 1;                    /* /bin/sh exists */
        }
        return exitstatus;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ansi/tolower.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include        <ctype.h>

int tolower(int c) {
        return isupper(c) ? c - 'A' + 'a' : c ;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/toupper.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include        <ctype.h>

int toupper(int c) {
        return islower(c) ? c - 'a' + 'A' : c ;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/tzset.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * tzset - set timezone information
 */
/* $Header: tzset.c,v 1.3 91/04/22 13:21:11 ceriel Exp $ */

/* This function is present for System V && POSIX */

#include        <time.h>
#include        "loc_time.h"

void
tzset(void)
{
        _tzset();       /* does the job */
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/wcstombs.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: wcstombs.c,v 1.3 90/03/28 16:37:07 eck Exp $ */

#include        <stdlib.h>
#include        <locale.h>
#include        <limits.h>

size_t
wcstombs(register char *s, register const wchar_t *pwcs, size_t n)
{
        register int i = n;

```

```
        while (--i >= 0) {
                if (!(*s++ = *pwcs++))
                        break;
        }
        return n - i - 1;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ansi/wctomb.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: wctomb.c,v 1.4 91/01/15 11:55:33 ceriel Exp $ */

#include        <stdlib.h>
#include        <limits.h>

int
/* was: wctomb(char *s, wchar_t wchar)
 * This conflicts with prototype, so it was changed to:
 */
wctomb(char *s, wchar_t wchar)
{
        if (!s) return 0;               /* no state dependent codings */

        *s = wchar;
        return 1;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/curspriv.h
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/* Constants */
#define _SUBWIN         1               /* window is a subwindow */
#define _ENDLINE        2               /* last winline is last screen line */
#define _FULLWIN        4               /* window fills screen */
#define _SCROLLWIN      8               /* window lwr rgt is screen lwr rgt */

#define _NO_CHANGE      -1              /* flags line edge unchanged */
#define _BREAKCHAR      0x03            /* ^C character */
#define _DCCHAR         0x08            /* Delete Char char (BS) */
#define _DLCHAR         0x1b            /* Delete Line char (ESC) */
#define _GOCHAR         0x11            /* ^Q character */
#define _PRINTCHAR      0x10            /* ^P character */
#define _STOPCHAR       0x13            /* ^S character */
#define  NUNGETCH       10              /* max # chars to ungetch() */

```

```
#define max(a,b) (((a) > (b)) ? (a) : (b))
#define min(a,b) (((a) < (b)) ? (a) : (b))

/* Character mask definitions. */
#define CHR_MSK ((int) 0x00ff)          /* ASCIIZ character mask */
#define ATR_MSK ((int) 0xff00)          /* attribute mask */
#define ATR_NRM ((int) 0x0000)          /* no special attributes */

/* Type declarations. */

typedef struct {
  WINDOW  *tmpwin;                      /* window used for updates */
  int      cursrow;                     /* position of physical cursor */
  int      curscol;
  bool     rawmode;
  bool     cbrkmode;
  bool     echoit;
} cursv;

/* External variables */
extern  cursv   _cursvar;               /* curses variables */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/beep.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <curses.h>
#include "curspriv.h"
#include <termcap.h>

extern char *bl, *vb;

/* Beep() sounds the terminal bell. */
void beep()
{
  if (bl)
        tputs(bl, 1, outc);
  else if (vb)
        tputs(vb, 1, outc);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/charpick.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <curses.h>
#include "curspriv.h"

/****************************************************************/
/* Winch(win) returns the character at the current position in  */
/* Window 'win'.                                                */
/****************************************************************/

int winch(win)
WINDOW *win;
```

```
{
  return((win->_line[win->_cury][win->_curx]) & 0xff);
}                                /* winch */

/****************************************************************/
/* Mvinch() moves the stdscr cursor to a new position, then     */
/* Returns the character at that position.                      */
/****************************************************************/

int mvinch(y, x)
int y;
int x;
{
  if (wmove(stdscr, y, x) == ERR) return(ERR);
  return((stdscr->_line[stdscr->_cury][stdscr->_curx]) & 0xff);
}

/****************************************************************/
/* Mvwinch() moves the cursor of window 'win' to a new posi-    */
/* Tion, then returns the character at that position.           */
/****************************************************************/

int mvwinch(win, y, x)
WINDOW *win;
int y;
int x;
{
  if (wmove(win, y, x) == ERR) return(ERR);
  return((win->_line[win->_cury][win->_curx]) & 0xff);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/curs_set.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <curses.h>
#include "curspriv.h"
#include <termcap.h>

extern char *vi, *ve, *vs;

/* Sets cursor visibility to unvisible=0; normal visible=1 or very good
 * visible=2.
*/
void curs_set(visibility)
int visibility;
{
  switch (visibility) {
      case 0:
        if (vi) tputs(vi, 1, outc);
        break;
      case 1:
        if (ve) tputs(ve, 1, outc);
        break;
      case 2:
```

```
        if (vs)
                tputs(vs, 1, outc);
        else if (ve)
                tputs(ve, 1, outc);
  }
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/cursesio.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <stdlib.h>
#include <termcap.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <curses.h>
#include "curspriv.h"

struct termios _orig_tty, _tty;
cursv _cursvar;

WINDOW *stdscr, *curscr;
int LINES, COLS;
bool NONL;

char termcap[1024];             /* termcap buffer */
char tc[200];                   /* area to hold string capabilities */
char *ttytype;                  /* terminal type from env */
char *arp;                      /* pointer for use in tgetstr */
char *cp;                       /* character pointer */

char *cl;                       /* clear screen capability */
char *cm;                       /* cursor motion capability */
char *so;                       /* start standout capability */
char *se;                       /* end standout capability */
char *mr;                       /* start of reverse */
char *me;                       /* revert to normal */
char *mb;                       /* start of blink */
char *md;                       /* start of bold */
char *us;                       /* start of underscore */
char *ue;                       /* end of underscore */
char *vi;                       /* cursor invisible */
char *ve;                       /* cursor normal */
char *vs;                       /* cursor good visible */
char *as;                       /* alternative charset start */
char *ae;                       /* alternative charset end */
char *bl;                       /* ring the bell */
char *vb;                       /* visual bell */

/* fatal - report error and die. Never returns */
void fatal(s)
char *s;
{
  (void) fprintf(stderr, "curses: %s\n", s);
  exit(1);
}
```

```
/* Outc - call putchar, necessary because putchar is a macro. */
void outc(c)
int c;
{
  putchar(c);
}

/* Move cursor to r,c */
void poscur(r, c)
int r, c;
{
  tputs(tgoto(cm, c, r), 1, outc);
}

/* Clear the screen */
void clrscr()
{
  tputs(cl, 1, outc);
}

/* This are terminal independent characters which can be used in curses */

unsigned int ACS_ULCORNER;
unsigned int ACS_LLCORNER;
unsigned int ACS_URCORNER;
unsigned int ACS_LRCORNER;
unsigned int ACS_RTEE;
unsigned int ACS_LTEE;
unsigned int ACS_BTEE;
unsigned int ACS_TTEE;
unsigned int ACS_HLINE;
unsigned int ACS_VLINE;
unsigned int ACS_PLUS;
unsigned int ACS_S1;
unsigned int ACS_S9;
unsigned int ACS_DIAMOND;
unsigned int ACS_CKBOARD;
unsigned int ACS_DEGREE;
unsigned int ACS_PLMINUS;
unsigned int ACS_BULLET;
unsigned int ACS_LARROW;
unsigned int ACS_RARROW;
unsigned int ACS_DARROW;
unsigned int ACS_UARROW;
unsigned int ACS_BOARD;
unsigned int ACS_LANTERN;
unsigned int ACS_BLOCK;

/* These defines describe the full set of grafic block characters which
 * can be defined via termcap.
 */

#define RIGHTARROW  0
#define LEFTARROW   1
#define DOWNARROW   2
#define UPARROW     3
#define FULLSQUARE  4
#define GREYSQUARE  5
#define EMPTYSQUARE 6
```

```
#define LATERN      7
#define DIAMOND     8
#define DEGREE      9
#define PLUSMINUS  10
#define DOWNRIGHT  11
#define UPRIGHT    12
#define UPLEFT     13
#define DOWNLEFT   14
#define CROSS      15
#define UPLINE     16
#define UPMIDLINE  17
#define MIDLINE    18
#define DOMIDLINE  19
#define DOWNLINE   20
#define TEELEFT    21
#define TEERIGHT   22
#define TEEHEAD    23
#define TEENORMAL  24
#define VERTLINE   25
#define PARAGRAPH  26

unsigned int _cursgraftable[27] =
{
 '>', '<', 'v', '^', '#', ':', ' ', '#', '+', '\'', '#', '+', '+',
 '+', '+', '+', '-', ' ', '-', ' ', '_', '+', '+', '+', '+', '|'
};
char _cursident[28] = "+,.-0ahI`fgjklmnopqrstuvwx~";

int setterm(type)
char *type;
{
  unsigned char *ac;
  int i;
#ifdef TIOCGWINSZ
  struct winsize wsize;
#endif

  if (tgetent(termcap, type) != 1) return ERR;

#ifdef TIOCGWINSZ
  if (ioctl(0, TIOCGWINSZ, &wsize) == 0) {
        LINES = wsize.ws_row != 0 ? wsize.ws_row : tgetnum("li");
        COLS = wsize.ws_col != 0 ? wsize.ws_col : tgetnum("co");
  } else {
#endif
        LINES = tgetnum("li");
        COLS = tgetnum("co");
#ifdef TIOCGWINSZ
  }
#endif
  arp = tc;
  cl = tgetstr("cl", &arp);
  so = tgetstr("so", &arp);
  se = tgetstr("se", &arp);
  cm = tgetstr("cm", &arp);
  mr = tgetstr("mr", &arp);
  me = tgetstr("me", &arp);
  mb = tgetstr("mb", &arp);
  md = tgetstr("md", &arp);
  us = tgetstr("us", &arp);
```

```
  ue = tgetstr("ue", &arp);
  vi = tgetstr("vi", &arp);
  ve = tgetstr("ve", &arp);
  vs = tgetstr("vs", &arp);
  as = tgetstr("as", &arp);
  ae = tgetstr("ae", &arp);
  ac = (unsigned char *) tgetstr("ac", &arp);
  bl = tgetstr("bl", &arp);
  vb = tgetstr("vb", &arp);

  if (ac) {
        while (*ac) {
                i = 0;
                while (*ac != _cursident[i]) i++;
                _cursgraftable[i] = *++ac | A_ALTCHARSET;
                ac++;
        }
  }

  ACS_ULCORNER = _cursgraftable[UPLEFT];
  ACS_LLCORNER = _cursgraftable[DOWNLEFT];
  ACS_URCORNER = _cursgraftable[UPRIGHT];
  ACS_LRCORNER = _cursgraftable[DOWNRIGHT];
  ACS_RTEE = _cursgraftable[TEERIGHT];
  ACS_LTEE = _cursgraftable[TEELEFT];
  ACS_BTEE = _cursgraftable[TEEHEAD];
  ACS_TTEE = _cursgraftable[TEENORMAL];
  ACS_HLINE = _cursgraftable[MIDLINE];
  ACS_VLINE = _cursgraftable[VERTLINE];
  ACS_PLUS = _cursgraftable[CROSS];
  ACS_S1 = _cursgraftable[UPLINE];
  ACS_S9 = _cursgraftable[DOWNLINE];
  ACS_DIAMOND = _cursgraftable[DIAMOND];
  ACS_CKBOARD = _cursgraftable[GREYSQUARE];
  ACS_DEGREE = _cursgraftable[DEGREE];
  ACS_PLMINUS = _cursgraftable[PLUSMINUS];
  ACS_BULLET = 'o';             /* where the hell is a bullet defined in
                                 * termcap ??? */
  ACS_LARROW = _cursgraftable[LEFTARROW];
  ACS_RARROW = _cursgraftable[RIGHTARROW];
  ACS_DARROW = _cursgraftable[DOWNARROW];
  ACS_UARROW = _cursgraftable[UPARROW];
  ACS_BOARD = _cursgraftable[EMPTYSQUARE];
  ACS_LANTERN = _cursgraftable[LATERN];
  ACS_BLOCK = _cursgraftable[FULLSQUARE];
  /* Wow, I got it! */
  return OK;
}

void gettmode()
{
  tcgetattr(0, &_orig_tty);
  tcgetattr(0, &_tty);
  _cursvar.echoit = (_tty.c_lflag & ECHO) != 0;
  _cursvar.rawmode = (_tty.c_lflag & (ICANON|ISIG)) == 0;
  _cursvar.cbrkmode = (_tty.c_lflag & (ICANON|ISIG)) == ISIG;
  NONL = (_tty.c_iflag & ICRNL) != 0;
}
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/endwin.c
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <curses.h>
#include "curspriv.h"
#include <termcap.h>

int endwin()
{
  extern char *me;

  curs_set(1);
  poscur(LINES - 1, 0);
  refresh();
  tputs(me, 1, outc);
  delwin(stdscr);
  delwin(curscr);
  delwin(_cursvar.tmpwin);
  resetty();
  return(OK);
}
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/flash.c
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <curses.h>
#include "curspriv.h"
#include <termcap.h>

extern char *bl, *vb;

/* Flash() flashes the terminal screen. */
void flash()
{
  if (vb)
        tputs(vb, 1, outc);
  else if (bl)
        tputs(bl, 1, outc);
}
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/initscr.c
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/* initscr.c - initialize the curses library */

#include <stdlib.h>
#include <curses.h>
#include "curspriv.h"

WINDOW *initscr()
{
  char *term;

```

```
  if ((term = getenv("TERM")) == NULL) return NULL;
  setterm(term);
  gettmode();
  if ((_cursvar.tmpwin = newwin(LINES, COLS, 0, 0)) == NULL) return NULL;
  if ((curscr = newwin(LINES, COLS, 0, 0)) == NULL) return NULL;
  if ((stdscr = newwin(LINES, COLS, 0, 0)) == NULL) return NULL;
  clearok(curscr, TRUE);
  return(stdscr);
}
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/longname.c
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <curses.h>
#include "curspriv.h"

/****************************************************************/
/* Longname() returns a pointer to a string describing the      */
/* User terminal.                                               */
/****************************************************************/

char *longname()
{
  return("not implemented");
}
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/move.c
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <curses.h>
#include "curspriv.h"

/****************************************************************/
/* Wmove() moves the cursor in window 'win' to position (x,y).  */
/****************************************************************/

int wmove(win, y, x)
WINDOW *win;
int y;
int x;
{
  if ((x<0) || (x>win->_maxx) || (y<win->_regtop) || (y>win->_regbottom))
        return(ERR);
  win->_curx = x;
  win->_cury = y;
  return(OK);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/mvcursor.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <curses.h>
#include "curspriv.h"

/****************************************************************/
/* Mvcur(oldy,oldx,newy,newx) the display cursor to <newy,newx> */
/****************************************************************/

int mvcur(oldy, oldx, newy, newx)
int oldy;
int oldx;
int newy;
int newx;
{
  if ((newy >= LINES) || (newx >= COLS) || (newy < 0) || (newx < 0))
        return(ERR);
  poscur(newy, newx);
  _cursvar.cursrow = newy;
  _cursvar.curscol = newx;
  return(OK);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/newwin.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <stdlib.h>
#include <curses.h>
#include "curspriv.h"

/****************************************************************/
/* Makenew() allocates all data for a new window except the     */
/* Actual lines themselves.                                      */
/****************************************************************/

_PROTOTYPE(static WINDOW *makenew, (int nlines, int ncols, int begy,int begx));

static WINDOW *makenew(num_lines, num_columns, begy, begx)
int num_lines, num_columns, begy, begx;
{
  int i;
  WINDOW *win;

  /* Allocate the window structure itself */
  if ((win = (WINDOW *) malloc(sizeof(WINDOW))) == NULL)
        return((WINDOW *) ERR);

  /* Allocate the line pointer array */
  if ((win->_line = (int **) calloc(num_lines, sizeof(int *))) == NULL) {
        free(win);
        return((WINDOW *) ERR);
```

```
  }

  /* Allocate the minchng and maxchng arrays */
  if ((win->_minchng = (int *) calloc(num_lines, sizeof(int))) == NULL) {
        free(win);
        free(win->_line);
        return((WINDOW *) ERR);
  }
  if ((win->_maxchng = (int *) calloc(num_lines, sizeof(int))) == NULL) {
        free(win);
        free(win->_line);
        free(win->_minchng);
        return((WINDOW *) ERR);
  }

  /* Initialize window variables */
  win->_curx = 0;
  win->_cury = 0;
  win->_maxy = num_lines - 1;
  win->_maxx = num_columns - 1;
  win->_begy = begy;
  win->_begx = begx;
  win->_flags = 0;
  win->_attrs = ATR_NRM;
  win->_tabsize = 8;
  win->_clear = FALSE;
  win->_leave = FALSE;
  win->_scroll = FALSE;
  win->_nodelay = FALSE;
  win->_keypad = FALSE;
  win->_regtop = 0;
  win->_regbottom = num_lines - 1;

  /* Init to say window unchanged */
  for (i = 0; i < num_lines; i++) {
        win->_minchng[i] = 0;
        win->_maxchng[i] = num_columns - 1;
  }

  /* Set flags for window properties */
  if ((begy + num_lines) == LINES) {
        win->_flags |= _ENDLINE;
        if ((begx == 0) && (num_columns == COLS) && (begy == 0))
                win->_flags |= _FULLWIN;
  }                                /* if */
  if (((begy + num_lines) == LINES) && ((begx + num_columns) == COLS))
        win->_flags |= _SCROLLWIN;
  return(win);
}


/****************************************************************/
/* Newwin() creates a new window with size num_lines * num_co-  */
/* Lumns, and origin begx,begy relative to the SCREEN. Special  */
/* Case: if num_lines and/or num_columns is 0, the remainder of */
/* The screen is used.                                          */
/****************************************************************/
WINDOW *newwin(num_lines, num_columns, begy, begx)
int num_lines, num_columns, begy, begx;
{
```

```
  WINDOW *win;
  int *ptr;
  int i, j;

  if (num_lines == 0) num_lines = LINES - begy;
  if (num_columns == 0) num_columns = COLS - begx;
  if ((win = makenew(num_lines, num_columns, begy, begx)) == (WINDOW *) ERR)
        return((WINDOW *) ERR);
  for (i = 0; i < num_lines; i++) {     /* make and clear the lines */
        if ((win->_line[i] = (int *)calloc(num_columns, sizeof(int))) == NULL){
                for (j = 0; j < i; j++) /* if error, free all the data */
                        free(win->_line[j]);
                free(win->_minchng);
                free(win->_maxchng);
                free(win->_line);
                free(win);
                return((WINDOW *) ERR);
        } else {
                for (ptr = win->_line[i]; ptr < win->_line[i] + num_columns;)
                        *ptr++ = ' ' | ATR_NRM;
        }
  }
  return(win);
}


/****************************************************************/
/* Subwin() creates a sub-window in the 'orig' window, with     */
/* Size num_lines * num_columns, and with origin begx, begy     */
/* Relative to the SCREEN. Special case: if num_lines and/or    */
/* Num_columns is 0, the remainder of the original window is    */
/* Used. The subwindow uses the original window's line buffers  */
/* To store it's own lines.                                     */
/****************************************************************/
WINDOW *subwin(orig, num_lines, num_columns, begy, begx)
WINDOW *orig;
int num_lines, num_columns, begy, begx;
{
  WINDOW *win;
  int i, j, k;

  /* Make sure window fits inside the original one */
  if (begy < orig->_begy || begx < orig->_begx ||
                    (begy + num_lines) > (orig->_begy + orig->_maxy) ||
                    (begx + num_columns) > (orig->_begx + orig->_maxx) )
        return((WINDOW *) ERR);

  if (num_lines == 0) num_lines = orig->_maxy - (begy - orig->_begy);
  if (num_columns == 0) num_columns = orig->_maxx - (begx - orig->_begx);
  if ((win = makenew(num_lines, num_columns, begy, begx)) == (WINDOW *) ERR)
        return((WINDOW *) ERR);

  /* Set line pointers the same as in the original window */
  j = begy - orig->_begy;
  k = begx - orig->_begx;
  for (i = 0; i < num_lines; i++) win->_line[i] = (orig->_line[j++]) + k;
  win->_flags |= _SUBWIN;
  return(win);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/options.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <curses.h>
#include "curspriv.h"

static bool hasold = FALSE;     /* for remembering old cursor type */
static int oldmode;

/****************************************************************/
/* Idlok() is used to set  flag for using the terminal insert/  */
/* Delete line capabilities. This is not relevant for the PC    */
/* Version of curses, and thus nothing is done.                 */
/****************************************************************/
void idlok(win, flag)
WINDOW *win;
bool flag;
{
}

/****************************************************************/
/* Clearok() marks window 'win' to cause screen clearing and    */
/* Redraw the next time a refresh is done.                      */
/****************************************************************/
void clearok(win, flag)
WINDOW *win;
bool flag;
{
  if (win == curscr)
        _cursvar.tmpwin->_clear = flag;
  else
        win->_clear = flag;
}

/****************************************************************/
/* Leaveok() marks window 'win' to allow the update routines    */
/* To leave the hardware cursor where it happens to be at the   */
/* End of update. Usually used in combination with cursoff().   */
/****************************************************************/

void leaveok(win, flag)
WINDOW *win;
bool flag;
{
  win->_leave = flag;
}

/****************************************************************/
/* Scrollok() marks window 'win' to allow the scrolling region  */
/* Of it to actually scroll.                                    */
/****************************************************************/
void scrollok(win, flag)
WINDOW *win;
bool flag;
{
  win->_scroll = flag;
}
```

```
/****************************************************************/
/* Nodelay() marks the window to make character input non-      */
/* Waiting, i.e. if there is no character to get, -1 will be    */
/* Returned.                                                    */
/****************************************************************/
void nodelay(win, flag)
WINDOW *win;
bool flag;
{
  win->_nodelay = flag;
}

/****************************************************************/
/* Keypad() marks window 'win' to use the special keypad mode.  */
/****************************************************************/
void keypad(win, flag)
WINDOW *win;
bool flag;
{
  win->_keypad = flag;
}

/****************************************************************/
/* Meta() allows use of any alternate character set allowed by  */
/* The terminal. We always allow this on the PC, so this one    */
/* Does nothing.                                                */
/****************************************************************/
void meta(win, flag)
WINDOW *win;
bool flag;
{
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/curses/overlay.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/****************************************************************/
/* Overlay() and overwrite() functions of the PCcurses package  */
/*                                                              */
/****************************************************************/
/* This version of curses is based on ncurses, a curses version */
/* Originally written by Pavel Curtis at Cornell University.    */
/* I have made substantial changes to make it run on IBM PC's,  */
/* And therefore consider myself free to make it public domain. */
/*              Bjorn Larsson (...mcvax!enea!infovax!bl)        */
/****************************************************************/
/* 1.0: Release:                                        870515  */
/****************************************************************/
/* Modified to run under the MINIX operating system by Don Cope */
/* These changes are also released into the public domain.      */
/*                                                      900906  */
/****************************************************************/

#include <curses.h>
#include "curspriv.h"

```

```
/****************************************************************/
/* Overlay() overwrites 'win1' upon 'win2', with origins alig-  */
/* Ned. Overlay is transparent; blanks from 'win1' are not      */
/* Copied to 'win2'.                                            */
/****************************************************************/
void overlay(win1, win2)
WINDOW *win1, *win2;
{
  int *minchng;
  int *maxchng;
  int *w1ptr;
  int *w2ptr;
  int attrs;
  int col;
  int line;
  int last_line;
  int last_col;

  last_col = min(win1->_maxx, win2->_maxx);
  last_line = min(win1->_maxy, win2->_maxy);
  attrs = win2->_attrs & ATR_MSK;
  minchng = win2->_minchng;
  maxchng = win2->_maxchng;

  for (line = 0; line <= last_line; line++) {
	register short fc, lc = 0;
	w1ptr = win1->_line[line];
	w2ptr = win2->_line[line];
	fc = _NO_CHANGE;
	for (col = 0; col <= last_col; col++) {
		if ((*w1ptr & CHR_MSK) != ' ') {
			*w2ptr = (*w1ptr & CHR_MSK) | attrs;
			if (fc == _NO_CHANGE) fc = col;
			lc = col;
		}
		w1ptr++;
		w2ptr++;
	}

	if (*minchng == _NO_CHANGE) {
		*minchng = fc;
		*maxchng = lc;
	} else if (fc != _NO_CHANGE) {
		if (fc < *minchng) *minchng = fc;
		if (lc > *maxchng) *maxchng = lc;
	}
	minchng++;
	maxchng++;
  }				/* for */
}				/* overlay */

/****************************************************************/
/* Overwrite() overwrites 'win1' upon 'win2', with origins      */
/* Aligned. Overwrite is non-transparent; blanks from 'win1'    */
/* Are copied to 'win2'.                                        */
/****************************************************************/
void overwrite(win1, win2)
WINDOW *win1, *win2;
{
  int *minchng;
```

```
  int *maxchng;
  int *w1ptr;
  int *w2ptr;
  int attrs;
  int col;
  int line;
  int last_line;
  int last_col;

  last_col = min(win1->_maxx, win2->_maxx);
  last_line = min(win1->_maxy, win2->_maxy);
  attrs = win2->_attrs & ATR_MSK;
  minchng = win2->_minchng;
  maxchng = win2->_maxchng;

  for (line = 0; line <= last_line; line++) {
	register short fc, lc = 0;

	w1ptr = win1->_line[line];
	w2ptr = win2->_line[line];
	fc = _NO_CHANGE;

	for (col = 0; col <= last_col; col++) {
		if ((*w1ptr & CHR_MSK) != (*w2ptr & CHR_MSK)) {
			*w2ptr = (*w1ptr & CHR_MSK) | attrs;

			if (fc == _NO_CHANGE) fc = col;
			lc = col;
		}
		w1ptr++;
		w2ptr++;
	}			/* for */

	if (*minchng == _NO_CHANGE) {
		*minchng = fc;
		*maxchng = lc;
	} else if (fc != _NO_CHANGE) {
		if (fc < *minchng) *minchng = fc;
		if (lc > *maxchng) *maxchng = lc;
	}
	minchng++;
	maxchng++;
  }
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/prntscan.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <string.h>
#include <curses.h>
#include "curspriv.h"

static char printscanbuf[513];  /* buffer used during I/O */

/****************************************************************/
/* Wprintw(win,fmt,args) does a printf() in window 'win'.       */
/****************************************************************/
int wprintw(WINDOW *win, const char *fmt, ...)
```

```c
{
  va_list args;

  va_start(args, fmt);
  vsprintf(printscanbuf, fmt, args);
  if (waddstr(win, printscanbuf) == ERR) return(ERR);
  return(strlen(printscanbuf));
}

/****************************************************************/
/* Printw(fmt,args) does a printf() in stdscr.                  */
/****************************************************************/
int printw(const char *fmt, ...)
{
  va_list args;

  va_start(args, fmt);
  vsprintf(printscanbuf, fmt, args);
  if (waddstr(stdscr, printscanbuf) == ERR) return(ERR);
  return(strlen(printscanbuf));
}                               /* printw */

/****************************************************************/
/* Mvprintw(fmt,args) moves the stdscr cursor to a new posi-    */
/* tion, then does a printf() in stdscr.                        */
/****************************************************************/
int mvprintw(int y, int x, const char *fmt, ...)
{
  va_list args;

  va_start(args, fmt);
  if (wmove(stdscr, y, x) == ERR) return(ERR);
  vsprintf(printscanbuf, fmt, args);
  if (waddstr(stdscr, printscanbuf) == ERR) return(ERR);
  return(strlen(printscanbuf));
}

/****************************************************************/
/* Mvwprintw(win,fmt,args) moves the window 'win's cursor to    */
/* A new position, then does a printf() in window 'win'.        */
/****************************************************************/
int mvwprintw(WINDOW *win, int y, int x, const char *fmt, ...)
{
  va_list args;

  va_start(args, fmt);
  if (wmove(win, y, x) == ERR) return(ERR);
  vsprintf(printscanbuf, fmt, args);
  if (waddstr(win, printscanbuf) == ERR) return(ERR);
  return(strlen(printscanbuf));
}                               /* mvwprintw */

/****************************************************************/
/* Wscanw(win,fmt,args) gets a string via window 'win', then    */
/* Scans the string using format 'fmt' to extract the values    */
/* And put them in the variables pointed to the arguments.      */
/****************************************************************/
int wscanw(WINDOW *win, const char *fmt, ...)
{
  va_list args;
```

```
  va_start(args, fmt);
  wrefresh(win);                /* set cursor */
  if (wgetstr(win, printscanbuf) == ERR)        /* get string */
        return(ERR);
  return(vsscanf(printscanbuf, fmt, args));
}                               /* wscanw */

/****************************************************************/
/* Scanw(fmt,args) gets a string via stdscr, then scans the     */
/* String using format 'fmt' to extract the values and put them */
/* In the variables pointed to the arguments.                   */
/****************************************************************/
int scanw(const char *fmt, ...)
{
  va_list args;

  va_start(args, fmt);
  wrefresh(stdscr);             /* set cursor */
  if (wgetstr(stdscr, printscanbuf) == ERR)     /* get string */
        return(ERR);
  return(vsscanf(printscanbuf, fmt, args));
}                               /* scanw */

/****************************************************************/
/* Mvscanw(y,x,fmt,args) moves stdscr's cursor to a new posi-   */
/* Tion, then gets a string via stdscr and scans the string     */
/* Using format 'fmt' to extract the values and put them in the */
/* Variables pointed to the arguments.                          */
/****************************************************************/
int mvscanw(int y, int x, const char *fmt, ...)
{
  va_list args;

  va_start(args, fmt);
  if (wmove(stdscr, y, x) == ERR) return(ERR);
  wrefresh(stdscr);             /* set cursor */
  if (wgetstr(stdscr, printscanbuf) == ERR)     /* get string */
        return(ERR);
  return(vsscanf(printscanbuf, fmt, args));
}                               /* mvscanw */

/****************************************************************/
/* Mvwscanw(win,y,x,fmt,args) moves window 'win's cursor to a   */
/* New position, then gets a string via 'win' and scans the     */
/* String using format 'fmt' to extract the values and put them */
/* In the variables pointed to the arguments.                   */
/****************************************************************/
int mvwscanw(WINDOW *win, int y, int x, const char *fmt, ...)
{
  va_list args;

  va_start(args, fmt);
  if (wmove(win, y, x) == ERR) return(ERR);
  wrefresh(win);                /* set cursor */
  if (wgetstr(win, printscanbuf) == ERR)        /* get string */
        return(ERR);
  return(vsscanf(printscanbuf, fmt, args));
}                               /* mvwscanw */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/refresh.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/* refresh.c */

#include <curses.h>
#include "curspriv.h"

/* Wrefresh() updates window win's area of the physical screen. */
void wrefresh(win)
WINDOW *win;
{
  if (win == curscr)
        curscr->_clear = TRUE;
  else
        wnoutrefresh(win);
  doupdate();
}

/****************************************************************/
/* Wnoutrefresh() updates the image of the desired screen,      */
/* Without doing physical update (copies window win's image to  */
/* The _cursvar.tmpwin window, which is hidden from the user).  */
/****************************************************************/

void wnoutrefresh(win)
register WINDOW *win;
{
  register int *dst;            /* start destination in temp window */
  register int *end;            /* end destination in temp window */
  register int *src;            /* source in user window */
  register int first;           /* first changed char on line */
  register int last;            /* last changed char on line */
  WINDOW *nscr;
  int begy;                     /* window's place on screen */
  int begx;
  int i;
  int j;

  nscr = _cursvar.tmpwin;
  begy = win->_begy;
  begx = win->_begx;

  for (i = 0, j = begy; i <= win->_maxy; i++, j++) {
        if (win->_minchng[i] != _NO_CHANGE) {
                first = win->_minchng[i];
                last = win->_maxchng[i];
                dst = &(nscr->_line[j][begx + first]);
                end = &(nscr->_line[j][begx + last]);
                src = &(win->_line[i][first]);

                while (dst <= end)      /* copy user line to temp window */
                        *dst++ = *src++;

                first += begx;  /* nscr's min/max change positions */
                last += begx;

                if ((nscr->_minchng[j] == _NO_CHANGE) || (nscr->_minchng[j] > first))
```

```
                        nscr->_minchng[j] = first;
                if (last > nscr->_maxchng[j]) nscr->_maxchng[j] = last;

                win->_minchng[i] = _NO_CHANGE;  /* updated now */
        }                       /* if */
        win->_maxchng[i] = _NO_CHANGE;  /* updated now */
  }                             /* for */

  if (win->_clear) {
        win->_clear = FALSE;
        nscr->_clear = TRUE;
  }                             /* if */
  if (!win->_leave) {
        nscr->_cury = win->_cury + begy;
        nscr->_curx = win->_curx + begx;
  }                             /* if */
}                               /* wnoutrefresh */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/scrreg.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/****************************************************************/
/* Wsetscrreg() routine of the PCcurses package                 */
/*                                                              */
/****************************************************************/
/* This version of curses is based on ncurses, a curses version */
/* Originally written by Pavel Curtis at Cornell University.    */
/* I have made substantial changes to make it run on IBM PC's,  */
/* And therefore consider myself free to make it public domain. */
/*              Bjorn Larsson (...mcvax!enea!infovax!bl)        */
/****************************************************************/
/* 1.0: Release:                                        870515  */
/****************************************************************/
/* Modified to run under the MINIX operating system by Don Cope */
/* These changes are also released into the public domain.      */
/*                                                      900906  */
/****************************************************************/

#include <curses.h>
#include "curspriv.h"

/****************************************************************/
/* Wsetscrreg() set the scrolling region of window 'win' to in- */
/* Clude all lines between 'top' and 'bottom'.                  */
/****************************************************************/

int wsetscrreg(win, top, bottom)
WINDOW *win;
int top;
int bottom;
{
  if ((0 <= top) &&
      (top <= win->_cury)
      &&
      (win->_cury <= bottom)
      &&
```

```
        (bottom <= win->_maxy)
          ) {
          win->_regtop = top;
          win->_regbottom = bottom;
          return(OK);
    }

     /* If */
    else
          return(ERR);
  }                                /* wsetscrreg */

  /****************************************************************/
  /* Setscrreg() set the scrolling region of stdscr to include    */
  /* All lines between 'top' and 'bottom'.                        */
  /****************************************************************/

  int setscrreg(top, bottom)
  int top;
  int bottom;
  {
    return(wsetscrreg(stdscr, top, bottom));
  }                                /* setscrreg */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/setterm.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <curses.h>
#include "curspriv.h"

_PROTOTYPE( static void ttysetflags, (void) );

static void ttysetflags()
{
  _tty.c_iflag |= ICRNL | IXON;
  _tty.c_oflag |= OPOST | ONLCR;
  _tty.c_lflag |= ECHO | ICANON | IEXTEN | ISIG;

  if (_cursvar.rawmode) {
        _tty.c_iflag &= ~(ICRNL | IXON);
        _tty.c_oflag &= ~(OPOST);
        _tty.c_lflag &= ~(ICANON | IEXTEN | ISIG);
  }
  if (_cursvar.cbrkmode) {
        _tty.c_lflag &= ~(ICANON);
  }
  if (!_cursvar.echoit) {
        _tty.c_lflag &= ~(ECHO | ECHONL);
  }
  if (NONL) {
        _tty.c_iflag &= ~(ICRNL);
        _tty.c_oflag &= ~(ONLCR);
  }
  tcsetattr(0, TCSANOW, &_tty);
}                                /* ttysetflags */

void raw()
```

```
{
  _cursvar.rawmode = TRUE;
  ttysetflags();
}                                /* raw */

void noraw()
{
  _cursvar.rawmode = FALSE;
  ttysetflags();
}                                /* noraw */

void echo()
{
  _cursvar.echoit = TRUE;
  ttysetflags();
}

void noecho()
{
  _cursvar.echoit = FALSE;
  ttysetflags();
}

void nl()
{
  NONL = FALSE;
  ttysetflags();
}                                /* nl */

void nonl()
{
  NONL = TRUE;
  ttysetflags();
}                                /* nonl */

void cbreak()
{
  _cursvar.cbrkmode = TRUE;
  ttysetflags();
}                                /* cbreak */

void nocbreak()
{
  _cursvar.cbrkmode = FALSE;
  ttysetflags();
}                                /* nocbreak */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/tabsize.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/****************************************************************/
/* Tabsize() routines of the PCcurses package                   */
/*                                                              */
/****************************************************************/
/* This version of curses is based on ncurses, a curses version */
/* Originally written by Pavel Curtis at Cornell University.    */
/* I have made substantial changes to make it run on IBM PC's,  */
/* And therefore consider myself free to make it public domain. */
/*              Bjorn Larsson (...mcvax!enea!infovax!bl)        */
/****************************************************************/
/* 1.0: Release:                                        870515  */
/****************************************************************/
/* Modified to run under the MINIX operating system by Don Cope */
/* These changes are also released into the public domain.      */
/*                                                      900906  */
/****************************************************************/

#include <curses.h>
#include "curspriv.h"

/****************************************************************/
/* Wtabsize(win,ts) sets the tabsize of window 'win' to 'ts',   */
/* And returns the original value.                              */
/****************************************************************/

int wtabsize(win, ts)
WINDOW *win;
int ts;
{
  int origval;

  origval = win->_tabsize;
  win->_tabsize = ts;
  return(origval);
}                               /* wtabsize */

/****************************************************************/
/* Tabsize(ts) sets the tabsize of stdscr to 'ts', and returns  */
/* The original value.                                          */
/****************************************************************/

int tabsize(ts)
int ts;
{
  int origval;

  origval = stdscr->_tabsize;
  stdscr->_tabsize = ts;
  return(origval);
}                               /* tabsize */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/termmisc.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <curses.h>
#include "curspriv.h"

/* Static variables or saving terminal modes */

int fixterm()
{
  return(OK);
}                               /* fixterm */

int resetterm()
{
  return(OK);
}

int saveoldterm()
{
  return(OK);
}                               /* saveoldterm */

int saveterm()
{
  return(OK);
}                               /* saveterm */

int baudrate()
{
  return(19200);
}                               /* baudrate */

/****************************************************************/
/* Erasechar(), killchar() returns std MSDOS erase chars.      */
/****************************************************************/

int erasechar()
{
  return(_DCCHAR);              /* character delete char */
}                               /* erasechar */

int killchar()
{
  return(_DLCHAR);              /* line delete char */
}                               /* killchar */

/****************************************************************/
/* Savetty() and resetty() saves and restores the terminal I/O  */
/* Settings.                                                    */
/****************************************************************/

int savetty()
{
  return(OK);
}                               /* savetty */

/****************************************************************/
```

```
/* Setupterm() sets up the terminal. On a PC, it is always suc- */
/* Cessful, and returns 1.                                      */
/****************************************************************/

int setupterm()
{
  return(1);
}                               /* setupterm */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/unctrl.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/****************************************************************/
/* Unctrl() routines of the PCcurses package                    */
/*                                                              */
/****************************************************************/
/* This version of curses is based on ncurses, a curses version */
/* Originally written by Pavel Curtis at Cornell University.    */
/* I have made substantial changes to make it run on IBM PC's,  */
/* And therefore consider myself free to make it public domain. */
/*              Bjorn Larsson (...mcvax!enea!infovax!bl)        */
/****************************************************************/
/* 1.0: Release:                                        870515  */
/****************************************************************/
/* Modified to run under the MINIX operating system by Don Cope */
/* These changes are also released into the public domain.      */
/*                                                      900906  */
/****************************************************************/

#include <curses.h>
#include "curspriv.h"

static char strbuf[3] = {0, 0, 0};

/****************************************************************/
/* Unctrl() returns a char pointer to a string corresponding to */
/* Argument character 'c'.                                      */
/****************************************************************/

char *unctrl(c)
char c;
{
  int ic = c;
  ic &= 0xff;

  if ((ic >= ' ') && (ic != 0x7f)) {    /* normal characters */
        strbuf[0] = ic;
        strbuf[1] = '\0';
        return(strbuf);
  }                             /* if */
  strbuf[0] = '^';              /* '^' prefix */
  if (c == 0x7f)                /* DEL */
        strbuf[1] = '?';
  else                          /* other control */
        strbuf[1] = ic + '@';
  return(strbuf);
}                               /* unctrl */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/update.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <curses.h>
#include "curspriv.h"
#include <termcap.h>

static WINDOW *twin;            /* used by many routines */

/****************************************************************/
/* Gotoxy() moves the physical cursor to the desired address on */
/* The screen. We don't optimize here - on a PC, it takes more  */
/* Time to optimize than to do things directly.                 */
/****************************************************************/

_PROTOTYPE(static void gotoxy, (int row, int col ));
_PROTOTYPE(static void newattr, (int ch ));
_PROTOTYPE(static void Putchar, (int ch ));
_PROTOTYPE(static void clrupdate, (WINDOW *scr ));
_PROTOTYPE(static void transformline, (int lineno ));

static void gotoxy(row, col)
int row, col;
{
  poscur(row, col);
  _cursvar.cursrow = row;
  _cursvar.curscol = col;
}

/* Update attributes */
static void newattr(ch)
int ch;
{
  extern char *me, *as, *ae, *mb, *md, *mr, *so, *us;
  static int lastattr = 0;

  if (lastattr != (ch &= ATR_MSK)) {
        lastattr = ch;

        tputs(me, 1, outc);
        if (ae) tputs(ae, 1, outc);

        if (ch & A_ALTCHARSET)
                if (as) tputs(as, 1, outc);
        if (ch & A_BLINK) tputs(mb, 1, outc);
        if (ch & A_BOLD) tputs(md, 1, outc);
        if (ch & A_REVERSE) tputs(mr, 1, outc);
        if (ch & A_STANDOUT) tputs(so, 1, outc);
        if (ch & A_UNDERLINE) tputs(us, 1, outc);
  }
}

/* Putchar() writes a character, with attributes, to the physical
   screen, but avoids writing to the lower right screen position.
   Should it care about am?
*/

/* Output char with attribute */
```

```
static void Putchar(ch)
int ch;
{
  if ((_cursvar.cursrow < LINES) || (_cursvar.curscol < COLS)) {
        newattr(ch);
        putchar(ch);
  }
}

/****************************************************************/
/* Clrupdate(scr) updates the screen by clearing it and then    */
/* Redraw it in it's entirety.                                  */
/****************************************************************/

static void clrupdate(scr)
WINDOW *scr;
{
  register int *src;
  register int *dst;
  register int i;
  register int j;
  WINDOW *w;

  w = curscr;

  if (scr != w) {               /* copy scr to curscr */
        for (i = 0; i < LINES; i++) {
                src = scr->_line[i];
                dst = w->_line[i];
                for (j = 0; j < COLS; j++) *dst++ = *src++;
        }                       /* for */
  }                             /* if */
  newattr(scr->_attrs);
  clrscr();
  scr->_clear = FALSE;
  for (i = 0; i < LINES; i++) { /* update physical screen */
        src = w->_line[i];
        j = 0;
        while (j < COLS) {
                if (*src != (' ' | ATR_NRM)) {
                        gotoxy(i, j);
                        while (j < COLS && (*src != (' ' | ATR_NRM))) {
                                Putchar(*src++);
                                j++;
                        }
                } else {
                        src++;
                        j++;
                }
        }                       /* for */
  }                             /* for */
  fflush(stdout);
}                               /* clrupdate */

/****************************************************************/
/* Transformline() updates the given physical line to look      */
/* Like the corresponding line in _cursvar.tmpwin.              */
/****************************************************************/

static void transformline(lineno)
```

```
register int lineno;
{
  register int *dstp;
  register int *srcp;
  register int dstc;
  register int srcc;
  int x;
  int endx;

  x = twin->_minchng[lineno];
  endx = twin->_maxchng[lineno];
  dstp = curscr->_line[lineno] + x;
  srcp = twin->_line[lineno] + x;

  while (x <= endx) {
        if ((*dstp != *srcp) || (dstc != srcc)) {
                gotoxy(lineno, x);
                while (x <= endx && ((*dstp != *srcp) || (dstc != srcc))) {
                        Putchar(*srcp);
                        *dstp++ = *srcp++;
                        x++;
                }
        } else {
                *dstp++ = *srcp++;
                x++;
        }
  }                             /* for */
  twin->_minchng[lineno] = _NO_CHANGE;
  twin->_maxchng[lineno] = _NO_CHANGE;
}                               /* transformline */

/****************************************************************/
/* Doupdate() updates the physical screen to look like _curs-   */
/* Var.tmpwin if curscr is not 'Clear-marked'. Otherwise it     */
/* Updates the screen to look like curscr.                      */
/****************************************************************/

void doupdate()
{
  int i;

  twin = _cursvar.tmpwin;
  if (curscr->_clear)
        clrupdate(curscr);
  else {
        if (twin->_clear)
                clrupdate(twin);
        else {
                for (i = 0; i < LINES; i++)
                        if (twin->_minchng[i] != _NO_CHANGE)
                                transformline(i);
        }
  }
  curscr->_curx = twin->_curx;
  curscr->_cury = twin->_cury;
  gotoxy(curscr->_cury, curscr->_curx);
  fflush(stdout);
}                               /* doupdate */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/waddch.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <curses.h>
#include "curspriv.h"

/****************************************************************/
/* Newline() does line advance and returns the new cursor line. */
/* If error, return -1.                                         */
/****************************************************************/

_PROTOTYPE( static short newline, (WINDOW *win, int lin));

static short newline(win, lin)
WINDOW *win;
int lin;
{
  if (++lin > win->_regbottom) {
        lin--;
        if (win->_scroll)
                scroll(win);
        else
                return(-1);
  }                             /* if */
  return(lin);
}                               /* newline */

/****************************************************************/
/* Waddch() inserts character 'c' at the current cursor posi-   */
/* Tion in window 'win', and takes any actions as dictated by   */
/* The character.                                               */
/****************************************************************/

int waddch(win, c)
WINDOW *win;
int c;
{
  int x = win->_curx;
  int y = win->_cury;
  int newx;
  int ch = c;
  int ts = win->_tabsize;

  ch &= (A_ALTCHARSET | 0xff);
  if (y > win->_maxy || x > win->_maxx || y < 0 || x < 0) return(ERR);
  switch (ch) {
      case '\t':
        for (newx = ((x / ts) + 1) * ts; x < newx; x++) {
                if (waddch(win, ' ') == ERR) return(ERR);
                if (win->_curx == 0)    /* if tab to next line */
                        return(OK);     /* exit the loop */
        }
        return(OK);

      case '\n':
        if (NONL) x = 0;
        if ((y = newline(win, y)) < 0) return (ERR);
        break;
```

```
      case '\r':        x = 0;  break;

      case '\b':
        if (--x < 0)            /* no back over left margin */
                x = 0;
        break;

      case 0x7f:
        {
                if (waddch(win, '^') == ERR) return(ERR);
                return(waddch(win, '?'));
        }

      default:
        if (ch < ' ') {         /* handle control chars */
                if (waddch(win, '^') == ERR) return(ERR);
                return(waddch(win, c + '@'));
        }
        ch |= (win->_attrs & ATR_MSK);
        if (win->_line[y][x] != ch) {   /* only if data change */
                if (win->_minchng[y] == _NO_CHANGE)
                        win->_minchng[y] = win->_maxchng[y] = x;
                else if (x < win->_minchng[y])
                        win->_minchng[y] = x;
                else if (x > win->_maxchng[y])
                        win->_maxchng[y] = x;
        }                       /* if */
        win->_line[y][x++] = ch;
        if (x > win->_maxx) {   /* wrap around test */
                x = 0;
                if ((y = newline(win, y)) < 0) return(ERR);
        }
        break;

  }                             /* switch */
  win->_curx = x;
  win->_cury = y;
  return(OK);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/waddstr.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <curses.h>
#include "curspriv.h"

/****************************************************************/
/* Waddstr() inserts string 'str' at the current cursor posi-   */
/* Tion in window 'win', and takes any actions as dictated by   */
/* The characters.                                               */
/****************************************************************/

int waddstr(win, str)
```

```
WINDOW *win;
char *str;
{
  while (*str) {
        if (waddch(win, *str++) == ERR) return(ERR);
  }
  return(OK);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/wbox.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <curses.h>
#include "curspriv.h"

/****************************************************************/
/* Wbox(win,ymin,xmin,ymax,xmax,v,h) draws a box in window      */
/* 'win', enclosing the area xmin-xmax and ymin-xmax. If        */
/* Xmax and/or ymax is 0, the window max value is used. 'v' and */
/* 'h' are the vertical and horizontal characters to use. If    */
/* 'v' and 'h' are 0, wbox will use the alternate character set */
/* In a pretty way.                                             */
/****************************************************************/

int wbox(win, ymin, xmin, ymax, xmax, v, h)
WINDOW *win;
int ymin, xmin, ymax, xmax;
unsigned int v;
unsigned int h;
{
  unsigned int vc, hc, ulc, urc, llc, lrc;      /* corner chars */
  int i;

  if (ymax == 0) ymax = win->_maxy;
  if (xmax == 0) xmax = win->_maxx;

  if (ymin >= win->_maxy || ymax > win->_maxy ||
      xmin >= win->_maxx || xmax > win->_maxx ||
      ymin >= ymax || xmin >= xmax)
        return(ERR);

  vc = v;
  hc = h;
  ulc = urc = llc = lrc = vc;   /* default same as vertical */

  if (v == 0 && h == 0) {
        ulc = ACS_ULCORNER;
        urc = ACS_URCORNER;
        llc = ACS_LLCORNER;
        lrc = ACS_LRCORNER;
        hc = ACS_HLINE;
        vc = ACS_VLINE;
  }
  for (i = xmin + 1; i <= xmax - 1; i++) {
        win->_line[ymin][i] = hc | win->_attrs;
        win->_line[ymax][i] = hc | win->_attrs;
  }
```

```
  for (i = ymin + 1; i <= ymax - 1; i++) {
        win->_line[i][xmin] = vc | win->_attrs;
        win->_line[i][xmax] = vc | win->_attrs;
  }
  win->_line[ymin][xmin] = ulc | win->_attrs;
  win->_line[ymin][xmax] = urc | win->_attrs;
  win->_line[ymax][xmin] = llc | win->_attrs;
  win->_line[ymax][xmax] = lrc | win->_attrs;

  for (i = ymin; i <= ymax; i++) {
        if (win->_minchng[i] == _NO_CHANGE) {
                win->_minchng[i] = xmin;
                win->_maxchng[i] = xmax;
        } else {
                win->_minchng[i] = min(win->_minchng[i], xmin);
                win->_maxchng[i] = max(win->_maxchng[i], xmax);
        }
  }
  return(OK);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/wclear.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <curses.h>
#include "curspriv.h"

/****************************************************************/
/* Wclear() fills all lines of window 'win' with blanks, and    */
/* Marks the window to be cleared at next refresh operation.    */
/****************************************************************/

void wclear(win)
WINDOW *win;
{
  werase(win);
  win->_clear = TRUE;
}                               /* wclear */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/wclrtobot.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <curses.h>
#include "curspriv.h"

/****************************************************************/
/* Wclrtobot() fills the right half of the cursor line of       */
/* Window 'win', and all lines below it with blanks.            */
/****************************************************************/

int wclrtobot(win)
WINDOW *win;
```

```
{
  int y, minx, startx, *ptr, *end, *maxx, blank;

  blank = ' ' | (win->_attrs & ATR_MSK);
  startx = win->_curx;
  for (y = win->_cury; y <= win->_regbottom; y++) {
        minx = _NO_CHANGE;
        end = &win->_line[y][win->_maxx];
        for (ptr = &win->_line[y][startx]; ptr <= end; ptr++) {
                if (*ptr != blank) {
                        maxx = ptr;
                        if (minx == _NO_CHANGE) minx = ptr - win->_line[y];
                        *ptr = blank;
                }               /* if */
        }                       /* for */
        if (minx != _NO_CHANGE) {
                if ((win->_minchng[y] > minx) || (win->_minchng[y] == _NO_CHANGE))
                        win->_minchng[y] = minx;
                if (win->_maxchng[y] < maxx - win->_line[y])
                        win->_maxchng[y] = maxx - win->_line[y];
        }                       /* if */
        startx = 0;
  }
  return(OK);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/wclrtoeol.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <curses.h>
#include "curspriv.h"

/****************************************************************/
/* Wclrtoeol() fills the half of the cursor line to the right   */
/* Of the cursor in window 'win' with blanks.                   */
/****************************************************************/

int wclrtoeol(win)
WINDOW *win;
{
  int *maxx, *ptr, *end, y, x, minx, blank;

  y = win->_cury;
  x = win->_curx;
  blank = ' ' | (win->_attrs & ATR_MSK);

  end = &win->_line[y][win->_maxx];
  minx = _NO_CHANGE;
  maxx = &win->_line[y][x];
  for (ptr = maxx; ptr <= end; ptr++) {
        if (*ptr != blank) {
                maxx = ptr;
                if (minx == _NO_CHANGE) minx = ptr - win->_line[y];
                *ptr = blank;
```

```
        }                       /* if */
  }                             /* for */

  if (minx != _NO_CHANGE) {
        if (win->_minchng[y] > minx || win->_minchng[y] == _NO_CHANGE)
                win->_minchng[y] = minx;
        if (win->_maxchng[y] < maxx - win->_line[y])
                win->_maxchng[y] = maxx - win->_line[y];
  }
  return(OK);
}
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/wdelch.c
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <curses.h>
#include "curspriv.h"

/* Wdelch() deletes the character at the window cursor, and the
   characters to the right of it are shifted left, inserting a
   space at the last position of the line.
*/

int wdelch(win)
WINDOW *win;
{
  int *temp1;
  int *temp2;
  int *end;
  int y = win->_cury;
  int x = win->_curx;
  int maxx = win->_maxx;

  end = &win->_line[y][maxx];
  temp1 = &win->_line[y][x];
  temp2 = temp1 + 1;
  while (temp1 < end) *temp1++ = *temp2++;
  *temp1 = ' ' | (win->_attrs & ATR_MSK);
  win->_maxchng[y] = maxx;
  if (win->_minchng[y] == _NO_CHANGE || win->_minchng[y] > x)
        win->_minchng[y] = x;
  return(OK);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/curses/wdeleteln.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <curses.h>
#include "curspriv.h"

/****************************************************************/
/* Wdeleteln() deletes the line at the window cursor, and the   */
/* Lines below it are shifted up, inserting a blank line at     */
/* The bottom of the window.                                    */
/****************************************************************/

int wdeleteln(win)
WINDOW *win;
{
  int *end, *temp, y, blank;

  blank = ' ' | (win->_attrs & ATR_MSK);

  temp = win->_line[win->_cury];
  for (y = win->_cury; y < win->_regbottom; y++) {
        win->_line[y] = win->_line[y + 1];
        win->_minchng[y] = 0;
        win->_maxchng[y] = win->_maxx;
  }
  win->_minchng[y] = 0;
  win->_maxchng[y] = win->_maxx;
  win->_line[win->_regbottom] = temp;
  for (end = &(temp[win->_maxx]); temp <= end;) *temp++ = blank;
  return(OK);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/curses/werase.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <curses.h>
#include "curspriv.h"

/****************************************************************/
/* Werase() fills all lines of window 'win' with blanks and po- */
/* Sitions the cursor at home in the scroll region.             */
/****************************************************************/

void werase(win)
WINDOW *win;
{
  int *end, *start, y, blank;

  blank = ' ' | (win->_attrs & ATR_MSK);

  for (y = win->_regtop; y <= win->_regbottom; y++) {   /* clear all lines */
        start = win->_line[y];
        end = &start[win->_maxx];
        while (start <= end)    /* clear all line */
                *start++ = blank;
```

```
          win->_minchng[y] = 0;
          win->_maxchng[y] = win->_maxx;
    }
  win->_cury = win->_regtop;    /* cursor home */
  win->_curx = 0;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/wgetch.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <curses.h>
#include <stdio.h>
#include "curspriv.h"

int wgetch(win)
WINDOW *win;
{
  bool weset = FALSE;
  char inp;

  if (!win->_scroll && (win->_flags & _FULLWIN)
      && win->_curx == win->_maxx - 1 && win->_cury == win->_maxy - 1)
        return ERR;
  if (_cursvar.echoit && !_cursvar.rawmode) {
        cbreak();
        weset++;
  }
  inp = getchar();
  if (_cursvar.echoit) {
        mvwaddch(curscr, win->_cury + win->_begy,
                 win->_curx + win->_begx, inp);
        waddch(win, inp);
  }
  if (weset) nocbreak();
  return inp;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/wgetstr.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <curses.h>
#include "curspriv.h"

/****************************************************************/
/* Wgetstr(win,str) reads in a string (terminated by \n or \r)  */
/* To the buffer pointed to by 'str', and displays the input    */
/* In window 'win'. The user's erase and kill characters are    */
/* Active.                                                      */
/****************************************************************/

```

```
int wgetstr(win, str)
WINDOW *win;
char *str;
{
  while ((*str = wgetch(win)) != ERR && *str != '\n') str++;
  if (*str == ERR) {
        *str = '\0';
        return ERR;
  }
  *str = '\0';
  return OK;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/windel.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/****************************************************************/
/* Delwin() routine of the PCcurses package.                   */
/*                                                              */
/****************************************************************/
/* This version of curses is based on ncurses, a curses version */
/* Originally written by Pavel Curtis at Cornell University.    */
/* I have made substantial changes to make it run on IBM PC's,  */
/* And therefore consider myself free to make it public domain. */
/*              Bjorn Larsson (...mcvax!enea!infovax!bl)        */
/****************************************************************/
/* 1.0: Release:                                        870515  */
/****************************************************************/
/* Modified to run under the MINIX operating system by Don Cope */
/* These changes are also released into the public domain.      */
/*                                                      900906  */
/****************************************************************/

#include <stdlib.h>
#include <curses.h>
#include "curspriv.h"

/****************************************************************/
/* Delwin() deallocates all data allocated by 'win'. If 'win'   */
/* Is a subwindow, it uses the original window's lines for sto- */
/* Rage, and thus the line arrays are not deallocated.          */
/****************************************************************/

void delwin(win)
WINDOW *win;
{
  int i;

  if (!(win->_flags & _SUBWIN)) {       /* subwindow uses 'parent's' lines */
        for (i = 0; i <= win->_maxy && win->_line[i]; i++)
                free(win->_line[i]);
  }
  free(win->_minchng);
  free(win->_maxchng);
  free(win->_line);
  free(win);
```

```
}                               /* delwin */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/winmove.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/****************************************************************/
/* Mvwin() routine of the PCcurses package                      */
/*                                                              */
/****************************************************************/
/* This version of curses is based on ncurses, a curses version */
/* Originally written by Pavel Curtis at Cornell University.    */
/* I have made substantial changes to make it run on IBM PC's,  */
/* And therefore consider myself free to make it public domain. */
/*              Bjorn Larsson (...mcvax!enea!infovax!bl)        */
/****************************************************************/
/* 1.0: Release:                                        870515  */
/****************************************************************/
/* Modified to run under the MINIX operating system by Don Cope */
/* These changes are also released into the public domain.      */
/*                                                      900906  */
/****************************************************************/

#include <curses.h>
#include "curspriv.h"

/****************************************************************/
/* Mvwin() moves window 'win' to position (begx, begy) on the   */
/* Screen.                                                      */
/****************************************************************/

int mvwin(win, begy, begx)
WINDOW *win;
int begy, begx;
{
  if ((begy + win->_maxy) > (LINES - 1) || (begx + win->_maxx) > (COLS - 1))
        return(ERR);
  win->_begy = begy;
  win->_begx = begx;
  touchwin(win);
  return(OK);
}                               /* mvwin */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/winsch.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <curses.h>
#include "curspriv.h"

/* Winsch() inserts character 'c' at the cursor position in
   window 'win'. The cursor is advanced.
*/

int winsch(win, c)
WINDOW *win;
char c;
{
  int *temp1;
  int *temp2;
  int *end;
  int x = win->_curx;
  int y = win->_cury;
  int maxx = win->_maxx;

  if ((c < ' ') && (c == '\n' || c == '\r' || c == '\t' || c == '\b'))
        return(waddch(win, c));
  end = &win->_line[y][x];
  temp1 = &win->_line[y][maxx];
  temp2 = temp1 - 1;
  if (c < ' ')                  /* if CTRL-char make space for 2 */
        temp2--;
  while (temp1 > end) *temp1-- = *temp2--;
  win->_maxchng[y] = maxx;
  if ((win->_minchng[y] == _NO_CHANGE) || (win->_minchng[y] > x))
        win->_minchng[y] = x;
  return(waddch(win, c));       /* fixes CTRL-chars too */
}                               /* winsch */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/winscrol.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/****************************************************************/
/* Scroll() routine of the PCcurses package                     */
/*                                                              */
/****************************************************************/
/* This version of curses is based on ncurses, a curses version */
/* Originally written by Pavel Curtis at Cornell University.    */
/* I have made substantial changes to make it run on IBM PC's,  */
/* And therefore consider myself free to make it public domain. */
/*              Bjorn Larsson (...mcvax!enea!infovax!bl)        */
/****************************************************************/
/* 1.0: Release:                                        870515  */
/****************************************************************/
/* Modified to run under the MINIX operating system by Don Cope */
/* These changes are also released into the public domain.      */
/*                                                      900906  */
```

```
/****************************************************************/

#include <curses.h>
#include "curspriv.h"

/****************************************************************/
/* Scroll() scrolls the scrolling region of 'win', but only if  */
/* Scrolling is allowed and if the cursor is inside the scrol-  */
/* Ling region.                                                 */
/****************************************************************/

void scroll(win)
WINDOW *win;
{
  int i;
  int *ptr;
  int *temp;
  static int blank;

  blank = ' ' | (win->_attrs & ATR_MSK);
  if ((!win->_scroll)           /* check if window scrolls */
      ||(win->_cury < win->_regtop)     /* and cursor in region */
      ||(win->_cury > win->_regbottom)
        )
        return;

  temp = win->_line[win->_regtop];
  for (i = win->_regtop; i < win->_regbottom; i++) {
        win->_line[i] = win->_line[i + 1];      /* re-arrange line pointers */
        win->_minchng[i] = 0;
        win->_maxchng[i] = win->_maxx;
  }
  for (ptr = temp; ptr - temp <= win->_maxx; ptr++)
        *ptr = blank;           /* make a blank line */
  win->_line[win->_regbottom] = temp;
  if (win->_cury > win->_regtop)/* if not on top line */
        win->_cury--;           /* cursor scrolls too */
  win->_minchng[win->_regbottom] = 0;
  win->_maxchng[win->_regbottom] = win->_maxx;
}                               /* scroll */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/curses/winsertln.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <curses.h>
#include "curspriv.h"

/****************************************************************/
/* Winsertln() inserts a blank line instead of the cursor line  */
/* In window 'win' and pushes other lines down.                 */
/****************************************************************/

int winsertln(win)
WINDOW *win;
```

```
{
  int *temp, *end, y, blank;

  blank = ' ' | (win->_attrs & ATR_MSK);
  temp = win->_line[win->_regbottom];
  for (y = win->_regbottom; y > win->_cury; y--) {
        win->_line[y] = win->_line[y - 1];
        win->_minchng[y] = 0;
        win->_maxchng[y] = win->_maxx;
  }
  win->_line[win->_cury] = temp;
  for (end = &temp[win->_maxx]; temp <= end; temp++) *temp = blank;
  win->_minchng[win->_cury] = 0;
  win->_maxchng[win->_cury] = win->_maxx;
  return(OK);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/curses/wintouch.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/****************************************************************/
/* Touchwin() routine of the PCcurses package                   */
/*                                                              */
/****************************************************************/
/* This version of curses is based on ncurses, a curses version */
/* Originally written by Pavel Curtis at Cornell University.    */
/* I have made substantial changes to make it run on IBM PC's,  */
/* And therefore consider myself free to make it public domain. */
/*              Bjorn Larsson (...mcvax!enea!infovax!bl)        */
/****************************************************************/
/* 1.0: Release:                                        870515  */
/****************************************************************/
/* Modified to run under the MINIX operating system by Don Cope */
/* These changes are also released into the public domain.      */
/*                                                      900906  */
/****************************************************************/

#include <curses.h>
#include "curspriv.h"

/****************************************************************/
/* Touchwin() marks all lines of window 'win' as changed, from  */
/* The first to the last character on the line.                 */
/****************************************************************/

void touchwin(win)
WINDOW *win;
{
  int y;
  int maxy;
  int maxx;

  maxy = win->_maxy;
  maxx = win->_maxx;

```

```
  for (y = 0; y <= maxy; y++) {
        win->_minchng[y] = 0;
        win->_maxchng[y] = maxx;
  }                             /* for */
}                               /* touchwin */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/editline/editline.h
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*  $Revision: 1.3 $
**
**  Internal header file for editline library.
*/
#include <stdio.h>
#if     defined(HAVE_STDLIB)
#include <stdlib.h>
#include <string.h>
#endif  /* defined(HAVE_STDLIB) */
#if     defined(SYS_UNIX)
#include "unix.h"
#endif  /* defined(SYS_UNIX) */
#if     defined(SYS_OS9)
#include "os9.h"
#endif  /* defined(SYS_OS9) */

#if     !defined(SIZE_T)
#define SIZE_T  unsigned int
#endif  /* !defined(SIZE_T) */

typedef unsigned char   CHAR;

#if     defined(HIDE)
#define STATIC  static
#else
#define STATIC  /* NULL */
#endif  /* !defined(HIDE) */

#if     !defined(CONST)
#if     defined(__STDC__)
#define CONST   const
#else
#define CONST
#endif  /* defined(__STDC__) */
#endif  /* !defined(CONST) */


#define MEM_INC         64
#define SCREEN_INC      256

#define DISPOSE(p)      free((char *)(p))
#define NEW(T, c)       \
        ((T *)malloc((unsigned int)(sizeof (T) * (c))))
#define RENEW(p, T, c)  \
        (p = (T *)realloc((char *)(p), (unsigned int)(sizeof (T) * (c))))
```

```
#define COPYFROMTO(new, p, len) \
        (void)memcpy((char *)(new), (char *)(p), (int)(len))


/*
**  Variables and routines internal to this package.
*/
extern int      rl_eof;
extern int      rl_erase;
extern int      rl_intr;
extern int      rl_kill;
extern int      rl_quit;
extern char     *rl_complete();
extern int      rl_list_possib();
extern void     rl_ttyset();
extern void     rl_add_slash();

#if     !defined(HAVE_STDLIB)
extern char     *getenv();
extern char     *malloc();
extern char     *realloc();
extern char     *memcpy();
extern char     *strcat();
extern char     *strchr();
extern char     *strrchr();
extern char     *strcpy();
extern int      strcmp();
extern int      strlen();
extern int      strncmp();
#endif  /* !defined(HAVE_STDLIB) */

#if     defined(NEED_STRDUP)
extern char     *strdup();
#endif
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/editline/os9.h
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*  $Revision: 1.1 $
**
**  Editline system header file for OS-9 (on 68k).
*/

#define CRLF            "\r\l"
#define FORWARD         extern

#include <dir.h>
typedef struct direct   DIRENTRY;
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/editline/unix.h
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*  $Revision: 1.1 $
**
**  Editline system header file for Unix.
*/

#define CRLF            "\r\n"
#define FORWARD         STATIC

#include <sys/types.h>
#include <sys/stat.h>

#if     defined(USE_DIRENT)
#include <dirent.h>
typedef struct dirent   DIRENTRY;
#else
#include <sys/dir.h>
typedef struct direct   DIRENTRY;
#endif  /* defined(USE_DIRENT) */

#if     !defined(S_ISDIR)
#define S_ISDIR(m)              (((m) & S_IFMT) == S_IFDIR)
#endif  /* !defined(S_ISDIR) */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/editline/complete.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*  $Revision: 1.4 $
**
**  History and file completion functions for editline library.
*/
#include "editline.h"


#if     defined(NEED_STRDUP)
/*
**  Return an allocated copy of a string.
*/
char *
strdup(p)
    char        *p;
{
    char        *new;

    if ((new = NEW(char, strlen(p) + 1)) != NULL)
        (void)strcpy(new, p);
    return new;
}
#endif  /* defined(NEED_STRDUP) */

/*
**  strcmp-like sorting predicate for qsort.
```

```
*/
STATIC int
compare(p1, p2)
    CONST void  *p1;
    CONST void  *p2;
{
    CONST char  **v1;
    CONST char  **v2;

    v1 = (CONST char **)p1;
    v2 = (CONST char **)p2;
    return strcmp(*v1, *v2);
}

/*
**  Fill in *avp with an array of names that match file, up to its length.
**  Ignore . and .. .
*/
STATIC int
FindMatches(dir, file, avp)
    char        *dir;
    char        *file;
    char        ***avp;
{
    char        **av;
    char        **new;
    char        *p;
    DIR         *dp;
    DIRENTRY    *ep;
    SIZE_T      ac;
    SIZE_T      len;
    SIZE_T      choices;
    SIZE_T      total;

    if ((dp = opendir(dir)) == NULL)
        return 0;

    av = NULL;
    ac = 0;
    len = strlen(file);
    choices = 0;
    total = 0;
    while ((ep = readdir(dp)) != NULL) {
        p = ep->d_name;
        if (p[0] == '.' && (p[1] == '\0' || (p[1] == '.' && p[2] == '\0')))
            continue;
        if (len && strncmp(p, file, len) != 0)
            continue;

        choices++;
        if ((total += strlen(p)) > 1024) {
            /* This is a bit too much. */
            while (ac > 0) DISPOSE(av[--ac]);
            continue;
        }

        if ((ac % MEM_INC) == 0) {
            if ((new = NEW(char*, ac + MEM_INC)) == NULL) {
                total = 0;
                break;
```

```
            }
            if (ac) {
                COPYFROMTO(new, av, ac * sizeof (char **));
                DISPOSE(av);
            }
            *avp = av = new;
        }

        if ((av[ac] = strdup(p)) == NULL) {
            if (ac == 0)
                DISPOSE(av);
            total = 0;
            break;
        }
        ac++;
    }

    /* Clean up and return. */
    (void)closedir(dp);
    if (total > 1024) {
        char many[sizeof(total) * 3];
        p = many + sizeof(many);
        *--p = '\0';
        while (choices > 0) {
           *--p = '0' + choices % 10;
           choices /= 10;
        }
        while (p > many + sizeof(many) - 8) *--p = ' ';
        if ((p = strdup(p)) != NULL) av[ac++] = p;
        if ((p = strdup("choices")) != NULL) av[ac++] = p;
    } else {
        if (ac)
            qsort(av, ac, sizeof (char **), compare);
    }
    return ac;
}

/*
**  Split a pathname into allocated directory and trailing filename parts.
*/
STATIC int
SplitPath(path, dirpart, filepart)
    char        *path;
    char        **dirpart;
    char        **filepart;
{
    static char DOT[] = ".";
    char        *dpart;
    char        *fpart;

    if ((fpart = strrchr(path, '/')) == NULL) {
        if ((dpart = strdup(DOT)) == NULL)
            return -1;
        if ((fpart = strdup(path)) == NULL) {
            DISPOSE(dpart);
            return -1;
        }
    }
    else {
        if ((dpart = strdup(path)) == NULL)
```

```
                return -1;
            dpart[fpart - path + 1] = '\0';
            if ((fpart = strdup(++fpart)) == NULL) {
                DISPOSE(dpart);
                return -1;
            }
        }
        *dirpart = dpart;
        *filepart = fpart;
        return 0;
    }

    /*
    **  Attempt to complete the pathname, returning an allocated copy.
    **  Fill in *unique if we completed it, or set it to 0 if ambiguous.
    */
    char *
    rl_complete(pathname, unique)
        char        *pathname;
        int         *unique;
    {
        char        **av;
        char        *dir;
        char        *file;
        char        *new;
        char        *p;
        SIZE_T      ac;
        SIZE_T      end;
        SIZE_T      i;
        SIZE_T      j;
        SIZE_T      len;

        if (SplitPath(pathname, &dir, &file) < 0)
            return NULL;
        if ((ac = FindMatches(dir, file, &av)) == 0) {
            DISPOSE(dir);
            DISPOSE(file);
            return NULL;
        }

        p = NULL;
        len = strlen(file);
        if (ac == 1) {
            /* Exactly one match -- finish it off. */
            *unique = 1;
            j = strlen(av[0]) - len + 2;
            if ((p = NEW(char, j + 1)) != NULL) {
                COPYFROMTO(p, av[0] + len, j);
                if ((new = NEW(char, strlen(dir) + strlen(av[0]) + 2)) != NULL) {
                    (void)strcpy(new, dir);
                    (void)strcat(new, "/");
                    (void)strcat(new, av[0]);
                    rl_add_slash(new, p);
                    DISPOSE(new);
                }
            }
        }
        else {
            *unique = 0;
            if (len) {
```

```
                /* Find largest matching substring. */
                for (i = len, end = strlen(av[0]); i < end; i++)
                    for (j = 1; j < ac; j++)
                        if (av[0][i] != av[j][i])
                            goto breakout;
  breakout:
                if (i > len) {
                    j = i - len + 1;
                    if ((p = NEW(char, j)) != NULL) {
                        COPYFROMTO(p, av[0] + len, j);
                        p[j - 1] = '\0';
                    }
                }
            }
        }

        /* Clean up and return. */
        DISPOSE(dir);
        DISPOSE(file);
        for (i = 0; i < ac; i++)
            DISPOSE(av[i]);
        DISPOSE(av);
        return p;
    }

    /*
    **  Return all possible completions.
    */
    int
    rl_list_possib(pathname, avp)
        char        *pathname;
        char        ***avp;
    {
        char        *dir;
        char        *file;
        int         ac;

        if (SplitPath(pathname, &dir, &file) < 0)
            return 0;
        ac = FindMatches(dir, file, avp);
        DISPOSE(dir);
        DISPOSE(file);
        return ac;
    }
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/editline/editline.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
    /*  $Revision: 1.7 $
    **
    **  Main editing routines for editline library.
    */
    #include "editline.h"
    #include <signal.h>
    #include <ctype.h>

    /*
    **  Manifest constants.
```

```
*/
#define SCREEN_WIDTH    80
#define SCREEN_ROWS     24
#define NO_ARG          (-1)
#define DEL             127
#define CTL(x)          ((x) & 0x1F)
#define ISCTL(x)        ((x) && (x) < ' ')
#define UNCTL(x)        ((x) + 64)
#define META(x)         ((x) | 0x80)
#define ISMETA(x)       ((x) & 0x80)
#define UNMETA(x)       ((x) & 0x7F)
#if     !defined(HIST_SIZE)
#define HIST_SIZE       20
#endif  /* !defined(HIST_SIZE) */

/*
**  Command status codes.
*/
typedef enum _STATUS {
    CSdone, CSeof, CSmove, CSdispatch, CSstay, CSsignal
} STATUS;

/*
**  The type of case-changing to perform.
*/
typedef enum _CASE {
    TOupper, TOlower
} CASE;

/*
**  Key to command mapping.
*/
typedef struct _KEYMAP {
    CHAR        Key;
    STATUS      (*Function)();
} KEYMAP;

/*
**  Command history structure.
*/
typedef struct _HISTORY {
    int         Size;
    int         Pos;
    CHAR        *Lines[HIST_SIZE];
} HISTORY;

/*
**  Globals.
*/
int             rl_eof;
int             rl_erase;
int             rl_intr;
int             rl_kill;
int             rl_quit;

STATIC CHAR             NIL[] = "";
STATIC CONST CHAR       *Input = NIL;
STATIC CHAR             *Line;
STATIC CONST char       *Prompt;
STATIC CHAR             *Yanked;
```

```
STATIC char             *Screen;
STATIC char             NEWLINE[]= CRLF;
STATIC HISTORY          H;
STATIC int              Repeat;
STATIC int              End;
STATIC int              Mark;
STATIC int              OldPoint;
STATIC int              Point;
STATIC int              PushBack;
STATIC int              Pushed;
STATIC int              Signal;
FORWARD KEYMAP          Map[33];
FORWARD KEYMAP          MetaMap[17];
STATIC SIZE_T           Length;
STATIC SIZE_T           ScreenCount;
STATIC SIZE_T           ScreenSize;
STATIC char             *backspace;
STATIC int              TTYwidth;
STATIC int              TTYrows;

/* Display print 8-bit chars as M-x' or as the actual 8-bit char? */
int             rl_meta_chars = 0;

/*
**  Declarations.
*/
STATIC CHAR     *editinput();
extern int      read();
extern int      write();
#if     defined(USE_TERMCAP)
extern char     *getenv();
extern char     *tgetstr();
extern int      tgetent();
#endif  /* defined(USE_TERMCAP) */

/*
**  TTY input/output functions.
*/

STATIC void
TTYflush()
{
    if (ScreenCount) {
        (void)write(1, Screen, ScreenCount);
        ScreenCount = 0;
    }
}

STATIC void
TTYput(c)
    CHAR        c;
{
    Screen[ScreenCount] = c;
    if (++ScreenCount >= ScreenSize - 1) {
        ScreenSize += SCREEN_INC;
        RENEW(Screen, char, ScreenSize);
    }
}

STATIC void
```

```
TTYputs(p)
    CHAR        *p;
{
    while (*p)
        TTYput(*p++);
}

STATIC void
TTYshow(c)
    CHAR        c;
{
    if (c == DEL) {
        TTYput('^');
        TTYput('?');
    }
    else if (ISCTL(c)) {
        TTYput('^');
        TTYput(UNCTL(c));
    }
    else if (rl_meta_chars && ISMETA(c)) {
        TTYput('M');
        TTYput('-');
        TTYput(UNMETA(c));
    }
    else
        TTYput(c);
}

STATIC void
TTYstring(p)
    CHAR        *p;
{
    while (*p)
        TTYshow(*p++);
}

STATIC unsigned int
TTYget()
{
    CHAR        c;

    TTYflush();
    if (Pushed) {
        Pushed = 0;
        return PushBack;
    }
    if (*Input)
        return *Input++;
    return read(0, &c, (SIZE_T)1) == 1 ? c : EOF;
}

#define TTYback()       (backspace ? TTYputs((CHAR *)backspace) : TTYput('\b'))

STATIC void
TTYbackn(n)
    int         n;
{
    while (--n >= 0)
        TTYback();
}
```

```
STATIC void
TTYinfo()
{
    static int          init;
#if     defined(USE_TERMCAP)
    char                *term;
    char                buff[2048];
    char                *bp;
#endif  /* defined(USE_TERMCAP) */
#if     defined(TIOCGWINSZ)
    struct winsize      W;
#endif  /* defined(TIOCGWINSZ) */

    if (init) {
#if     defined(TIOCGWINSZ)
        /* Perhaps we got resized. */
        if (ioctl(0, TIOCGWINSZ, &W) >= 0
         && W.ws_col > 0 && W.ws_row > 0) {
            TTYwidth = (int)W.ws_col;
            TTYrows = (int)W.ws_row;
        }
#endif  /* defined(TIOCGWINSZ) */
        return;
    }
    init++;

    TTYwidth = TTYrows = 0;
#if     defined(USE_TERMCAP)
    bp = &buff[0];
    if ((term = getenv("TERM")) == NULL)
        term = "dumb";
    if (tgetent(buff, term) < 0) {
       TTYwidth = SCREEN_WIDTH;
       TTYrows = SCREEN_ROWS;
       return;
    }
    if ((backspace = tgetstr("le", &bp)) != NULL)
        backspace = strdup(backspace);
    TTYwidth = tgetnum("co");
    TTYrows = tgetnum("li");
#endif  /* defined(USE_TERMCAP) */

#if     defined(TIOCGWINSZ)
    if (ioctl(0, TIOCGWINSZ, &W) >= 0) {
        TTYwidth = (int)W.ws_col;
        TTYrows = (int)W.ws_row;
    }
#endif  /* defined(TIOCGWINSZ) */

    if (TTYwidth <= 0 || TTYrows <= 0) {
        TTYwidth = SCREEN_WIDTH;
        TTYrows = SCREEN_ROWS;
    }
}


/*
**  Print an array of words in columns.
*/
```

```
STATIC void
columns(ac, av)
    int         ac;
    CHAR        **av;
{
    CHAR        *p;
    int         i;
    int         j;
    int         k;
    int         len;
    int         skip;
    int         longest;
    int         cols;

    /* Find longest name, determine column count from that. */
    for (longest = 0, i = 0; i < ac; i++)
        if ((j = strlen((char *)av[i])) > longest)
            longest = j;
    cols = TTYwidth / (longest + 3);

    TTYputs((CHAR *)NEWLINE);
    for (skip = ac / cols + 1, i = 0; i < skip; i++) {
        for (j = i; j < ac; j += skip) {
            for (p = av[j], len = strlen((char *)p), k = len; --k >= 0; p++)
                TTYput(*p);
            if (j + skip < ac)
                while (++len < longest + 3)
                    TTYput(' ');
        }
        TTYputs((CHAR *)NEWLINE);
    }
}

STATIC void
reposition()
{
    int         i;
    CHAR        *p;

    TTYput('\r');
    TTYputs((CONST CHAR *)Prompt);
    for (i = Point, p = Line; --i >= 0; p++)
        TTYshow(*p);
}

STATIC void
left(Change)
    STATUS      Change;
{
    TTYback();
    if (Point) {
        if (ISCTL(Line[Point - 1]))
            TTYback();
        else if (rl_meta_chars && ISMETA(Line[Point - 1])) {
            TTYback();
            TTYback();
        }
    }
    if (Change == CSmove)
        Point--;
```

```
}

STATIC void
right(Change)
    STATUS      Change;
{
    TTYshow(Line[Point]);
    if (Change == CSmove)
        Point++;
}

STATIC STATUS
ring_bell()
{
    TTYput('\07');
    TTYflush();
    return CSstay;
}

STATIC STATUS
do_macro(c)
    unsigned int        c;
{
    CHAR                name[4];

    name[0] = '_';
    name[1] = c;
    name[2] = '_';
    name[3] = '\0';

    if ((Input = (CHAR *)getenv((char *)name)) == NULL) {
        Input = NIL;
        return ring_bell();
    }
    return CSstay;
}

STATIC STATUS
do_forward(move)
    STATUS      move;
{
    int         i;
    CHAR        *p;

    i = 0;
    do {
        p = &Line[Point];
        for ( ; Point < End && (*p == ' ' || !isalnum(*p)); Point++, p++)
            if (move == CSmove)
                right(CSstay);

        for (; Point < End && isalnum(*p); Point++, p++)
            if (move == CSmove)
                right(CSstay);

        if (Point == End)
            break;
    } while (++i < Repeat);

    return CSstay;
```

```
}

STATIC STATUS
do_case(type)
    CASE        type;
{
    int         i;
    int         end;
    int         count;
    CHAR        *p;

    (void)do_forward(CSstay);
    if (OldPoint != Point) {
        if ((count = Point - OldPoint) < 0)
            count = -count;
        Point = OldPoint;
        if ((end = Point + count) > End)
            end = End;
        for (i = Point, p = &Line[i]; i < end; i++, p++) {
            if (type == TOupper) {
                if (islower(*p))
                    *p = toupper(*p);
            }
            else if (isupper(*p))
                *p = tolower(*p);
            right(CSmove);
        }
    }
    return CSstay;
}

STATIC STATUS
case_down_word()
{
    return do_case(TOlower);
}

STATIC STATUS
case_up_word()
{
    return do_case(TOupper);
}

STATIC void
ceol()
{
    int         extras;
    int         i;
    CHAR        *p;

    for (extras = 0, i = Point, p = &Line[i]; i <= End; i++, p++) {
        TTYput(' ');
        if (ISCTL(*p)) {
            TTYput(' ');
            extras++;
        }
        else if (rl_meta_chars && ISMETA(*p)) {
            TTYput(' ');
            TTYput(' ');
            extras += 2;
```

```
            }
        }

        for (i += extras; i > Point; i--)
            TTYback();
    }

    STATIC void
    clear_line()
    {
        Point = -strlen(Prompt);
        TTYput('\r');
        ceol();
        Point = 0;
        End = 0;
        Line[0] = '\0';
    }

    STATIC STATUS
    insert_string(p)
        CHAR        *p;
    {
        SIZE_T      len;
        int         i;
        CHAR        *new;
        CHAR        *q;

        len = strlen((char *)p);
        if (End + len >= Length) {
            if ((new = NEW(CHAR, Length + len + MEM_INC)) == NULL)
                return CSstay;
            if (Length) {
                COPYFROMTO(new, Line, Length);
                DISPOSE(Line);
            }
            Line = new;
            Length += len + MEM_INC;
        }

        for (q = &Line[Point], i = End - Point; --i >= 0; )
            q[len + i] = q[i];
        COPYFROMTO(&Line[Point], p, len);
        End += len;
        Line[End] = '\0';
        TTYstring(&Line[Point]);
        Point += len;

        return Point == End ? CSstay : CSmove;
    }

    STATIC STATUS
    redisplay()
    {
        TTYputs((CONST CHAR *)NEWLINE);
        TTYputs((CONST CHAR *)Prompt);
        TTYstring(Line);
        return CSmove;
    }

    STATIC STATUS
```

```
toggle_meta_mode()
{
    rl_meta_chars = ! rl_meta_chars;
    return redisplay();
}


STATIC CHAR *
next_hist()
{
    return H.Pos >= H.Size - 1 ? NULL : H.Lines[++H.Pos];
}

STATIC CHAR *
prev_hist()
{
    return H.Pos == 0 ? NULL : H.Lines[--H.Pos];
}

STATIC STATUS
do_insert_hist(p)
    CHAR        *p;
{
    if (p == NULL)
        return ring_bell();
    Point = 0;
    reposition();
    ceol();
    End = 0;
    return insert_string(p);
}

STATIC STATUS
do_hist(move)
    CHAR        *(*move)();
{
    CHAR        *p;
    int         i;

    i = 0;
    do {
        if ((p = (*move)()) == NULL)
            return ring_bell();
    } while (++i < Repeat);
    return do_insert_hist(p);
}

STATIC STATUS
h_next()
{
    return do_hist(next_hist);
}

STATIC STATUS
h_prev()
{
    return do_hist(prev_hist);
}

STATIC STATUS
```

```
h_first()
{
    return do_insert_hist(H.Lines[H.Pos = 0]);
}

STATIC STATUS
h_last()
{
    return do_insert_hist(H.Lines[H.Pos = H.Size - 1]);
}

/*
**  Return zero if pat appears as a substring in text.
*/
STATIC int
substrcmp(text, pat, len)
    char        *text;
    char        *pat;
    int         len;
{
    char        c;

    if ((c = *pat) == '\0')
        return *text == '\0';
    for ( ; *text; text++)
        if (*text == c && strncmp(text, pat, len) == 0)
            return 0;
    return 1;
}

STATIC CHAR *
search_hist(search, move)
    CHAR        *search;
    CHAR        *(*move)();
{
    static CHAR *old_search;
    int         len;
    int         pos;
    int         (*match)();
    char        *pat;

    /* Save or get remembered search pattern. */
    if (search && *search) {
        if (old_search)
            DISPOSE(old_search);
        old_search = (CHAR *)strdup((char *)search);
    }
    else {
        if (old_search == NULL || *old_search == '\0')
            return NULL;
        search = old_search;
    }

    /* Set up pattern-finder. */
    if (*search == '^') {
        match = strncmp;
        pat = (char *)(search + 1);
    }
    else {
        match = substrcmp;
```

```
            pat = (char *)search;
        }
        len = strlen(pat);

        for (pos = H.Pos; (*move)() != NULL; )
            if ((*match)((char *)H.Lines[H.Pos], pat, len) == 0)
                return H.Lines[H.Pos];
        H.Pos = pos;
        return NULL;
    }

    STATIC STATUS
    h_search()
    {
        static int  Searching;
        CONST char  *old_prompt;
        CHAR        *(*move)();
        CHAR        *p;

        if (Searching)
            return ring_bell();
        Searching = 1;

        clear_line();
        old_prompt = Prompt;
        Prompt = "Search: ";
        TTYputs((CONST CHAR *)Prompt);
        move = Repeat == NO_ARG ? prev_hist : next_hist;
        p = editinput();
        Prompt = old_prompt;
        Searching = 0;
        TTYputs((CONST CHAR *)Prompt);
        if (p == NULL && Signal > 0) {
            Signal = 0;
            clear_line();
            return redisplay();
        }
        p = search_hist(p, move);
        clear_line();
        if (p == NULL) {
            (void)ring_bell();
            return redisplay();
        }
        return do_insert_hist(p);
    }

    STATIC STATUS
    fd_char()
    {
        int         i;

        i = 0;
        do {
            if (Point >= End)
                break;
            right(CSmove);
        } while (++i < Repeat);
        return CSstay;
    }
```

```
STATIC void
save_yank(begin, i)
    int         begin;
    int         i;
{
    if (Yanked) {
        DISPOSE(Yanked);
        Yanked = NULL;
    }

    if (i < 1)
        return;

    if ((Yanked = NEW(CHAR, (SIZE_T)i + 1)) != NULL) {
        COPYFROMTO(Yanked, &Line[begin], i);
        Yanked[i] = '\0';
    }
}

STATIC STATUS
delete_string(count)
    int         count;
{
    int         i;
    CHAR        *p;

    if (count <= 0 || End == Point)
        return ring_bell();

    if (count == 1 && Point == End - 1) {
        /* Optimize common case of delete at end of line. */
        End--;
        p = &Line[Point];
        i = 1;
        TTYput(' ');
        if (ISCTL(*p)) {
            i = 2;
            TTYput(' ');
        }
        else if (rl_meta_chars && ISMETA(*p)) {
            i = 3;
            TTYput(' ');
            TTYput(' ');
        }
        TTYbackn(i);
        *p = '\0';
        return CSmove;
    }
    if (Point + count > End && (count = End - Point) <= 0)
        return CSstay;

    if (count > 1)
        save_yank(Point, count);

    for (p = &Line[Point], i = End - (Point + count) + 1; --i >= 0; p++)
        p[0] = p[count];
    ceol();
    End -= count;
    TTYstring(&Line[Point]);
    return CSmove;
```

```
}

STATIC STATUS
bk_char()
{
    int         i;

    i = 0;
    do {
        if (Point == 0)
            break;
        left(CSmove);
    } while (++i < Repeat);

    return CSstay;
}

STATIC STATUS
bk_del_char()
{
    int         i;

    i = 0;
    do {
        if (Point == 0)
            break;
        left(CSmove);
    } while (++i < Repeat);

    return delete_string(i);
}

STATIC STATUS
kill_line()
{
    int         i;

    if (Repeat != NO_ARG) {
        if (Repeat < Point) {
            i = Point;
            Point = Repeat;
            reposition();
            (void)delete_string(i - Point);
        }
        else if (Repeat > Point) {
            right(CSmove);
            (void)delete_string(Repeat - Point - 1);
        }
        return CSmove;
    }

    save_yank(Point, End - Point);
    Line[Point] = '\0';
    ceol();
    End = Point;
    return CSstay;
}

STATIC STATUS
insert_char(c)
```

```
    int         c;
{
    STATUS      s;
    CHAR        buff[2];
    CHAR        *p;
    CHAR        *q;
    int         i;

    if (Repeat == NO_ARG || Repeat < 2) {
        buff[0] = c;
        buff[1] = '\0';
        return insert_string(buff);
    }

    if ((p = NEW(CHAR, Repeat + 1)) == NULL)
        return CSstay;
    for (i = Repeat, q = p; --i >= 0; )
        *q++ = c;
    *q = '\0';
    Repeat = 0;
    s = insert_string(p);
    DISPOSE(p);
    return s;
}

STATIC STATUS
meta()
{
    unsigned int        c;
    KEYMAP              *kp;

    if ((c = TTYget()) == EOF)
        return CSeof;
#if     defined(ANSI_ARROWS)
    /* Also include VT-100 arrows. */
    if (c == '[' || c == 'O')
        switch (c = TTYget()) {
        default:        return ring_bell();
        case EOF:       return CSeof;
        case 'A':       return h_prev();
        case 'B':       return h_next();
        case 'C':       return fd_char();
        case 'D':       return bk_char();
        }
#endif  /* defined(ANSI_ARROWS) */

    if (isdigit(c)) {
        for (Repeat = c - '0'; (c = TTYget()) != EOF && isdigit(c); )
            Repeat = Repeat * 10 + c - '0';
        Pushed = 1;
        PushBack = c;
        return CSstay;
    }

    if (isupper(c))
        return do_macro(c);
    for (kp = MetaMap; kp->Function; kp++)
        if (kp->Key == c)
            return (*kp->Function)();

```

```
    return ring_bell();
}

STATIC STATUS
emacs(c)
    unsigned int        c;
{
    STATUS              s;
    KEYMAP              *kp;

    OldPoint = Point;
    if (rl_meta_chars && ISMETA(c)) {
        Pushed = 1;
        PushBack = UNMETA(c);
        return meta();
    }
    for (kp = Map; kp->Function; kp++)
        if (kp->Key == c)
            break;
    s = kp->Function ? (*kp->Function)() : insert_char((int)c);
    if (!Pushed)
        /* No pushback means no repeat count; hacky, but true. */
        Repeat = NO_ARG;
    return s;
}

STATIC STATUS
TTYspecial(c)
    unsigned int        c;
{
    if (ISMETA(c))
        return CSdispatch;

    if (c == rl_erase || c == DEL)
        return bk_del_char();
    if (c == rl_kill) {
        if (Point != 0) {
            Point = 0;
            reposition();
        }
        Repeat = NO_ARG;
        return kill_line();
    }
    if (c == rl_eof && Point == 0 && End == 0)
        return CSeof;
    if (c == rl_intr) {
        Signal = SIGINT;
        return CSsignal;
    }
    if (c == rl_quit) {
        Signal = SIGQUIT;
        return CSeof;
    }

    return CSdispatch;
}

STATIC CHAR *
editinput()
{
```

```
    unsigned int        c;

    Repeat = NO_ARG;
    OldPoint = Point = Mark = End = 0;
    Line[0] = '\0';

    Signal = -1;
    while ((c = TTYget()) != EOF)
        switch (TTYspecial(c)) {
        case CSdone:
            return Line;
        case CSeof:
            return NULL;
        case CSsignal:
            return (CHAR *)"";
        case CSmove:
            reposition();
            break;
        case CSdispatch:
            switch (emacs(c)) {
            case CSdone:
                return Line;
            case CSeof:
                return NULL;
            case CSsignal:
                return (CHAR *)"";
            case CSmove:
                reposition();
                break;
            case CSdispatch:
            case CSstay:
                break;
            }
            break;
        case CSstay:
            break;
        }
    return NULL;
}

STATIC void
hist_add(p)
    CHAR        *p;
{
    int         i;

    if ((p = (CHAR *)strdup((char *)p)) == NULL)
        return;
    if (H.Size < HIST_SIZE)
        H.Lines[H.Size++] = p;
    else {
        DISPOSE(H.Lines[0]);
        for (i = 0; i < HIST_SIZE - 1; i++)
            H.Lines[i] = H.Lines[i + 1];
        H.Lines[i] = p;
    }
    H.Pos = H.Size - 1;
}

/*
```

```
**  For compatibility with FSF readline.
*/
/* ARGSUSED0 */
void
rl_reset_terminal(p)
    char        *p;
{
}

void
rl_initialize()
{
}

char *
readline(prompt)
    CONST char  *prompt;
{
    CHAR        *line;
    int         s;

    if (Line == NULL) {
        Length = MEM_INC;
        if ((Line = NEW(CHAR, Length)) == NULL)
            return NULL;
    }

    TTYinfo();
    rl_ttyset(0);
    hist_add(NIL);
    ScreenSize = SCREEN_INC;
    Screen = NEW(char, ScreenSize);
    Prompt = prompt ? prompt : (char *)NIL;
    TTYputs((CONST CHAR *)Prompt);
    if ((line = editinput()) != NULL) {
        line = (CHAR *)strdup((char *)line);
        TTYputs((CHAR *)NEWLINE);
        TTYflush();
    }
    rl_ttyset(1);
    DISPOSE(Screen);
    DISPOSE(H.Lines[--H.Size]);
    if (Signal > 0) {
        s = Signal;
        Signal = 0;
        (void)kill(getpid(), s);
    }
    return (char *)line;
}

void
add_history(p)
    char        *p;
{
    if (p == NULL || *p == '\0')
        return;

#if     defined(UNIQUE_HISTORY)
    if (H.Pos && strcmp(p, (char *) H.Lines[H.Pos - 1]) == 0)
        return;
```

```
#endif  /* defined(UNIQUE_HISTORY) */
    if (H.Size && strcmp(p, (char *) H.Lines[H.Size - 1]) == 0)
        return;
    hist_add((CHAR *)p);
}


STATIC STATUS
beg_line()
{
    if (Point) {
        Point = 0;
        return CSmove;
    }
    return CSstay;
}

STATIC STATUS
del_char()
{
    return delete_string(Repeat == NO_ARG ? 1 : Repeat);
}

STATIC STATUS
end_line()
{
    if (Point != End) {
        Point = End;
        return CSmove;
    }
    return CSstay;
}

/*
**  Move back to the beginning of the current word and return an
**  allocated copy of it.
*/
STATIC CHAR *
find_word()
{
    static char SEPS[] = "#:;&|^$=`{}()<>\n\t ";
    CHAR        *p;
    CHAR        *new;
    SIZE_T      len;

    for (p = &Line[Point]; p > Line && strchr(SEPS, (char)p[-1]) == NULL; p--)
        continue;
    len = Point - (p - Line) + 1;
    if ((new = NEW(CHAR, len)) == NULL)
        return NULL;
    COPYFROMTO(new, p, len);
    new[len - 1] = '\0';
    return new;
}

STATIC STATUS
c_possible()
{
    CHAR        **av;
    CHAR        *word;
```

```
    int         ac;

    word = find_word();
    ac = rl_list_possib((char *)word, (char ***)&av);
    if (word)
        DISPOSE(word);
    if (ac) {
        columns(ac, av);
        while (--ac >= 0)
            DISPOSE(av[ac]);
        DISPOSE(av);
        return CSmove;
    }
    return ring_bell();
}

STATIC STATUS
c_complete()
{
    CHAR        *p;
    CHAR        *word;
    int         unique;
    STATUS      s;

    word = find_word();
    p = (CHAR *)rl_complete((char *)word, &unique);
    if (word)
        DISPOSE(word);
    if (p && *p) {
        s = insert_string(p);
#if ANNOYING_NOISE
        if (!unique)
            (void)ring_bell();
#endif
        DISPOSE(p);
        return s;
    }
    return c_possible();
}

STATIC STATUS
accept_line()
{
    Line[End] = '\0';
    return CSdone;
}

STATIC STATUS
transpose()
{
    CHAR        c;

    if (Point) {
        if (Point == End)
            left(CSmove);
        c = Line[Point - 1];
        left(CSstay);
        Line[Point - 1] = Line[Point];
        TTYshow(Line[Point - 1]);
        Line[Point++] = c;
```

```
            TTYshow(c);
        }
        return CSstay;
    }

    STATIC STATUS
    quote()
    {
        unsigned int        c;

        return (c = TTYget()) == EOF ? CSeof : insert_char((int)c);
    }

    STATIC STATUS
    wipe()
    {
        int         i;

        if (Mark > End)
            return ring_bell();

        if (Point > Mark) {
            i = Point;
            Point = Mark;
            Mark = i;
            reposition();
        }

        return delete_string(Mark - Point);
    }

    STATIC STATUS
    mk_set()
    {
        Mark = Point;
        return CSstay;
    }

    STATIC STATUS
    exchange()
    {
        unsigned int        c;

        if ((c = TTYget()) != CTL('X'))
            return c == EOF ? CSeof : ring_bell();

        if ((c = Mark) <= End) {
            Mark = Point;
            Point = c;
            return CSmove;
        }
        return CSstay;
    }

    STATIC STATUS
    yank()
    {
        if (Yanked && *Yanked)
            return insert_string(Yanked);
        return CSstay;
```

```
}

STATIC STATUS
copy_region()
{
    if (Mark > End)
        return ring_bell();

    if (Point > Mark)
        save_yank(Mark, Point - Mark);
    else
        save_yank(Point, Mark - Point);

    return CSstay;
}

STATIC STATUS
move_to_char()
{
    unsigned int        c;
    int                 i;
    CHAR                *p;

    if ((c = TTYget()) == EOF)
        return CSeof;
    for (i = Point + 1, p = &Line[i]; i < End; i++, p++)
        if (*p == c) {
            Point = i;
            return CSmove;
        }
    return CSstay;
}

STATIC STATUS
fd_word()
{
    return do_forward(CSmove);
}

STATIC STATUS
fd_kill_word()
{
    int         i;

    (void)do_forward(CSstay);
    if (OldPoint != Point) {
        i = Point - OldPoint;
        Point = OldPoint;
        return delete_string(i);
    }
    return CSstay;
}

STATIC STATUS
bk_word()
{
    int         i;
    CHAR        *p;

    i = 0;
```

```
    do {
        for (p = &Line[Point]; p > Line && !isalnum(p[-1]); p--)
            left(CSmove);

        for (; p > Line && p[-1] != ' ' && isalnum(p[-1]); p--)
            left(CSmove);

        if (Point == 0)
            break;
    } while (++i < Repeat);

    return CSstay;
}

STATIC STATUS
bk_kill_word()
{
    (void)bk_word();
    if (OldPoint != Point)
        return delete_string(OldPoint - Point);
    return CSstay;
}

STATIC int
argify(line, avp)
    CHAR        *line;
    CHAR        ***avp;
{
    CHAR        *c;
    CHAR        **p;
    CHAR        **new;
    int         ac;
    int         i;

    i = MEM_INC;
    if ((*avp = p = NEW(CHAR*, i))== NULL)
         return 0;

    for (c = line; isspace(*c); c++)
        continue;
    if (*c == '\n' || *c == '\0')
        return 0;

    for (ac = 0, p[ac++] = c; *c && *c != '\n'; ) {
        if (isspace(*c)) {
            *c++ = '\0';
            if (*c && *c != '\n') {
                if (ac + 1 == i) {
                    new = NEW(CHAR*, i + MEM_INC);
                    if (new == NULL) {
                        p[ac] = NULL;
                        return ac;
                    }
                    COPYFROMTO(new, p, i * sizeof (char **));
                    i += MEM_INC;
                    DISPOSE(p);
                    *avp = p = new;
                }
                p[ac++] = c;
            }
```

```
        }
        else
            c++;
    }
    *c = '\0';
    p[ac] = NULL;
    return ac;
}

STATIC STATUS
last_argument()
{
    CHAR        **av;
    CHAR        *p;
    STATUS      s;
    int         ac;

    if (H.Size == 1 || (p = H.Lines[H.Size - 2]) == NULL)
        return ring_bell();

    if ((p = (CHAR *)strdup((char *)p)) == NULL)
        return CSstay;
    ac = argify(p, &av);

    if (Repeat != NO_ARG)
        s = Repeat < ac ? insert_string(av[Repeat]) : ring_bell();
    else
        s = ac ? insert_string(av[ac - 1]) : CSstay;

    if (ac)
        DISPOSE(av);
    DISPOSE(p);
    return s;
}

STATIC KEYMAP   Map[33] = {
    {   CTL('@'),       mk_set          },
    {   CTL('A'),       beg_line        },
    {   CTL('B'),       bk_char         },
    {   CTL('D'),       del_char        },
    {   CTL('E'),       end_line        },
    {   CTL('F'),       fd_char         },
    {   CTL('G'),       ring_bell       },
    {   CTL('H'),       bk_del_char     },
    {   CTL('I'),       c_complete      },
    {   CTL('J'),       accept_line     },
    {   CTL('K'),       kill_line       },
    {   CTL('L'),       redisplay       },
    {   CTL('M'),       accept_line     },
    {   CTL('N'),       h_next          },
    {   CTL('O'),       ring_bell       },
    {   CTL('P'),       h_prev          },
    {   CTL('Q'),       ring_bell       },
    {   CTL('R'),       h_search        },
    {   CTL('S'),       ring_bell       },
    {   CTL('T'),       transpose       },
    {   CTL('U'),       ring_bell       },
    {   CTL('V'),       quote           },
    {   CTL('W'),       bk_kill_word    },
    {   CTL('X'),       exchange        },
```

```
        {   CTL('Y'),       yank            },
        {   CTL('Z'),       end_line        },
        {   CTL('['),       meta            },
        {   CTL(']'),       move_to_char    },
        {   CTL('^'),       ring_bell       },
        {   CTL('_'),       ring_bell       },
        {   0,              NULL            }
};

STATIC KEYMAP   MetaMap[17]= {
        {   CTL('H'),       wipe            },
        {   DEL,            wipe            },
        {   ' ',            mk_set          },
        {   '.',            last_argument   },
        {   '<',            h_first         },
        {   '>',            h_last          },
        {   '?',            c_possible      },
        {   'b',            bk_word         },
        {   'd',            fd_kill_word    },
        {   'f',            fd_word         },
        {   'l',            case_down_word  },
        {   'm',            toggle_meta_mode },
        {   'u',            case_up_word    },
        {   'y',            yank            },
        {   'w',            copy_region     },
        {   0,              NULL            }
};
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/editline/sysos9.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*  $Revision: 1.1 $
**
**  OS-9 system-dependant routines for editline library.
*/
#include "editline.h"
#include <sgstat.h>
#include <modes.h>


void
rl_ttyset(Reset)
    int                 Reset;
{
    static struct sgbuf old;
    struct sgbuf        new;


    if (Reset == 0) {
        _gs_opt(0, &old);
        _gs_opt(0, &new);
        new.sg_backsp = 0;      new.sg_delete = 0;      new.sg_echo = 0;
        new.sg_alf = 0;         new.sg_nulls = 0;       new.sg_pause = 0;
        new.sg_page = 0;        new.sg_bspch = 0;       new.sg_dlnch = 0;
        new.sg_eorch = 0;       new.sg_eofch = 0;       new.sg_rlnch = 0;
        new.sg_dulnch = 0;      new.sg_psch = 0;        new.sg_kbich = 0;
```

```
        new.sg_kbach = 0;       new.sg_bsech = 0;       new.sg_bellch = 0;
        new.sg_xon = 0;         new.sg_xoff = 0;        new.sg_tabcr = 0;
        new.sg_tabsiz = 0;
        _ss_opt(0, &new);
        rl_erase = old.sg_bspch;
        rl_kill = old.sg_dlnch;
        rl_eof = old.sg_eofch;
        rl_intr = old.sg_kbich;
        rl_quit = -1;
    }
    else
        _ss_opt(0, &old);
}

void
rl_add_slash(path, p)
    char        *path;
    char        *p;
{
    (void)strcat(p, access(path, S_IREAD | S_IFDIR) ? " " : "/");
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/editline/sysunix.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*  $Revision: 1.2 $
**
**  Unix system-dependant routines for editline library.
*/
#include "editline.h"

#if     defined(HAVE_TCGETATTR)
#include <termios.h>

void
rl_ttyset(Reset)
    int                         Reset;
{
    static struct termios       old;
    struct termios              new;

    if (Reset == 0) {
        (void)tcgetattr(0, &old);
        rl_erase = old.c_cc[VERASE];
        rl_kill = old.c_cc[VKILL];
        rl_eof = old.c_cc[VEOF];
        rl_intr = old.c_cc[VINTR];
        rl_quit = old.c_cc[VQUIT];

        new = old;
        new.c_lflag &= ~(ECHO | ICANON | ISIG | IEXTEN);
        new.c_cc[VMIN] = 1;
        new.c_cc[VTIME] = 0;
        (void)tcsetattr(0, TCSADRAIN, &new);
    }
```

```
    else
        (void)tcsetattr(0, TCSADRAIN, &old);
}

#else
#if     defined(HAVE_TERMIO)
#include <termio.h>

void
rl_ttyset(Reset)
    int                         Reset;
{
    static struct termio        old;
    struct termio               new;

    if (Reset == 0) {
        (void)ioctl(0, TCGETA, &old);
        rl_erase = old.c_cc[VERASE];
        rl_kill = old.c_cc[VKILL];
        rl_eof = old.c_cc[VEOF];
        rl_intr = old.c_cc[VINTR];
        rl_quit = old.c_cc[VQUIT];

        new = old;
        new.c_cc[VINTR] = -1;
        new.c_cc[VQUIT] = -1;
        new.c_lflag &= ~(ECHO | ICANON);
        new.c_cc[VMIN] = 1;
        new.c_cc[VTIME] = 0;
        (void)ioctl(0, TCSETAW, &new);
    }
    else
        (void)ioctl(0, TCSETAW, &old);
}

#else
#include <sgtty.h>

void
rl_ttyset(Reset)
    int                         Reset;
{
    static struct sgttyb        old_sgttyb;
    static struct tchars        old_tchars;
    struct sgttyb               new_sgttyb;
    struct tchars               new_tchars;

    if (Reset == 0) {
        (void)ioctl(0, TIOCGETP, &old_sgttyb);
        rl_erase = old_sgttyb.sg_erase;
        rl_kill = old_sgttyb.sg_kill;

        (void)ioctl(0, TIOCGETC, &old_tchars);
        rl_eof = old_tchars.t_eofc;
        rl_intr = old_tchars.t_intrc;
        rl_quit = old_tchars.t_quitc;

        new_sgttyb = old_sgttyb;
        new_sgttyb.sg_flags &= ~ECHO;
        new_sgttyb.sg_flags |= RAW;
```

```
        (void)ioctl(0, TIOCSETP, &new_sgttyb);

        new_tchars = old_tchars;
        new_tchars.t_intrc = -1;
        new_tchars.t_quitc = -1;
        (void)ioctl(0, TIOCSETC, &new_tchars);
    }
    else {
        (void)ioctl(0, TIOCSETP, &old_sgttyb);
        (void)ioctl(0, TIOCSETC, &old_tchars);
    }
}
#endif  /* defined(HAVE_TERMIO) */
#endif  /* defined(HAVE_TCGETATTR) */

void
rl_add_slash(path, p)
    char        *path;
    char        *p;
{
    struct stat Sb;

    if (stat(path, &Sb) >= 0)
        (void)strcat(p, S_ISDIR(Sb.st_mode) ? "/" : " ");
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/editline/testit.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*  $Revision: 1.3 $
**
**  A "micro-shell" to test editline library.
**  If given any arguments, commands aren't executed.
*/
#include <stdio.h>
#if     defined(HAVE_STDLIB)
#include <stdlib.h>
#endif  /* defined(HAVE_STDLIB) */

extern char     *readline();
extern void     add_history();

#if     !defined(HAVE_STDLIB)
extern int      chdir();
extern int      free();
extern int      strncmp();
extern int      system();
extern void     exit();
extern char     *getenv();
#endif  /* !defined(HAVE_STDLIB) */


#if     defined(NEED_PERROR)
void
```

```
perror(s)
    char        *s;
{
    extern int  errno;

    (voidf)printf(stderr, "%s: error %d\n", s, errno);
}
#endif  /* defined(NEED_PERROR) */


/* ARGSUSED1 */
int
main(ac, av)
    int         ac;
    char        *av[];
{
    char        *prompt;
    char        *p;
    int         doit;

    doit = ac == 1;
    if ((prompt = getenv("TESTPROMPT")) == NULL)
        prompt = "testit>  ";

    while ((p = readline(prompt)) != NULL) {
        (void)printf("\t\t\t|%s|\n", p);
        if (doit)
            if (strncmp(p, "cd ", 3) == 0) {
                if (chdir(&p[3]) < 0)
                    perror(&p[3]);
            }
            else if (system(p) != 0)
                perror(p);
        add_history(p);
        free(p);
    }
    exit(0);
    /* NOTREACHED */
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/end/edata.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#
.sect .text
.sect .rom
.sect .data
.sect .bss
.define _edata
.sect .data
        .align _EM_WSIZE
_edata:
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/end/em_end.s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#
.sect .text
.sect .rom
.sect .data
.sect .bss
.define endtext,enddata,endbss,__end
.sect .text
        .align _EM_WSIZE
.sect .rom
        .align _EM_WSIZE
.sect .data
        .align _EM_WSIZE
.sect .bss
        .align _EM_WSIZE
.sect .end ! only for declaration of _end, __end and endbss.

        .sect .text
endtext:
        .sect .rom
endrom:
        .sect .data
enddata:
        .sect .end
__end:
endbss:
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/end/end.s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
.sect .text
.sect .rom
.sect .data
.sect .bss
.define _end
.sect .end ! only for declaration of _end, __end and endbss.
_end:
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/end/etext.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#
.sect .text
.sect .rom
.sect .data
.sect .bss
.define _etext
.sect .text
        .align _EM_WSIZE
        .sect .text
_etext:
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/FP_bias.h
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: FP_bias.h,v 1.4 89/07/25 14:16:55 ceriel Exp $ */

/*
        include file for floating point package
*/

                /*      FLOAT FORMAT EXPONENT BIAS      */

#define SGL_BIAS         127    /* excess  128 notation used    */
#define DBL_BIAS        1023    /* excess 1024 notation used    */
#define EXT_BIAS           0    /* 2s-complement notation used  */
                                /* this is possible because the */
                                /* sign is in a seperate word   */

                /*      VARIOUS MAX AND MIN VALUES      */
                /*      1) FOR THE DIFFERENT FORMATS    */

#define SGL_MAX            254  /*      standard definition     */
#define SGL_MIN              1  /*      standard definition     */
#define DBL_MAX           2046  /*      standard definition     */
#define DBL_MIN              1  /*      standard definition     */
#define EXT_MAX          16383  /*      standard minimum        */
#define EXT_MIN         -16382  /*      standard minimum        */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/float/FP_shift.h
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: FP_shift.h,v 1.3 89/07/25 14:17:04 ceriel Exp $ */

/*
        include file for floating point package
*/

# define        CARRYBIT        0x80000000L
# define        NORMBIT         0x80000000L
# define        EXP_STORE       16


                                /* parameters for Single Precision */
#define SGL_EXPSHIFT    7
#define SGL_M1LEFT      8
#define SGL_ZERO        0xffffff80L
#define SGL_EXACT       0xff
#define SGL_RUNPACK     SGL_M1LEFT

#define SGL_ROUNDUP     0x80
#define SGL_CARRYOUT    0x01000000L
#define SGL_MASK        0x007fffffL

                                /* parameters for Double Precision */
                                /* used in extend.c */

#define DBL_EXPSHIFT    4

#define DBL_M1LEFT      11

#define DBL_RPACK       (32-DBL_M1LEFT)
#define DBL_LPACK       DBL_M1LEFT

                                /* used in compact.c */

#define DBL_ZERO        0xfffffd00L

#define DBL_EXACT       0x7ff

#define DBL_RUNPACK     DBL_M1LEFT
#define DBL_LUNPACK     (32-DBL_RUNPACK)

#define DBL_ROUNDUP     0x400
#define DBL_CARRYOUT    0x00200000L
#define DBL_MASK        0x000fffffL
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/float/FP_trap.h
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: FP_trap.h,v 1.2 88/04/07 11:33:06 ceriel Exp $ */

/*
        include file for floating point package
*/

                        /*      EM TRAPS        */

#define EIOVFL  3       /* Integer  Overflow            */
#define EFOVFL  4       /* Floating Overflow            */
#define EFUNFL  5       /* Floating Underflow           */
#define EIDIVZ  6       /* Integer  Divide by 0         */
#define EFDIVZ  7       /* Floating Divide by 0.0       */
#define EIUND   8       /* Integer  Undefined Number    */
#define EFUND   9       /* Floating Undefined Number    */
#define ECONV   10      /* Conversion Error             */
# define trap(x) _fptrp(x)
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/float/FP_types.h
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: FP_types.h,v 1.4 93/01/05 12:03:05 ceriel Exp $ */

/********************************************************/
/*
        Type definitions for C Floating Point Package
        include file for floating point package
*/
/********************************************************/
/*
        THESE STRUCTURES ARE USED TO ADDRESS THE INDIVIDUAL
        PARTS OF THE FLOATING POINT NUMBER REPRESENTATIONS.

        THREE STRUCTURES ARE DEFINED:
                SINGLE: single precision floating format
                DOUBLE: double precision floating format
                EXTEND: double precision extended format
*/
/********************************************************/

#ifndef __FPTYPES
```

```
#define __FPTYPES

typedef struct  {
        unsigned long   h_32;   /* higher 32 bits of 64 */
        unsigned long   l_32;   /* lower  32 bits of 64 */
}       B64;

typedef unsigned long   SINGLE;

typedef struct  {
        unsigned long   d[2];
}       DOUBLE;

typedef struct  {       /* expanded float format        */
        short   sign;
        short   exp;
        B64     mantissa;
#define m1 mantissa.h_32
#define m2 mantissa.l_32
} EXTEND;

struct  fef4_returns {
        int     e;
        SINGLE  f;
};

struct  fef8_returns {
        int     e;
        DOUBLE  f;
};

struct fif4_returns {
        SINGLE ipart;
        SINGLE fpart;
};

struct fif8_returns {
        DOUBLE ipart;
        DOUBLE fpart;
};

#if __STDC__
#define _PROTOTYPE(function, params)    function params
#else
#define _PROTOTYPE(function, params)    function()
#endif
_PROTOTYPE( void add_ext, (EXTEND *e1, EXTEND *e2));
_PROTOTYPE( void mul_ext, (EXTEND *e1, EXTEND *e2));
_PROTOTYPE( void div_ext, (EXTEND *e1, EXTEND *e2));
_PROTOTYPE( void sub_ext, (EXTEND *e1, EXTEND *e2));
_PROTOTYPE( void sft_ext, (EXTEND *e1, EXTEND *e2));
_PROTOTYPE( void nrm_ext, (EXTEND *e1));
_PROTOTYPE( void zrf_ext, (EXTEND *e1));
_PROTOTYPE( void extend, (unsigned long *from, EXTEND *to, int size));
_PROTOTYPE( void compact, (EXTEND *from, unsigned long *to, int size));
_PROTOTYPE( void _fptrp, (int));
_PROTOTYPE( void adf4, (SINGLE s2, SINGLE s1));
_PROTOTYPE( void adf8, (DOUBLE s2, DOUBLE s1));
_PROTOTYPE( void sbf4, (SINGLE s2, SINGLE s1));
_PROTOTYPE( void sbf8, (DOUBLE s2, DOUBLE s1));
```

```
_PROTOTYPE( void dvf4, (SINGLE s2, SINGLE s1));
_PROTOTYPE( void dvf8, (DOUBLE s2, DOUBLE s1));
_PROTOTYPE( void mlf4, (SINGLE s2, SINGLE s1));
_PROTOTYPE( void mlf8, (DOUBLE s2, DOUBLE s1));
_PROTOTYPE( void ngf4, (SINGLE f));
_PROTOTYPE( void ngf8, (DOUBLE f));
_PROTOTYPE( void zrf4, (SINGLE *l));
_PROTOTYPE( void zrf8, (DOUBLE *z));
_PROTOTYPE( void cff4, (DOUBLE src));
_PROTOTYPE( void cff8, (SINGLE src));
_PROTOTYPE( void cif4, (int ss, long src));
_PROTOTYPE( void cif8, (int ss, long src));
_PROTOTYPE( void cuf4, (int ss, long src));
_PROTOTYPE( void cuf8, (int ss, long src));
_PROTOTYPE( long cfu, (int ds, int ss, DOUBLE src));
_PROTOTYPE( long cfi, (int ds, int ss, DOUBLE src));
_PROTOTYPE( int cmf4, (SINGLE s2, SINGLE s1));
_PROTOTYPE( int cmf8, (DOUBLE d1, DOUBLE d2));
_PROTOTYPE( void fef4, (struct fef4_returns *r, SINGLE s1));
_PROTOTYPE( void fef8, (struct fef8_returns *r, DOUBLE s1));
_PROTOTYPE( void fif4, (struct fif4_returns *p, SINGLE x, SINGLE y));
_PROTOTYPE( void fif8, (struct fif8_returns *p, DOUBLE x, DOUBLE y));

_PROTOTYPE( void b64_sft, (B64 *, int));
_PROTOTYPE( void b64_lsft, (B64 *));
_PROTOTYPE( void b64_rsft, (B64 *));
_PROTOTYPE( int b64_add, (B64 *, B64 *));
#endif
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/float/adder.h
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: adder.h,v 1.2 92/02/20 18:20:25 philip Exp $ */

/*
 *      include file for 32 & 64 bit addition
 */

typedef struct  B64 {
        unsigned long   h_32;   /* higher 32 bits of 64 */
        unsigned long   l_32;   /* lower  32 bits of 64 */
}       B64;
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/byte_order.h
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#define CHAR_UNSIGNED   0
#define MSB_AT_LOW_ADDRESS      0
#define MSW_AT_LOW_ADDRESS      0
#define FL_MSB_AT_LOW_ADDRESS   0
#define FL_MSW_AT_LOW_ADDRESS   0
#define FL_MSL_AT_LOW_ADDRESS   0
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/get_put.h
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: get_put.h,v 1.3 89/10/25 17:15:01 ceriel Exp $ */

#include <byte_order.h>

#if CHAR_UNSIGNED
#define Xchar(ch)       (ch)
#else
#define Xchar(ch)       ((ch) & 0377)
#endif

#define BYTES_REVERSED (MSB_AT_LOW_ADDRESS != FL_MSB_AT_LOW_ADDRESS)
#define WORDS_REVERSED (MSW_AT_LOW_ADDRESS != FL_MSW_AT_LOW_ADDRESS)
#define LONGS_REVERSED (FL_MSL_AT_LOW_ADDRESS)

#if BYTES_REVERSED
#define uget2(c)        (Xchar((c)[1]) | ((unsigned) Xchar((c)[0]) << 8))
#define Xput2(i, c)     (((c)[1] = (i)), ((c)[0] = (i) >> 8))
#define put2(i, c)      { register int j = (i); Xput2(j, c); }
#else
#define uget2(c)        (* ((unsigned short *) (c)))
#define Xput2(i, c)     (* ((short *) (c)) = (i))
#define put2(i, c)      Xput2(i, c)
#endif

#define get2(c)         ((short) uget2(c))

#if WORDS_REVERSED || BYTES_REVERSED
#define get4(c)         (uget2((c)+2) | ((long) uget2(c) << 16))
#define put4(l, c)      { register long x=(l); \
                          Xput2((int)x,(c)+2); \
                          Xput2((int)(x>>16),(c)); \
                        }
#else
#define get4(c)         (* ((long *) (c)))
#define put4(l, c)      (* ((long *) (c)) = (l))
```

```
#endif
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/add_ext.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: add_ext.c,v 1.7 93/01/05 12:03:11 ceriel Exp $ */

/*
        ADD TWO EXTENDED FORMAT NUMBERS
*/

#include "FP_types.h"

void
add_ext(e1,e2)
register EXTEND *e1,*e2;
{
        if ((e2->m1 | e2->m2) == 0L) {
                return;
        }
        if ((e1->m1 | e1->m2) == 0L) {
                *e1 = *e2;
                return;
        }
        sft_ext(e1, e2);        /* adjust mantissas to equal powers */
        if (e1->sign != e2->sign) {
                /* e1 + e2 = e1 - (-e2) */
                if (e2->m1 > e1->m1 ||
                    (e2->m1 == e1->m1 && e2->m2 > e1->m2)) {
                        /*      abs(e2) > abs(e1) */
                        EXTEND x;

                        x = *e1;
                        *e1 = *e2;
                        if (x.m2 > e1->m2) {
                                e1->m1 -= 1;    /* carry in */
                        }
                        e1->m1 -= x.m1;
                        e1->m2 -= x.m2;
                }
                else {
                        if (e2->m2 > e1->m2)
                                e1->m1 -= 1;    /* carry in */
                        e1->m1 -= e2->m1;
                        e1->m2 -= e2->m2;
                }
        }
        else {
                if (b64_add(&e1->mantissa,&e2->mantissa)) {     /* addition carry */
                        b64_rsft(&e1->mantissa);        /* shift mantissa one bit RIGHT */
```

```
                                e1->m1 |= 0x80000000L;  /* set max bit  */
                                e1->exp++;              /* increase the exponent */
                        }
                }
                nrm_ext(e1);
        }
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/float/adder.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: adder.c,v 1.6 93/01/05 12:03:17 ceriel Exp $ */

/*
 *      these are the routines the routines to do 32 and  64-bit addition
 */

# ifdef EXT_DEBUG
# include <stdio.h>
# endif

# include "FP_types.h"
# define        UNKNOWN -1
# define        TRUE     1
# define        FALSE    0
# define        MAXBIT  0x80000000L

        /*
         *      add 64 bits
         */
int
b64_add(e1,e2)
                /*
                 * pointers to 64 bit 'registers'
                 */
register        B64     *e1,*e2;
{
                register        int     overflow;
                                int     carry;

                        /* add higher pair of 32 bits */
        overflow = ((unsigned long) 0xFFFFFFFF - e1->h_32 < e2->h_32);
        e1->h_32 += e2->h_32;

                        /* add lower pair of 32 bits */
        carry = ((unsigned long) 0xFFFFFFFF - e1->l_32 < e2->l_32);
        e1->l_32 += e2->l_32;
# ifdef EXT_DEBUG
        printf("\t\t\t\t\tb64_add: overflow (%d); internal carry(%d)\n",
                                        overflow,carry);
        fflush(stdout);
```

```
# endif
        if ((carry) && (++e1->h_32 == 0))
                return(TRUE);           /* had a 64 bit overflow */
        return(overflow);               /* return status from higher add */
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/adf4.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: adf4.c,v 1.7 93/01/05 12:03:23 ceriel Exp $ */

/*
        ADD TWO FLOATS - SINGLE (ADF 4)
*/

#include        "FP_types.h"

void
adf4(s2,s1)
SINGLE  s1,s2;
{
        EXTEND  e1,e2;
        int     swap = 0;

        if (s1 == (SINGLE) 0) {
                s1 = s2;
                return;
        }
        if (s2 == (SINGLE) 0) {
                return;
        }
        extend(&s1,&e1,sizeof(SINGLE));
        extend(&s2,&e2,sizeof(SINGLE));
        add_ext(&e1,&e2);
        compact(&e1,&s1,sizeof(SINGLE));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/adf8.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: adf8.c,v 1.7 93/01/05 12:03:29 ceriel Exp $ */

/*
        ADD TWO FLOATS - DOUBLE (ADF 8)
*/

#include        "FP_types.h"

void
adf8(s2,s1)
DOUBLE  s1,s2;
{
        EXTEND  e1,e2;

        if (s1.d[0] == 0 && s1.d[1] == 0) {
                s1 = s2;
                return;
        }
        if (s2.d[0] == 0 && s2.d[1] == 0) {
                return;
        }

        extend(&s1.d[0],&e1,sizeof(DOUBLE));
        extend(&s2.d[0],&e2,sizeof(DOUBLE));
        add_ext(&e1,&e2);
        compact(&e1,&s1.d[0],sizeof(DOUBLE));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/cff4.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: cff4.c,v 1.5 93/01/05 12:03:36 ceriel Exp $ */

/*
                CONVERT DOUBLE TO SINGLE (CFF 8 4)

        This routine works quite simply. A floating point
        of size 08 is converted to extended format.
        This extended variable is converted back to
        a floating point of size 04.

```

```
*/

#include        "FP_types.h"

void
cff4(src)
DOUBLE  src;    /* the source itself -  THIS TIME it's DOUBLE */
{
        EXTEND  buf;

        extend(&src.d[0],&buf,sizeof(DOUBLE));  /* no matter what */
        compact(&buf,&(src.d[1]),sizeof(SINGLE));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/cff8.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: cff8.c,v 1.5 93/01/05 12:03:41 ceriel Exp $ */

/*
                CONVERT SINGLE TO DOUBLE (CFF 4 8)

        This routine works quite simply. A floating point
        of size 04 is converted to extended format.
        This extended variable is converted back to
        a floating point of size 08.

*/

#include "FP_types.h"

void
cff8(src)
SINGLE  src;
{
        EXTEND  buf;

        extend(&src,&buf,sizeof(SINGLE));       /* no matter what */
        compact(&buf, &src,sizeof(DOUBLE));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/float/cfi.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: cfi.c,v 1.5 93/01/05 12:03:48 ceriel Exp $ */

/*
                CONVERT FLOAT TO SIGNED (CFI m n)

                N.B. The caller must know what it is getting.
                     A LONG is always returned. If it is an
                     integer the high byte is cleared first.
*/

#include "FP_trap.h"
#include "FP_types.h"
#include "FP_shift.h"

long
cfi(ds,ss,src)
int     ds;     /* destination size (2 or 4) */
int     ss;     /* source size      (4 or 8) */
DOUBLE  src;    /* assume worst case */
{
        EXTEND  buf;
        long    new;
        short   max_exp;

        extend(&src.d[0],&buf,ss);      /* get extended format */
        if (buf.exp < 0) {      /* no conversion needed */
                src.d[ss == 8] = 0L;
                return(0L);
        }
        max_exp = (ds << 3) - 2;        /* signed numbers */
                                /* have more limited max_exp */
        if (buf.exp > max_exp) {
                if (buf.exp == max_exp+1 && buf.sign && buf.m1 == NORMBIT &&
                    buf.m2 == 0L) {
                }
                else {
                        trap(EIOVFL);   /* integer overflow     */
                        buf.exp %= max_exp; /* truncate */
                }
        }
        new = buf.m1 >> (31-buf.exp);
        if (buf.sign)
                new = -new;
done:
        src.d[ss == 8] = new;
        return(new);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/cfu.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: cfu.c,v 1.5 93/01/05 12:03:55 ceriel Exp $ */

/*
                CONVERT FLOAT TO UNSIGNED (CFU m n)

                N.B. The caller must know what it is getting.
                     A LONG is always returned. If it is an
                     integer the high byte is cleared first.
*/

#include "FP_trap.h"
#include "FP_types.h"

long
cfu(ds,ss,src)
int     ds;     /* destination size (2 or 4) */
int     ss;     /* source size      (4 or 8) */
DOUBLE  src;    /* assume worst case */
{
        EXTEND  buf;
        long    new;
        short   newint, max_exp;

        extend(&src.d[0],&buf,ss);      /* get extended format  */
        if (buf.exp < 0) {      /* no conversion needed */
                src.d[ss == 8] = 0L;
                return(0L);
        }
        max_exp = (ds << 3) - 1;
        if (buf.exp > max_exp) {
                trap(EIOVFL);   /* integer overflow     */
                buf.exp %= max_exp;
        }
        new = buf.m1 >> (31-buf.exp);
done:
        src.d[ss == 8] = new;
        return(new);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/float/cif4.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: cif4.c,v 1.5 93/01/05 12:04:01 ceriel Exp $ */

/*
        CONVERT INTEGER TO SINGLE (CIF n 4)

        THIS ROUTINE WORKS BY FILLING AN EXTENDED
        WITH THE INTEGER VALUE IN EXTENDED FORMAT
        AND USES COMPACT() TO PUT IT INTO THE PROPER
        FLOATING POINT PRECISION.
*/

#include "FP_types.h"

void
cif4(ss,src)
int     ss;     /* source size */
long    src;    /* largest possible integer to convert */
{
        EXTEND  buf;
        short   *ipt;
        long    i_src;
        SINGLE  *result;

        zrf_ext(&buf);
        if (ss == sizeof(long)) {
                buf.exp = 31;
                i_src = src;
                result = (SINGLE *) &src;
        }
        else    {
                ipt = (short *) &src;
                i_src = (long) *ipt;
                buf.exp = 15;
                result = (SINGLE *) &ss;
        }
        if (i_src == 0) {
                *result = (SINGLE) 0L;
                return;
        }
                        /* ESTABLISHED THAT src != 0    */
                        /* adjust exponent field        */
        buf.sign = (i_src < 0) ? 0x8000 : 0;
                        /* clear sign bit of integer    */
                        /* move to mantissa field       */
        buf.m1 = (i_src < 0) ? -i_src : i_src;
                        /* adjust mantissa field        */
        if (ss != sizeof(long))
                buf.m1 <<= 16;
        nrm_ext(&buf);          /* adjust mantissa field        */
        compact(&buf, result,sizeof(SINGLE));   /* put on stack */
```

```
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/cif8.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: cif8.c,v 1.5 93/01/05 12:04:07 ceriel Exp $ */

/*
        CONVERT INTEGER TO FLOAT (CIF n 8)

        THIS ROUTINE WORKS BY FILLING AN EXTENDED
        WITH THE INTEGER VALUE IN EXTENDED FORMAT
        AND USES COMPACT() TO PUT IT INTO THE PROPER
        FLOATING POINT PRECISION.
*/

#include "FP_types.h"

void
cif8(ss,src)
int     ss;     /* source size */
long    src;    /* largest possible integer to convert */
{
        EXTEND  buf;
        DOUBLE  *result;        /* for return value */
        short   *ipt;
        long    i_src;

        result = (DOUBLE *) ((void *) &ss);     /* always */
        zrf_ext(&buf);
        if (ss == sizeof(long)) {
                buf.exp = 31;
                i_src = src;
        }
        else    {
                ipt = (short *) &src;
                i_src = (long) *ipt;
                buf.exp = 15;
        }
        if (i_src == 0) {
                zrf8(result);
                return;
        }
                        /* ESTABLISHED THAT src != 0    */
                        /* adjust exponent field        */
        buf.sign = (i_src < 0) ? 0x8000 : 0;
                        /* clear sign bit of integer    */
                        /* move to mantissa field       */
        buf.m1 = (i_src < 0) ? -i_src : i_src;
                        /* adjust mantissa field        */
```

```
        if (ss != sizeof(long))
                buf.m1 <<= 16;
        nrm_ext(&buf);
        compact(&buf,&result->d[0],8);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/float/cmf4.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: cmf4.c,v 1.6 93/01/05 12:04:14 ceriel Exp $ */

/*
        COMPARE SINGLES (CMF 4)
*/

#include        "FP_types.h"
#include        "get_put.h"

int
cmf4(f1,f2)
SINGLE  f1,f2;
{
                /*
                 * return ((f1 < f2) ? 1 : (f1 - f2))
                 */
#define SIGN(x) (((x) < 0) ? -1 : 1)
        int     sign1,sign2;
        long    l1,l2;

        l1 = get4((char *) &f1);
        l2 = get4((char *) &f2);

        if (l1 == l2) return 0;

        sign1 = SIGN(l1);
        sign2 = SIGN(l2);
        if (sign1 != sign2) {
                if ((l1 & 0x7fffffff) == 0 &&
                    (l2 & 0x7fffffff) == 0) return 0;
                return ((sign1 > 0) ? -1 : 1);
        }

        return (sign1 * ((l1 < l2) ? 1 : -1));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/cmf8.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: cmf8.c,v 1.9 93/01/05 12:04:22 ceriel Exp $ */

/*
        COMPARE DOUBLES (CMF 8)
*/

#include        "FP_types.h"
#include        "get_put.h"

int
cmf8(d1,d2)
DOUBLE  d1,d2;
{
#define SIGN(x) (((x) < 0) ? -1 : 1)
                /*
                 * return ((d1 < d2) ? 1 : (d1 > d2) ? -1 : 0))
                 */
        long    l1,l2;
        int     sign1,sign2;
        int     rv;

#if FL_MSL_AT_LOW_ADDRESS
        l1 = get4((char *)&d1);
        l2 = get4((char *)&d2);
#else
        l1 = get4(((char *)&d1+4));
        l2 = get4(((char *)&d2+4));
#endif
        sign1 = SIGN(l1);
        sign2 = SIGN(l2);
        if (sign1 != sign2) {
                l1 &= 0x7fffffff;
                l2 &= 0x7fffffff;
                if (l1 != 0 || l2 != 0) {
                        return ((sign1 > 0) ? -1 : 1);
                }
        }
        if (l1 != l2)   {       /* we can decide here */
                rv = l1 < l2 ? 1 : -1;
        }
        else    {               /* decide in 2nd half */
                unsigned long u1, u2;
#if FL_MSL_AT_LOW_ADDRESS
                u1 = get4(((char *)&d1 + 4));
                u2 = get4(((char *)&d2 + 4));
#else
                u1 = get4((char *)&d1);
                u2 = get4((char *)&d2);
#endif
                if (u1 == u2)
```

```
                        return(0);
                if (u1 < u2) rv = 1;
                else rv = -1;
        }
        return sign1 * rv;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/float/compact.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: compact.c,v 1.13 93/01/05 12:04:28 ceriel Exp $ */

/*
        COMPACT EXTEND FORMAT INTO FLOAT OF PROPER SIZE
*/

# include "FP_bias.h"
# include "FP_shift.h"
# include "FP_trap.h"
# include "FP_types.h"
# include "get_put.h"

void
compact(f,to,size)
EXTEND  *f;
unsigned long   *to;
int     size;
{
        int     error = 0;

        if (size == sizeof(DOUBLE)) {
        /*
         * COMPACT EXTENDED INTO DOUBLE
         */
                DOUBLE *DBL = (DOUBLE *) (void *) to;

                if ((f->m1|(f->m2 & DBL_ZERO)) == 0L)   {
                        zrf8(DBL);
                        return;
                }
                f->exp += DBL_BIAS;     /* restore proper bias  */
                if (f->exp > DBL_MAX)   {
dbl_over:                       trap(EFOVFL);
                        f->exp = DBL_MAX+1;
                        f->m1 = 0;
                        f->m2 = 0;
                        if (error++)
                                return;
                }
                else if (f->exp < DBL_MIN)      {
```

```
                        b64_rsft(&(f->mantissa));
                        if (f->exp < 0) {
                                b64_sft(&(f->mantissa), -f->exp);
                                f->exp = 0;
                        }
                        /* underflow ??? */
                }

                /* local CAST conversion                */

                /* because of special format shift only 10 bits */
                /* bit shift mantissa 10 bits           */

                /* first align within words, then do store operation */

                DBL->d[0] = f->m1 >> DBL_RUNPACK;   /* plus 22 == 32 */
                DBL->d[1] = f->m2 >> DBL_RUNPACK;   /* plus 22 == 32 */
                DBL->d[1] |= (f->m1 << DBL_LUNPACK); /* plus 10 == 32 */

                /* if not exact then round to nearest   */
                /* on a tie, round to even */

#ifdef EXCEPTION_INEXACT
                if ((f->m2 & DBL_EXACT) != 0) {
                    INEXACT();
#endif
                    if (((f->m2 & DBL_EXACT) > DBL_ROUNDUP)
                        || ((f->m2 & DBL_EXACT) == DBL_ROUNDUP
                            && (f->m2 & (DBL_ROUNDUP << 1)))) {
                        DBL->d[1]++;    /* rounding up  */
                        if (DBL->d[1] == 0L) { /* carry out     */
                            DBL->d[0]++;

                            if (f->exp == 0 && (DBL->d[0] & ~DBL_MASK)) {
                                        f->exp++;
                                }
                            if (DBL->d[0] & DBL_CARRYOUT) { /* carry out */
                                if (DBL->d[0] & 01)
                                    DBL->d[1] = CARRYBIT;
                                DBL->d[0] >>= 1;
                                f->exp++;
                            }
                        }
                        /*      check for overflow                      */
                        if (f->exp > DBL_MAX)
                                goto dbl_over;
                    }
#ifdef EXCEPTION_INEXACT
                }
#endif

                /*
                 * STORE EXPONENT AND SIGN:
                 *
                 * 1) clear leading bits (B4-B15)
                 * 2) shift and store exponent
                 */

                DBL->d[0] &= DBL_MASK;
                DBL->d[0] |=
```

```
                        ((long) (f->exp << DBL_EXPSHIFT) << EXP_STORE);
                if (f->sign)
                        DBL->d[0] |= CARRYBIT;

                /*
                 * STORE MANTISSA
                 */

#if FL_MSL_AT_LOW_ADDRESS
                put4(DBL->d[0], (char *) &DBL->d[0]);
                put4(DBL->d[1], (char *) &DBL->d[1]);
#else
                { unsigned long l;
                  put4(DBL->d[1], (char *) &l);
                  put4(DBL->d[0], (char *) &DBL->d[1]);
                  DBL->d[0] = l;
                }
#endif
        }
        else {
                /*
                 * COMPACT EXTENDED INTO FLOAT
                 */
                SINGLE  *SGL;

                /* local CAST conversion                */
                SGL = (SINGLE *) (void *) to;
                if ((f->m1 & SGL_ZERO) == 0L)   {
                        *SGL = 0L;
                        return;
                }
                f->exp += SGL_BIAS;     /* restore bias */
                if (f->exp > SGL_MAX)   {
sgl_over:                       trap(EFOVFL);
                        f->exp = SGL_MAX+1;
                        f->m1 = 0L;
                        f->m2 = 0L;
                        if (error++)
                                return;
                }
                else if (f->exp < SGL_MIN)      {
                        b64_rsft(&(f->mantissa));
                        if (f->exp < 0) {
                                b64_sft(&(f->mantissa), -f->exp);
                                f->exp = 0;
                        }
                        /* underflow ??? */
                }

                /* shift mantissa and store     */
                *SGL = (f->m1 >> SGL_RUNPACK);

                /* check for rounding to nearest        */
                /* on a tie, round to even              */
#ifdef EXCEPTION_INEXACT
                if (f->m2 != 0 ||
                    (f->m1 & SGL_EXACT) != 0L) {
                        INEXACT();
#endif
                        if (((f->m1 & SGL_EXACT) > SGL_ROUNDUP)
```

```
                                 || ((f->m1 & SGL_EXACT) == SGL_ROUNDUP
                                    && (f->m1 & (SGL_ROUNDUP << 1)))) {
                                    (*SGL)++;
                                    if (f->exp == 0 && (*SGL & ~SGL_MASK)) {
                                            f->exp++;
                                    }
                        /* check normal */
                                    if (*SGL & SGL_CARRYOUT)        {
                                            *SGL >>= 1;
                                            f->exp++;
                                    }
                                    if (f->exp > SGL_MAX)
                                            goto sgl_over;
                        }
#ifdef EXCEPTION_INEXACT
                }
#endif

                /*
                 * STORE EXPONENT AND SIGN:
                 *
                 * 1) clear leading bit of fraction
                 * 2) shift and store exponent
                 */

                *SGL &= SGL_MASK; /* B23-B31 are 0 */
                *SGL |= ((long) (f->exp << SGL_EXPSHIFT) << EXP_STORE);
                if (f->sign)
                        *SGL |= CARRYBIT;

                /*
                 * STORE MANTISSA
                 */

                put4(*SGL, (char *) &SGL);
        }
}
```

++
src/lib/float/cuf4.c
++

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: cuf4.c,v 1.6 93/01/05 12:04:35 ceriel Exp $ */

/*
        CONVERT INTEGER TO SINGLE (CUF n 4)

        THIS ROUTINE WORKS BY FILLING AN EXTENDED
        WITH THE INTEGER VALUE IN EXTENDED FORMAT
        AND USES COMPACT() TO PUT IT INTO THE PROPER
        FLOATING POINT PRECISION.
*/
```

```
#include "FP_types.h"

void
cuf4(ss,src)
int     ss;     /* source size */
long    src;    /* largest possible integer to convert */
{
        EXTEND  buf;
        short   *ipt;
        SINGLE  *result;
        long    i_src;

        zrf_ext(&buf);
        if (ss == sizeof(long)) {
                buf.exp = 31;
                i_src = src;
                result = (SINGLE *) &src;
        }
        else    {
                ipt = (short *) &src;
                i_src = (long) *ipt;
                buf.exp = 15;
                result = (SINGLE *) ((void *) &ss);
        }
        if (i_src == 0) {
                *result = (SINGLE) 0L;
                return;
        }
                        /* ESTABLISHED THAT src != 0    */

                        /* adjust exponent field        */
        if (ss != sizeof(long))
                i_src <<= 16;

                        /* move to mantissa field       */
        buf.m1 = i_src;

                        /* adjust mantissa field        */
        nrm_ext(&buf);
        compact(&buf,result,4);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/cuf8.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: cuf8.c,v 1.6 93/01/05 12:04:41 ceriel Exp $ */

/*
        CONVERT INTEGER TO FLOAT (CUF n 8)

```

```
        THIS ROUTINE WORKS BY FILLING AN EXTENDED
        WITH THE INTEGER VALUE IN EXTENDED FORMAT
        AND USES COMPACT() TO PUT IT INTO THE PROPER
        FLOATING POINT PRECISION.
*/

#include "FP_types.h"

void
cuf8(ss,src)
int     ss;     /* source size */
long    src;    /* largest possible integer to convert */
{
        EXTEND  buf;
        short   *ipt;
        long    i_src;

        zrf_ext(&buf);
        if (ss == sizeof(long)) {
                buf.exp = 31;
                i_src = src;
        }
        else    {
                ipt = (short *) &src;
                i_src = (long) *ipt;
                buf.exp = 15;
        }
        if (i_src == 0) {
                zrf8((DOUBLE *)((void *)&ss));
                return;
        }
                        /* ESTABLISHED THAT src != 0    */

                        /* adjust exponent field        */
        if (ss != sizeof(long))
                i_src <<= 16;

                        /* move to mantissa field       */
        buf.m1 = i_src;

                        /* adjust mantissa field        */
        nrm_ext(&buf);
        compact(&buf,(unsigned long *) (void *)&ss,8);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/div_ext.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: div_ext.c,v 1.10 93/01/05 12:04:47 ceriel Exp $ */

/*
        DIVIDE EXTENDED FORMAT
*/
```

```

#include "FP_bias.h"
#include "FP_trap.h"
#include "FP_types.h"

/*
        November 15, 1984

        This is a routine to do the work.
        There are two versions:
        One is based on the partial products method
        and makes no use possible machine instructions
        to divide (hardware dividers).
        The other is used when USE_DIVIDE is defined. It is much faster on
        machines with fast 4 byte operations.
*/
/********************************************************/

void
div_ext(e1,e2)
EXTEND  *e1,*e2;
{
        short   error = 0;
        B64             result;
        register        unsigned long   *lp;
#ifndef USE_DIVIDE
        short   count;
#else
        unsigned short u[9], v[5];
        register int j;
        register unsigned short *u_p = u;
        int maxv = 4;
#endif

        if ((e2->m1 | e2->m2) == 0) {
                /*
                 * Exception 8.2 - Divide by zero
                 */
                trap(EFDIVZ);
                e1->m1 = e1->m2 = 0L;
                e1->exp = EXT_MAX;
                return;
        }
        if ((e1->m1 | e1->m2) == 0) {   /* 0 / anything == 0 */
                e1->exp = 0;    /* make sure */
                return;
        }
#ifndef USE_DIVIDE
        /*
         * numbers are right shifted one bit to make sure
         * that m1 is quaranteed to be larger if its
         * maximum bit is set
         */
        b64_rsft(&e1->mantissa);        /* 64 bit shift right */
        b64_rsft(&e2->mantissa);        /* 64 bit shift right */
        e1->exp++;
        e2->exp++;
#endif
        /*      check for underflow, divide by zero, etc        */
        e1->sign ^= e2->sign;
```

```
        e1->exp -= e2->exp;

#ifndef USE_DIVIDE
                /* do division of mantissas     */
                /* uses partial product method  */
                /* init control variables       */

        count = 64;
        result.h_32 = 0L;
        result.l_32 = 0L;

                /* partial product division loop */

        while (count--) {
                /* first left shift result 1 bit        */
                /* this is ALWAYS done                  */

                b64_lsft(&result);

                /* compare dividend and divisor         */
                /* if dividend >= divisor add a bit     */
                /* and subtract divisior from dividend  */

                if ( (e1->m1 < e2->m1) ||
                        ((e1->m1 == e2->m1) && (e1->m2 < e2->m2) ))
                        ;       /* null statement */
                                /* i.e., don't add or subtract */
                else    {
                        result.l_32++;  /* ADD  */
                        if (e2->m2 > e1->m2)
                                e1->m1 -= 1;    /* carry in */
                        e1->m1 -= e2->m1;       /* do SUBTRACTION */
                        e1->m2 -= e2->m2;       /*    SUBTRACTION */
                }

                /*      shift dividend left one bit OR  */
                /*      IF it equals ZERO we can break out      */
                /*      of the loop, but still must shift       */
                /*      the quotient the remaining count bits   */
                /* NB   save the results of this test in error  */
                /*      if not zero, then the result is inexact. */
                /*      this would be reported in IEEE standard */

                /*      lp points to dividend                   */
                lp = &e1->m1;

                error = ((*lp | *(lp+1)) != 0L) ? 1 : 0;
                if (error)      {       /* more work */
                        /*      assume max bit == 0 (see above) */
                        b64_lsft(&e1->mantissa);
                        continue;
                }
                else
                        break;  /* leave loop   */
        }       /* end of divide by subtraction loop    */

        if (count > 0)  {
                lp = &result.h_32;
                if (count > 31) {       /* move to higher word */
                        *lp = *(lp+1);
```

```
                        count -= 32;
                        *(lp+1) = 0L;   /* clear low word       */
                }
                if (*lp)
                        *lp <<= count;  /* shift rest of way    */
                lp++;   /*  == &result.l_32     */
                if (*lp) {
                        result.h_32 |= (*lp >> 32-count);
                        *lp <<= count;
                }
        }
#else /* USE_DIVIDE */

        u[4] = (e1->m2 & 1) << 15;
        b64_rsft(&(e1->mantissa));
        u[0] = e1->m1 >> 16;
        u[1] = e1->m1;
        u[2] = e1->m2 >> 16;
        u[3] = e1->m2;
        u[5] = 0; u[6] = 0; u[7] = 0;
        v[1] = e2->m1 >> 16;
        v[2] = e2->m1;
        v[3] = e2->m2 >> 16;
        v[4] = e2->m2;
        while (! v[maxv]) maxv--;
        result.h_32 = 0;
        result.l_32 = 0;
        lp = &result.h_32;

        /*
         * Use an algorithm of Knuth (The art of programming, Seminumerical
         * algorithms), to divide u by v. u and v are both seen as numbers
         * with base 65536.
         */
        for (j = 0; j <= 3; j++, u_p++) {
                unsigned long q_est, temp;

                if (j == 2) lp++;
                if (u_p[0] == 0 && u_p[1] < v[1]) continue;
                temp = ((unsigned long)u_p[0] << 16) + u_p[1];
                if (u_p[0] >= v[1]) {
                        q_est = 0x0000FFFFL;
                }
                else {
                        q_est = temp / v[1];
                }
                temp -= q_est * v[1];
                while (temp < 0x10000 && v[2]*q_est > ((temp<<16)+u_p[2])) {
                        q_est--;
                        temp += v[1];
                }
                /*      Now, according to Knuth, we have an estimate of the
                        quotient, that is either correct or one too big, but
                        almost always correct.
                */
                if (q_est != 0)  {
                        int i;
                        unsigned long k = 0;
                        int borrow = 0;

```

```
				for (i = maxv; i > 0; i--) {
					unsigned long tmp = q_est * v[i] + k + borrow;
					unsigned short md = tmp;

					borrow = (md > u_p[i]);
					u_p[i] -= md;
					k = tmp >> 16;
				}
				k += borrow;
				borrow = u_p[0] < k;
				u_p[0] -= k;

				if (borrow) {
					/* So, this does not happen often; the estimate
					   was one too big; correct this
					*/
					*lp |= (j & 1) ? (q_est - 1) : ((q_est-1)<<16);
					borrow = 0;
					for (i = maxv; i > 0; i--) {
						unsigned long tmp
						    = v[i]+(unsigned long)u_p[i]+borrow;

						u_p[i] = tmp;
						borrow = tmp >> 16;
					}
					u_p[0] += borrow;
				}
				else *lp |= (j & 1) ? q_est : (q_est<<16);
			}
		}
#ifdef	EXCEPTION_INEXACT
		u_p = &u[0];
		for (j = 7; j >= 0; j--) {
			if (*u_p++) {
				error = 1;
				break;
			}
		}
#endif
#endif

#ifdef	EXCEPTION_INEXACT
		if (error)	{
			/*
			 * report here exception 8.5 - Inexact
			 * from Draft 8.0 of IEEE P754:
			 * In the absence of an invalid operation exception,
			 * if the rounded result of an operation is not exact or if
			 * it overflows without a trap, then the inexact exception
			 * shall be assigned. The rounded or overflowed result
			 * shall be delivered to the destination.
			 */
			INEXACT();
#endif
		e1->mantissa = result;

		nrm_ext(e1);
		if (e1->exp < EXT_MIN)	{
			/*
			 * Exception 8.4 - Underflow
```

```
                         */
                        trap(EFUNFL);   /* underflow */
                        e1->exp = EXT_MIN;
                        e1->m1 = e1->m2 = 0L;
                        return;
                }
                if (e1->exp >= EXT_MAX) {
                        /*
                         * Exception 8.3 - Overflow
                         */
                        trap(EFOVFL);   /* overflow */
                        e1->exp = EXT_MAX;
                        e1->m1 = e1->m2 = 0L;
                        return;
                }
        }
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                src/lib/float/dvf4.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
        /*
          (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
          See the copyright notice in the ACK home directory, in the file "Copyright".
        */

        /* $Header: dvf4.c,v 1.5 93/01/05 12:04:53 ceriel Exp $ */

        /*
                DIVIDE TWO SINGLES - SINGLE Precision (dvf 4)
        */

        #include        "FP_types.h"

        void
        dvf4(s2,s1)
        SINGLE  s1,s2;
        {
                EXTEND  e1,e2;

                extend(&s1,&e1,sizeof(SINGLE));
                extend(&s2,&e2,sizeof(SINGLE));

                        /* do a divide */
                div_ext(&e1,&e2);
                compact(&e1,&s1,sizeof(SINGLE));
        }
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/dvf8.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: dvf8.c,v 1.5 93/01/05 12:04:59 ceriel Exp $ */

/*
        DIVIDE TWO FLOATS - DOUBLE Precision (DVF 8)
*/

#include        "FP_types.h"

void
dvf8(s2,s1)
DOUBLE  s1,s2;
{
        EXTEND  e1,e2;

        extend(&s1.d[0],&e1,sizeof(DOUBLE));
        extend(&s2.d[0],&e2,sizeof(DOUBLE));

                /* do a divide */
        div_ext(&e1,&e2);
        compact(&e1,&s1.d[0],sizeof(DOUBLE));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/extend.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: extend.c,v 1.11 93/01/05 12:05:05 ceriel Exp $ */

/*
        CONVERTS FLOATING POINT TO EXTENDED FORMAT

        Two sizes of FLOATING Point are known:
                SINGLE and DOUBLE
*/
/********************************************************/
/*
        It is not required to normalize in extended
        format, but it has been chosen to do so.
        Extended Format is as follows (at exit):

->sign  S000 0000 | 0000 0000           <SIGN>
```

```
->exp   0EEE EEEE | EEEE EEEE           <EXPONENT>
->m1    LFFF FFFF | FFFF FFFF           <L.Fraction>
        FFFF FFFF | FFFF FFFF           <Fraction>
->m2    FFFF FFFF | FFFF FFFF           <Fraction>
        FFFF F000 | 0000 0000           <Fraction>
*/
/********************************************************/

#include "FP_bias.h"
#include "FP_shift.h"
#include "FP_types.h"
#include "get_put.h"
/********************************************************/

void
extend(from,to,size)
unsigned long   *from;
EXTEND  *to;
int     size;
{
        register char *cpt1;
        unsigned long   tmp;
        int     leadbit = 0;

        cpt1 = (char *) from;

#if FL_MSL_AT_LOW_ADDRESS
#if FL_MSW_AT_LOW_ADDRESS
        to->exp = uget2(cpt1);
#else
        to->exp = uget2(cpt1+2);
#endif
#else
#if FL_MSW_AT_LOW_ADDRESS
        to->exp = uget2(cpt1+(size == sizeof(DOUBLE) ? 4 : 0));
#else
        to->exp = uget2(cpt1+(size == sizeof(DOUBLE) ? 6 : 2));
#endif
#endif
        to->sign = (to->exp & 0x8000);  /* set sign bit */
        to->exp ^= to->sign;
        if (size == sizeof(DOUBLE))
                to->exp >>= DBL_EXPSHIFT;
        else
                to->exp >>= SGL_EXPSHIFT;
        if (to->exp > 0)
                leadbit++;      /* will set Lead bit later      */
        else to->exp++;

        if (size == sizeof(DOUBLE))     {
#if FL_MSL_AT_LOW_ADDRESS
                to->m1 = get4(cpt1);
                cpt1 += 4;
                tmp = get4(cpt1);
#else
                tmp = get4(cpt1);
                cpt1 += 4;
                to->m1 = get4(cpt1);
#endif
                if (to->exp == 1 && to->m1 == 0 && tmp == 0) {
```

```
                        to->exp = 0;
                        to->sign = 0;
                        to->m1 = 0;
                        to->m2 = 0;
                        return;
                }
                to->m1 <<= DBL_M1LEFT;          /* shift        */
                to->exp -= DBL_BIAS;            /* remove bias  */
                to->m1 |= (tmp>>DBL_RPACK);     /* plus 10 == 32        */
                to->m2 = (tmp<<DBL_LPACK);      /* plus 22 == 32        */
        }
        else    {       /* size == sizeof(SINGLE)               */
                to->m1 = get4(cpt1);
                to->m1  <<= SGL_M1LEFT; /* shift        */
                if (to->exp == 1 && to->m1 == 0) {
                        to->exp = 0;
                        to->sign = 0;
                        to->m1 = 0;
                        to->m2 = 0;
                        return;
                }
                to->exp -= SGL_BIAS;            /* remove bias  */
                to->m2 = 0L;
        }

        to->m1 |= NORMBIT;                              /* set bit L    */
        if (leadbit == 0) {             /* set or clear Leading Bit     */
                to->m1 &= ~NORMBIT;                     /* clear bit L  */
                nrm_ext(to);                            /* and normalize */
        }
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/fef4.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: fef4.c,v 1.7 93/01/05 12:05:12 ceriel Exp $ */

/*
        SEPERATE INTO EXPONENT AND FRACTION (FEF 4)
*/

#include        "FP_types.h"

void
fef4(r,s1)
SINGLE  s1;
struct fef4_returns     *r;
{
        EXTEND  buf;
        register struct fef4_returns    *p = r; /* make copy; r might refer
```

```
                                                  to itself (see table)
                                               */

        extend(&s1,&buf,sizeof(SINGLE));
        if (buf.exp == 0 && buf.m1 == 0 && buf.m2 == 0) {
                p->e = 0;
        }
        else {
                p->e = buf.exp+1;
                buf.exp = -1;
        }
        compact(&buf,&p->f,sizeof(SINGLE));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/fef8.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: fef8.c,v 1.7 93/01/05 12:05:18 ceriel Exp $ */

/*
        SEPERATE DOUBLE INTO EXPONENT AND FRACTION (FEF 8)
*/

#include        "FP_types.h"

void
fef8(r, s1)
DOUBLE  s1;
struct fef8_returns *r;
{
        EXTEND  buf;
        register struct fef8_returns *p = r;    /* make copy, r might refer
                                                  to itself (see table)
                                               */

        extend(&s1.d[0],&buf,sizeof(DOUBLE));
        if (buf.exp == 0 && buf.m1 == 0 && buf.m2 == 0) {
                p->e = 0;
        }
        else {
                p->e = buf.exp + 1;
                buf.exp = -1;
        }
        compact(&buf,&p->f.d[0],sizeof(DOUBLE));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/float/fif4.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: fif4.c,v 1.7 93/01/05 12:05:24 ceriel Exp $ */

/*
        MULTIPLY AND DISMEMBER PARTS (FIF 4)
*/

#include "FP_types.h"
#include "FP_shift.h"

void
fif4(p,x,y)
SINGLE  x,y;
struct fif4_returns *p;
{

        EXTEND  e1,e2;

        extend(&y,&e1,sizeof(SINGLE));
        extend(&x,&e2,sizeof(SINGLE));
                /* do a multiply */
        mul_ext(&e1,&e2);
        e2 = e1;
        compact(&e2,&y,sizeof(SINGLE));
        if (e1.exp < 0) {
                p->ipart = 0;
                p->fpart = y;
                return;
        }
        if (e1.exp > 30 - SGL_M1LEFT) {
                p->ipart = y;
                p->fpart = 0;
                return;
        }
        b64_sft(&e1.mantissa, 63 - e1.exp);
        b64_sft(&e1.mantissa, e1.exp - 63);     /* "loose" low order bits */
        compact(&e1,&(p->ipart),sizeof(SINGLE));
        extend(&(p->ipart), &e2, sizeof(SINGLE));
        extend(&y, &e1, sizeof(SINGLE));
        sub_ext(&e1, &e2);
        compact(&e1, &(p->fpart), sizeof(SINGLE));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/fif8.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: fif8.c,v 1.7 93/01/05 12:05:30 ceriel Exp $ */

/*
        MULTIPLY AND DISMEMBER PARTS (FIF 8)
*/

#include "FP_types.h"
#include "FP_shift.h"

void
fif8(p,x,y)
DOUBLE  x,y;
struct fif8_returns *p;
{

        EXTEND  e1,e2;

        extend(&y.d[0],&e1,sizeof(DOUBLE));
        extend(&x.d[0],&e2,sizeof(DOUBLE));
                /* do a multiply */
        mul_ext(&e1,&e2);
        e2 = e1;
        compact(&e2, &y.d[0], sizeof(DOUBLE));
        if (e1.exp < 0) {
                p->ipart.d[0] = 0;
                p->ipart.d[1] = 0;
                p->fpart = y;
                return;
        }
        if (e1.exp > 62 - DBL_M1LEFT) {
                p->ipart = y;
                p->fpart.d[0] = 0;
                p->fpart.d[1] = 0;
                return;
        }
        b64_sft(&e1.mantissa, 63 - e1.exp);
        b64_sft(&e1.mantissa, e1.exp - 63);     /* "loose" low order bits */
        compact(&e1, &(p->ipart.d[0]), sizeof(DOUBLE));
        extend(&(p->ipart.d[0]), &e2, sizeof(DOUBLE));
        extend(&y.d[0], &e1, sizeof(DOUBLE));
        sub_ext(&e1, &e2);
        compact(&e1, &(p->fpart.d[0]), sizeof(DOUBLE));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/float/mlf4.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: mlf4.c,v 1.4 93/01/05 12:05:37 ceriel Exp $ */

/*
 * Multiply Single Precesion Float (MLF 4)
 */

#include        "FP_types.h"

void
mlf4(s2,s1)
SINGLE  s1,s2;
{
        EXTEND  e1,e2;

        extend(&s1,&e1,sizeof(SINGLE));
        extend(&s2,&e2,sizeof(SINGLE));
                /* do a multiply */
        mul_ext(&e1,&e2);
        compact(&e1,&s1,sizeof(SINGLE));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/float/mlf8.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: mlf8.c,v 1.4 93/01/05 12:05:44 ceriel Exp $ */

/*
 * Multiply Double Precision Float (MLF 8)
 */

#include        "FP_types.h"

void
mlf8(s2,s1)
DOUBLE  s1,s2;
{
        EXTEND  e1,e2;

        extend(&s1.d[0],&e1,sizeof(DOUBLE));
```

```
        extend(&s2.d[0],&e2,sizeof(DOUBLE));
                /* do a multiply */
        mul_ext(&e1,&e2);
        compact(&e1,&s1.d[0],sizeof(DOUBLE));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/float/mul_ext.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: mul_ext.c,v 1.6 93/01/05 12:05:51 ceriel Exp $ */

/*
        ROUTINE TO MULTIPLY TWO EXTENDED FORMAT NUMBERS
*/

# include "FP_bias.h"
# include "FP_trap.h"
# include "FP_types.h"
# include "FP_shift.h"

void
mul_ext(e1,e2)
EXTEND  *e1,*e2;
{
        register int    i,j;            /* loop control */
        unsigned short  mp[4];          /* multiplier */
        unsigned short  mc[4];          /* multipcand */
        unsigned short  result[8];      /* result */
        register unsigned short *pres;

        /* first save the sign (XOR)                    */
        e1->sign ^= e2->sign;

        /* compute new exponent */
        e1->exp += e2->exp + 1;
        /* 128 bit multiply of mantissas                        */

                /* assign unknown long formats          */
                /* to known unsigned word formats       */
        mp[0] = e1->m1 >> 16;
        mp[1] = (unsigned short) e1->m1;
        mp[2] = e1->m2 >> 16;
        mp[3] = (unsigned short) e1->m2;
        mc[0] = e2->m1 >> 16;
        mc[1] = (unsigned short) e2->m1;
        mc[2] = e2->m2 >> 16;
        mc[3] = (unsigned short) e2->m2;
        for (i = 8; i--;) {
                result[i] = 0;
```

```
	}
	/*
	 *	fill registers with their components
	 */
	for(i=4, pres = &result[4];i--;pres--) if (mp[i]) {
		unsigned short k = 0;
		unsigned long mpi = mp[i];
		for(j=4;j--;) {
			unsigned long tmp = (unsigned long)pres[j] + k;
			if (mc[j]) tmp += mpi * mc[j];
			pres[j] = tmp;
			k = tmp >> 16;
		}
		pres[-1] = k;
	}
	if (! (result[0] & 0x8000)) {
		e1->exp--;
		for (i = 0; i <= 3; i++) {
			result[i] <<= 1;
			if (result[i+1]&0x8000) result[i] |= 1;
		}
		result[4] <<= 1;
	}

	/*
	 *	combine the registers to a total
	 */
	e1->m1 = ((unsigned long)(result[0]) << 16) + result[1];
	e1->m2 = ((unsigned long)(result[2]) << 16) + result[3];
	if (result[4] & 0x8000) {
		if (++e1->m2 == 0)
			if (++e1->m1 == 0) {
				e1->m1 = NORMBIT;
				e1->exp++;
			}
	}

					/* check for overflow	*/
	if (e1->exp >= EXT_MAX) {
		trap(EFOVFL);
			/* if caught			*/
			/* return signed infinity	*/
		e1->exp = EXT_MAX;
infinity:	e1->m1 = e1->m2 =0L;
		return;
	}
				/* check for underflow	*/
	if (e1->exp < EXT_MIN)	{
		trap(EFUNFL);
		e1->exp = EXT_MIN;
		goto infinity;
	}
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/ngf4.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: ngf4.c,v 1.7 93/01/05 12:05:57 ceriel Exp $ */

/*
                NEGATE A FLOATING POINT (NGF 4)
*/
/********************************************************/

#include "FP_types.h"
#include "get_put.h"

#define OFF ((FL_MSW_AT_LOW_ADDRESS ? 0 : 2) + (FL_MSB_AT_LOW_ADDRESS ? 0 : 1))
void
ngf4(f)
SINGLE  f;
{
        unsigned char *p;

        if (f != (SINGLE) 0) {
                p = (unsigned char *) &f + OFF;
                *p ^= 0x80;
        }
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/ngf8.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: ngf8.c,v 1.8 93/01/05 12:06:05 ceriel Exp $ */

/*
                NEGATE A FLOATING POINT (NGF 8)
*/
/********************************************************/

#include "FP_types.h"
#include "get_put.h"

#define OFF ((FL_MSL_AT_LOW_ADDRESS ? 0 : 4) + (FL_MSW_AT_LOW_ADDRESS ? 0 : 2) + (FL_MSB_AT_LOW_ADDRESS ? 0 : 1))

void
ngf8(f)
DOUBLE  f;
```

```
{
        unsigned char   *p;

        if (f.d[0] != 0 || f.d[1] != 0) {
                p = (unsigned char *) &f + OFF;
                *p ^= 0x80;
        }
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/nrm_ext.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: nrm_ext.c,v 1.5 93/01/05 12:06:11 ceriel Exp $ */

/********************************************************/
/*
        NORMALIZE an EXTENDED FORMAT NUMBER
*/
/********************************************************/

#include "FP_shift.h"
#include "FP_types.h"

void
nrm_ext(e1)
EXTEND  *e1;
{
                /* we assume that the mantissa != 0     */
                /* if it is then just return            */
                /* to let it be a problem elsewhere     */
                /* THAT IS, The exponent is not set to  */
                /* zero. If we don't test here an       */
                /* infinite loop is generated when      */
                /* mantissa is zero                     */

        if ((e1->m1 | e1->m2) == 0L)
                return;

                /* if top word is zero mov low word     */
                /* to top word, adjust exponent value   */
        if (e1->m1 == 0L)       {
                e1->m1 = e1->m2;
                e1->m2 = 0L;
                e1->exp -= 32;
        }
        if ((e1->m1 & NORMBIT) == 0) {
                unsigned long l = ((unsigned long)NORMBIT >> 1);
                int cnt = -1;

                while (! (l & e1->m1)) {
                        l >>= 1;
                        cnt--;
```

```
                        }
                        e1->exp += cnt;
                        b64_sft(&(e1->mantissa), cnt);
                }
        }
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/float/sbf4.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: sbf4.c,v 1.8 93/01/05 12:06:16 ceriel Exp $ */

/*
        SUBTRACT TWO FLOATS - SINGLE Precision (SBF 4)
*/

#include        "FP_types.h"

void
sbf4(s2,s1)
SINGLE  s1,s2;
{
        EXTEND e1,e2;

        if (s2 == (SINGLE) 0) {
                return;
        }
        extend(&s1,&e1,sizeof(SINGLE));
        extend(&s2,&e2,sizeof(SINGLE));
        sub_ext(&e1,&e2);
        compact(&e1,&s1,sizeof(SINGLE));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/float/sbf8.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: sbf8.c,v 1.8 93/01/05 12:06:22 ceriel Exp $ */

/*
        SUBTRACT TWO FLOATS - DOUBLE Precision (SBF 8)
*/
```

```
#include        "FP_types.h"

void
sbf8(s2,s1)
DOUBLE  s1,s2;
{
        EXTEND e1, e2;

        if (s2.d[0] == 0 && s2.d[1] == 0) {
                return;
        }
        extend(&s1.d[0],&e1,sizeof(DOUBLE));
        extend(&s2.d[0],&e2,sizeof(DOUBLE));
        sub_ext(&e1,&e2);
        compact(&e1,&s1.d[0],sizeof(DOUBLE));
}
```

++
src/lib/float/sft_ext.c
++

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: sft_ext.c,v 1.5 93/01/05 12:06:28 ceriel Exp $ */

/*
        SHIFT TWO EXTENDED NUMBERS INTO PROPER
        ALIGNMENT FOR ADDITION (exponents are equal)
        Numbers should not be zero on entry.
*/

#include "FP_types.h"

void
sft_ext(e1,e2)
EXTEND  *e1,*e2;
{
        register        EXTEND  *s;
        register        int     diff;

        diff = e1->exp - e2->exp;

        if (!diff)
                return; /* exponents are equal  */

        if (diff < 0)   { /* e2 is larger       */
                        /* shift e1             */
                diff = -diff;
                s = e1;
        }
        else            /* e1 is larger         */
                        /* shift e2             */
                s = e2;
```

```

        s->exp += diff;
        b64_sft(&(s->mantissa), diff);
}
```

++
src/lib/float/shifter.c
++

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: shifter.c,v 1.6 93/01/05 12:06:34 ceriel Exp $ */

# include "FP_types.h"

void
b64_sft(e1,n)
B64     *e1;
int     n;
{
        if (n > 0) {
                if (n > 63) {
                        e1->l_32 = 0;
                        e1->h_32 = 0;
                        return;
                }
                if (n >= 32) {
                        e1->l_32 = e1->h_32;
                        e1->h_32 = 0;
                        n -= 32;
                }
                if (n > 0) {
                        e1->l_32 >>= n;
                        if (e1->h_32 != 0) {
                                e1->l_32 |= (e1->h_32 << (32 - n));
                                e1->h_32 >>= n;
                        }
                }
                return;
        }
        n = -n;
        if (n > 0) {
                if (n > 63) {
                        e1->l_32 = 0;
                        e1->h_32 = 0;
                        return;
                }
                if (n >= 32) {
                        e1->h_32 = e1->l_32;
                        e1->l_32 = 0;
                        n -= 32;
                }
                if (n > 0) {
                        e1->h_32 <<= n;
                        if (e1->l_32 != 0) {
                                e1->h_32 |= (e1->l_32 >> (32 - n));
```

```
                                e1->l_32 <<= n;
                        }
                }
        }
}

void
b64_lsft(e1)
B64     *e1;
{
        /*      shift left 1 bit */
        e1->h_32 <<= 1;
        if (e1->l_32 & 0x80000000L) e1->h_32 |= 1;
        e1->l_32 <<= 1;
}

void
b64_rsft(e1)
B64     *e1;
{
        /*      shift right 1 bit */
        e1->l_32 >>= 1;
        if (e1->h_32 & 1) e1->l_32 |= 0x80000000L;
        e1->h_32 >>= 1;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/sub_ext.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: sub_ext.c,v 1.6 93/01/05 12:06:40 ceriel Exp $ */

/*
        SUBTRACT 2 EXTENDED FORMAT NUMBERS
*/

#include "FP_types.h"

void
sub_ext(e1,e2)
EXTEND  *e1,*e2;
{
        if ((e2->m1 | e2->m2) == 0L) {
                return;
        }
        if ((e1->m1 | e1->m2) == 0L) {
                *e1 = *e2;
                e1->sign = e2->sign ? 0 : 1;
                return;
        }
```

```
        sft_ext(e1, e2);
        if (e1->sign != e2->sign) {
                /* e1 - e2 = e1 + (-e2) */
                if (b64_add(&e1->mantissa,&e2->mantissa)) { /* addition carry */
                        b64_rsft(&e1->mantissa);      /* shift mantissa one bit RIGHT */
                        e1->m1 |= 0x80000000L;  /* set max bit  */
                        e1->exp++;              /* increase the exponent */
                }
        }
        else if (e2->m1 > e1->m1 ||
                 (e2->m1 == e1->m1 && e2->m2 > e1->m2)) {
                /*      abs(e2) > abs(e1) */
                if (e1->m2 > e2->m2) {
                        e2->m1 -= 1;    /* carry in */
                }
                e2->m1 -= e1->m1;
                e2->m2 -= e1->m2;
                *e1 = *e2;
                e1->sign = e2->sign ? 0 : 1;
        }
        else {
                if (e2->m2 > e1->m2)
                        e1->m1 -= 1;    /* carry in */
                e1->m1 -= e2->m1;
                e1->m2 -= e2->m2;
        }
        nrm_ext(e1);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/zrf4.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: zrf4.c,v 1.5 93/01/05 12:06:46 ceriel Exp $ */

/*
        return a zero float (ZRF 4)
*/

#include "FP_types.h"

void
zrf4(l)
SINGLE  *l;
{
        *l = 0L;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/zrf8.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: zrf8.c,v 1.4 93/01/05 12:06:52 ceriel Exp $ */

/*
        return a zero double (ZRF 8)
*/

#include "FP_types.h"

void
zrf8(z)
DOUBLE  *z;
{

        z->d[0] = 0L;
        z->d[1] = 0L;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/float/zrf_ext.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
  (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
  See the copyright notice in the ACK home directory, in the file "Copyright".
*/

/* $Header: zrf_ext.c,v 1.5 93/01/05 12:06:58 ceriel Exp $ */

/*
        ZERO and return EXTEND FORMAT FLOAT
*/

#include "FP_types.h"

void
zrf_ext(e)
EXTEND  *e;
{
        e->m1 = 0;
        e->m2 = 0;
        e->exp = 0;
        e->sign = 0;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/float/fptrp.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#
.sect .text; .sect .rom; .sect .data; .sect .bss
.define __fptrp
.sect .text
__fptrp:
#if __i386
        push    ebp
        mov     ebp, esp
        mov     eax, 8(bp)
        call    .Xtrp
        leave
        ret
#else /* i86 */
        push    bp
        mov     bp, sp
        mov     ax, 4(bp)
        call    .Xtrp
        jmp     .cret
#endif
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/fphook/fltpr.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include        <stdio.h>
#include        <stdlib.h>
#include        "../stdio/loc_incl.h"

int _fp_hook = 1;

char *
_f_print(va_list *ap, int flags, char *s, char c, int precision)
{
        fprintf(stderr,"cannot print floating point\n");
        exit(EXIT_FAILURE);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/fphook/fphook.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * fltpr.c - print floating point numbers
 */
/* $Header: fltpr.c,v 1.4 90/02/27 16:47:40 eck Exp $ */

#ifndef NOFLOAT
#include        <string.h>
#include        <stdarg.h>
#include        "../stdio/loc_incl.h"
int _fp_hook = 1;
```

```

static char *
_pfloat(long double r, register char *s, int n, int flags)
{
        register char *s1;
        int sign, dp;
        register int i;

        s1 = _fcvt(r, n, &dp, &sign);
        if (sign)
                *s++ = '-';
        else if (flags & FL_SIGN)
                *s++ = '+';
        else if (flags & FL_SPACE)
                *s++ = ' ';

        if (dp<=0)
                *s++ = '0';
        for (i=dp; i>0; i--)
                if (*s1) *s++ = *s1++;
                else *s++ = '0';
        if (((i=n) > 0) || (flags & FL_ALT))
                *s++ = '.';
        while (++dp <= 0) {
                if (--i<0)
                        break;
                *s++ = '0';
        }
        while (--i >= 0)
                if (*s1) *s++ = *s1++;
                else *s++ = '0';
        return s;
}

static char *
_pscien(long double r, register char *s, int n, int flags)
{
        int sign, dp;
        register char *s1;

        s1 = _ecvt(r, n + 1, &dp, &sign);
        if (sign)
                *s++ = '-';
        else if (flags & FL_SIGN)
                *s++ = '+';
        else if (flags & FL_SPACE)
                *s++ = ' ';

        *s++ = *s1++;
        if ((n > 0) || (flags & FL_ALT))
                *s++ = '.';
        while (--n >= 0)
                if (*s1) *s++ = *s1++;
                else *s++ = '0';
        *s++ = 'e';
        if ( r != 0 ) --dp ;
        if ( dp<0 ) {
                *s++ = '-' ; dp= -dp ;
        } else {
                *s++ = '+' ;
```

```
        }
        if (dp >= 100) {
                *s++ = '0' + (dp / 100);
                dp %= 100;
        }
        *s++ = '0' + (dp/10);
        *s++ = '0' + (dp%10);
        return s;
}

#define NDIGINEXP(exp)          (((exp) >= 100 || (exp) <= -100) ? 3 : 2)
#define LOW_EXP                 -4
#define USE_EXP(exp, ndigits)   (((exp) < LOW_EXP + 1) || (exp >= ndigits + 1))

static char *
_gcvt(long double value, int ndigit, char *s, int flags)
{
        int sign, dp;
        register char *s1, *s2;
        register int i;
        register int nndigit = ndigit;

        s1 = _ecvt(value, ndigit, &dp, &sign);
        s2 = s;
        if (sign) *s2++ = '-';
        else if (flags & FL_SIGN)
                *s2++ = '+';
        else if (flags & FL_SPACE)
                *s2++ = ' ';

        if (!(flags & FL_ALT))
                for (i = nndigit - 1; i > 0 && s1[i] == '0'; i--)
                        nndigit--;

        if (USE_EXP(dp,ndigit)) {
                /* Use E format */
                dp--;
                *s2++ = *s1++;
                if ((nndigit > 1) || (flags & FL_ALT)) *s2++ = '.';
                while (--nndigit > 0) *s2++ = *s1++;
                *s2++ = 'e';
                if (dp < 0) {
                        *s2++ = '-';
                        dp = -dp;
                }
                else     *s2++ = '+';
                s2 += NDIGINEXP(dp);
                *s2 = 0;
                for (i = NDIGINEXP(dp); i > 0; i--) {
                        *--s2 = dp % 10 + '0';
                        dp /= 10;
                }
                return s;
        }
        /* Use f format */
        if (dp <= 0) {
                if (*s1 != '0') {
                        /* otherwise the whole number is 0 */
                        *s2++ = '0';
                        *s2++ = '.';
```

```
                }
                while (dp < 0) {
                        dp++;
                        *s2++ = '0';
                }
        }
        for (i = 1; i <= nndigit; i++) {
                *s2++ = *s1++;
                if (i == dp) *s2++ = '.';
        }
        if (i <= dp) {
                while (i++ <= dp) *s2++ = '0';
                *s2++ = '.';
        }
        if ((s2[-1]=='.') && !(flags & FL_ALT)) s2--;
        *s2 = '\0';
        return s;
}

char *
_f_print(va_list *ap, int flags, char *s, char c, int precision)
{
        register char *old_s = s;
        long double ld_val;

        if (flags & FL_LONGDOUBLE) ld_val = va_arg(*ap, long double);
        else ld_val = (long double) va_arg(*ap, double);

        switch(c) {
        case 'f':
                s = _pfloat(ld_val, s, precision, flags);
                break;
        case 'e':
        case 'E':
                s = _pscien(ld_val, s, precision , flags);
                break;
        case 'g':
        case 'G':
                s = _gcvt(ld_val, precision, s, flags);
                s += strlen(s);
                break;
        }
        if ( c == 'E' || c == 'G') {
                while (*old_s && *old_s != 'e') old_s++;
                if (*old_s == 'e') *old_s = 'E';
        }
        return s;
}
#endif  /* NOFLOAT */
/* $Header: strtod.c,v 1.3 90/09/07 11:00:24 eck Exp $ */

#include <stdlib.h>
#include "../ansi/ext_fmt.h"

void _str_ext_cvt(const char *s, char **ss, struct EXTEND *e);
double _ext_dbl_cvt(struct EXTEND *e);

double
strtod(const char *p, char **pp)
{
```

```
        struct EXTEND e;

        _str_ext_cvt(p, pp, &e);
        return _ext_dbl_cvt(&e);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/fphook/strtod.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include        <stdio.h>
#include        <stdlib.h>

double
strtod(const char *p, char **pp)
{
        fprintf(stderr,"cannot print floating point\n");
        exit(EXIT_FAILURE);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ip/ether.h
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/* $Id: ether.h,v 2.2 89/10/23 15:42:19 dupuy Exp $ */

/* Interface definitions for ethernet access library */

typedef union etheraddr
{
    unsigned char bytes[6];             /* byteorder safe initialization */
    unsigned short shorts[3];           /* force 2-byte alignment */
}
          ether_addr;

typedef struct etherpacket
{
    ether_addr dest;
    ether_addr src;
    unsigned char type[2];              /* in network byte order! */
    unsigned short pktlen;              /* length of pktbuf ONLY */
    char *pktbuf;
}
            ether_packet;

typedef struct ethervec
{
    ether_addr dest;
    ether_addr src;
    unsigned char type[2];              /* in network byte order! */
    unsigned short iovcnt;              /* number of iovec to use */
    struct iovec *iov;                  /* ptr to array of iovec */
}
         ether_vec;
```

```
#ifndef __ETHER_BCAST_ADDR__
extern ether_addr ether_bcast_addr;
#endif

#ifdef __STDC__

int ether_open (char *name, unsigned type, ether_addr * address);

ether_addr *ether_address (int fd, ether_addr * address);

ether_addr *ether_intfaddr (char *intf, ether_addr * address);

char **ether_interfaces (void);

int ether_write (int fd, ether_packet * packet);

int ether_writev (int fd, ether_vec * packet);

int ether_read (int fd, ether_packet * packet);

int ether_readv (int fd, ether_vec * packet);

int ether_blocking (int fd, int state);

int ether_send_self (int fd);

int ether_mcast_self (int fd);

int ether_bcast_self (int fd);

char *ether_ntoa (ether_addr *);

ether_addr *ether_aton (char *);

#ifdef __GNUC__

/*
 * Avoid stupid warnings if structs aren't defined
 */

typedef struct in_addr *_ether_NoNsEnSe;
typedef struct hostent *_ether_nOnSeNsE;

#endif

char *ether_e2a (ether_addr *, char *);

ether_addr *ether_a2e (char *, ether_addr *);

struct in_addr *ether_e2ip (ether_addr *, struct in_addr *);

ether_addr *ether_ip2e (struct in_addr *, ether_addr *);

char *ether_e2host (ether_addr *, char *);

ether_addr *ether_host2e (char *, ether_addr *);

ether_addr *ether_hostent2e (struct hostent *, ether_addr *);
```

```
#else

int ether_open ();
ether_addr *ether_address ();
ether_addr *ether_intfaddr ();
char **ether_interfaces ();
int ether_write ();
int ether_writev ();
int ether_read ();
int ether_readv ();
int ether_blocking ();
int ether_send_self ();
int ether_mcast_self ();
int ether_bcast_self ();

char *ether_ntoa ();
ether_addr *ether_aton ();
char *ether_e2a ();
ether_addr *ether_a2e ();
struct in_addr *ether_e2ip ();
ether_addr *ether_ip2e ();
char *ether_e2host ();
ether_addr *ether_host2e ();
ether_addr *ether_hostent2e ();

#endif

#undef ether_cmp                        /* lose def from netinet/if_ether.h */

#define ether_cmp(addr1,addr2) \
 ((addr1)->shorts[0] != (addr2)->shorts[0] \
  || (addr1)->shorts[1] != (addr2)->shorts[1] \
  || (addr1)->shorts[2] != (addr2)->shorts[2])

#define ETHERSTRLEN 18                  /* max length of "xx:xx:xx:xx:xx:xx" */

#ifdef NOFILE                           /* i.e. we have included sys/param.h */
#ifndef MAXHOSTNAMELEN                  /* but MAXHOSTNAMELEN still isnt set */
#define MAXHOSTNAMELEN 64
#endif
#endif

/* should be defined in terms of ether_packet struct; need offsetof() macro */

#define ETHER_DST       0
#define ETHER_SRC       6
#define ETHER_TYPE      12
#define ETHER_PKT       14
#define ETHER_MIN       46
#define ETHER_MAX       1500

#define ETHER_MINTYPE   0x5DD           /* lowest protocol not valid IEEE802 */
#define ETHER_MAXTYPE   0xFFFF          /* largest possible protocol */

#define ETHER_MCAST(addr) (((unsigned char *) (addr))[0] & 0x01)

#ifdef NT_ALLTYPES
#define ETHER_ALLTYPES NT_ALLTYPES
#else
#define ETHER_ALLTYPES ((unsigned) -1)
```

```
#endif
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ip/domainname.c
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
domainname.c
*/

#include <stdio.h>
#include <string.h>
#include <net/netlib.h>

int getdomainname(domain, size)
char *domain;
size_t size;
{
        FILE *domainfile;
        char *line;

        domainfile= fopen("/etc/domainname", "r");
        if (!domainfile)
        {
                return -1;
        }

        line= fgets(domain, size, domainfile);
        fclose(domainfile);
        if (!line)
                return -1;
        line= strchr(domain, '\n');
        if (line)
                *line= '\0';
        return 0;
}
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ip/ether_line.c
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
**  ETHER_LINE
**
**      This routine parses the array pointed to by "line" (which should be
**      from a file in the format of /etc/ethers) and returns in "eaddr" the
**      ethernet address at the start of the line and the corresponding host
**      name in "hostname".  It assumes either tabs or spaces separate the
**      two.  The buffer pointed to by "hostname" must be big enough to hold
**      the host name plus a NULL byte.
**      The function returns 0 on success and 1 on failure.
```

```
**      Arguments are assumed sensible.  Null pointers will probably cause
**      exceptions.
**      Author: Gregory J. Sharp, July 1990
**      Adapted to MINIX: Philip Homburg, May 1992
*/

#include <sys/types.h>
#include <ctype.h>
#include <stdlib.h>
#include <net/gen/ether.h>
#include <net/gen/if_ether.h>

int
ether_line(line, eaddr, hostname)
char *                  line;
struct ether_addr *     eaddr;
char *                  hostname;
{
    register int i;
    register unsigned long val;

/* skip leading white space */
    while (*line != '\n' && (*line == ' ' || *line == '\t'))
        line++;

/* read the ethernet address */
    for (i = 0; i < 5; i++)
    {
        val = (unsigned long) strtol(line, &line, 16);
        if (val > 255 || *line++ != ':')
            return 1;
        eaddr->ea_addr[i] = val & 0xff;
    }
    val = (unsigned long) strtol(line, &line, 16);
    if (val > 255 || (*line != ' ' && *line != '\t'))
        return 1;
    eaddr->ea_addr[i] = val & 0xff;

/* skip leading white space */
    while (*line != '\n' && (*line == ' ' || *line == '\t'))
        line++;

/* read in the hostname */
    while (!isspace(*line))
        *hostname++ = *line++;
    *hostname = '\0';
    return 0;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/ip/ethera2n.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
ethera2n.c

Convert an ASCII string with an ethernet address into a struct ether_addr.

Created:        Nov 17, 1992 by Philip Homburg
*/

#include <sys/types.h>
#include <stdlib.h>
#include <net/netlib.h>
#include <net/gen/ether.h>
#include <net/gen/if_ether.h>

struct ether_addr *ether_aton(s)
char *s;
{
        static struct ether_addr ea;

        int i;
        long v;
        char *check;

        if (s == NULL)
                return NULL;

        for (i=0; i<6; i++)
        {
                v= strtol(s, &check, 16);
                if (v<0 || v>255)
                        return NULL;
                if ((i<5 && check[0] != ':') || (i == 5 && check[0] != '\0'))
                        return NULL;
                ea.ea_addr[i]= v;
                s= check+1;
        }
        return &ea;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/ip/ethere2a.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/* $Id: ethere2a.c,v 2.1 89/10/23 15:42:28 dupuy Exp $ */
/* This file was part of the etherlib package. */

#include <stdio.h>

#ifdef _MINIX
#include <sys/types.h>
#include <stdlib.h>

#include <net/gen/ether.h>
```

```
#include <net/gen/if_ether.h>

#define ETHERSTRLEN 18                  /* max length of "xx:xx:xx:xx:xx:xx" */
#define ether_addr      ether_addr_t
#define bytes           ea_addr
char *ether_e2a _ARGS(( ether_addr_t *a, char *e ));
#else
#include "libether.h"
#endif

char *
ether_e2a (addr, estring)
ether_addr *addr;
char *estring;
{
#ifdef lint
    char *sprintf ();
#endif
    if (estring == NULL)
        estring = (char *) malloc (ETHERSTRLEN);

    if (estring != NULL)
        (void) sprintf (estring, "%x:%x:%x:%x:%x:%x",
                        addr->bytes[0], addr->bytes[1], addr->bytes[2],
                        addr->bytes[3], addr->bytes[4], addr->bytes[5]);
    return (estring);
}

#ifndef ETHERDB

char *
ether_ntoa (addr)
ether_addr *addr;
{
    static char estring[ETHERSTRLEN];

    return (ether_e2a (addr, estring));
}

#endif
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/ip/etherh2n.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
etherh2n.c

Created:        May 20, 1992 by Philip Homburg
*/

#include <stdio.h>
#include <string.h>
#include <net/gen/if_ether.h>

```

```
int
ether_hostton(hostname, e)
char *hostname;
struct ether_addr *e;
{
        FILE *etherf;
        char b[256], hn[256];

        etherf= fopen(_PATH_ETHERS, "r");
        if (etherf == NULL)
                return 1;

        while(fgets(b, sizeof(b), etherf) != NULL)
        {
                if (ether_line(b, e, hn) == 0 && strcmp(hn, hostname) == 0)
                {
                        fclose(etherf);
                        return 0;
                }
        }
        fclose(etherf);
        return 1;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ip/ethern2h.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
ethern2h.c

Created:        Nov 12, 1992 by Philip Homburg
*/

#include <sys/types.h>
#include <stdio.h>
#include <string.h>
#include <net/gen/ether.h>
#include <net/gen/if_ether.h>

int
ether_ntohost(hostname, e)
char *hostname;
struct ether_addr *e;
{
        FILE *etherf;
        char b[256];
        struct ether_addr e_tmp;

        etherf= fopen(_PATH_ETHERS, "r");
        if (etherf == NULL)
                return 1;

        while(fgets(b, sizeof(b), etherf) != NULL)
        {
                if (ether_line(b, &e_tmp, hostname) == 0 &&
                memcmp(&e_tmp, e, sizeof(e_tmp)) == 0)
                {
```

```
                        fclose(etherf);
                        return 0;
                }
        }
        fclose(etherf);
        return 1;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ip/getdomain.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*      getdomainname()                                 Author: Kees J. Bot
 *                                                               2 Dec 1994
 */
#define nil 0
#include <sys/types.h>
#include <sys/utsname.h>
#include <unistd.h>
#include <string.h>

int getdomainname(char *domain, size_t size)
{
        char nodename[256];
        char *dot;

        if (gethostname(nodename, sizeof(nodename)) < 0)
                return -1;
        nodename[sizeof(nodename)-1]= 0;
        if ((dot= strchr(nodename, '.')) == nil) dot= ".";

        strncpy(domain, dot+1, size);
        if (size > 0) domain[size-1]= 0;
        return 0;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ip/gethnmadr.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * Copyright (c) 1985, 1988 Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms are permitted
 * provided that: (1) source distributions retain this entire copyright
 * notice and comment, and (2) distributions including binaries display
 * the following acknowledgement:  ‘This product includes software
 * developed by the University of California, Berkeley and its contributors’’
 * in the documentation or other materials provided with the distribution
 * and in all advertising materials mentioning features or use of this
 * software. Neither the name of the University nor the names of its
 * contributors may be used to endorse or promote products derived
 * from this software without specific prior written permission.
 * THIS SOFTWARE IS PROVIDED ‘AS IS’’ AND WITHOUT ANY EXPRESS OR
```

```
 * IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 */

#if defined(LIBC_SCCS) && !defined(lint)
static char sccsid[] = "@(#)gethostnamadr.c     6.41 (Berkeley) 6/1/90";
#endif /* LIBC_SCCS and not lint */

#ifdef _MINIX
#include <sys/types.h>
#include <ctype.h>
#include <errno.h>
#include <stdio.h>
#include <string.h>

#include <net/hton.h>
#include <net/gen/nameser.h>
#include <net/gen/netdb.h>
#include <net/gen/in.h>
#include <net/gen/inet.h>
#include <net/gen/resolv.h>
#include <net/gen/socket.h>
#else
#include <sys/param.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <ctype.h>
#include <netdb.h>
#include <stdio.h>
#include <errno.h>
#include <arpa/inet.h>
#include <arpa/nameser.h>
#include <resolv.h>
#endif /* AMOEABA */

#define MAXALIASES      35
#define MAXADDRS        35

static char *h_addr_ptrs[MAXADDRS + 1];

#ifdef _MINIX
struct in_addr
{
        ipaddr_t s_addr;
};
typedef u32_t u_long;
typedef u16_t u_short;
typedef u8_t u_char;
union querybuf;

extern int dn_skipname _ARGS(( const u_char *comp_dn, const u_char *eom ));
#define getshort _getshort
static struct hostent *getanswer _ARGS(( union querybuf *answer, int anslen,
        int iquery ));
#define bcmp memcmp
#define bcopy(s, d, l) memcpy(d, s, l)
#endif /* _MINIX */

static struct hostent host;
static char *host_aliases[MAXALIASES];
```

```
static char hostbuf[BUFSIZ+1];
static struct in_addr host_addr;

#ifndef _MINIX
char *strpbrk();
#endif /* !_MINIX */

#if PACKETSZ > 1024
#define MAXPACKET       PACKETSZ
#else
#define MAXPACKET       1024
#endif

typedef union querybuf
{
        dns_hdr_t hdr;
        u_char buf[MAXPACKET];
} querybuf_t;

typedef union align {
    long al;
    char ac;
} align_t;

static struct hostent *
getanswer(answer, anslen, iquery)
        querybuf_t *answer;
        int anslen;
        int iquery;
{
        register dns_hdr_t *hp;
        register u_char *cp;
        register int n;
        u_char *eom;
        char *bp, **ap;
        int type, class, buflen, ancount, qdcount;
        int haveanswer, getclass = C_ANY;
        char **hap;

        eom = answer->buf + anslen;
        /*
         * find first satisfactory answer
         */
        hp = &answer->hdr;
        ancount = ntohs(hp->dh_ancount);
        qdcount = ntohs(hp->dh_qdcount);
        bp = hostbuf;
        buflen = sizeof(hostbuf);
        cp = answer->buf + sizeof(dns_hdr_t);
        if (qdcount) {
                if (iquery) {
                        if ((n = dn_expand((u_char *)answer->buf, eom,
                             cp, (u_char *)bp, buflen)) < 0) {
                                h_errno = NO_RECOVERY;
                                return ((struct hostent *) NULL);
                        }
                        cp += n + QFIXEDSZ;
                        host.h_name = bp;
                        n = strlen(bp) + 1;
                        bp += n;
```

```
                        buflen -= n;
                } else
                        cp += dn_skipname(cp, eom) + QFIXEDSZ;
                while (--qdcount > 0)
                        cp += dn_skipname(cp, eom) + QFIXEDSZ;
        } else if (iquery) {
                if (hp->dh_flag1 & DHF_AA)
                        h_errno = HOST_NOT_FOUND;
                else
                        h_errno = TRY_AGAIN;
                return ((struct hostent *) NULL);
        }
        ap = host_aliases;
        *ap = NULL;
        host.h_aliases = host_aliases;
        hap = h_addr_ptrs;
        *hap = NULL;
#if BSD >= 43 || defined(h_addr)        /* new-style hostent structure */
        host.h_addr_list = h_addr_ptrs;
#endif
        haveanswer = 0;
        while (--ancount >= 0 && cp < eom) {
                if ((n = dn_expand((u_char *)answer->buf, eom, cp, (u_char *)bp,
                        buflen)) < 0)
                        break;
                cp += n;
                type = getshort(cp);
                cp += sizeof(u_short);
                class = getshort(cp);
                cp += sizeof(u_short) + sizeof(u_long);
                n = getshort(cp);
                cp += sizeof(u_short);
                if (type == T_CNAME) {
                        cp += n;
                        if (ap >= &host_aliases[MAXALIASES-1])
                                continue;
                        *ap++ = bp;
                        n = strlen(bp) + 1;
                        bp += n;
                        buflen -= n;
                        continue;
                }
                if (iquery && type == T_PTR) {
                        if ((n = dn_expand((u8_t *)answer->buf, eom,
                            cp, (u8_t *)bp, buflen)) < 0) {
                                cp += n;
                                continue;
                        }
                        cp += n;
                        host.h_name = bp;
                        return(&host);
                }
                if (iquery || type != T_A)  {
#ifdef DEBUG
                        if (_res.options & RES_DEBUG)
                                printf("unexpected answer type %d, size %d\n",
                                        type, n);
#endif
                        cp += n;
                        continue;
```

```
		}
		if (haveanswer) {
			if (n != host.h_length) {
				cp += n;
				continue;
			}
			if (class != getclass) {
				cp += n;
				continue;
			}
		} else {
			host.h_length = n;
			getclass = class;
			host.h_addrtype = (class == C_IN) ? AF_INET : AF_UNSPEC;
			if (!iquery) {
				host.h_name = bp;
				bp += strlen(bp) + 1;
			}
		}

		bp += (size_t)(sizeof(align_t) -
					((u_long)bp % sizeof(align_t)));

		if (bp + n >= &hostbuf[sizeof(hostbuf)]) {
#ifdef DEBUG
			if (_res.options & RES_DEBUG)
				printf("size (%d) too big\n", n);
#endif
			break;
		}
		bcopy(cp, *hap++ = bp, n);
		bp +=n;
		cp += n;
		haveanswer++;
	}
	if (haveanswer) {
		*ap = NULL;
#if BSD >= 43 || defined(h_addr)	/* new-style hostent structure */
		*hap = NULL;
#else
		host.h_addr = h_addr_ptrs[0];
#endif
		return (&host);
	} else {
		h_errno = TRY_AGAIN;
		return ((struct hostent *) NULL);
	}
}

struct hostent *
gethostbyname(name)
	char *name;
{
	querybuf_t buf;
	register char *cp;
	int n;

	/*
	 * disallow names consisting only of digits/dots, unless
	 * they end in a dot.
```

```
         */
        if (isdigit(name[0]))
                for (cp = name;; ++cp) {
                        if (!*cp) {
                                if (*--cp == '.')
                                        break;
                                /*
                                 * All-numeric, no dot at the end.
                                 * Fake up a hostent as if we'd actually
                                 * done a lookup.  What if someone types
                                 * 255.255.255.255?  The test below will
                                 * succeed spuriously... ???
                                 */
                                if ((host_addr.s_addr = inet_addr(name)) == -1) {
                                        h_errno = HOST_NOT_FOUND;
                                        return((struct hostent *) NULL);
                                }
                                host.h_name = name;
                                host.h_aliases = host_aliases;
                                host_aliases[0] = NULL;
                                host.h_addrtype = AF_INET;
                                host.h_length = sizeof(u_long);
                                h_addr_ptrs[0] = (char *)&host_addr;
                                h_addr_ptrs[1] = (char *)0;
#if BSD >= 43 || defined(h_addr)        /* new-style hostent structure */
                                host.h_addr_list = h_addr_ptrs;
#else
                                host.h_addr = h_addr_ptrs[0];
#endif
                                return (&host);
                        }
                        if (!isdigit(*cp) && *cp != '.')
                                break;
                }

        if ((n = res_search(name, C_IN, T_A, buf.buf, sizeof(buf))) < 0) {
#ifdef DEBUG
                if (_res.options & RES_DEBUG)
                        printf("res_search failed\n");
#endif
                return ((struct hostent *) NULL);
        }
        return (getanswer(&buf, n, 0));
}

struct hostent *
gethostbyaddr(addr, len, type)
        const char *addr;
        int len, type;
{
        int n;
        querybuf_t buf;
        register struct hostent *hp;
        char qbuf[MAXDNAME];

        if (type != AF_INET)
                return ((struct hostent *) NULL);
        (void)sprintf(qbuf, "%u.%u.%u.%u.in-addr.arpa",
                ((unsigned)addr[3] & 0xff),
                ((unsigned)addr[2] & 0xff),
```

```
                ((unsigned)addr[1] & 0xff),
                ((unsigned)addr[0] & 0xff));
        n = res_query(qbuf, C_IN, T_PTR, (u8_t *)&buf, sizeof(buf));
        if (n < 0) {
#ifdef DEBUG
                if (_res.options & RES_DEBUG)
                        printf("res_query failed\n");
#endif
                return ((struct hostent *) NULL);
        }
        hp = getanswer(&buf, n, 1);
        if (hp == NULL)
                return ((struct hostent *) NULL);
        hp->h_addrtype = type;
        hp->h_length = len;
        h_addr_ptrs[0] = (char *)&host_addr;
        h_addr_ptrs[1] = (char *)0;
        host_addr = *(struct in_addr *)addr;
#if BSD < 43 && !defined(h_addr)        /* new-style hostent structure */
        hp->h_addr = h_addr_ptrs[0];
#endif
        return(hp);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ip/gethostent.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * Copyright (c) 1985, 1988 Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms are permitted
 * provided that: (1) source distributions retain this entire copyright
 * notice and comment, and (2) distributions including binaries display
 * the following acknowledgement:  ‘This product includes software
 * developed by the University of California, Berkeley and its contributors’’
 * in the documentation or other materials provided with the distribution
 * and in all advertising materials mentioning features or use of this
 * software. Neither the name of the University nor the names of its
 * contributors may be used to endorse or promote products derived
 * from this software without specific prior written permission.
 * THIS SOFTWARE IS PROVIDED ‘AS IS’’ AND WITHOUT ANY EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 */

#if defined(LIBC_SCCS) && !defined(lint)
static char sccsid[] = "@(#)gethostnamadr.c     6.41 (Berkeley) 6/1/90";
#endif /* LIBC_SCCS and not lint */

/* Prefix the functions defined here with underscores to distinguish them
 * from the newer replacements in the resolver library.
 */
#define sethostent      _sethostent
#define endhostent      _endhostent
#define gethostent      _gethostent
#define gethostbyname   _gethostbyname
```

```
#define gethostbyaddr   _gethostbyaddr

#ifdef _MINIX
#include <sys/types.h>
#include <ctype.h>
#include <errno.h>
#include <stdio.h>
#include <string.h>

#include <net/hton.h>
#include <net/gen/nameser.h>
#include <net/gen/netdb.h>
#include <net/gen/in.h>
#include <net/gen/inet.h>
#include <net/gen/resolv.h>
#include <net/gen/socket.h>
#else
#include <sys/param.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <ctype.h>
#include <netdb.h>
#include <stdio.h>
#include <errno.h>
#include <arpa/inet.h>
#include <arpa/nameser.h>
#include <resolv.h>
#endif /* !_MINIX */

#define MAXALIASES      35
#define MAXADDRS        35

#ifdef _MINIX
typedef u32_t u_long;
typedef u16_t u_short;
typedef u8_t u_char;

#define bcmp    memcmp
#endif /* _MINIX */

static struct hostent host;
static char *host_aliases[MAXALIASES];
static char hostbuf[BUFSIZ+1];
static FILE *hostf = NULL;
static u_long hostaddr[(MAXADDRS+sizeof(u_long)-1)/sizeof(u_long)];
static char *host_addrs[2];
static int stayopen = 0;

#ifndef _MINIX
char *strpbrk();
#endif /* !_MINIX */

void
sethostent(f)
        int f;
{
        if (hostf == NULL)
                hostf = fopen(_PATH_HOSTS, "r" );
        else
                rewind(hostf);
```

```
		stayopen |= f;
	}

	void
	endhostent()
	{
		if (hostf && !stayopen) {
			(void) fclose(hostf);
			hostf = NULL;
		}
	}

	struct hostent *
	gethostent()
	{
		char *p;
		register char *cp, **q;

		if (hostf == NULL && (hostf = fopen(_PATH_HOSTS, "r" )) == NULL)
			return (NULL);
	again:
		if ((p = fgets(hostbuf, BUFSIZ, hostf)) == NULL)
			return (NULL);
		if (*p == '#')
			goto again;
		cp = strpbrk(p, "#\n");
		if (cp == NULL)
			goto again;
		*cp = '\0';
		cp = strpbrk(p, " \t");
		if (cp == NULL)
			goto again;
		*cp++ = '\0';
		/* THIS STUFF IS INTERNET SPECIFIC */
	#if BSD >= 43 || defined(h_addr)	/* new-style hostent structure */
		host.h_addr_list = host_addrs;
	#endif
		host.h_addr = (char *) hostaddr;
		*((u_long *)host.h_addr) = inet_addr(p);
		host.h_length = sizeof (u_long);
		host.h_addrtype = AF_INET;
		while (*cp == ' ' || *cp == '\t')
			cp++;
		host.h_name = cp;
		q = host.h_aliases = host_aliases;
		cp = strpbrk(cp, " \t");
		if (cp != NULL)
			*cp++ = '\0';
		while (cp && *cp) {
			if (*cp == ' ' || *cp == '\t') {
				cp++;
				continue;
			}
			if (q < &host_aliases[MAXALIASES - 1])
				*q++ = cp;
			cp = strpbrk(cp, " \t");
			if (cp != NULL)
				*cp++ = '\0';
		}
		*q = NULL;
```

```
        return (&host);
}

struct hostent *
gethostbyname(name)
        char *name;
{
        register struct hostent *p;
        register char **cp;

        sethostent(0);
        while (p = gethostent()) {
                if (strcasecmp(p->h_name, name) == 0)
                        break;
                for (cp = p->h_aliases; *cp != 0; cp++)
                        if (strcasecmp(*cp, name) == 0)
                                goto found;
        }
found:
        endhostent();
        return (p);
}

struct hostent *
gethostbyaddr(addr, len, type)
        const char *addr;
        int len, type;
{
        register struct hostent *p;

        sethostent(0);
        while (p = gethostent())
                if (p->h_addrtype == type && !bcmp(p->h_addr, addr, len))
                        break;
        endhostent();
        return (p);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ip/gethostname.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/* gethostname(2) system call emulation */

#include <sys/types.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <net/gen/netdb.h>

#define HOSTNAME_FILE "/etc/hostname.file"

int gethostname(char *buf, size_t len)
{
        int fd;
        int r;
```

```c
        char *nl;

        if ((fd= open(HOSTNAME_FILE, O_RDONLY)) < 0) return -1;

        r= read(fd, buf, len);
        close(fd);
        if (r == -1) return -1;

        buf[len-1]= '\0';
        if ((nl= strchr(buf, '\n')) != NULL) *nl= '\0';
        return 0;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ip/getproto.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * Copyright (c) 1983 Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms are permitted
 * provided that: (1) source distributions retain this entire copyright
 * notice and comment, and (2) distributions including binaries display
 * the following acknowledgement:  ``This product includes software
 * developed by the University of California, Berkeley and its contributors''
 * in the documentation or other materials provided with the distribution
 * and in all advertising materials mentioning features or use of this
 * software. Neither the name of the University nor the names of its
 * contributors may be used to endorse or promote products derived
 * from this software without specific prior written permission.
 * THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 */

#if defined(LIBC_SCCS) && !defined(lint)
static char sccsid[] = "@(#)getproto.c  5.6 (Berkeley) 6/1/90";
#endif /* LIBC_SCCS and not lint */

#include <stddef.h>

#ifdef _MINIX
#include <ansi.h>
#include <net/gen/netdb.h>
#endif

extern int _proto_stayopen;

struct protoent *
getprotobynumber(proto)
        register int proto;
{
        register struct protoent *p;

        setprotoent(_proto_stayopen);
        while (p = getprotoent())
```

```
                if (p->p_proto == proto)
                        break;
        if (!_proto_stayopen)
                endprotoent();
        return (p);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ip/getprotoent.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * Copyright (c) 1983 Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms are permitted
 * provided that: (1) source distributions retain this entire copyright
 * notice and comment, and (2) distributions including binaries display
 * the following acknowledgement:  ‘This product includes software
 * developed by the University of California, Berkeley and its contributors’’
 * in the documentation or other materials provided with the distribution
 * and in all advertising materials mentioning features or use of this
 * software. Neither the name of the University nor the names of its
 * contributors may be used to endorse or promote products derived
 * from this software without specific prior written permission.
 * THIS SOFTWARE IS PROVIDED ‘AS IS’’ AND WITHOUT ANY EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 */

#if defined(LIBC_SCCS) && !defined(lint)
static char sccsid[] = "@(#)getprotoent.c       5.7 (Berkeley) 6/1/90";
#endif /* LIBC_SCCS and not lint */

#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>

#ifdef _MINIX
#include <net/gen/netdb.h>

static char *any _ARGS(( char *cp, char *match ));
#endif

#define MAXALIASES      35

static FILE *protof = NULL;
static char line[BUFSIZ+1];
static struct protoent proto;
static char *proto_aliases[MAXALIASES];
int _proto_stayopen;

void
setprotoent(f)
        int f;
{
```

```
        if (protof == NULL)
                protof = fopen(_PATH_PROTOCOLS, "r" );
        else
                rewind(protof);
        _proto_stayopen |= f;
}

void
endprotoent()
{
        if (protof) {
                fclose(protof);
                protof = NULL;
        }
        _proto_stayopen = 0;
}

struct protoent *
getprotoent()
{
        char *p;
        register char *cp, **q;

        if (protof == NULL && (protof = fopen(_PATH_PROTOCOLS, "r" )) == NULL)
                return (NULL);
again:
        if ((p = fgets(line, BUFSIZ, protof)) == NULL)
                return (NULL);
        if (*p == '#')
                goto again;
        cp = any(p, "#\n");
        if (cp == NULL)
                goto again;
        *cp = '\0';
        proto.p_name = p;
        cp = any(p, " \t");
        if (cp == NULL)
                goto again;
        *cp++ = '\0';
        while (*cp == ' ' || *cp == '\t')
                cp++;
        p = any(cp, " \t");
        if (p != NULL)
                *p++ = '\0';
        proto.p_proto = atoi(cp);
        q = proto.p_aliases = proto_aliases;
        if (p != NULL) {
                cp = p;
                while (cp && *cp) {
                        if (*cp == ' ' || *cp == '\t') {
                                cp++;
                                continue;
                        }
                        if (q < &proto_aliases[MAXALIASES - 1])
                                *q++ = cp;
                        cp = any(cp, " \t");
                        if (cp != NULL)
                                *cp++ = '\0';
                }
        }
```

```
        *q = NULL;
        return (&proto);
}

static char *
any(cp, match)
        register char *cp;
        char *match;
{
        register char *mp, c;

        while (c = *cp) {
                for (mp = match; *mp; mp++)
                        if (*mp == c)
                                return (cp);
                cp++;
        }
        return ((char *)0);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ip/getservent.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * Copyright (c) 1983 Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms are permitted
 * provided that: (1) source distributions retain this entire copyright
 * notice and comment, and (2) distributions including binaries display
 * the following acknowledgement:  ‘This product includes software
 * developed by the University of California, Berkeley and its contributors’’
 * in the documentation or other materials provided with the distribution
 * and in all advertising materials mentioning features or use of this
 * software. Neither the name of the University nor the names of its
 * contributors may be used to endorse or promote products derived
 * from this software without specific prior written permission.
 * THIS SOFTWARE IS PROVIDED ‘AS IS’’ AND WITHOUT ANY EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 */

#if defined(LIBC_SCCS) && !defined(lint)
static char sccsid[] = "@(#)getservent.c        5.8 (Berkeley) 6/1/90";
#endif /* LIBC_SCCS and not lint */

#include <sys/types.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>

#include <net/hton.h>
#include <net/gen/netdb.h>

#define MAXALIASES      35

static FILE *servf = NULL;
static char line[BUFSIZ+1];
```

```
static struct servent serv;
static char *serv_aliases[MAXALIASES];
int _serv_stayopen;

static char *any _ARGS(( char *cp, char *match ));

void
setservent(f)
        int f;
{
        if (servf == NULL)
                servf = fopen(_PATH_SERVICES, "r" );
        else
                rewind(servf);
        _serv_stayopen |= f;
}

void
endservent()
{
        if (servf) {
                fclose(servf);
                servf = NULL;
        }
        _serv_stayopen = 0;
}

struct servent *
getservent()
{
        char *p;
        register char *cp, **q;

        if (servf == NULL && (servf = fopen(_PATH_SERVICES, "r" )) == NULL)
                return (NULL);
again:
        if ((p = fgets(line, BUFSIZ, servf)) == NULL)
                return (NULL);
        if (*p == '#')
                goto again;
        cp = any(p, "#\n");
        if (cp == NULL)
                goto again;
        *cp = '\0';
        serv.s_name = p;
        p = any(p, " \t");
        if (p == NULL)
                goto again;
        *p++ = '\0';
        while (*p == ' ' || *p == '\t')
                p++;
        cp = any(p, ",/");
        if (cp == NULL)
                goto again;
        *cp++ = '\0';
        serv.s_port = htons((u16_t)atoi(p));
        serv.s_proto = cp;
        q = serv.s_aliases = serv_aliases;
        cp = any(cp, " \t");
        if (cp != NULL)
```

```
                *cp++ = '\0';
        while (cp && *cp) {
                if (*cp == ' ' || *cp == '\t') {
                        cp++;
                        continue;
                }
                if (q < &serv_aliases[MAXALIASES - 1])
                        *q++ = cp;
                cp = any(cp, " \t");
                if (cp != NULL)
                        *cp++ = '\0';
        }
        *q = NULL;
        return (&serv);
}

static char *
any(cp, match)
        register char *cp;
        char *match;
{
        register char *mp, c;

        while (c = *cp) {
                for (mp = match; *mp; mp++)
                        if (*mp == c)
                                return (cp);
                cp++;
        }
        return ((char *)0);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ip/getsrvbyname.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * Copyright (c) 1983 Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms are permitted
 * provided that: (1) source distributions retain this entire copyright
 * notice and comment, and (2) distributions including binaries display
 * the following acknowledgement:  ``This product includes software
 * developed by the University of California, Berkeley and its contributors''
 * in the documentation or other materials provided with the distribution
 * and in all advertising materials mentioning features or use of this
 * software. Neither the name of the University nor the names of its
 * contributors may be used to endorse or promote products derived
 * from this software without specific prior written permission.
 * THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 */

#if defined(LIBC_SCCS) && !defined(lint)
```

```
static char sccsid[] = "@(#)getservbyname.c     5.6 (Berkeley) 6/1/90";
#endif /* LIBC_SCCS and not lint */

#include <string.h>

#include <net/gen/netdb.h>

extern int _serv_stayopen;

struct servent *
getservbyname(name, proto)
        const char *name, *proto;
{
        register struct servent *p;
        register char **cp;

        setservent(_serv_stayopen);
        while (p = getservent()) {
                if (strcmp(name, p->s_name) == 0)
                        goto gotname;
                for (cp = p->s_aliases; *cp; cp++)
                        if (strcmp(name, *cp) == 0)
                                goto gotname;
                continue;
gotname:
                if (proto == 0 || strcmp(p->s_proto, proto) == 0)
                        break;
        }
        if (!_serv_stayopen)
                endservent();
        return (p);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ip/getsrvbyport.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * Copyright (c) 1983 Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms are permitted
 * provided that: (1) source distributions retain this entire copyright
 * notice and comment, and (2) distributions including binaries display
 * the following acknowledgement:  ‘This product includes software
 * developed by the University of California, Berkeley and its contributors’’
 * in the documentation or other materials provided with the distribution
 * and in all advertising materials mentioning features or use of this
 * software. Neither the name of the University nor the names of its
 * contributors may be used to endorse or promote products derived
 * from this software without specific prior written permission.
 * THIS SOFTWARE IS PROVIDED ‘AS IS’’ AND WITHOUT ANY EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 */

#if defined(LIBC_SCCS) && !defined(lint)
```

```
static char sccsid[] = "@(#)getservbyport.c     5.6 (Berkeley) 6/1/90";
#endif /* LIBC_SCCS and not lint */

#include <stddef.h>
#include <string.h>

#ifdef _MINIX
#include <net/gen/netdb.h>
#endif

extern int _serv_stayopen;

struct servent *
getservbyport(port, proto)
        int port;
        const char *proto;
{
        register struct servent *p;

        setservent(_serv_stayopen);
        while (p = getservent()) {
                if (p->s_port != port)
                        continue;
                if (proto == 0 || strcmp(p->s_proto, proto) == 0)
                        break;
        }
        if (!_serv_stayopen)
                endservent();
        return (p);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ip/hton.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
hton.c
*/

#include <sys/types.h>
#include <minix/config.h>
#include <net/hton.h>

u16_t _tmp;
u32_t _tmp_l;
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ip/inet_addr.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * Copyright (c) 1983, 1990 Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms are permitted provided
 * that: (1) source distributions retain this entire copyright notice and
 * comment, and (2) distributions including binaries display the following
 * acknowledgement:  `This product includes software developed by the
 * University of California, Berkeley and its contributors'' in the
 * documentation or other materials provided with the distribution and in
 * all advertising materials mentioning features or use of this software.
 * Neither the name of the University nor the names of its contributors may
 * be used to endorse or promote products derived from this software without
 * specific prior written permission.
 * THIS SOFTWARE IS PROVIDED `AS IS'' AND WITHOUT ANY EXPRESS OR IMPLIED
 * WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 */

#if defined(LIBC_SCCS) && !defined(lint)
static char sccsid[] = "@(#)inet_addr.c 5.8 (Berkeley) 6/23/90";
#endif /* LIBC_SCCS and not lint */

#if _MINIX
#include <sys/types.h>
#include <ctype.h>
#include <errno.h>

#include <net/hton.h>
#include <net/gen/in.h>
#include <net/gen/inet.h>
#endif

#ifdef __STDC__
#define _CONST  const
#else
#define _CONST
#endif

/*
 * Ascii internet address interpretation routine.
 * The value returned is in network order.
 */
ipaddr_t
inet_addr(cp)
        register _CONST char *cp;
{
        ipaddr_t val;

        if (inet_aton(cp, &val))
                return (val);
        errno= EINVAL;
        return (ipaddr_t)-1;
}
```

```
/*
 * Check whether "cp" is a valid ascii representation
 * of an Internet address and convert to a binary address.
 * Returns 1 if the address is valid, 0 if not.
 * This replaces inet_addr, the return value from which
 * cannot distinguish between failure and a local broadcast address.
 */

int
inet_aton(cp, addr)
        register _CONST char *cp;
        ipaddr_t *addr;
{
        register u32_t val, base, n;
        register char c;
        u32_t parts[4], *pp = parts;

        for (;;) {
                /*
                 * Collect number up to `.''.
                 * Values are specified as for C:
                 * 0x=hex, 0=octal, other=decimal.
                 */
                val = 0; base = 10;
                if (*cp == '0') {
                        if (*++cp == 'x' || *cp == 'X')
                                base = 16, cp++;
                        else
                                base = 8;
                }
                while ((c = *cp) != '\0') {
                        if (isascii(c) && isdigit(c)) {
                                val = (val * base) + (c - '0');
                                cp++;
                                continue;
                        }
                        if (base == 16 && isascii(c) && isxdigit(c)) {
                                val = (val << 4) +
                                        (c + 10 - (islower(c) ? 'a' : 'A'));
                                cp++;
                                continue;
                        }
                        break;
                }
                if (*cp == '.') {
                        /*
                         * Internet format:
                         *      a.b.c.d
                         *      a.b.c   (with c treated as 16-bits)
                         *      a.b     (with b treated as 24 bits)
                         */
                        if (pp >= parts + 3 || val > 0xff)
                                return (0);
                        *pp++ = val, cp++;
                } else
                        break;
        }
        /*
         * Check for trailing characters.
         */
```

```
        if (*cp && (!isascii(*cp) || !isspace(*cp)))
                return (0);
        /*
         * Concoct the address according to
         * the number of parts specified.
         */
        n = pp - parts + 1;
        switch (n) {

        case 1:                         /* a -- 32 bits */
                break;

        case 2:                         /* a.b -- 8.24 bits */
                if (val > 0xffffff)
                        return (0);
                val |= parts[0] << 24;
                break;

        case 3:                         /* a.b.c -- 8.8.16 bits */
                if (val > 0xffff)
                        return (0);
                val |= (parts[0] << 24) | (parts[1] << 16);
                break;

        case 4:                         /* a.b.c.d -- 8.8.8.8 bits */
                if (val > 0xff)
                        return (0);
                val |= (parts[0] << 24) | (parts[1] << 16) | (parts[2] << 8);
                break;
        }
        if (addr)
                *addr = htonl(val);
        return (1);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ip/inet_ntoa.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * Copyright (c) 1983 Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms are permitted
 * provided that: (1) source distributions retain this entire copyright
 * notice and comment, and (2) distributions including binaries display
 * the following acknowledgement:  ‘This product includes software
 * developed by the University of California, Berkeley and its contributors’’
 * in the documentation or other materials provided with the distribution
 * and in all advertising materials mentioning features or use of this
 * software. Neither the name of the University nor the names of its
 * contributors may be used to endorse or promote products derived
 * from this software without specific prior written permission.
 * THIS SOFTWARE IS PROVIDED ‘AS IS’’ AND WITHOUT ANY EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 */

#if defined(LIBC_SCCS) && !defined(lint)
```

```
static char sccsid[] = "@(#)inet_ntoa.c 5.5 (Berkeley) 6/1/90";
#endif /* LIBC_SCCS and not lint */

/*
 * Convert network-format internet address
 * to base 256 d.d.d.d representation.
 */

#include <sys/types.h>
#include <stdio.h>

#include <net/gen/in.h>
#include <net/gen/inet.h>

char *
inet_ntoa(in)
        ipaddr_t in;
{
        static char b[18];
        register u8_t *p;

        p = (u8_t *)&in;
        sprintf(b, "%d.%d.%d.%d", p[0], p[1], p[2], p[3]);
        return (b);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                           src/lib/ip/memcspn.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: strcspn.c,v 1.1 91/12/19 13:20:40 philip Exp $ */

#include        <string.h>

size_t
memcspn(const char *string, size_t strlen, const char *notin, size_t notinlen)
{
        register const char *s1, *s2;
        int i,j;

        for (s1 = string, i = 0; i<strlen; s1++, i++) {
                for(s2 = notin, j = 0; *s2 != *s1 && j < notinlen; s2++, j++)
                        /* EMPTY */ ;
                if (j != notinlen)
                        break;
        }
        return s1 - string;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ip/oneC_sum.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*      oneC_sum() - One complement's checksum          Author: Kees J. Bot
 *                                                              8 May 1995
 * See RFC 1071, "Computing the Internet checksum"
 */

#include <sys/types.h>
#include <net/gen/oneCsum.h>

u16_t oneC_sum(U16_t prev, void *data, size_t size)
{
        u8_t *dptr;
        size_t n;
        u16_t word;
        u32_t sum;
        int swap= 0;

        sum= prev;
        dptr= data;
        n= size;

        swap= ((size_t) dptr & 1);
        if (swap) {
                sum= ((sum & 0xFF) << 8) | ((sum & 0xFF00) >> 8);
                if (n > 0) {
                        ((u8_t *) &word)[0]= 0;
                        ((u8_t *) &word)[1]= dptr[0];
                        sum+= (u32_t) word;
                        dptr+= 1;
                        n-= 1;
                }
        }

        while (n >= 8) {
                sum+= (u32_t) ((u16_t *) dptr)[0]
                    + (u32_t) ((u16_t *) dptr)[1]
                    + (u32_t) ((u16_t *) dptr)[2]
                    + (u32_t) ((u16_t *) dptr)[3];
                dptr+= 8;
                n-= 8;
        }

        while (n >= 2) {
                sum+= (u32_t) ((u16_t *) dptr)[0];
                dptr+= 2;
                n-= 2;
        }

        if (n > 0) {
                ((u8_t *) &word)[0]= dptr[0];
                ((u8_t *) &word)[1]= 0;
                sum+= (u32_t) word;
        }

        sum= (sum & 0xFFFF) + (sum >> 16);
        if (sum > 0xFFFF) sum++;
```

```

        if (swap) {
                sum= ((sum & 0xFF) << 8) | ((sum & 0xFF00) >> 8);
        }
        return sum;
}
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ip/rcmd.c
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * Copyright (c) 1983 Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms are permitted
 * provided that: (1) source distributions retain this entire copyright
 * notice and comment, and (2) distributions including binaries display
 * the following acknowledgement:  ‘This product includes software
 * developed by the University of California, Berkeley and its contributors’’
 * in the documentation or other materials provided with the distribution
 * and in all advertising materials mentioning features or use of this
 * software. Neither the name of the University nor the names of its
 * contributors may be used to endorse or promote products derived
 * from this software without specific prior written permission.
 * THIS SOFTWARE IS PROVIDED ‘AS IS’’ AND WITHOUT ANY EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 */

#if defined(LIBC_SCCS) && !defined(lint)
static char sccsid[] = "@(#)rcmd.c      5.22 (Berkeley) 6/1/90";
#endif /* LIBC_SCCS and not lint */

#if _MINIX
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <ctype.h>
#include <errno.h>
#include <fcntl.h>
#include <limits.h>
#include <pwd.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <unistd.h>
#include <net/gen/netdb.h>
#include <net/gen/in.h>
#include <net/gen/tcp.h>
#include <net/gen/tcp_io.h>
#include <net/hton.h>
#include <net/netlib.h>

#define MAXHOSTNAMELEN  256
```

```
#define MAXPATHLEN PATH_MAX
#else
#include <stdio.h>
#include <ctype.h>
#include <pwd.h>
#include <sys/param.h>
#include <sys/file.h>
#include <sys/signal.h>
#include <sys/socket.h>
#include <sys/stat.h>

#include <netinet/in.h>

#include <netdb.h>
#include <errno.h>
#endif

#ifdef __STDC__
#define CONST   const
#else
#define CONST
#endif

extern  errno;
#if _MINIX
int _validuser _ARGS(( FILE *hostf, const char *rhost, const char *luser,
                                        const char *ruser, int baselen ));
static int _checkhost _ARGS(( const char *rhost, const char *lhost, int len ));
#else
char    *index();
#endif

#if _MINIX
int rcmd(ahost, rport, locuser, remuser, cmd, fd2p)
char **ahost;
int rport;
CONST char *locuser, *remuser, *cmd;
int *fd2p;
{
        int fd, fd2, result;
        struct hostent *hp;
        tcpport_t lport;
        nwio_tcpconf_t tcpconf;
        nwio_tcpcl_t tcpconnopt;
        pid_t pid;
        char num[8];
        char c;
        char *tcp_device;

        fd= -1;
        fd2= -1;

        tcp_device= getenv("TCP_DEVICE");
        if (tcp_device == NULL)
                tcp_device= TCP_DEVICE;
        hp= gethostbyname(*ahost);
        if (!hp)
        {
                fprintf(stderr, "%s: unknown host\n", *ahost);
                return -1;
```

```
	}
	*ahost= hp->h_name;
	for (lport= TCPPORT_RESERVED-1; lport >= TCPPORT_RESERVED/2; lport--)
	{
		fd= open (tcp_device, O_RDWR);
		if (fd<0)
		{
			fprintf(stderr, "unable to open %s: %s\n",
				tcp_device, strerror(errno));
			goto bad;
		}
		tcpconf.nwtc_flags= NWTC_LP_SET | NWTC_SET_RA | NWTC_SET_RP |
			NWTC_EXCL;
		tcpconf.nwtc_locport= htons(lport);
		tcpconf.nwtc_remport= rport;
		tcpconf.nwtc_remaddr= *(ipaddr_t *)hp->h_addr;

		result= ioctl(fd, NWIOSTCPCONF, &tcpconf);
		if (result<0)
		{
			if (errno == EADDRINUSE)
			{
				close(fd);
				continue;
			}
			fprintf(stderr, "unable to ioctl(NWIOSTCPCONF): %s\n",
				strerror(errno));
			goto bad;
		}
		tcpconf.nwtc_flags= NWTC_SHARED;
		result= ioctl(fd, NWIOSTCPCONF, &tcpconf);
		if (result<0)
		{
			fprintf(stderr, "unable to ioctl(NWIOSTCPCONF): %s\n",
				strerror(errno));
			goto bad;
		}
		tcpconnopt.nwtcl_flags= 0;

		do
		{
			result= ioctl (fd, NWIOTCPCONN, &tcpconnopt);
			if (result<0 && errno == EAGAIN)
			{
				sleep(2);
			}
		} while (result<0 && errno == EAGAIN);
		if (result<0 && errno != EADDRINUSE)
		{
			fprintf(stderr,
				"unable to ioctl(NWIOTCPCONN): %s\n",
				strerror(errno));
			goto bad;
		}
		if (result>=0)
			break;
	}
	if (lport<TCPPORT_RESERVED/2)
	{
		fprintf(stderr, "can't get port\n");
```

```
                        return -1;
                }
                if (!fd2p)
                {
                        if (write(fd, "", 1) != 1)
                        {
                                fprintf(stderr, "unable to write: %s", strerror(errno));
                                goto bad;
                        }
                }
                else
                {
                        fd2= open (tcp_device, O_RDWR);
                        if (fd2<0)
                        {
                                fprintf(stderr, "unable to open %s: %s\n",
                                        tcp_device, strerror(errno));
                                goto bad;
                        }
                        tcpconf.nwtc_flags= NWTC_LP_SET | NWTC_UNSET_RA |
                                NWTC_UNSET_RP | NWTC_SHARED;
                        tcpconf.nwtc_locport= htons(lport);

                        result= ioctl(fd2, NWIOSTCPCONF, &tcpconf);
                        if (result<0)
                        {
                                fprintf(stderr,
                                        "unable to ioctl(NWIOSTCPCONF): %s\n",
                                        strerror(errno));
                                goto bad;
                        }
                        pid= fork();
                        if (pid<0)
                        {
                                fprintf(stderr, "unable to fork: %s\n",
                                        strerror(errno));
                                goto bad;
                        }
                        if (!pid)
                        {
                                alarm(0);
                                signal(SIGALRM, SIG_DFL);
                                alarm(30); /* give up after half a minute */
                                tcpconnopt.nwtcl_flags= 0;

                                do
                                {
                                        result= ioctl (fd2, NWIOTCPLISTEN,
                                                &tcpconnopt);
                                        if (result<0 && errno == EAGAIN)
                                        {
                                                sleep(2);
                                        }
                                } while (result<0 && errno == EAGAIN);
                                if (result<0 && errno != EADDRINUSE)
                                {
                                        fprintf(stderr,
                                                "unable to ioctl(NWIOTCPLISTEN): %s\n",
                                                strerror(errno));
                                        exit(1);
```

```
				}
				if (result>=0)
					exit(0);
				else
					exit(1);
			}
			/*
			 * This sleep is a HACK.  The command that we are starting
			 * will try to connect to the fd2 port.  It seems that for
			 * this to succeed the child process must have already made
			 * the call to ioctl above (the NWIOTCPLISTEN) call.
			 * The sleep gives the child a chance to make the call
			 * before the parent sends the port number to the
			 * command being started.
			 */
			sleep(1);

			sprintf(num, "%d", lport);
			if (write(fd, num, strlen(num)+1) != strlen(num)+1)
			{
				fprintf(stderr, "unable to write: %s\n",
					strerror(errno));
				goto bad;
			}

		}
		write (fd, locuser, strlen(locuser)+1);
		write (fd, remuser, strlen(remuser)+1);
		write (fd, cmd, strlen(cmd)+1);
		if (read(fd, &c, 1) != 1)
		{
			fprintf(stderr, "unable to read: %s\n", strerror(errno) );
			goto bad;
		}
		if (c != 0)
		{
			while (read(fd, &c, 1) == 1)
			{
				write(2, &c, 1);
				if (c == '\n')
					break;
			}
			goto bad;
		}
		if (fd2p)
		{
			*fd2p= fd2;
			result= ioctl(fd2, NWIOGTCPCONF, &tcpconf);
			if (result<0)
			{
				fprintf(stderr, "unable to ioctl(NWIOGTCPCONF): %s\n",
					strerror(errno) );
				goto bad;
			}
			if (ntohs(tcpconf.nwtc_remport) >= TCPPORT_RESERVED)
			{
				fprintf(stderr, "unable to setup 2nd channel\n");
				goto bad;
			}
		}
```

```
	return fd;

bad:
	if (fd>=0)
		close(fd);
	if (fd2>=0)
		close(fd2);
	return -1;
}
#else /* _MINIX */
rcmd(ahost, rport, locuser, remuser, cmd, fd2p)
	char **ahost;
	u_short rport;
	char *locuser, *remuser, *cmd;
	int *fd2p;
{
	int s, timo = 1, pid;
	long oldmask;
	struct sockaddr_in sin, sin2, from;
	char c;
	int lport = IPPORT_RESERVED - 1;
	struct hostent *hp;
	fd_set reads;

	pid = getpid();
	hp = gethostbyname(*ahost);
	if (hp == 0) {
		herror(*ahost);
		return (-1);
	}
	*ahost = hp->h_name;
	oldmask = sigblock(sigmask(SIGURG));
	for (;;) {
		s = rresvport(&lport);
		if (s < 0) {
			if (errno == EAGAIN)
				fprintf(stderr, "socket: All ports in use\n");
			else
				perror("rcmd: socket");
			sigsetmask(oldmask);
			return (-1);
		}
		fcntl(s, F_SETOWN, pid);
		sin.sin_family = hp->h_addrtype;
		bcopy(hp->h_addr_list[0], (caddr_t)&sin.sin_addr, hp->h_length);
		sin.sin_port = rport;
		if (connect(s, (caddr_t)&sin, sizeof (sin), 0) >= 0)
			break;
		(void) close(s);
		if (errno == EADDRINUSE) {
			lport--;
			continue;
		}
		if (errno == ECONNREFUSED && timo <= 16) {
			sleep(timo);
			timo *= 2;
			continue;
		}
		if (hp->h_addr_list[1] != NULL) {
			int oerrno = errno;
```

```

                        fprintf(stderr,
                            "connect to address %s: ", inet_ntoa(sin.sin_addr));
                        errno = oerrno;
                        perror(0);
                        hp->h_addr_list++;
                        bcopy(hp->h_addr_list[0], (caddr_t)&sin.sin_addr,
                            hp->h_length);
                        fprintf(stderr, "Trying %s...\n",
                                inet_ntoa(sin.sin_addr));
                        continue;
                }
                perror(hp->h_name);
                sigsetmask(oldmask);
                return (-1);
        }
        lport--;
        if (fd2p == 0) {
                write(s, "", 1);
                lport = 0;
        } else {
                char num[8];
                int s2 = rresvport(&lport), s3;
                int len = sizeof (from);

                if (s2 < 0)
                        goto bad;
                listen(s2, 1);
                (void) sprintf(num, "%d", lport);
                if (write(s, num, strlen(num)+1) != strlen(num)+1) {
                        perror("write: setting up stderr");
                        (void) close(s2);
                        goto bad;
                }
                FD_ZERO(&reads);
                FD_SET(s, &reads);
                FD_SET(s2, &reads);
                errno = 0;
                if (select(32, &reads, 0, 0, 0) < 1 ||
                    !FD_ISSET(s2, &reads)) {
                        if (errno != 0)
                                perror("select: setting up stderr");
                        else
                            fprintf(stderr,
                                "select: protocol failure in circuit setup.\n");
                        (void) close(s2);
                        goto bad;
                }
                s3 = accept(s2, &from, &len, 0);
                (void) close(s2);
                if (s3 < 0) {
                        perror("accept");
                        lport = 0;
                        goto bad;
                }
                *fd2p = s3;
                from.sin_port = ntohs((u_short)from.sin_port);
                if (from.sin_family != AF_INET ||
                    from.sin_port >= IPPORT_RESERVED ||
                    from.sin_port < IPPORT_RESERVED / 2) {
```

```
                                fprintf(stderr,
                                    "socket: protocol failure in circuit setup.\n");
                                goto bad2;
                        }
                }
                (void) write(s, locuser, strlen(locuser)+1);
                (void) write(s, remuser, strlen(remuser)+1);
                (void) write(s, cmd, strlen(cmd)+1);
                if (read(s, &c, 1) != 1) {
                        perror(*ahost);
                        goto bad2;
                }
                if (c != 0) {
                        while (read(s, &c, 1) == 1) {
                                (void) write(2, &c, 1);
                                if (c == '\n')
                                        break;
                        }
                        goto bad2;
                }
                sigsetmask(oldmask);
                return (s);
        bad2:
                if (lport)
                        (void) close(*fd2p);
        bad:
                (void) close(s);
                sigsetmask(oldmask);
                return (-1);
        }

        rresvport(alport)
                int *alport;
        {
                struct sockaddr_in sin;
                int s;

                sin.sin_family = AF_INET;
                sin.sin_addr.s_addr = INADDR_ANY;
                s = socket(AF_INET, SOCK_STREAM, 0);
                if (s < 0)
                        return (-1);
                for (;;) {
                        sin.sin_port = htons((u_short)*alport);
                        if (bind(s, (caddr_t)&sin, sizeof (sin)) >= 0)
                                return (s);
                        if (errno != EADDRINUSE) {
                                (void) close(s);
                                return (-1);
                        }
                        (*alport)--;
                        if (*alport == IPPORT_RESERVED/2) {
                                (void) close(s);
                                errno = EAGAIN;         /* close */
                                return (-1);
                        }
                }
        }
        #endif /* _MINIX */
```

```
int     _check_rhosts_file = 1;

ruserok(rhost, superuser, ruser, luser)
        CONST char *rhost;
        int superuser;
        CONST char *ruser, *luser;
{
        FILE *hostf;
        char fhost[MAXHOSTNAMELEN];
        int first = 1;
        register CONST char *sp;
        register char *p;
        int baselen = -1;

        sp = rhost;
        p = fhost;
        while (*sp) {
                if (*sp == '.') {
                        if (baselen == -1)
                                baselen = sp - rhost;
                        *p++ = *sp++;
                } else {
                        *p++ = isupper(*sp) ? tolower(*sp++) : *sp++;
                }
        }
        *p = '\0';
        hostf = superuser ? (FILE *)0 : fopen(_PATH_HEQUIV, "r");
again:
        if (hostf) {
                if (!_validuser(hostf, fhost, luser, ruser, baselen)) {
                        (void) fclose(hostf);
                        return(0);
                }
                (void) fclose(hostf);
        }
        if (first == 1 && (_check_rhosts_file || superuser)) {
                struct stat sbuf;
                struct passwd *pwd;
                char pbuf[MAXPATHLEN];

                first = 0;
                if ((pwd = getpwnam(luser)) == NULL)
                        return(-1);
                (void)strcpy(pbuf, pwd->pw_dir);
                (void)strcat(pbuf, "/.rhosts");
                if ((hostf = fopen(pbuf, "r")) == NULL)
                        return(-1);
                /*
                 * if owned by someone other than user or root or if
                 * writeable by anyone but the owner, quit
                 */
                if (fstat(fileno(hostf), &sbuf) ||
                    sbuf.st_uid && sbuf.st_uid != pwd->pw_uid ||
                    sbuf.st_mode&022) {
                        fclose(hostf);
                        return(-1);
                }
                goto again;
        }
        return (-1);
```

```
}

/* don't make static, used by lpd(8) */
int _validuser(hostf, rhost, luser, ruser, baselen)
        FILE *hostf;
        CONST char *rhost, *luser, *ruser;
        int baselen;
{
        char *user;
        char ahost[MAXHOSTNAMELEN];
        register char *p;

        while (fgets(ahost, sizeof (ahost), hostf)) {
                p = ahost;
                while (*p != '\n' && *p != ' ' && *p != '\t' && *p != '\0') {
                        *p = isupper(*p) ? tolower(*p) : *p;
                        p++;
                }
                if (*p == ' ' || *p == '\t') {
                        *p++ = '\0';
                        while (*p == ' ' || *p == '\t')
                                p++;
                        user = p;
                        while (*p != '\n' && *p != ' ' && *p != '\t' && *p != '\0')
                                p++;
                } else
                        user = p;
                *p = '\0';
                if (_checkhost(rhost, ahost, baselen) &&
                    !strcmp(ruser, *user ? user : luser)) {
                        return (0);
                }
        }
        return (-1);
}

static int
_checkhost(rhost, lhost, len)
        CONST char *rhost, *lhost;
        int len;
{
        static char ldomain[MAXHOSTNAMELEN + 1];
        static char *domainp = NULL;
        static int nodomain = 0;
        register char *cp;

        if (len == -1)
                return(!strcmp(rhost, lhost));
        if (strncmp(rhost, lhost, len))
                return(0);
        if (!strcmp(rhost, lhost))
                return(1);
        if (*(lhost + len) != '\0')
                return(0);
        if (nodomain)
                return(0);
        if (!domainp) {
                if (gethostname(ldomain, sizeof(ldomain)) == -1) {
                        nodomain = 1;
                        return(0);
```

```
                }
                ldomain[MAXHOSTNAMELEN] = 0;
                if ((domainp = index(ldomain, '.')) == (char *)NULL) {
                        nodomain = 1;
                        return(0);
                }
                for (cp = ++domainp; *cp; ++cp)
                        if (isupper(*cp))
                                *cp = tolower(*cp);
        }
        return(!strcmp(domainp, rhost + len +1));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ip/res_comp.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * Copyright (c) 1985 Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms are permitted provided
 * that: (1) source distributions retain this entire copyright notice and
 * comment, and (2) distributions including binaries display the following
 * acknowledgement:  `This product includes software developed by the
 * University of California, Berkeley and its contributors'' in the
 * documentation or other materials provided with the distribution and in
 * all advertising materials mentioning features or use of this software.
 * Neither the name of the University nor the names of its contributors may
 * be used to endorse or promote products derived from this software without
 * specific prior written permission.
 * THIS SOFTWARE IS PROVIDED `AS IS'' AND WITHOUT ANY EXPRESS OR IMPLIED
 * WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 */

#if defined(LIBC_SCCS) && !defined(lint)
static char sccsid[] = "@(#)res_comp.c  6.18 (Berkeley) 6/27/90";
#endif /* LIBC_SCCS and not lint */

#if _MINIX
#include <sys/types.h>
#include <stdlib.h>

#include <net/gen/in.h>
#include <net/gen/nameser.h>
#include <net/gen/resolv.h>

typedef u8_t u_char;
typedef u16_t u_short;
typedef u32_t u_long;

static int dn_find _ARGS(( const u_char *exp_dn, const u_char *msg,
        u_char **dnptrs, u_char **lastdnptr ));
int dn_skipname _ARGS(( const u_char *comp_dn, const u_char *eom ));

#define getshort _getshort
```

```
#define getlong _getlong
#define putshort __putshort
#define putlong __putlong
#else
#include <sys/types.h>
#include <stdio.h>
#include <arpa/nameser.h>

static dn_find();
#endif

#ifdef __STDC__
#define CONST   const
#else
#define CONST
#endif

/*
 * Expand compressed domain name 'comp_dn' to full domain name.
 * 'msg' is a pointer to the begining of the message,
 * 'eomorig' points to the first location after the message,
 * 'exp_dn' is a pointer to a buffer of size 'length' for the result.
 * Return size of compressed name or -1 if there was an error.
 */
dn_expand(msg, eomorig, comp_dn, exp_dn, length)
        CONST u_char *msg, *eomorig, *comp_dn;
        u_char *exp_dn;
        int length;
{
        register CONST u_char *cp;
        register u_char *dn;
        register int n, c;
        CONST u_char *eom;
        int len = -1, checked = 0;

        dn = exp_dn;
        cp = comp_dn;
        eom = exp_dn + length;
        /*
         * fetch next label in domain name
         */
        while (n = *cp++) {
                /*
                 * Check for indirection
                 */
                switch (n & INDIR_MASK) {
                case 0:
                        if (dn != exp_dn) {
                                if (dn >= eom)
                                        return (-1);
                                *dn++ = '.';
                        }
                        if (dn+n >= eom)
                                return (-1);
                        checked += n + 1;
                        while (--n >= 0) {
                                if ((c = *cp++) == '.') {
                                        if (dn + n + 2 >= eom)
                                                return (-1);
                                        *dn++ = '\\';
```

```
                                }
                                *dn++ = c;
                                if (cp >= eomorig)      /* out of range */
                                        return(-1);
                        }
                        break;

                case INDIR_MASK:
                        if (len < 0)
                                len = cp - comp_dn + 1;
                        cp = msg + (((n & 0x3f) << 8) | (*cp & 0xff));
                        if (cp < msg || cp >= eomorig)  /* out of range */
                                return(-1);
                        checked += 2;
                        /*
                         * Check for loops in the compressed name;
                         * if we've looked at the whole message,
                         * there must be a loop.
                         */
                        if (checked >= eomorig - msg)
                                return (-1);
                        break;

                default:
                        return (-1);                    /* flag error */
                }
        }
        *dn = '\0';
        if (len < 0)
                len = cp - comp_dn;
        return (len);
}

/*
 * Compress domain name 'exp_dn' into 'comp_dn'.
 * Return the size of the compressed name or -1.
 * 'length' is the size of the array pointed to by 'comp_dn'.
 * 'dnptrs' is a list of pointers to previous compressed names. dnptrs[0]
 * is a pointer to the beginning of the message. The list ends with NULL.
 * 'lastdnptr' is a pointer to the end of the arrary pointed to
 * by 'dnptrs'. Side effect is to update the list of pointers for
 * labels inserted into the message as we compress the name.
 * If 'dnptr' is NULL, we don't try to compress names. If 'lastdnptr'
 * is NULL, we don't update the list.
 */
int
dn_comp(exp_dn, comp_dn, length, dnptrs, lastdnptr)
        CONST u_char *exp_dn;
        u_char *comp_dn;
        int length;
        u_char **dnptrs, **lastdnptr;
{
        register u_char *cp;
        register CONST u_char *dn;
        register int c, l;
        u_char **cpp, **lpp, *sp, *eob;
        u_char *msg;

        dn = exp_dn;
        cp = comp_dn;
```

```
        eob = cp + length;
        if (dnptrs != NULL) {
                if ((msg = *dnptrs++) != NULL) {
                        for (cpp = dnptrs; *cpp != NULL; cpp++)
                                ;
                        lpp = cpp;      /* end of list to search */
                }
        } else
                msg = NULL;
        for (c = *dn++; c != '\0'; ) {
                /* look to see if we can use pointers */
                if (msg != NULL) {
                        if ((l = dn_find(dn-1, msg, dnptrs, lpp)) >= 0) {
                                if (cp+1 >= eob)
                                        return (-1);
                                *cp++ = (l >> 8) | INDIR_MASK;
                                *cp++ = l % 256;
                                return (cp - comp_dn);
                        }
                        /* not found, save it */
                        if (lastdnptr != NULL && cpp < lastdnptr-1) {
                                *cpp++ = cp;
                                *cpp = NULL;
                        }
                }
                sp = cp++;      /* save ptr to length byte */
                do {
                        if (c == '.') {
                                c = *dn++;
                                break;
                        }
                        if (c == '\\') {
                                if ((c = *dn++) == '\0')
                                        break;
                        }
                        if (cp >= eob) {
                                if (msg != NULL)
                                        *lpp = NULL;
                                return (-1);
                        }
                        *cp++ = c;
                } while ((c = *dn++) != '\0');
                /* catch trailing '.'s but not '..' */
                if ((l = cp - sp - 1) == 0 && c == '\0') {
                        cp--;
                        break;
                }
                if (l <= 0 || l > MAXLABEL) {
                        if (msg != NULL)
                                *lpp = NULL;
                        return (-1);
                }
                *sp = l;
        }
        if (cp >= eob) {
                if (msg != NULL)
                        *lpp = NULL;
                return (-1);
        }
        *cp++ = '\0';
```

```
        return (cp - comp_dn);
}

/*
 * Skip over a compressed domain name. Return the size or -1.
 */
dn_skipname(comp_dn, eom)
        CONST u_char *comp_dn, *eom;
{
        register CONST u_char *cp;
        register int n;

        cp = comp_dn;
        while (cp < eom && (n = *cp++)) {
                /*
                 * check for indirection
                 */
                switch (n & INDIR_MASK) {
                case 0:         /* normal case, n == len */
                        cp += n;
                        continue;
                default:        /* illegal type */
                        return (-1);
                case INDIR_MASK:        /* indirection */
                        cp++;
                }
                break;
        }
        return (cp - comp_dn);
}

/*
 * Search for expanded name from a list of previously compressed names.
 * Return the offset from msg if found or -1.
 * dnptrs is the pointer to the first name on the list,
 * not the pointer to the start of the message.
 */
static int
dn_find(exp_dn, msg, dnptrs, lastdnptr)
        CONST u_char *exp_dn, *msg;
        u_char **dnptrs, **lastdnptr;
{
        CONST register u_char *dn, *cp;
        register u_char **cpp;
        register int n;
        CONST u_char *sp;

        for (cpp = dnptrs; cpp < lastdnptr; cpp++) {
                dn = exp_dn;
                sp = cp = *cpp;
                while (n = *cp++) {
                        /*
                         * check for indirection
                         */
                        switch (n & INDIR_MASK) {
                        case 0:         /* normal case, n == len */
                                while (--n >= 0) {
                                        if (*dn == '.')
                                                goto next;
                                        if (*dn == '\\')
```

```
                                                dn++;
                                        if (*dn++ != *cp++)
                                                goto next;
                                }
                                if ((n = *dn++) == '\0' && *cp == '\0')
                                        return (sp - msg);
                                if (n == '.')
                                        continue;
                                goto next;

                        default:        /* illegal type */
                                return (-1);

                        case INDIR_MASK:        /* indirection */
                                cp = msg + (((n & 0x3f) << 8) | *cp);
                        }
                }
                if (*dn == '\0')
                        return (sp - msg);
        next:   ;
        }
        return (-1);
}

/*
 * Routines to insert/extract short/long's. Must account for byte
 * order and non-alignment problems. This code at least has the
 * advantage of being portable.
 *
 * used by sendmail.
 */

u16_t
getshort(msgp)
        CONST u8_t *msgp;
{
        return ((msgp[0] << 8) | (msgp[1] << 0));
}

u32_t
getlong(msgp)
        CONST u8_t *msgp;
{
        return (  ((u32_t) msgp[0] << 24)
                | ((u32_t) msgp[1] << 16)
                | ((u32_t) msgp[2] <<  8)
                | ((u32_t) msgp[3] <<  0));
}


void
putshort(s, msgp)
        register U16_t s;
        register u8_t *msgp;
{

        msgp[1] = s;
        msgp[0] = s >> 8;
}
```

```
void
putlong(l, msgp)
        register u32_t l;
        register u8_t *msgp;
{

        msgp[3] = l;
        msgp[2] = (l >>= 8);
        msgp[1] = (l >>= 8);
        msgp[0] = l >> 8;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ip/res_init.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*-
 * Copyright (c) 1985, 1989 Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms are permitted provided
 * that: (1) source distributions retain this entire copyright notice and
 * comment, and (2) distributions including binaries display the following
 * acknowledgement:  `This product includes software developed by the
 * University of California, Berkeley and its contributors'' in the
 * documentation or other materials provided with the distribution and in
 * all advertising materials mentioning features or use of this software.
 * Neither the name of the University nor the names of its contributors may
 * be used to endorse or promote products derived from this software without
 * specific prior written permission.
 * THIS SOFTWARE IS PROVIDED `AS IS'' AND WITHOUT ANY EXPRESS OR IMPLIED
 * WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 */

#if defined(LIBC_SCCS) && !defined(lint)
static char sccsid[] = "@(#)res_init.c  6.14 (Berkeley) 6/27/90";
#endif /* LIBC_SCCS and not lint */

#if _MINIX
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#include <net/hton.h>
#include <net/gen/in.h>
#include <net/gen/inet.h>
#include <net/gen/nameser.h>
#include <net/gen/netdb.h>
#include <net/gen/resolv.h>
#include <net/gen/socket.h>

#define index(s,c) strchr(s,c)
#else
```

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdio.h>
#include <arpa/nameser.h>
#include <resolv.h>
#endif

/*
 * Resolver state
 */
struct state _res;

/*
 * Set up default settings.  If the configuration file exist, the values
 * there will have precedence.  Otherwise, the server address is set to
 * 127.0.0.1 (localhost) and the default domain name comes from gethostname().
 *
 * The configuration file should only be used if you want to redefine your
 * domain or run without a server on your machine.
 *
 * Return 0 if completes successfully, -1 on error
 */
int
res_init()
{
        register FILE *fp;
        register char *cp, **pp;
        register int n;
        char buf[BUFSIZ];
        int haveenv = 0;
        int havesearch = 0;
        struct servent* servent;
        u16_t nameserver_port;

        /* Resolver state default settings */
        _res.retrans = RES_TIMEOUT;     /* retransmition time interval */
        _res.retry = 4;                 /* number of times to retransmit */
        _res.options = RES_DEFAULT;     /* options flags */
        _res.nscount = 0;               /* number of name servers */

        servent= getservbyname("domain", NULL);
        if (!servent)
        {
                h_errno= NO_RECOVERY;
                return -1;
        }
        nameserver_port= servent->s_port;

        /* Allow user to override the local domain definition */
        if ((cp = getenv("LOCALDOMAIN")) != NULL) {
                (void)strncpy(_res.defdname, cp, sizeof(_res.defdname));
                haveenv++;
        }

        if ((fp = fopen(_PATH_RESCONF, "r")) != NULL) {
            /* read the config file */
            while (fgets(buf, sizeof(buf), fp) != NULL) {
                /* read default domain name */
                if (!strncmp(buf, "domain", sizeof("domain") - 1)) {
```

```
                        if (haveenv)        /* skip if have from environ */
                                continue;
                        cp = buf + sizeof("domain") - 1;
                        while (*cp == ' ' || *cp == '\t')
                                cp++;
                        if ((*cp == '\0') || (*cp == '\n'))
                                continue;
                        (void)strncpy(_res.defdname, cp, sizeof(_res.defdname) - 1);
                        if ((cp = index(_res.defdname, '\n')) != NULL)
                                *cp = '\0';
                        havesearch = 0;
                        continue;
                    }
                    /* set search list */
                    if (!strncmp(buf, "search", sizeof("search") - 1)) {
                        if (haveenv)        /* skip if have from environ */
                                continue;
                        cp = buf + sizeof("search") - 1;
                        while (*cp == ' ' || *cp == '\t')
                                cp++;
                        if ((*cp == '\0') || (*cp == '\n'))
                                continue;
                        (void)strncpy(_res.defdname, cp, sizeof(_res.defdname) - 1);
                        if ((cp = index(_res.defdname, '\n')) != NULL)
                                *cp = '\0';
                        /*
                         * Set search list to be blank-separated strings
                         * on rest of line.
                         */
                        cp = _res.defdname;
                        pp = _res.dnsrch;
                        *pp++ = cp;
                        for (n = 0; *cp && pp < _res.dnsrch + MAXDNSRCH; cp++) {
                                if (*cp == ' ' || *cp == '\t') {
                                        *cp = 0;
                                        n = 1;
                                } else if (n) {
                                        *pp++ = cp;
                                        n = 0;
                                }
                        }
                        /* null terminate last domain if there are excess */
                        while (*cp != '\0' && *cp != ' ' && *cp != '\t')
                                cp++;
                        *cp = '\0';
                        *pp++ = 0;
                        havesearch = 1;
                        continue;
                    }
                    /* read nameservers to query */
                    if (!strncmp(buf, "nameserver", sizeof("nameserver") - 1) &&
                       _res.nscount < MAXNS) {
                        cp = buf + sizeof("nameserver") - 1;
                        while (*cp == ' ' || *cp == '\t')
                                cp++;
                        if ((*cp == '\0') || (*cp == '\n'))
                                continue;
                        if (!inet_aton(cp, &_res.nsaddr_list[_res.nscount]))
                                continue;
                        _res.nsport_list[_res.nscount]= nameserver_port;
```

```
                        _res.nscount++;
                        continue;
                    }
                }
                (void) fclose(fp);
            }
            if (_res.nscount == 0) {
                    /* "localhost" is the default nameserver. */
                    _res.nsaddr_list[0]= HTONL(0x7F000001);
                    _res.nsport_list[0]= nameserver_port;
                    _res.nscount= 1;
            }
            if (_res.defdname[0] == 0) {
                    if (gethostname(buf, sizeof(_res.defdname)) == 0 &&
                       (cp = index(buf, '.')))
                            (void)strcpy(_res.defdname, cp + 1);
            }

            /* find components of local domain that might be searched */
            if (havesearch == 0) {
                    pp = _res.dnsrch;
                    *pp++ = _res.defdname;
                    for (cp = _res.defdname, n = 0; *cp; cp++)
                            if (*cp == '.')
                                    n++;
                    cp = _res.defdname;
                    for (; n >= LOCALDOMAINPARTS && pp < _res.dnsrch + MAXDFLSRCH;
                        n--) {
                            cp = index(cp, '.');
                            *pp++ = ++cp;
                    }
                    *pp++ = 0;
            }
            _res.options |= RES_INIT;
            return (0);
    }
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ip/res_mkquery.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * Copyright (c) 1985 Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms are permitted
 * provided that: (1) source distributions retain this entire copyright
 * notice and comment, and (2) distributions including binaries display
 * the following acknowledgement:  ``This product includes software
 * developed by the University of California, Berkeley and its contributors''
 * in the documentation or other materials provided with the distribution
 * and in all advertising materials mentioning features or use of this
 * software. Neither the name of the University nor the names of its
 * contributors may be used to endorse or promote products derived
 * from this software without specific prior written permission.
 * THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY EXPRESS OR
```

```
 * IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 */

#if defined(LIBC_SCCS) && !defined(lint)
static char sccsid[] = "@(#)res_mkquery.c       6.12 (Berkeley) 6/1/90";
#endif /* LIBC_SCCS and not lint */

#if _MINIX
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <net/hton.h>
#include <net/gen/in.h>
#include <net/gen/nameser.h>
#include <net/gen/resolv.h>

typedef u16_t u_short;
typedef unsigned u_int;
typedef u32_t u_long;

#define bzero(b,l) memset(b,0,l)
#define bcopy(s,d,l) memcpy(d,s,l)

#define putshort __putshort
#define putlong __putlong
#else
#include <stdio.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/nameser.h>
#include <resolv.h>
#endif

#ifdef __STDC__
#define _CONST  const
#else
#define _CONST
#endif

/*
 * Form all types of queries.
 * Returns the size of the result or -1.
 */
res_mkquery(op, dname, class, type, data, datalen, newrr, buf, buflen)
        int op;                 /* opcode of query */
        _CONST char *dname;     /* domain name */
        int class, type;        /* class and type of query */
        _CONST char *data;      /* resource record data */
        int datalen;            /* length of data */
        _CONST struct rrec *newrr; /* new rr for modify or append */
        char *buf;              /* buffer to put query */
        int buflen;             /* size of buffer */
{
        register dns_hdr_t *hp;
        register char *cp;
        register int n;
        char *dnptrs[10], **dpp, **lastdnptr;
```

```

#ifdef DEBUG
        if (_res.options & RES_DEBUG)
                printf("res_mkquery(%d, %s, %d, %d)\n", op, dname, class, type);
#endif /* DEBUG */
        /*
         * Initialize header fields.
         */
        if ((buf == NULL) || (buflen < sizeof(dns_hdr_t)))
                return(-1);
        bzero(buf, sizeof(dns_hdr_t));
        hp = (dns_hdr_t *) buf;
        hp->dh_id = htons(++_res.id);
        hp->dh_flag1= 0;
        hp->dh_flag2= 0;
        hp->dh_flag1 |= (op << 3) & DHF_OPCODE;
        hp->dh_flag2 |= ((_res.options & RES_PRIMARY) != 0 ? 1 : 0) << 6;
        hp->dh_flag1 |= (_res.options & RES_RECURSE) != 0 ? 1 : 0;
        hp->dh_flag2 |= NOERROR & DHF_RCODE;
        cp = buf + sizeof(dns_hdr_t);
        buflen -= sizeof(dns_hdr_t);
        dpp = dnptrs;
        *dpp++ = buf;
        *dpp++ = NULL;
        lastdnptr = dnptrs + sizeof(dnptrs)/sizeof(dnptrs[0]);
        /*
         * perform opcode specific processing
         */
        switch (op) {
        case QUERY:
                if ((buflen -= QFIXEDSZ) < 0)
                        return(-1);
                if ((n = dn_comp((u8_t *)dname, (u8_t *)cp, buflen,
                        (u8_t **)dnptrs, (u8_t **)lastdnptr)) < 0)
                        return (-1);
                cp += n;
                buflen -= n;
                putshort(type, (u8_t *)cp);
                cp += sizeof(u_short);
                putshort(class, (u8_t *)cp);
                cp += sizeof(u_short);
                hp->dh_qdcount = htons(1);
                if (op == QUERY || data == NULL)
                        break;
                /*
                 * Make an additional record for completion domain.
                 */
                buflen -= RRFIXEDSZ;
                if ((n = dn_comp((u8_t *)data, (u8_t *)cp, buflen,
                        (u8_t **)dnptrs, (u8_t **)lastdnptr)) < 0)
                        return (-1);
                cp += n;
                buflen -= n;
                putshort(T_NULL, (u8_t *)cp);
                cp += sizeof(u_short);
                putshort(class, (u8_t *)cp);
                cp += sizeof(u_short);
                putlong(0, (u8_t *)cp);
                cp += sizeof(u_long);
                putshort(0, (u8_t *)cp);
```

```
                        cp += sizeof(u_short);
                        hp->dh_arcount = htons(1);
                        break;

                case IQUERY:
                        /*
                         * Initialize answer section
                         */
                        if (buflen < 1 + RRFIXEDSZ + datalen)
                                return (-1);
                        *cp++ = '\0';   /* no domain name */
                        putshort(type, (u8_t *)cp);
                        cp += sizeof(u_short);
                        putshort(class, (u8_t *)cp);
                        cp += sizeof(u_short);
                        putlong(0, (u8_t *)cp);
                        cp += sizeof(u_long);
                        putshort(datalen, (u8_t *)cp);
                        cp += sizeof(u_short);
                        if (datalen) {
                                bcopy(data, cp, datalen);
                                cp += datalen;
                        }
                        hp->dh_ancount = htons(1);
                        break;

        #ifdef ALLOW_UPDATES
                /*
                 * For UPDATEM/UPDATEMA, do UPDATED/UPDATEDA followed by UPDATEA
                 * (Record to be modified is followed by its replacement in msg.)
                 */
                case UPDATEM:
                case UPDATEMA:

                case UPDATED:
                        /*
                         * The res code for UPDATED and UPDATEDA is the same; user
                         * calls them differently: specifies data for UPDATED; server
                         * ignores data if specified for UPDATEDA.
                         */
                case UPDATEDA:
                        buflen -= RRFIXEDSZ + datalen;
                        if ((n = dn_comp(dname, cp, buflen, dnptrs, lastdnptr)) < 0)
                                return (-1);
                        cp += n;
                        putshort(type, cp);
                        cp += sizeof(u_short);
                        putshort(class, cp);
                        cp += sizeof(u_short);
                        putlong(0, cp);
                        cp += sizeof(u_long);
                        putshort(datalen, cp);
                        cp += sizeof(u_short);
                        if (datalen) {
                                bcopy(data, cp, datalen);
                                cp += datalen;
                        }
                        if ( (op == UPDATED) || (op == UPDATEDA) ) {
                                hp->ancount = htons(0);
                                break;
```

```
                }
                /* Else UPDATEM/UPDATEMA, so drop into code for UPDATEA */

        case UPDATEA:   /* Add new resource record */
                buflen -= RRFIXEDSZ + datalen;
                if ((n = dn_comp(dname, cp, buflen, dnptrs, lastdnptr)) < 0)
                        return (-1);
                cp += n;
                putshort(newrr->r_type, cp);
                cp += sizeof(u_short);
                putshort(newrr->r_class, cp);
                cp += sizeof(u_short);
                putlong(0, cp);
                cp += sizeof(u_long);
                putshort(newrr->r_size, cp);
                cp += sizeof(u_short);
                if (newrr->r_size) {
                        bcopy(newrr->r_data, cp, newrr->r_size);
                        cp += newrr->r_size;
                }
                hp->ancount = htons(0);
                break;

#endif /* ALLOW_UPDATES */
        }
        return (cp - buf);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ip/res_query.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * Copyright (c) 1988 Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms are permitted
 * provided that: (1) source distributions retain this entire copyright
 * notice and comment, and (2) distributions including binaries display
 * the following acknowledgement:  ‘This product includes software
 * developed by the University of California, Berkeley and its contributors’’
 * in the documentation or other materials provided with the distribution
 * and in all advertising materials mentioning features or use of this
 * software. Neither the name of the University nor the names of its
 * contributors may be used to endorse or promote products derived
 * from this software without specific prior written permission.
 * THIS SOFTWARE IS PROVIDED ‘AS IS’’ AND WITHOUT ANY EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
 */

#if defined(LIBC_SCCS) && !defined(lint)
static char sccsid[] = "@(#)res_query.c 5.7 (Berkeley) 6/1/90";
#endif /* LIBC_SCCS and not lint */

#if _MINIX
#include <sys/types.h>
```

```
#include <ctype.h>
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <net/hton.h>
#include <net/gen/in.h>
#include <net/gen/nameser.h>
#include <net/gen/netdb.h>
#include <net/gen/resolv.h>

typedef u8_t u_char;

#define bcopy(s,d,l) memcpy(d,s,l)

#define hostalias __hostalias
#else
#include <sys/param.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <ctype.h>
#include <netdb.h>
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <arpa/inet.h>
#include <arpa/nameser.h>
#include <resolv.h>

extern int errno;
#endif

#if __STDC__
#define CONST   const
#else
#define CONST
#endif

#if PACKETSZ > 1024
#define MAXPACKET       PACKETSZ
#else
#define MAXPACKET       1024
#endif

int h_errno;

/*
 * Formulate a normal query, send, and await answer.
 * Returned answer is placed in supplied buffer "answer".
 * Perform preliminary check of answer, returning success only
 * if no error is indicated and the answer count is nonzero.
 * Return the size of the response on success, -1 on error.
 * Error number is left in h_errno.
 * Caller must parse answer and determine whether it answers the question.
 */
int
res_query(name, class, type, answer, anslen)
        char *name;             /* domain name */
        int class, type;        /* class and type of query */
```

```
        u_char *answer;         /* buffer to put answer */
        int anslen;             /* size of answer buffer */
{
        char buf[MAXPACKET];
        dns_hdr_t *hp;
        int n;

        if ((_res.options & RES_INIT) == 0 && res_init() == -1)
                return (-1);
#ifdef DEBUG
        if (_res.options & RES_DEBUG)
                printf("res_query(%s, %d, %d)\n", name, class, type);
#endif
        n = res_mkquery(QUERY, name, class, type, (char *)NULL, 0, NULL,
            buf, sizeof(buf));

        if (n <= 0) {
#ifdef DEBUG
                if (_res.options & RES_DEBUG)
                        printf("res_query: mkquery failed\n");
#endif
                h_errno = NO_RECOVERY;
                return (n);
        }
        n = res_send(buf, n, (char *)answer, anslen);
        if (n < 0) {
#ifdef DEBUG
                if (_res.options & RES_DEBUG)
                        printf("res_query: send error(%d)\n", errno);
#endif
                h_errno = TRY_AGAIN;
                return(n);
        }

        hp = (dns_hdr_t *) answer;
        if ((hp->dh_flag2 & DHF_RCODE) != NOERROR ||
                ntohs(hp->dh_ancount) == 0) {
#ifdef DEBUG
                if (_res.options & RES_DEBUG)
                        printf("rcode = %d, ancount=%d\n",
                                hp->dh_flag2 & DHF_RCODE,
                            ntohs(hp->dh_ancount));
#endif
                switch (hp->dh_flag2 & DHF_RCODE) {
                        case NXDOMAIN:
                                h_errno = HOST_NOT_FOUND;
                                break;
                        case SERVFAIL:
                                h_errno = TRY_AGAIN;
                                break;
                        case NOERROR:
                                h_errno = NO_DATA;
                                break;
                        case FORMERR:
                        case NOTIMP:
                        case REFUSED:
                        default:
                                h_errno = NO_RECOVERY;
                                break;
                }
```

```
                        return (-1);
                }
                return(n);
        }

        /*
         * Formulate a normal query, send, and retrieve answer in supplied buffer.
         * Return the size of the response on success, -1 on error.
         * If enabled, implement search rules until answer or unrecoverable failure
         * is detected.  Error number is left in h_errno.
         * Only useful for queries in the same name hierarchy as the local host
         * (not, for example, for host address-to-name lookups in domain in-addr.arpa).
         */
        res_search(name, class, type, answer, anslen)
                char *name;             /* domain name */
                int class, type;        /* class and type of query */
                u_char *answer;         /* buffer to put answer */
                int anslen;             /* size of answer */
        {
                register char *cp, **domain;
                int n, ret, got_nodata = 0;

                if ((_res.options & RES_INIT) == 0 && res_init() == -1)
                        return (-1);

                errno = 0;
                h_errno = HOST_NOT_FOUND;               /* default, if we never query */
                for (cp = name, n = 0; *cp; cp++)
                        if (*cp == '.')
                                n++;
                if (n == 0 && (cp = hostalias(name)))
                        return (res_query(cp, class, type, answer, anslen));

                /*
                 * We do at least one level of search if
                 *      - there is no dot and RES_DEFNAME is set, or
                 *      - there is at least one dot, there is no trailing dot,
                 *        and RES_DNSRCH is set.
                 */
                if ((n == 0 && _res.options & RES_DEFNAMES) ||
                   (n != 0 && *--cp != '.' && _res.options & RES_DNSRCH))
                     for (domain = _res.dnsrch; *domain; domain++) {
                        ret = res_querydomain(name, *domain, class, type,
                            answer, anslen);
                        if (ret > 0)
                                return (ret);
                        /*
                         * If no server present, give up.
                         * If name isn't found in this domain,
                         * keep trying higher domains in the search list
                         * (if that's enabled).
                         * On a NO_DATA error, keep trying, otherwise
                         * a wildcard entry of another type could keep us
                         * from finding this entry higher in the domain.
                         * If we get some other error (negative answer or
                         * server failure), then stop searching up,
                         * but try the input name below in case it's fully-qualified.
                         */
                        if (errno == ECONNREFUSED) {
                                h_errno = TRY_AGAIN;
```

```
                        return (-1);
                }
                if (h_errno == NO_DATA)
                        got_nodata++;
                if ((h_errno != HOST_NOT_FOUND && h_errno != NO_DATA) ||
                    (_res.options & RES_DNSRCH) == 0)
                        break;
        }
        /*
         * If the search/default failed, try the name as fully-qualified,
         * but only if it contained at least one dot (even trailing).
         * This is purely a heuristic; we assume that any reasonable query
         * about a top-level domain (for servers, SOA, etc) will not use
         * res_search.
         */
        if (n && (ret = res_querydomain(name, (char *)NULL, class, type,
            answer, anslen)) > 0)
                return (ret);
        if (got_nodata)
                h_errno = NO_DATA;
        return (-1);
}

/*
 * Perform a call on res_query on the concatenation of name and domain,
 * removing a trailing dot from name if domain is NULL.
 */
int
res_querydomain(name, domain, class, type, answer, anslen)
        char *name, *domain;
        int class, type;        /* class and type of query */
        u_char *answer;         /* buffer to put answer */
        int anslen;             /* size of answer */
{
        char nbuf[2*MAXDNAME+2];
        char *longname = nbuf;
        int n;

#ifdef DEBUG
        if (_res.options & RES_DEBUG)
                printf("res_querydomain(%s, %s, %d, %d)\n",
                    name, domain, class, type);
#endif
        if (domain == NULL) {
                /*
                 * Check for trailing '.';
                 * copy without '.' if present.
                 */
                n = strlen(name) - 1;
                if (name[n] == '.' && n < sizeof(nbuf) - 1) {
                        bcopy(name, nbuf, n);
                        nbuf[n] = '\0';
                } else
                        longname = name;
        } else
                (void)sprintf(nbuf, "%.*s.%.*s",
                    MAXDNAME, name, MAXDNAME, domain);

        return (res_query(longname, class, type, answer, anslen));
}
```

```
char *
hostalias(name)
        register CONST char *name;
{
        register char *C1, *C2;
        FILE *fp;
        char *file;
        char buf[BUFSIZ];
        static char abuf[MAXDNAME];

        file = getenv("HOSTALIASES");
        if (file == NULL || (fp = fopen(file, "r")) == NULL)
                return (NULL);
        buf[sizeof(buf) - 1] = '\0';
        while (fgets(buf, sizeof(buf), fp)) {
                for (C1 = buf; *C1 && !isspace(*C1); ++C1);
                if (!*C1)
                        break;
                *C1 = '\0';
                if (!strcasecmp(buf, name)) {
                        while (isspace(*++C1));
                        if (!*C1)
                                break;
                        for (C2 = C1 + 1; *C2 && !isspace(*C2); ++C2);
                        abuf[sizeof(abuf) - 1] = *C2 = '\0';
                        (void)strncpy(abuf, C1, sizeof(abuf) - 1);
                        fclose(fp);
                        return (abuf);
                }
        }
        fclose(fp);
        return (NULL);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/ip/res_send.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * Copyright (c) 1985, 1989 Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
```

```
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS `AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */

#if defined(LIBC_SCCS) && !defined(lint)
static char sccsid[] = "@(#)res_send.c  6.27 (Berkeley) 2/24/91";
#endif /* LIBC_SCCS and not lint */

/*
 * Send query to name server and wait for reply.
 */

#if !_MINIX
#include <sys/param.h>
#include <sys/time.h>
#include <sys/socket.h>
#include <sys/uio.h>
#include <netinet/in.h>
#include <arpa/nameser.h>
#include <stdio.h>
#include <errno.h>
#include <resolv.h>
#include <unistd.h>
#include <string.h>

#else /* _MINIX */

#include <sys/types.h>
#include <sys/ioctl.h>
#include <sys/stat.h>
#include <assert.h>
#include <errno.h>
#include <fcntl.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#include <net/hton.h>

#include <net/netlib.h>
#include <net/gen/in.h>
#include <net/gen/inet.h>
#include <net/gen/netdb.h>
#include <net/gen/nameser.h>
#include <net/gen/resolv.h>
#include <net/gen/tcp.h>
#include <net/gen/tcp_io.h>
#include <net/gen/udp.h>
#include <net/gen/udp_hdr.h>
```

```
#include <net/gen/udp_io.h>

typedef u16_t u_short;

static int tcp_connect _ARGS(( ipaddr_t host, Tcpport_t port, int *terrno ));
static int tcpip_writeall _ARGS(( int fd, const char *buf, size_t siz ));
static int udp_connect _ARGS(( void ));
static int udp_sendto _ARGS(( int fd, const char *buf, unsigned buflen,
                                ipaddr_t addr, Udpport_t port ));
static int udp_receive _ARGS(( int fd, char *buf, unsigned buflen,
                                time_t timeout ));
static void alarm_handler _ARGS(( int sig ));

#endif /* !_MINIX */

static int s = -1;      /* socket used for communications */
#if !_MINIX
static struct sockaddr no_addr;

#ifndef FD_SET
#define NFDBITS         32
#define FD_SETSIZE      32
#define FD_SET(n, p)    ((p)->fds_bits[(n)/NFDBITS] |= (1 << ((n) % NFDBITS)))
#define FD_CLR(n, p)    ((p)->fds_bits[(n)/NFDBITS] &= ~(1 << ((n) % NFDBITS)))
#define FD_ISSET(n, p)  ((p)->fds_bits[(n)/NFDBITS] & (1 << ((n) % NFDBITS)))
#define FD_ZERO(p)      bzero((char *)(p), sizeof(*(p)))
#endif /* FD_SET */
#endif /* _MINIX */

res_send(buf, buflen, answer, anslen)
        const char *buf;
        int buflen;
        char *answer;
        int anslen;
{
        register int n;
        int try, v_circuit, resplen, ns;
        int gotsomewhere = 0, connected = 0;
        int connreset = 0;
#if !_MINIX
        u_short id, len;
#else /* _MINIX */
        u16_t id, len;
#endif /* !_MINIX */
        char *cp;
#if !_MINIX
        fd_set dsmask;
        struct timeval timeout;
        HEADER *hp = (HEADER *) buf;
        HEADER *anhp = (HEADER *) answer;
        struct iovec iov[2];
#else /* _MINIX */
        time_t timeout;
        dns_hdr_t *hp = (dns_hdr_t *) buf;
        dns_hdr_t *anhp = (dns_hdr_t *) answer;
#endif /* !_MINIX */
        int terrno = ETIMEDOUT;
        char junk[512];

#ifdef DEBUG
```

```
        if (_res.options & RES_DEBUG) {
                printf("res_send()\n");
                __p_query(buf);
        }
#endif /* DEBUG */
        if (!(_res.options & RES_INIT))
                if (res_init() == -1) {
                        return(-1);
                }

        v_circuit = (_res.options & RES_USEVC) || buflen > PACKETSZ;
#if !_MINIX
        id = hp->id;
#else /* _MINIX */
        id = hp->dh_id;
#endif /* !_MINIX */
        /*
         * Send request, RETRY times, or until successful
         */
        for (try = 0; try < _res.retry; try++) {
            for (ns = 0; ns < _res.nscount; ns++) {
#ifdef DEBUG
#if !_MINIX
                if (_res.options & RES_DEBUG)
                        printf("Querying server (# %d) address = %s\n", ns+1,
                              inet_ntoa(_res.nsaddr_list[ns].sin_addr));
#else /* _MINIX */
                if (_res.options & RES_DEBUG)
                        printf("Querying server (# %d) address = %s\n", ns+1,
                              inet_ntoa(_res.nsaddr_list[ns]));
#endif /* !_MINIX */
#endif /* DEBUG */
        usevc:
                if (v_circuit) {
#if !_MINIX
                        int truncated = 0;

                        /*
                         * Use virtual circuit;
                         * at most one attempt per server.
                         */
                        try = _res.retry;
                        if (s < 0) {
                                s = socket(AF_INET, SOCK_STREAM, 0);
                                if (s < 0) {
                                        terrno = errno;
#ifdef DEBUG
                                        if (_res.options & RES_DEBUG)
                                            perror("socket (vc) failed");
#endif /* DEBUG */
                                        continue;
                                }
                                if (connect(s,
                                    (struct sockaddr *)&(_res.nsaddr_list[ns]),
                                    sizeof(struct sockaddr)) < 0) {
                                        terrno = errno;
#ifdef DEBUG
                                        if (_res.options & RES_DEBUG)
                                            perror("connect failed");
#endif /* DEBUG */
```

```
                                                (void) close(s);
                                                s = -1;
                                                continue;
                                        }
                                }
                                /*
                                 * Send length & message
                                 */
                                len = htons((u_short)buflen);
                                iov[0].iov_base = (caddr_t)&len;
                                iov[0].iov_len = sizeof(len);
                                iov[1].iov_base = (char *)buf;
                                iov[1].iov_len = buflen;
                                if (writev(s, iov, 2) != sizeof(len) + buflen) {
                                        terrno = errno;
#ifdef DEBUG
                                        if (_res.options & RES_DEBUG)
                                                perror("write failed");
#endif /* DEBUG */
                                        (void) close(s);
                                        s = -1;
                                        continue;
                                }
                                /*
                                 * Receive length & response
                                 */
                                cp = answer;
                                len = sizeof(short);
                                while (len != 0 &&
                                    (n = read(s, (char *)cp, (int)len)) > 0) {
                                        cp += n;
                                        len -= n;
                                }
                                if (n <= 0) {
                                        terrno = errno;
#ifdef DEBUG
                                        if (_res.options & RES_DEBUG)
                                                perror("read failed");
#endif /* DEBUG */
                                        (void) close(s);
                                        s = -1;
                                        /*
                                         * A long running process might get its TCP
                                         * connection reset if the remote server was
                                         * restarted.  Requery the server instead of
                                         * trying a new one.  When there is only one
                                         * server, this means that a query might work
                                         * instead of failing.  We only allow one reset
                                         * per query to prevent looping.
                                         */
                                        if (terrno == ECONNRESET && !connreset) {
                                                connreset = 1;
                                                ns--;
                                        }
                                        continue;
                                }
                                cp = answer;
                                if ((resplen = ntohs(*(u_short *)cp)) > anslen) {
#ifdef DEBUG
                                        if (_res.options & RES_DEBUG)
```

```
                                        fprintf(stderr, "response truncated\n");
#endif /* DEBUG */
                                len = anslen;
                                truncated = 1;
                        } else
                                len = resplen;
                        while (len != 0 &&
                           (n = read(s, (char *)cp, (int)len)) > 0) {
                                cp += n;
                                len -= n;
                        }
                        if (n <= 0) {
                                terrno = errno;
#ifdef DEBUG
                                if (_res.options & RES_DEBUG)
                                        perror("read failed");
#endif /* DEBUG */
                                (void) close(s);
                                s = -1;
                                continue;
                        }
                        if (truncated) {
                                /*
                                 * Flush rest of answer
                                 * so connection stays in synch.
                                 */
                                anhp->tc = 1;
                                len = resplen - anslen;
                                while (len != 0) {
                                        n = (len > sizeof(junk) ?
                                            sizeof(junk) : len);
                                        if ((n = read(s, junk, n)) > 0)
                                                len -= n;
                                        else
                                                break;
                                }
                        }
#else /* _MINIX */
                        int truncated = 0;
                        int nbytes;

                        /*
                         * Use virtual circuit;
                         * at most one attempt per server.
                         */
                        try = _res.retry;
                        if (s < 0)
                        {
                                s= tcp_connect(_res.nsaddr_list[ns],
                                        _res.nsport_list[ns], &terrno);
                                if (s == -1)
                                        continue;
                        }
                        /*
                         * Send length & message
                         */
                        len = htons((u_short)buflen);
                        nbytes= tcpip_writeall(s, (char *)&len,
                                sizeof(len));
                        if (nbytes != sizeof(len))
```

```
			{
				terrno= errno;
#ifdef DEBUG
				if (_res.options & RES_DEBUG)
					fprintf(stderr, "write failed: %s\n",
					strerror(terrno));
#endif /* DEBUG */
				close(s);
				s= -1;
				continue;
			}
			nbytes= tcpip_writeall(s, buf, buflen);
			if (nbytes != buflen)
			{
				terrno= errno;
#ifdef DEBUG
				if (_res.options & RES_DEBUG)
					fprintf(stderr, "write failed: %s\n",
					strerror(terrno));
#endif /* DEBUG */
				close(s);
				s= -1;
				continue;
			}
			/*
			 * Receive length & response
			 */
			cp = answer;
			len = sizeof(short);
			while (len != 0)
			{
				n = read(s, (char *)cp, (int)len);
				if (n <= 0)
					break;
				cp += n;
				assert(len >= n);
				len -= n;
			}
			if (len) {
				terrno = errno;
#ifdef DEBUG
				if (_res.options & RES_DEBUG)
					fprintf(stderr, "read failed: %s\n",
						strerror(terrno));
#endif /* DEBUG */
				close(s);
				s= -1;
				/*
				 * A long running process might get its TCP
				 * connection reset if the remote server was
				 * restarted.  Requery the server instead of
				 * trying a new one.  When there is only one
				 * server, this means that a query might work
				 * instead of failing.  We only allow one reset
				 * per query to prevent looping.
				 */
				if (terrno == ECONNRESET && !connreset) {
					connreset = 1;
					ns--;
				}
```

```
				continue;
			}
			cp = answer;
			if ((resplen = ntohs(*(u_short *)cp)) > anslen) {
#ifdef DEBUG
				if (_res.options & RES_DEBUG)
					fprintf(stderr, "response truncated\n");
#endif /* DEBUG */
				len = anslen;
				truncated = 1;
			} else
				len = resplen;
			while (len != 0)
			{
				n= read(s, (char *)cp, (int)len);
				if (n <= 0)
					break;
				cp += n;
				assert(len >= n);
				len -= n;
			}
			if (len) {
				terrno = errno;
#ifdef DEBUG
				if (_res.options & RES_DEBUG)
					fprintf(stderr, "read failed: %s\n",
						strerror(terrno));
#endif /* DEBUG */
				close(s);
				s= -1;
				continue;
			}
			if (truncated) {
				/*
				 * Flush rest of answer
				 * so connection stays in synch.
				 */
				anhp->dh_flag1 |= DHF_TC;
				len = resplen - anslen;
				while (len != 0) {
					n = (len > sizeof(junk) ?
					    sizeof(junk) : len);
					n = read(s, junk, n);
					if (n <= 0)
					{
						assert(len >= n);
						len -= n;
					}
					else
						break;
				}
			}
#endif /* _MINIX */
		} else {
#if !_MINIX
			/*
			 * Use datagrams.
			 */
			if (s < 0) {
				s = socket(AF_INET, SOCK_DGRAM, 0);
```

```
				if (s < 0) {
					terrno = errno;
#ifdef DEBUG
					if (_res.options & RES_DEBUG)
					    perror("socket (dg) failed");
#endif /* DEBUG */
					continue;
				}
			}
#if	BSD >= 43
			/*
			 * I’m tired of answering this question, so:
			 * On a 4.3BSD+ machine (client and server,
			 * actually), sending to a nameserver datagram
			 * port with no nameserver will cause an
			 * ICMP port unreachable message to be returned.
			 * If our datagram socket is "connected" to the
			 * server, we get an ECONNREFUSED error on the next
			 * socket operation, and select returns if the
			 * error message is received.  We can thus detect
			 * the absence of a nameserver without timing out.
			 * If we have sent queries to at least two servers,
			 * however, we don’t want to remain connected,
			 * as we wish to receive answers from the first
			 * server to respond.
			 */
			if (_res.nscount == 1 || (try == 0 && ns == 0)) {
				/*
				 * Don’t use connect if we might
				 * still receive a response
				 * from another server.
				 */
				if (connected == 0) {
			if (connect(s, (struct sockaddr *)&_res.nsaddr_list[ns],
					    sizeof(struct sockaddr)) < 0) {
#ifdef DEBUG
						if (_res.options & RES_DEBUG)
							perror("connect");
#endif /* DEBUG */
						continue;
					}
					connected = 1;
				}
				if (send(s, buf, buflen, 0) != buflen) {
#ifdef DEBUG
					if (_res.options & RES_DEBUG)
						perror("send");
#endif /* DEBUG */
					continue;
				}
			} else {
				/*
				 * Disconnect if we want to listen
				 * for responses from more than one server.
				 */
				if (connected) {
					(void) connect(s, &no_addr,
					    sizeof(no_addr));
					connected = 0;
				}
```

```
#endif /* BSD */
                                if (sendto(s, buf, buflen, 0,
                                    (struct sockaddr *)&_res.nsaddr_list[ns],
                                    sizeof(struct sockaddr)) != buflen) {
#ifdef DEBUG
                                        if (_res.options & RES_DEBUG)
                                                perror("sendto");
#endif /* DEBUG */
                                        continue;
                                }
#if     BSD >= 43
                        }
#endif /* BSD */

                        /*
                         * Wait for reply
                         */
                        timeout.tv_sec = (_res.retrans << try);
                        if (try > 0)
                                timeout.tv_sec /= _res.nscount;
                        if (timeout.tv_sec <= 0)
                                timeout.tv_sec = 1;
                        timeout.tv_usec = 0;
wait:
                        FD_ZERO(&dsmask);
                        FD_SET(s, &dsmask);
                        n = select(s+1, &dsmask, (fd_set *)NULL,
                                (fd_set *)NULL, &timeout);
                        if (n < 0) {
#ifdef DEBUG
                                if (_res.options & RES_DEBUG)
                                        perror("select");
#endif /* DEBUG */
                                continue;
                        }
                        if (n == 0) {
                                /*
                                 * timeout
                                 */
#ifdef DEBUG
                                if (_res.options & RES_DEBUG)
                                        printf("timeout\n");
#endif /* DEBUG */
#if BSD >= 43
                                gotsomewhere = 1;
#endif
                                continue;
                        }
                        if ((resplen = recv(s, answer, anslen, 0)) <= 0) {
#ifdef DEBUG
                                if (_res.options & RES_DEBUG)
                                        perror("recvfrom");
#endif /* DEBUG */
                                continue;
                        }
                        gotsomewhere = 1;
                        if (id != anhp->id) {
                                /*
                                 * response from old query, ignore it
                                 */
```

```
#ifdef DEBUG
                                if (_res.options & RES_DEBUG) {
                                        printf("old answer:\n");
                                        __p_query(answer);
                                }
#endif /* DEBUG */
                                goto wait;
                        }
                        if (!(_res.options & RES_IGNTC) && anhp->tc) {
                                /*
                                 * get rest of answer;
                                 * use TCP with same server.
                                 */
#ifdef DEBUG
                                if (_res.options & RES_DEBUG)
                                        printf("truncated answer\n");
#endif /* DEBUG */
                                (void) close(s);
                                s = -1;
                                v_circuit = 1;
                                goto usevc;
                        }
#else /* _MINIX */
                        /*
                         * Use datagrams.
                         */
                        if (s < 0) {
                                s = udp_connect();
                                if (s < 0) {
                                        terrno = errno;
#ifdef DEBUG
                                        if (_res.options & RES_DEBUG)
                                            perror("udp_connect failed");
#endif /* DEBUG */
                                        continue;
                                }
                        }
                        if (udp_sendto(s, buf, buflen, _res.nsaddr_list[ns],
                                _res.nsport_list[ns]) != buflen) {
#ifdef DEBUG
                                if (_res.options & RES_DEBUG)
                                        perror("sendto");
#endif /* DEBUG */
                                continue;
                        }

                        /*
                         * Wait for reply
                         */
                        timeout= (_res.retrans << try);
                        if (try > 0)
                                timeout /= _res.nscount;
                        if (timeout <= 0)
                                timeout= 1;
wait:
                        if ((resplen= udp_receive(s, answer, anslen, timeout))
                                == -1)
                        {
                                if (errno == EINTR)
                                {
```

```
                                        /*
                                         * timeout
                                         */
#ifdef DEBUG
                                                if (_res.options & RES_DEBUG)
                                                        printf("timeout\n");
#endif /* DEBUG */
                                                gotsomewhere = 1;
                                        }
                                        else
                                        {
#ifdef DEBUG
                                        if (_res.options & RES_DEBUG)
                                                perror("udp_receive");
#endif /* DEBUG */
                                        }
                                        continue;
                                }
                                gotsomewhere = 1;
                                if (id != anhp->dh_id) {
                                        /*
                                         * response from old query, ignore it
                                         */
#ifdef DEBUG
                                        if (_res.options & RES_DEBUG) {
                                                printf("old answer:\n");
                                                __p_query(answer);
                                        }
#endif /* DEBUG */
                                        goto wait;
                                }
                                if (!(_res.options & RES_IGNTC) &&
                                        (anhp->dh_flag1 & DHF_TC)) {
                                        /*
                                         * get rest of answer;
                                         * use TCP with same server.
                                         */
#ifdef DEBUG
                                        if (_res.options & RES_DEBUG)
                                                printf("truncated answer\n");
#endif /* DEBUG */
                                        (void) close(s);
                                        s = -1;
                                        v_circuit = 1;
                                        goto usevc;
                                }
#endif /* !_MINIX */
                        }
#ifdef DEBUG
                        if (_res.options & RES_DEBUG) {
                                printf("got answer:\n");
                                __p_query(answer);
                        }
#endif /* DEBUG */
                        /*
                         * If using virtual circuits, we assume that the first server
                         * is preferred * over the rest (i.e. it is on the local
                         * machine) and only keep that one open.
                         * If we have temporarily opened a virtual circuit,
                         * or if we haven't been asked to keep a socket open,
```

```
                         * close the socket.
                         */
                        if ((v_circuit &&
                            ((_res.options & RES_USEVC) == 0 || ns != 0)) ||
                            (_res.options & RES_STAYOPEN) == 0) {
                                (void) close(s);
                                s = -1;
                        }
                        return (resplen);
                }
        }
        if (s >= 0) {
                (void) close(s);
                s = -1;
        }
        if (v_circuit == 0)
                if (gotsomewhere == 0)
                        errno = ECONNREFUSED;   /* no nameservers found */
                else
                        errno = ETIMEDOUT;      /* no answer obtained */
        else
                errno = terrno;
        return (-1);
}

/*
 * This routine is for closing the socket if a virtual circuit is used and
 * the program wants to close it.  This provides support for endhostent()
 * which expects to close the socket.
 *
 * This routine is not expected to be user visible.
 */
void
_res_close()
{
        if (s != -1) {
                (void) close(s);
                s = -1;
        }
}

#if _MINIX
static int tcp_connect(host, port, terrno)
ipaddr_t host;
tcpport_t port;
int *terrno;
{
        char *dev_name;
        int fd;
        int error;
        nwio_tcpconf_t tcpconf;
        nwio_tcpcl_t clopt;

        dev_name= getenv("TCP_DEVICE");
        if (!dev_name)
                dev_name= TCP_DEVICE;
        fd= open(dev_name, O_RDWR);
        if (fd == -1)
        {
                *terrno= errno;
```

```
                return -1;
        }
        tcpconf.nwtc_flags= NWTC_EXCL | NWTC_LP_SEL | NWTC_SET_RA | NWTC_SET_RP;
        tcpconf.nwtc_remaddr= host;
        tcpconf.nwtc_remport= port;
        error= ioctl(fd, NWIOSTCPCONF, &tcpconf);
        if (error == -1)
        {
                *terrno= errno;
                close(fd);
                return -1;
        }
        clopt.nwtcl_flags= 0;
        error= ioctl(fd, NWIOTCPCONN, &clopt);
        if (error == -1)
        {
                *terrno= errno;
                close(fd);
                return -1;
        }
        *terrno= 0;
        return fd;
}

static int tcpip_writeall(fd, buf, siz)
int fd;
const char *buf;
size_t siz;
{
        size_t siz_org;
        int nbytes;

        siz_org= siz;

        while (siz)
        {
                nbytes= write(fd, buf, siz);
                if (nbytes <= 0)
                        return siz_org-siz;
                assert(siz >= nbytes);
                buf += nbytes;
                siz -= nbytes;
        }
        return siz_org;
}


static int udp_connect()
{
        nwio_udpopt_t udpopt;
        char *dev_name;
        int fd, r, terrno;

        dev_name= getenv("UDP_DEVICE");
        if (!dev_name)
                dev_name= UDP_DEVICE;
        fd= open(dev_name, O_RDWR);
        if (fd == -1)
                return -1;

```

```
        udpopt.nwuo_flags= NWUO_COPY | NWUO_LP_SEL | NWUO_EN_LOC |
                NWUO_EN_BROAD | NWUO_RP_ANY | NWUO_RA_ANY | NWUO_RWDATALL |
                NWUO_DI_IPOPT;
        r= ioctl(fd, NWIOSUDPOPT, &udpopt);
        if (r == -1)
        {
                terrno= errno;
                close(fd);
                errno= terrno;
                return -1;
        }
        return fd;
}

static int udp_sendto(fd, buf, buflen, addr, port)
int fd;
const char *buf;
unsigned buflen;
ipaddr_t addr;
udpport_t port;
{
        char *newbuf;
        udp_io_hdr_t *udp_io_hdr;
        int r, terrno;

        newbuf= malloc(sizeof(*udp_io_hdr) + buflen);
        if (newbuf == NULL)
        {
                errno= ENOMEM;
                return -1;
        }
        udp_io_hdr= (udp_io_hdr_t *)newbuf;
        udp_io_hdr->uih_dst_addr= addr;
        udp_io_hdr->uih_dst_port= port;
        udp_io_hdr->uih_ip_opt_len= 0;
        udp_io_hdr->uih_data_len= buflen;

        memcpy(newbuf + sizeof(*udp_io_hdr), buf, buflen);
        r= write(fd, newbuf, sizeof(*udp_io_hdr) + buflen);
        terrno= errno;
        free(newbuf);
        if (r >= sizeof(*udp_io_hdr))
                r -= sizeof(*udp_io_hdr);
        errno= terrno;
        return r;
}

static void alarm_handler(sig)
int sig;
{
        signal(SIGALRM, alarm_handler);
        alarm(1);
}

static int udp_receive(fd, buf, buflen, timeout)
int fd;
char *buf;
unsigned buflen;
time_t timeout;
{
```

```
        char *newbuf;
        udp_io_hdr_t *udp_io_hdr;
        int r, terrno;
        void (*u_handler) _ARGS(( int sig ));
        time_t u_timeout;

        newbuf= malloc(sizeof(*udp_io_hdr) + buflen);
        if (newbuf == NULL)
        {
                errno= ENOMEM;
                return -1;
        }

        u_handler= signal(SIGALRM, alarm_handler);
        u_timeout= alarm(timeout);

        r= read(fd, newbuf, sizeof(*udp_io_hdr) + buflen);
        terrno= errno;

        if (r < 0 || r <= sizeof(*udp_io_hdr))
        {
                if (r > 0)
                        r= 0;
                free(newbuf);


                alarm(0);
                signal(SIGALRM, u_handler);
                alarm(u_timeout);

                errno= terrno;
                return r;
        }

        memcpy(buf, newbuf + sizeof(*udp_io_hdr), r - sizeof(*udp_io_hdr));
        free(newbuf);

        alarm(0);
        signal(SIGALRM, u_handler);
        alarm(u_timeout);

        return r-sizeof(*udp_io_hdr);
}

#endif
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/ip/strcasecmp.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
strcasecmp.c

Created Oct 14, 1991 by Philip Homburg
*/

#include <ctype.h>
#include <string.h>

#ifdef __STDC__
#define _CONST  const
#else
#define _CONST
#endif

int
strcasecmp(s1, s2)
_CONST char *s1, *s2;
{
        int c1, c2;
        while (c1= toupper(*s1++), c2= toupper(*s2++), c1 == c2 && (c1 & c2))
                ;
        if (c1 & c2)
                return c1 < c2 ? -1 : 1;
        return c1 ? 1 : (c2 ? -1 : 0);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/liby/main.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*-
 * Copyright (c) 1990 The Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
```

```
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ‘AS IS’’ AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */

#if defined(LIBC_SCCS) && !defined(lint)
static char sccsid[] = "@(#)main.c      5.3 (Berkeley) 1/13/91";
#endif /* not lint */

main()
{
        exit(yyparse());
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/liby/yyerror.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*-
 * Copyright (c) 1990 The Regents of the University of California.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ‘AS IS’’ AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
```

```
 * SUCH DAMAGE.
 */

#if defined(LIBC_SCCS) && !defined(lint)
static char sccsid[] = "@(#)yyerror.c   5.2 (Berkeley) 5/15/90";
#endif /* not lint */

#include <stdio.h>

yyerror(msg)
char *msg;
{
        (void)fprintf(stderr, "%s\n", msg);
        return(0);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/math/localmath.h
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * localmath.h - This header is used by the mathematical library.
 */
/* $Header: localmath.h,v 1.2 89/12/18 15:43:05 eck Exp $ */

/* some constants (Hart & Cheney) */
#define M_PI            3.14159265358979323846264338327950288
#define M_2PI           6.28318530717958647692528676655900576
#define M_3PI_4         2.35619449019234492884698253745962716
#define M_PI_2          1.57079632679489661923132169163975144
#define M_3PI_8         1.17809724509617246442349126872981358
#define M_PI_4          0.78539816339744830961566084581987572
#define M_PI_8          0.39269908169872415480783042290993786
#define M_1_PI          0.31830988618379067153776752674502872
#define M_2_PI          0.63661977236758134307553505349005744
#define M_4_PI          1.27323954473516268615107010698011488
#define M_E             2.71828182845904523536028747135266250
#define M_LOG2E         1.44269504088896340735992468100189213
#define M_LOG10E        0.43429448190325182765112891891660508
#define M_LN2           0.69314718055994530941723212145817657
#define M_LN10          2.30258509299404568401799145468436421
#define M_SQRT2         1.41421356237309504880168872420969808
#define M_1_SQRT2       0.70710678118654752440084436210484904
#define M_EULER         0.57721566490153286060651209008240243

/* macros for constructing polynomials */
#define POLYNOM1(x, a)  ((a)[1]*(x)+(a)[0])
#define POLYNOM2(x, a)  (POLYNOM1((x),(a)+1)*(x)+(a)[0])
#define POLYNOM3(x, a)  (POLYNOM2((x),(a)+1)*(x)+(a)[0])
#define POLYNOM4(x, a)  (POLYNOM3((x),(a)+1)*(x)+(a)[0])
#define POLYNOM5(x, a)  (POLYNOM4((x),(a)+1)*(x)+(a)[0])
#define POLYNOM6(x, a)  (POLYNOM5((x),(a)+1)*(x)+(a)[0])
#define POLYNOM7(x, a)  (POLYNOM6((x),(a)+1)*(x)+(a)[0])
#define POLYNOM8(x, a)  (POLYNOM7((x),(a)+1)*(x)+(a)[0])
#define POLYNOM9(x, a)  (POLYNOM8((x),(a)+1)*(x)+(a)[0])
```

```
#define POLYNOM10(x, a) (POLYNOM9((x),(a)+1)*(x)+(a)[0])
#define POLYNOM11(x, a) (POLYNOM10((x),(a)+1)*(x)+(a)[0])
#define POLYNOM12(x, a) (POLYNOM11((x),(a)+1)*(x)+(a)[0])
#define POLYNOM13(x, a) (POLYNOM12((x),(a)+1)*(x)+(a)[0])

#define M_LN_MAX_D      (M_LN2 * DBL_MAX_EXP)
#define M_LN_MIN_D      (M_LN2 * (DBL_MIN_EXP - 1))
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/math/asin.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 *
 * Author: Ceriel J.H. Jacobs
 */
/* $Header: asin.c,v 1.3 91/03/19 16:38:12 ceriel Exp $ */

#include        <math.h>
#include        <errno.h>
#include        "localmath.h"

static double
asin_acos(double x, int cosfl)
{
        int negative = x < 0;
        int     i;
        double  g;
        static double p[] = {
                -0.27368494524164255994e+2,
                 0.57208227877891731407e+2,
                -0.39688862997540877339e+2,
                 0.10152522233806463645e+2,
                -0.69674573447350646411e+0
        };
        static double q[] = {
                -0.16421096714498560795e+3,
                 0.41714430248260412556e+3,
                -0.38186303361750149284e+3,
                 0.15095270841030604719e+3,
                -0.23823859153670238830e+2,
                 1.0
        };

        if (__IsNan(x)) {
                errno = EDOM;
                return x;
        }

        if (negative) {
                x = -x;
        }
        if (x > 0.5) {
                i = 1;
                if (x > 1) {
```

```
                        errno = EDOM;
                        return 0;
                }
                g = 0.5 - 0.5 * x;
                x = - sqrt(g);
                x += x;
        }
        else {
                /* ??? avoid underflow ??? */
                i = 0;
                g = x * x;
        }
        x += x * g * POLYNOM4(g, p) / POLYNOM5(g, q);
        if (cosfl) {
                if (! negative) x = -x;
        }
        if ((cosfl == 0) == (i == 1)) {
                x = (x + M_PI_4) + M_PI_4;
        }
        else if (cosfl && negative && i == 1) {
                x = (x + M_PI_2) + M_PI_2;
        }
        if (! cosfl && negative) x = -x;
        return x;
}

double
asin(double x)
{
        return asin_acos(x, 0);
}

double
acos(double x)
{
        return asin_acos(x, 1);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/math/atan.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 *
 * Author: Ceriel J.H. Jacobs
 */
/* $Header: atan.c,v 1.3 91/03/19 16:38:21 ceriel Exp $ */

#include        <float.h>
#include        <math.h>
#include        <errno.h>
#include        "localmath.h"

double
atan(double x)
```

```
{
        /*      Algorithm and coefficients from:
                        "Software manual for the elementary functions"
                        by W.J. Cody and W. Waite, Prentice-Hall, 1980
        */

        static double p[] = {
                -0.13688768894191926929e+2,
                -0.20505855195861651981e+2,
                -0.84946240351320683534e+1,
                -0.83758299368150059274e+0
        };
        static double q[] = {
                 0.41066306682575781263e+2,
                 0.86157349597130242515e+2,
                 0.59578436142597344465e+2,
                 0.15024001160028576121e+2,
                 1.0
        };
        static double a[] = {
                0.0,
                0.52359877559829887307710723554658381,  /* pi/6 */
                M_PI_2,
                1.04719755119659774615421446109316763   /* pi/3 */
        };

        int     neg = x < 0;
        int     n;
        double  g;

        if (__IsNan(x)) {
                errno = EDOM;
                return x;
        }
        if (neg) {
                x = -x;
        }
        if (x > 1.0) {
                x = 1.0/x;
                n = 2;
        }
        else    n = 0;

        if (x > 0.26794919243112270647) {       /* 2-sqtr(3) */
                n = n + 1;
                x = (((0.73205080756887729353*x-0.5)-0.5)+x)/
                        (1.73205080756887729353+x);
        }

        /* ??? avoid underflow ??? */

        g = x * x;
        x += x * g * POLYNOM3(g, p) / POLYNOM4(g, q);
        if (n > 1) x = -x;
        x += a[n];
        return neg ? -x : x;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/math/atan2.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 *
 * Author: Ceriel J.H. Jacobs
 */
/* $Header: atan2.c,v 1.2 89/12/18 15:42:35 eck Exp $ */

#include        <math.h>
#include        <errno.h>
#include        "localmath.h"

double
atan2(double y, double x)
{
        double absx, absy, val;

        if (x == 0 && y == 0) {
                errno = EDOM;
                return 0;
        }
        absy = y < 0 ? -y : y;
        absx = x < 0 ? -x : x;
        if (absy - absx == absy) {
                /* x negligible compared to y */
                return y < 0 ? -M_PI_2 : M_PI_2;
        }
        if (absx - absy == absx) {
                /* y negligible compared to x */
                val = 0.0;
        }
        else    val = atan(y/x);
        if (x > 0) {
                /* first or fourth quadrant; already correct */
                return val;
        }
        if (y < 0) {
                /* third quadrant */
                return val - M_PI;
        }
        return val + M_PI;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/math/ceil.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 *
 * Author: Ceriel J.H. Jacobs
 */
/* $Header: ceil.c,v 1.1 89/05/10 16:00:46 eck Exp $ */

#include        <math.h>

double
ceil(double x)
{
        double val;

        return modf(x, &val) > 0 ? val + 1.0 : val ;
        /*      this also works if modf always returns a positive
                fractional part
        */
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/math/exp.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 *
 * Author: Ceriel J.H. Jacobs
 */
/* $Header: exp.c,v 1.5 91/03/19 16:38:29 ceriel Exp $ */

#include        <math.h>
#include        <float.h>
#include        <errno.h>
#include        "localmath.h"


double
exp(double x)
{
        /*      Algorithm and coefficients from:
                        "Software manual for the elementary functions"
                        by W.J. Cody and W. Waite, Prentice-Hall, 1980
        */

        static double p[] = {
                0.25000000000000000000e+0,
                0.75753180159422776666e-2,
```

```
                0.31555192765684646356e-4
        };

        static double q[] = {
                0.50000000000000000000e+0,
                0.56817302698551221787e-1,
                0.63121894374398503557e-3,
                0.75104028399870046114e-6
        };
        double  xn, g;
        int     n;
        int     negative = x < 0;

        if (__IsNan(x)) {
                errno = EDOM;
                return x;
        }
        if (x < M_LN_MIN_D) {
                errno = ERANGE;
                return 0.0;
        }
        if (x > M_LN_MAX_D) {
                errno = ERANGE;
                return HUGE_VAL;
        }

        if (negative) x = -x;

        /* ??? avoid underflow ??? */

        n = x * M_LOG2E + 0.5;  /* 1/ln(2) = log2(e), 0.5 added for rounding */
        xn = n;
        {
                double  x1 = (long) x;
                double  x2 = x - x1;

                g = ((x1-xn*0.693359375)+x2) - xn*(-2.1219444005469058277e-4);
        }
        if (negative) {
                g = -g;
                n = -n;
        }
        xn = g * g;
        x = g * POLYNOM2(xn, p);
        n += 1;
        return (ldexp(0.5 + x/(POLYNOM3(xn, q) - x), n));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/math/fabs.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 *
 * Author: Ceriel J.H. Jacobs
 */
/* $Header: fabs.c,v 1.1 89/05/10 16:01:17 eck Exp $ */

double
fabs(double x)
{
        return  x < 0 ? -x : x;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/math/floor.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 *
 * Author: Ceriel J.H. Jacobs
 */
/* $Header: floor.c,v 1.1 89/05/10 16:01:29 eck Exp $ */

#include        <math.h>

double
floor(double x)
{
        double val;

        return modf(x, &val) < 0 ? val - 1.0 : val ;
        /*      this also works if modf always returns a positive
                fractional part
        */
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/math/fmod.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 *
 * Author: Hans van Eck
 */
/* $Header: fmod.c,v 1.1 89/12/18 15:41:53 eck Exp $ */

#include        <math.h>
#include        <errno.h>

double
fmod(double x, double y)
{
        long    i;
        double val;
        double frac;

        if (y == 0) {
                errno = EDOM;
                return 0;
        }
        frac = modf( x / y, &val);

        return frac * y;

/*
        val = x / y;
        if (val > LONG_MIN && val < LONG_MAX) {
                i = val;
                return x - i * y;
        }
*/
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/math/frexp.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#
.sect .text; .sect .rom; .sect .data; .sect .bss
.extern _frexp
.sect .text
_frexp:
#if __i386
        push    ebp
        mov     ebp, esp
        push    12(ebp)
        push    8(ebp)
        mov     eax, esp
        add     eax, -4
        push    eax
        call    .fef8
        mov     eax, 16(ebp)
```

```
        pop     (eax)
        pop     eax
        pop     edx
        leave
        ret
#else /* i86 */
        push    bp
        mov     bp, sp
        lea     bx, 4(bp)
        mov     cx, #8
        call    .loi
        mov     ax, sp
        add     ax, #-2
        push    ax
        call    .fef8
        mov     bx, 12(bp)
        pop     (bx)
        call    .ret8
        jmp     .cret
#endif
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/math/hugeval.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1990 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 *
 * Author: Hans van Eck
 */
/* $Header: hugeval.c,v 1.1 90/10/24 14:29:42 eck Exp $ */
#include        <math.h>

double
__huge_val(void)
{
        return 1.0e+1000;       /* This will generate a warning */
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/math/isnan.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
int __IsNan(double d)
{
#if defined(vax) || defined(pdp)
#else
        float f = d;

        if ((*((long *) &f) & 0x7f800000) == 0x7f800000 &&
            (*((long *) &f) & 0x007fffff) != 0) return 1;
#endif
        return 0;
```

```
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/math/ldexp.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1987 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: ldexp.c,v 1.5 91/03/19 16:38:37 ceriel Exp $ */

#include        <math.h>
#include        <float.h>
#include        <errno.h>

double
ldexp(double fl, int exp)
{
        int sign = 1;
        int currexp;

        if (__IsNan(fl)) {
                errno = EDOM;
                return fl;
        }
        if (fl == 0.0) return 0.0;
        if (fl<0) {
                fl = -fl;
                sign = -1;
        }
        if (fl > DBL_MAX) {             /* for infinity */
                errno = ERANGE;
                return sign * fl;
        }
        fl = frexp(fl,&currexp);
        exp += currexp;
        if (exp > 0) {
                if (exp > DBL_MAX_EXP) {
                        errno = ERANGE;
                        return sign * HUGE_VAL;
                }
                while (exp>30) {
                        fl *= (double) (1L << 30);
                        exp -= 30;
                }
                fl *= (double) (1L << exp);
        }
        else    {
                /* number need not be normalized */
                if (exp < DBL_MIN_EXP - DBL_MANT_DIG) {
                        return 0.0;
                }
                while (exp<-30) {
                        fl /= (double) (1L << 30);
                        exp += 30;
```

```
                }
                fl /= (double) (1L << -exp);
        }
        return sign * fl;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/math/log.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 *
 * Author: Ceriel J.H. Jacobs
 */
/* $Header: log.c,v 1.3 91/03/19 16:38:45 ceriel Exp $ */

#include        <math.h>
#include        <float.h>
#include        <errno.h>
#include        "localmath.h"

double
log(double x)
{
        /*      Algorithm and coefficients from:
                        "Software manual for the elementary functions"
                        by W.J. Cody and W. Waite, Prentice-Hall, 1980
        */
        static double a[] = {
                -0.64124943423745581147e2,
                 0.16383943563021534222e2,
                -0.78956112887491257267e0
        };
        static double b[] = {
                -0.76949932108494879777e3,
                 0.31203222091924532844e3,
                -0.35667977739034646171e2,
                 1.0
        };

        double  znum, zden, z, w;
        int     exponent;

        if (__IsNan(x)) {
                errno = EDOM;
                return x;
        }
        if (x < 0) {
                errno = EDOM;
                return -HUGE_VAL;
        }
        else if (x == 0) {
                errno = ERANGE;
```

```
                return -HUGE_VAL;
        }

        if (x <= DBL_MAX) {
        }
        else return x;  /* for infinity and Nan */
        x = frexp(x, &exponent);
        if (x > M_1_SQRT2) {
                znum = (x - 0.5) - 0.5;
                zden = x * 0.5 + 0.5;
        }
        else {
                znum = x - 0.5;
                zden = znum * 0.5 + 0.5;
                exponent--;
        }
        z = znum/zden; w = z * z;
        x = z + z * w * (POLYNOM2(w,a)/POLYNOM3(w,b));
        z = exponent;
        x += z * (-2.121944400546905827679e-4);
        return x + z * 0.693359375;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/math/log10.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 *
 * Author: Ceriel J.H. Jacobs
 */
/* $Header: log10.c,v 1.3 91/03/19 16:38:55 ceriel Exp $ */

#include        <math.h>
#include        <errno.h>
#include        "localmath.h"

double
log10(double x)
{
        if (__IsNan(x)) {
                errno = EDOM;
                return x;
        }
        if (x < 0) {
                errno = EDOM;
                return -HUGE_VAL;
        }
        else if (x == 0) {
                errno = ERANGE;
                return -HUGE_VAL;
        }

        return log(x) / M_LN10;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/math/modf.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#
.sect .text; .sect .rom; .sect .data; .sect .bss
.extern _modf
.sect .text
_modf:
#if __i386
        push    ebp
        mov     ebp, esp
        push    12(ebp)
        push    8(ebp)
        push    1
        push    4
        call    .cif8
        mov     eax, esp
        push    eax
        call    .fif8
        pop     ecx
        mov     edx, 16(ebp)
        pop     ecx
        pop     ebx
        mov     0(edx), ecx
        mov     4(edx), ebx
        pop     eax
        pop     edx
        leave
        ret
#else /* i86 */
        push    bp
        mov     bp, sp
        lea     bx, 4(bp)
        mov     cx, #8
        call    .loi
        mov     dx, #1
        push    dx
        push    dx
        push    dx
        mov     ax, #2
        push    ax
        call    .cif8
        mov     ax, sp
        push    ax
        call    .fif8
        pop     bx
        mov     bx, 12(bp)
        mov     cx, #8
        call    .sti
        call    .ret8
        jmp     .cret
#endif
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/math/pow.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 *
 * Author: Ceriel J.H. Jacobs
 */
/* $Header: pow.c,v 1.3 90/08/28 13:59:36 eck Exp $ */

#include        <math.h>
#include        <float.h>
#include        <errno.h>
#include        "localmath.h"

double
pow(double x, double y)
{
        /*      Simple version for now. The Cody and Waite book has
                a very complicated, much more precise version, but
                this version has machine-dependent arrays A1 and A2,
                and I don’t know yet how to solve this ???
        */
        double dummy;
        int     result_neg = 0;

        if ((x == 0 && y == 0) ||
            (x < 0 && modf(y, &dummy) != 0)) {
                errno = EDOM;
                return 0;
        }

        if (x == 0) return x;

        if (x < 0) {
                if (modf(y/2.0, &dummy) != 0) {
                        /* y was odd */
                        result_neg = 1;
                }
                x = -x;
        }
        x = log(x);

        if (x < 0) {
                x = -x;
                y = -y;
        }
        /* Beware of overflow in the multiplication */
        if (x > 1.0 && y > DBL_MAX/x) {
                errno = ERANGE;
                return result_neg ? -HUGE_VAL : HUGE_VAL;
        }

        x = exp(x * y);
        return result_neg ? -x : x;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/math/sin.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 *
 * Author: Ceriel J.H. Jacobs
 */
/* $Header: sin.c,v 1.3 91/03/19 16:39:04 ceriel Exp $ */

#include        <math.h>
#include        <float.h>
#include        <errno.h>
#include        "localmath.h"

static double
sinus(double x, int cos_flag)
{
        /*      Algorithm and coefficients from:
                        "Software manual for the elementary functions"
                        by W.J. Cody and W. Waite, Prentice-Hall, 1980
        */

        static double r[] = {
                -0.16666666666666665052e+0,
                 0.83333333333331650314e-2,
                -0.19841269841201840457e-3,
                 0.27557319210152756119e-5,
                -0.25052106798274584544e-7,
                 0.16058936490371589114e-9,
                -0.76429178068910467734e-12,
                 0.27204790957888846175e-14
        };

        double  y;
        int     neg = 1;

        if (__IsNan(x)) {
                errno = EDOM;
                return x;
        }
        if (x < 0) {
                x = -x;
                neg = -1;
        }
        if (cos_flag) {
                neg = 1;
                y = M_PI_2 + x;
        }
        else    y = x;

        /* ??? avoid loss of significance, if y is too large, error ??? */

        y = y * M_1_PI + 0.5;

        if (y >= DBL_MAX/M_PI) return 0.0;

```

```
        /*      Use extended precision to calculate reduced argument.
                Here we used 12 bits of the mantissa for a1.
                Also split x in integer part x1 and fraction part x2.
        */
#define A1 3.1416015625
#define A2 -8.908910206761537356617e-6
        {
                double x1, x2;

                modf(y, &y);
                if (modf(0.5*y, &x1)) neg = -neg;
                if (cos_flag) y -= 0.5;
                x2 = modf(x, &x1);
                x = x1 - y * A1;
                x += x2;
                x -= y * A2;
#undef A1
#undef A2
        }

        if (x < 0) {
                neg = -neg;
                x = -x;
        }

        /* ??? avoid underflow ??? */

        y = x * x;
        x += x * y * POLYNOM7(y, r);
        return neg==-1 ? -x : x;
}

double
sin(double x)
{
        return sinus(x, 0);
}

double
cos(double x)
{
        if (x < 0) x = -x;
        return sinus(x, 1);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/math/sinh.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 *
 * Author: Ceriel J.H. Jacobs
 */
/* $Header: sinh.c,v 1.3 91/03/19 16:39:12 ceriel Exp $ */

#include        <math.h>
#include        <float.h>
```

```
#include        <errno.h>
#include        "localmath.h"

static double
sinh_cosh(double x, int cosh_flag)
{
        /*      Algorithm and coefficients from:
                        "Software manual for the elementary functions"
                        by W.J. Cody and W. Waite, Prentice-Hall, 1980
        */

        static double p[] = {
                -0.35181283430177117881e+6,
                -0.11563521196851768270e+5,
                -0.16375798202630751372e+3,
                -0.78966127417357099479e+0
        };
        static double q[] = {
                -0.21108770058106271242e+7,
                 0.36162723109421836460e+5,
                -0.27773523119650701167e+3,
                 1.0
        };
        int     negative = x < 0;
        double  y = negative ? -x : x;

        if (__IsNan(x)) {
                errno = EDOM;
                return x;
        }
        if (! cosh_flag && y <= 1.0) {
                /* ??? check for underflow ??? */
                y = y * y;
                return x + x * y * POLYNOM3(y, p)/POLYNOM3(y,q);
        }

        if (y >= M_LN_MAX_D) {
                /* exp(y) would cause overflow */
#define LNV     0.69316101074218750000e+0
#define VD2M1   0.52820835025874852469e-4
                double  w = y - LNV;

                if (w < M_LN_MAX_D+M_LN2-LNV) {
                        x = exp(w);
                        x += VD2M1 * x;
                }
                else {
                        errno = ERANGE;
                        x = HUGE_VAL;
                }
        }
        else {
                double  z = exp(y);

                x = 0.5 * (z + (cosh_flag ? 1.0 : -1.0)/z);
        }
        return negative ? -x : x;
}

double
```

```
sinh(double x)
{
        return sinh_cosh(x, 0);
}

double
cosh(double x)
{
        if (x < 0) x = -x;
        return sinh_cosh(x, 1);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/math/sqrt.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 *
 * Author: Ceriel J.H. Jacobs
 */
/* $Header: sqrt.c,v 1.3 91/03/19 16:39:21 ceriel Exp $ */

#include        <math.h>
#include        <float.h>
#include        <errno.h>

#define NITER   5

double
sqrt(double x)
{
        int exponent;
        double val;

        if (__IsNan(x)) {
                errno = EDOM;
                return x;
        }
        if (x <= 0) {
                if (x < 0) errno = EDOM;
                return 0;
        }

        if (x > DBL_MAX) return x;      /* for infinity */

        val = frexp(x, &exponent);
        if (exponent & 1) {
                exponent--;
                val *= 2;
        }
        val = ldexp(val + 1.0, exponent/2 - 1);
        /* was: val = (val + 1.0)/2.0; val = ldexp(val, exponent/2); */
        for (exponent = NITER - 1; exponent >= 0; exponent--) {
                val = (val + x / val) / 2.0;
```

```
        }
        return val;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/math/tan.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 *
 * Author: Ceriel J.H. Jacobs
 */
/* $Header: tan.c,v 1.3 91/03/19 16:39:30 ceriel Exp $ */

#include        <math.h>
#include        <float.h>
#include        <errno.h>
#include        "localmath.h"

double
tan(double x)
{
        /*      Algorithm and coefficients from:
                        "Software manual for the elementary functions"
                        by W.J. Cody and W. Waite, Prentice-Hall, 1980
        */

        int negative = x < 0;
        int invert = 0;
        double  y;
        static double   p[] = {
                 1.0,
                -0.13338350006421960681e+0,
                 0.34248878235890589960e-2,
                -0.17861707342254426711e-4
        };
        static double   q[] = {
                 1.0,
                -0.46671683339755294240e+0,
                 0.25663832289440112864e-1,
                -0.31181531907010027307e-3,
                 0.49819433993786512270e-6
        };

        if (__IsNan(x)) {
                errno = EDOM;
                return x;
        }
        if (negative) x = -x;

        /* ??? avoid loss of significance, error if x is too large ??? */

        y = x * M_2_PI + 0.5;

        if (y >= DBL_MAX/M_PI_2) return 0.0;

```

```
        /*      Use extended precision to calculate reduced argument.
                Here we used 12 bits of the mantissa for a1.
                Also split x in integer part x1 and fraction part x2.
        */
    #define A1 1.57080078125
    #define A2 -4.454455103380768678308e-6
        {
                double x1, x2;

                modf(y, &y);
                if (modf(0.5*y, &x1)) invert = 1;
                x2 = modf(x, &x1);
                x = x1 - y * A1;
                x += x2;
                x -= y * A2;
    #undef A1
    #undef A2
        }

        /* ??? avoid underflow ??? */
        y = x * x;
        x += x * y * POLYNOM2(y, p+1);
        y = POLYNOM4(y, q);
        if (negative) x = -x;
        return invert ? -y/x : x/y;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/math/tanh.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 *
 * Author: Ceriel J.H. Jacobs
 */
/* $Header: tanh.c,v 1.3 91/03/19 16:39:40 ceriel Exp $ */

#include        <float.h>
#include        <math.h>
#include        <errno.h>
#include        "localmath.h"

double
tanh(double x)
{
        /*      Algorithm and coefficients from:
                        "Software manual for the elementary functions"
                        by W.J. Cody and W. Waite, Prentice-Hall, 1980
        */

        static double p[] = {
                -0.16134119023996228053e+4,
                -0.99225929672236083313e+2,
                -0.96437492777225469787e+0
```

```
        };
        static double q[] = {
                 0.48402357071988688686e+4,
                 0.22337720718962312926e+4,
                 0.11274474380534949335e+3,
                 1.0
        };
        int     negative = x < 0;

        if (__IsNan(x)) {
                errno = EDOM;
                return x;
        }
        if (negative) x = -x;

        if (x >= 0.5*M_LN_MAX_D) {
                x = 1.0;
        }
#define LN3D2   0.54930614433405484570e+0       /* ln(3)/2 */
        else if (x > LN3D2) {
                x = 0.5 - 1.0/(exp(x+x)+1.0);
                x += x;
        }
        else {
                /* ??? avoid underflow ??? */
                double g = x*x;
                x += x * g * POLYNOM2(g, p)/POLYNOM3(g, q);
        }
        return negative ? -x : x;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/_brk.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define brk     _brk
#define sbrk    _sbrk
#include <unistd.h>

extern char *_brksize;

/* Both OSF/1 and SYSVR4 man pages specify that brk(2) returns int.
 * However, BSD4.3 specifies that brk() returns char*.  POSIX omits
 * brk() on the grounds that it imposes a memory model on an architecture.
 * For this reason, brk() and sbrk() are not in the lib/posix directory.
 * On the other hand, they are so crucial to correct operation of so many
 * parts of the system, that we have chosen to hide the name brk using _brk,
 * as with system calls.  In this way, if a user inadvertently defines a
 * procedure brk, MINIX may continue to work because the true call is _brk.
 */
PUBLIC int brk(addr)
char *addr;
{
  message m;
```

```
  m.m1_p1 = addr;
  if (_syscall(MM, BRK, &m) < 0) return(-1);
  _brksize = m.m2_p1;
  return(0);
}


PUBLIC char *sbrk(incr)
int incr;
{
  char *newsize, *oldsize;

  oldsize = _brksize;
  newsize = _brksize + incr;
  if (incr > 0 && newsize < oldsize || incr < 0 && newsize > oldsize)
        return( (char *) -1);
  if (brk(newsize) == 0)
        return(oldsize);
  else
        return( (char *) -1);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/other/_longjerr.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define longjerr        _longjerr
#define write           _write
#include <minix/minlib.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>


_PROTOTYPE( void longjerr, (void));

PUBLIC void longjerr()
{
  static char errmsg[] = "longj error\n";

  write(2, errmsg, strlen(errmsg));     /* hope it's stderr */
  while(1) abort();             /* XXX - maybe just exit */
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/_reboot.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/* reboot.c - Systemcall interface to mm/signal.c::do_reboot()

   author: Edvard Tuinder  v892231@si.hhs.NL
 */

#include <lib.h>
#define reboot  _reboot
#include <unistd.h>
#include <stdarg.h>

int reboot(int how, ...)
{
  message m;
  va_list ap;

  va_start(ap, how);
  if ((m.m1_i1 = how) == RBT_MONITOR) {
        m.m1_p1 = va_arg(ap, char *);
        m.m1_i2 = va_arg(ap, size_t);
  }
  va_end(ap);

  return _syscall(MM, REBOOT, &m);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/_seekdir.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*      seekdir()                                       Author: Kees J. Bot
 *                                                              24 Apr 1989
 */
#define nil 0
#include <lib.h>
#define lseek   _lseek
#define readdir _readdir
#define seekdir _seekdir
#include <sys/types.h>
#include <dirent.h>
#include <unistd.h>
#include <errno.h>

int seekdir(DIR *dp, off_t pos)
/* Seek to position pos in a directory. */
{
        int off;

        if (dp == nil) { errno= EBADF; return -1; }

        dp->_count= 0;
        dp->_ptr= dp->_buf;

        off= pos & (sizeof(dp->_buf) - 1);
        dp->_pos= pos - off;
```

```

        if (lseek(dp->_fd, dp->_pos, SEEK_SET) == -1) return -1;

        while (dp->_pos < pos && readdir(dp) != nil) {}

        return 0;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/asynchio.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*      asyn_init(), asyn_read(), asyn_write(), asyn_ioctl(),
 *      asyn_wait(), asyn_synch(), asyn_close()
 *                                                       Author: Kees J. Bot
 *                                                               26 Jan 1995
 * Thise are just stub routines that are call compatible with
 * the asynchio(3) library of Minix-vmd.  See asynchio.h.
 */
#define nil 0
#define alarm   _alarm
#define ioctl   _ioctl
#define read    _read
#define sigaction _sigaction
#define sigfillset _sigfillset
#define time    _time
#define write   _write
#include <lib.h>
#include <sys/ioctl.h>
#include <sys/asynchio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <time.h>
#include <signal.h>

#define IDLE            0
#define INPROGRESS      1
#define RESULT          2

#define OP_READ         0
#define OP_WRITE        1
#define OP_IOCTL        2

static int *asyn_current;
static int asyn_op;
static int asyn_fd;
static int asyn_req;
static void *asyn_data;
static ssize_t asyn_count;
static int asyn_errno;

void asyn_init(asynchio_t *asyn)
{
        *asyn= IDLE;
}
```

```
static ssize_t operation(int op, asynchio_t *asyn, int fd, int req,
                                                void *data, ssize_t count)
{
        switch (*asyn) {
        case INPROGRESS:
                if (asyn_current != asyn && asyn_op != op) abort();
                /*FALL THROUGH*/
        case IDLE:
                asyn_current= asyn;
                asyn_op= op;
                asyn_fd= fd;
                asyn_req= req;
                asyn_data= data;
                asyn_count= count;
                *asyn= INPROGRESS;
                errno= EINPROGRESS;
                return -1;
        case RESULT:
                if (asyn_current != asyn && asyn_op != op) abort();
                *asyn= IDLE;
                errno= asyn_errno;
                return asyn_count;
        }
}

ssize_t asyn_read(asynchio_t *asyn, int fd, void *buf, size_t len)
{
        return operation(OP_READ, asyn, fd, 0, buf, len);
}

ssize_t asyn_write(asynchio_t *asyn, int fd, const void *buf, size_t len)
{
        return operation(OP_WRITE, asyn, fd, 0, (void *) buf, len);
}

int asyn_ioctl(asynchio_t *asyn, int fd, unsigned long request, void *data)
{
        return operation(OP_IOCTL, asyn, fd, request, data, 0);
}

static void time_out(int sig)
{
        alarm(1);
}

int asyn_wait(asynchio_t *asyn, int flags, struct timeval *to)
{
        time_t now;
        unsigned old_timer, new_timer;
        struct sigaction old_sa, new_sa;

        if (*asyn == IDLE) return 0;
        if (asyn_current != asyn || *asyn != INPROGRESS) abort();
        if (flags & ASYN_NONBLOCK) abort();

        if (to != nil) {
                now= time(nil);
                if (to->tv_sec <= now) { errno= EINTR; return -1; }
                old_timer= alarm(0);
                new_sa.sa_handler= time_out;
```

```
                sigfillset(&new_sa.sa_mask);
                new_sa.sa_flags= 0;
                sigaction(SIGALRM, &new_sa, &old_sa);
                new_timer= to->tv_sec - now;
                if (new_timer < old_timer) {
                        new_timer= old_timer;
                }
                alarm(new_timer);
        }
        switch (asyn_op) {
        case OP_READ:
                asyn_count= read(asyn_fd, asyn_data, asyn_count);
                asyn_errno= errno;
                break;
        case OP_WRITE:
                asyn_count= write(asyn_fd, asyn_data, asyn_count);
                asyn_errno= errno;
                break;
        case OP_IOCTL:
                asyn_count= ioctl(asyn_fd, asyn_req, asyn_data);
                asyn_errno= errno;
                break;
        }
        if (to != nil) {
                alarm(0);
                sigaction(SIGALRM, &old_sa, (struct sigaction *)0);
                alarm(old_timer);
        }

        if (asyn_count == -1 && asyn_errno == EINTR) {
                errno= EINTR;
                return -1;
        } else {
                *asyn= RESULT;
                return 0;
        }
}

int asyn_synch(asynchio_t *asyn, int fd)
{
}

int asyn_close(asynchio_t *asyn, int fd)
{
        *asyn= IDLE;
}
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/bcmp.c
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
/* bcmp - Berklix equivalent of memcmp  */

#include <string.h>

int bcmp(s1, s2, length)        /* == 0 or != 0 for equality and inequality */
_CONST void *s1;
_CONST void *s2;
int length;
{
  return(memcmp((_CONST _VOIDSTAR) s1, (_CONST _VOIDSTAR) s2, (_SIZET) length));
}
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/bcopy.c
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
/* bcopy - Berklix equivalent of memcpy  */

#include <string.h>

void bcopy(src, dst, length)
_CONST void *src;
void *dst;
int length;
{
  (void) memcpy((_VOIDSTAR) dst, (_CONST _VOIDSTAR) src, (_SIZET) length);
}
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/bzero.c
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
/* bzero - Berklix subset of memset  */

#include <string.h>

void bzero(dst, length)
void *dst;
int length;
{
  (void) memset((_VOIDSTAR) dst, 0, (_SIZET) length);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/crypt.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*	crypt() - one-way password encryption function	Author: Kees J. Bot
 *								7 Feb 1994
 * This routine does not encrypt anything, it uses the pwdauth
 * program to do the hard work.
 */
#define nil 0
#define pipe _pipe
#define fork _fork
#define close _close
#define dup2 _dup2
#define execl _execl
#define read _read
#define _exit __exit
#define write _write
#define waitpid _waitpid
#include <sys/types.h>
#include <unistd.h>
#include <string.h>
#include <stdio.h>
#include <sys/wait.h>

/* Set-uid root program to read /etc/shadow or encrypt passwords. */
static char PWDAUTH[] = "/usr/lib/pwdauth";
#define LEN	1024

char *crypt(const char *key, const char *salt)
{
	pid_t pid;
	int status;
	int pfd[2];
	static char pwdata[LEN];
	char *p= pwdata;
	const char *k= key;
	const char *s= salt;
	int n;

	/* Fill pwdata[] with the key and salt. */
	while ((*p++ = *k++) != 0) if (p == pwdata+LEN-1) goto fail;
	while ((*p++ = *s++) != 0) if (p == pwdata+LEN-0) goto fail;

	if (pipe(pfd) < 0) goto fail;

	/* Prefill the pipe. */
	(void) write(pfd[1], pwdata, p - pwdata);

	switch ((pid= fork())) {
	case -1:
		close(pfd[0]);
		close(pfd[1]);
		goto fail;
	case 0:
		/* Connect both input and output to the pipe. */
		if (pfd[0] != 0) {
			dup2(pfd[0], 0);
			close(pfd[0]);
```

```
                }
                if (pfd[1] != 1) {
                        dup2(pfd[1], 1);
                        close(pfd[1]);
                }

                execl(PWDAUTH, PWDAUTH, (char *) nil);

                /* No pwdauth?  Fail! */
                (void) read(0, pwdata, LEN);
                _exit(1);
        }
        close(pfd[1]);

        if (waitpid(pid, &status, 0) < 0 || status != 0) {
                close(pfd[0]);
                goto fail;
        }

        /* Read and return the result.  Check if it contains exactly one
         * string.
         */
        n= read(pfd[0], pwdata, LEN);
        close(pfd[0]);
        if (n < 0) goto fail;
        p = pwdata + n;
        n = 0;
        while (p > pwdata) if (*--p == 0) n++;
        if (n != 1) goto fail;
        return pwdata;

fail:
        pwdata[0] = salt[0] ^ 1;                /* make result != salt */
        pwdata[1] = 0;
        return pwdata;
}

/*
 * $PchHeader: /mount/hd2/minix/lib/misc/RCS/crypt.c,v 1.3 1994/12/22 13:51:49 philip Exp $
 */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/ctermid.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*  ctermid(3)
 *
 *  Author: Terrence Holm          Aug. 1988
 *
 *
 *  Ctermid(3) returns a pointer to a string naming the controlling
 *  terminal. If <name_space> is NULL then local PRIVATE storage
 *  is used, otherwise <name_space> must point to storage of at
 *  least L_ctermid characters.
 *
```

```
 *  Returns a pointer to "/dev/tty".
 */

#include <lib.h>
#include <string.h>
#include <stdio.h>

_PROTOTYPE( char *ctermid, (char *name_space));

#ifndef L_ctermid
#define L_ctermid  9
#endif

char *ctermid(name_space)
char *name_space;
{
  PRIVATE char termid[L_ctermid];

  if (name_space == (char *)NULL) name_space = termid;
  strcpy(name_space, "/dev/tty");
  return(name_space);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/cuserid.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*  cuserid(3)
 *
 *  Author: Terrence W. Holm          Sept. 1987
 */

#include <lib.h>
#include <pwd.h>
#include <string.h>
#include <unistd.h>
#include <stdio.h>

#ifndef  L_cuserid
#define  L_cuserid   9
#endif

char *cuserid(user_name)
char *user_name;
{
  PRIVATE char userid[L_cuserid];
  struct passwd *pw_entry;

  if (user_name == (char *)NULL) user_name = userid;

  pw_entry = getpwuid(geteuid());

  if (pw_entry == (struct passwd *)NULL) {
        *user_name = '\0';
        return((char *)NULL);
  }
  strcpy(user_name, pw_entry->pw_name);
```

```

  return(user_name);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/environ.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * environ.c - define the variable environ
 */
/* $Header: environ.c,v 1.1 90/09/27 13:39:50 eck Exp $ */
/*
 * This file defines the variable environ and initializes it with a magic
 * value.  The C run-time start-off routine tests whether the variable
 * environ is initialized with this value.  If it is not, it is assumed
 * that it is defined by the user.  Only two bytes are tested, since we
 * don't know the endian-ness and alignment restrictions of the machine.
 * This means that the low-order two-bytes should be equal to the
 * high-order two-bytes on machines with four-byte pointers.  In fact, all
 * the bytes in the pointer are the same, just in case.
 */

#if _EM_PSIZE==2
char **environ = (char **) 0x5353;
#else
char **environ = (char **) 0x53535353;
#endif
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/errno.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
/* errno.c - declare variable errno             Author: F. Meulenbroeks */

int errno = 0;
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/execlp.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/* execlp(3) and execvp(3)
 *
 * Author: Terrence W. Holm      July 1988
 */

/* FIXES - Dec 1989 - Jan 1990 Bruce Evans.
 *       - Don't use search path when file name contains a '/' *anywhere*.
 *       - Invoke sh(1) on command files.
 *       - Use PATH_MAX and check strings fit in buffer.
 *       - Use stdargs, with the unjustified assumption that va_start() turns
```

```
 *         the arg list into a char *[].  Strictly, the arg list should be
 *         copied, "wasting" up to ARG_MAX bytes.
 */

/*  Execlp(3) and execvp(3) are like execl(3) and execv(3),
 *  except that they use the environment variable $PATH as
 *  a search list of possible locations for the executable
 *  file, if <file> does not contain a '/', and they attempt
 *  to run non-binary executable files using sh(1).
 *
 *  The path search list is a list of directory names separated
 *  by ':'s. If a colon appears at the beginning or end of the
 *  list, or two appear together, then an empty prefix is tried.
 *  If $PATH is not in the environment, it defaults to "".
 *
 *  For example, if <file> is "ls", and the $PATH is
 *  ":/usr/local/bin:/bin:/usr/bin", then  ./ls,
 *  /usr/local/bin/ls, /bin/ls and /usr/bin/ls are tried until
 *  an exectable one is found. If the direct attempt to exec it
 *  fails, the arg list is modified to begin with "sh" and the
 *  absolute name of <file>, and an exec of /bin/sh is tried.
 *  If this fails, no further attempts are made.
 *
 *  This function only returns after an error.  It returns -1
 *  and sets errno like execv().
 */

#include <sys/types.h>
#include <lib.h>
#include <errno.h>
#include <limits.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <stdarg.h>

#undef NULL
#define NULL 0                  /* kludge for ACK not understanding void * */

#define MAX_NUM_ARGS 512        /* maximum number of arguments to execvp */

extern char **environ;          /* environment pointer */

#ifdef _ANSI
int execlp(const char *file, const char *arg, ...)
#else
int execlp(file)
char *file;
#endif
{
  register va_list argp;
  register int result;

  va_start(argp, file);
  result = execvp(file, (char **) argp);
  va_end(argp);
  return(result);
}

#ifdef _ANSI
```

```
int execvp(const char *file, char *const argv[])
#else
int execvp(file, argv)
char *file;
char **argv;
#endif
{
  int i, best_errno;
  char **envtop;
  size_t flength;
  char *searchpath;
  size_t slength;
  char *split;
  char execpath[PATH_MAX + 1];
  char *arg2[MAX_NUM_ARGS + 3]; /* place to copy argv */

  /* POSIX requires argv to be immutable.  Unfortunately, we have to change it
   * during execution.  To keep POSIX happy, a copy is made and the copy
   * changed.  The question arises: how big should the copy be?  Allocating
   * space dynamically requires using malloc, which itself takes up a lot
   * of space.  The solution chosen here is to limit the number of arguments
   * to MAX_NUM_ARGS and set this value fairly high.  This solution is simpler
   * and is probably adequate.  Only programs with huge numbers of very short
   * arguments will get an error (if the arguments are large, ARG_MAX will
   * be exceeded.
   */

  if (strchr(file, '/') != NULL || (searchpath = getenv("PATH")) == NULL)
        searchpath = "";
  flength = strlen(file);
  best_errno = ENOENT;

  while (1) {
        split = strchr(searchpath, ':');
        if (split == NULL)
                slength = strlen(searchpath);
        else
                slength = split - searchpath;
        if (slength + flength >= sizeof execpath - 2) {
                errno = ENAMETOOLONG;   /* too bad if premature */
                return(-1);
        }
        strncpy(execpath, searchpath, slength);
        if (slength != 0) execpath[slength++] = '/';
        strcpy(execpath + slength, file);

        /* Don't try to avoid execv() for non-existent files, since the Minix
         * shell doesn't, and it is not clear whether access() or stat() work
         * right when this code is set-uid.
         */
        execv(execpath, argv);
        switch (errno) {
            case EACCES:
                best_errno = errno;     /* more useful than ENOENT */
            case ENOENT:
                if (split == NULL) {
                        /* No more path components. */
                        errno = best_errno;
                        return(-1);
                }
```

```
                searchpath = split + 1; /* try next in path */
                break;
            case ENOEXEC:
                /* Assume a command file and invoke sh(1) on it.  Replace arg0
                 * (which is usually a short name for the command) by the full
                 * name of the command file.
                 */

                /* Copy the arg pointers from argv to arg2,  moving them up by
                 * 1, overlaying the assumed NULL at the end, to make room
                 * for "sh" at the beginning.
                 */
                i = 0;
                if (argv != NULL)
                {
                        while (argv[i] != 0) {
                                if (i >= MAX_NUM_ARGS) {
                                        /* Copy failed.  Not enough room. */
                                        errno = ENOEXEC;
                                        return(-1);
                                }
                                arg2[i + 1] = argv[i];
                                i++;
                        }
                }
                arg2[0] = "sh";         /* exec the shell */
                arg2[1] = execpath;     /* full path */
                arg2[i + 1] = NULL;     /* terminator */

                /* Count the environment pointers. */
                for (envtop = environ; *envtop != NULL; ) envtop++;

                /* Try only /bin/sh, like the Minix shell.  Lose if the user
                 * has a different shell or the command has #!another/shell.
                 */
                __execve("/bin/sh", arg2, environ, i + 1, (int)(envtop - environ));

                /* Oops, no shell?   Give up. */
                errno = ENOEXEC;
                return(-1);
            default:
                return(-1);     /* probably  ENOMEM or E2BIG */
        }
    }
  }
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/fdopen.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * fdopen - convert a (UNIX) file descriptor into a FILE pointer
 */
/* $Header: fdopen.c,v 1.4 91/02/22 16:32:05 ceriel Exp $ */

#include        <stdlib.h>
#include        "../stdio/loc_incl.h"
#include        <stdio.h>

FILE *
fdopen(fd, mode)
int fd;
_CONST char *mode;
{
        register int i;
        FILE *stream;
        int flags = 0;

        if (fd < 0) return (FILE *)NULL;
        for (i = 0; __iotab[i] != 0 ; i++)
                if (i >= FOPEN_MAX-1)
                        return (FILE *)NULL;

        switch(*mode++) {
        case 'r':
                flags |= _IOREAD | _IOREADING;
                break;
        case 'a':
                flags |= _IOAPPEND;
        case 'w':
                flags |= _IOWRITE | _IOWRITING;
                break;
        default:
                return (FILE *)NULL;
        }
        while(*mode) {
                switch(*mode++) {
                case 'b':
                        continue;
                case '+':
                        flags |= _IOREAD | _IOWRITE;
                        continue;
                /* The sequence may be followed by aditional characters */
                default:
                        break;
                }
                break;
        }

        if ((stream = (FILE *) malloc(sizeof(FILE))) == NULL) {
                return (FILE *)NULL;
        }

        if ((flags & _IOREAD) && (flags & _IOWRITE))
                flags &= ~(_IOREADING | _IOWRITING);
```

```
        stream->_count = 0;
        stream->_fd = fd;
        stream->_flags = flags;
        stream->_buf = NULL;
        __iotab[i] = stream;
        return stream;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/other/ffs.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
/*  ffs(3)
 *
 *  Author: Terrence W. Holm          Sep. 1988
 */
_PROTOTYPE( int ffs, (int word));

int ffs(word)
int word;
{
  int i;

  if (word == 0) return(0);

  for (i = 1;; ++i, word >>= 1)
        if (word & 1) return(i);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/other/fslib.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/* fslib.c - routines needed by fs and fs utilities */

#include <minix/config.h>       /* for unused stuff in <minix/type.h> :-( */
#include <ansi.h>
#include <sys/types.h>
#include <minix/const.h>
#include <minix/type.h>         /* for unshort :-( */
#include "fs/const.h"           /* depends of -I flag in Makefile */
#include "fs/type.h"            /* ditto */
#include "fs/inode.h"           /* ditto */
#include <minix/fslib.h>

/* The next routine is copied from fsck.c and mkfs.c...  (Re)define some
 * things for consistency.  Some things should be done better.  The shifts
 * should be replaced by multiplications and divisions by MAP_BITS_PER_BLOCK
 * since log2 of this is too painful to get right.
 */
#define BITMAPSHIFT      13     /* = log2(MAP_BITS_PER_BLOCK) */

/* Convert from bit count to a block count. The usual expression
```

```
 *
 *      (nr_bits + (1 << BITMAPSHIFT) - 1) >> BITMAPSHIFT
 *
 * doesn't work because of overflow.
 *
 * Other overflow bugs, such as the expression for N_ILIST overflowing when
 * s_inodes is just over V*_INODES_PER_BLOCK less than the maximum+1, are not
 * fixed yet, because that number of inodes is silly.
 */
/* The above comment doesn't all apply now bit_t is long.  Overflow is now
 * unlikely, but negative bit counts are now possible (though unlikely)
 * and give silly results.
 */
PUBLIC int bitmapsize(nr_bits)
bit_t nr_bits;
{
  int nr_blocks;

  nr_blocks = (int) (nr_bits >> BITMAPSHIFT);
  if (((bit_t) nr_blocks << BITMAPSHIFT) < nr_bits) ++nr_blocks;
  return(nr_blocks);
}


/*===========================================================================*
 *                              conv2                                        *
 *===========================================================================*/
PUBLIC unsigned conv2(norm, w)
int norm;                       /* TRUE if no swap, FALSE for byte swap */
int w;                          /* promotion of 16-bit word to be swapped */
{
/* Possibly swap a 16-bit word between 8086 and 68000 byte order. */

  if (norm) return( (unsigned) w & 0xFFFF);
  return( ((w&BYTE) << 8) | ( (w>>8) & BYTE));
}


/*===========================================================================*
 *                              conv4                                        *
 *===========================================================================*/
PUBLIC long conv4(norm, x)
int norm;                       /* TRUE if no swap, FALSE for byte swap */
long x;                         /* 32-bit long to be byte swapped */
{
/* Possibly swap a 32-bit long between 8086 and 68000 byte order. */

  unsigned lo, hi;
  long l;

  if (norm) return(x);                  /* byte order was already ok */
  lo = conv2(FALSE, (int) x & 0xFFFF);  /* low-order half, byte swapped */
  hi = conv2(FALSE, (int) (x>>16) & 0xFFFF);    /* high-order half, swapped */
  l = ( (long) lo <<16) | hi;
  return(l);
}


/*===========================================================================*
 *                              conv_inode                                   *
```

```
 *===========================================================================*/
PUBLIC void conv_inode(rip, dip, dip2, rw_flag, magic)
register struct inode *rip;     /* pointer to the in-core inode struct */
register d1_inode *dip;         /* pointer to the V1 on-disk inode struct */
register d2_inode *dip2;        /* pointer to the V2 on-disk inode struct */
int rw_flag;                    /* READING or WRITING */
int magic;                      /* magic number of file system */
{
/* Copy the inode from the disk block to the in-core table or vice versa.
 * If the fourth parameter below is FALSE, the bytes are swapped.
 */
  switch (magic) {
        case SUPER_MAGIC:       old_icopy(rip, dip,  rw_flag, TRUE);    break;
        case SUPER_REV:         old_icopy(rip, dip,  rw_flag, FALSE);   break;
        case SUPER_V2:          new_icopy(rip, dip2, rw_flag, TRUE);    break;
        case SUPER_V2_REV:      new_icopy(rip, dip2, rw_flag, FALSE);   break;
  }
}


/*===========================================================================*
 *                              old_icopy                                    *
 *===========================================================================*/
PUBLIC void old_icopy(rip, dip, direction, norm)
register struct inode *rip;     /* pointer to the in-core inode struct */
register d1_inode *dip;         /* pointer to the d1_inode inode struct */
int direction;                  /* READING (from disk) or WRITING (to disk) */
int norm;                       /* TRUE = do not swap bytes; FALSE = swap */

{
/* 4 different on-disk inode layouts are supported, one for each combination
 * of V1.x/V2.x * bytes-swapped/not-swapped.  When an inode is read or written
 * this routine handles the conversions so that the information in the inode
 * table is independent of the disk structure from which the inode came.
 * The old_icopy routine copies to and from V1 disks.
 */

  int i;

  if (direction == READING) {
        /* Copy V1.x inode to the in-core table, swapping bytes if need be. */
        rip->i_mode    = conv2(norm, dip->d1_mode);
        rip->i_uid     = conv2(norm,dip->d1_uid );
        rip->i_size    = conv4(norm,dip->d1_size);
        rip->i_mtime   = conv4(norm,dip->d1_mtime);
        rip->i_atime   = 0;
        rip->i_ctime   = 0;
        rip->i_nlinks  = (nlink_t) dip->d1_nlinks;      /* 1 char */
        rip->i_gid     = (gid_t) dip->d1_gid;           /* 1 char */
        rip->i_ndzones = V1_NR_DZONES;
        rip->i_nindirs = V1_INDIRECTS;
        for (i = 0; i < V1_NR_TZONES; i++)
                rip->i_zone[i] = conv2(norm, (int) dip->d1_zone[i]);
  } else {
        /* Copying V1.x inode to disk from the in-core table. */
        dip->d1_mode   = conv2(norm,rip->i_mode);
        dip->d1_uid    = conv2(norm,rip->i_uid );
        dip->d1_size   = conv4(norm,rip->i_size);
        dip->d1_mtime  = conv4(norm,rip->i_mtime);
        dip->d1_nlinks = (nlink_t) rip->i_nlinks;       /* 1 char */
```

```
        dip->d1_gid    = (gid_t) rip->i_gid;           /* 1 char */
        for (i = 0; i < V1_NR_TZONES; i++)
                dip->d1_zone[i] = conv2(norm, (int) rip->i_zone[i]);
  }
}


/*===========================================================================*
 *                              new_icopy                                    *
 *===========================================================================*/
PUBLIC void new_icopy(rip, dip, direction, norm)
register struct inode *rip;     /* pointer to the in-core inode struct */
register d2_inode *dip; /* pointer to the d2_inode struct */
int direction;                  /* READING (from disk) or WRITING (to disk) */
int norm;                       /* TRUE = do not swap bytes; FALSE = swap */

{
/* Same as old_icopy, but to/from V2 disk layout. */

  int i;

  if (direction == READING) {
        /* Copy V2.x inode to the in-core table, swapping bytes if need be. */
        rip->i_mode    = conv2(norm,dip->d2_mode);
        rip->i_uid     = conv2(norm,dip->d2_uid );
        rip->i_nlinks  = conv2(norm,(int) dip->d2_nlinks);
        rip->i_gid     = conv2(norm,(int) dip->d2_gid );
        rip->i_size    = conv4(norm,dip->d2_size);
        rip->i_atime   = conv4(norm,dip->d2_atime);
        rip->i_ctime   = conv4(norm,dip->d2_ctime);
        rip->i_mtime   = conv4(norm,dip->d2_mtime);
        rip->i_ndzones = V2_NR_DZONES;
        rip->i_nindirs = V2_INDIRECTS;
        for (i = 0; i < V2_NR_TZONES; i++)
                rip->i_zone[i] = conv4(norm, (long) dip->d2_zone[i]);
  } else {
        /* Copying V2.x inode to disk from the in-core table. */
        dip->d2_mode   = conv2(norm,rip->i_mode);
        dip->d2_uid    = conv2(norm,rip->i_uid );
        dip->d2_nlinks = conv2(norm,rip->i_nlinks);
        dip->d2_gid    = conv2(norm,rip->i_gid );
        dip->d2_size   = conv4(norm,rip->i_size);
        dip->d2_atime  = conv4(norm,rip->i_atime);
        dip->d2_ctime  = conv4(norm,rip->i_ctime);
        dip->d2_mtime  = conv4(norm,rip->i_mtime);
        for (i = 0; i < V2_NR_TZONES; i++)
                dip->d2_zone[i] = conv4(norm, (long) rip->i_zone[i]);
  }
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/other/fsversion.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/* This procedure examines a file system and figures out whether it is
 * version 1 or version 2.  It returns the result as an int.  If the
 * file system is neither, it returns -1.  A typical call is:
 *
 *      n = fsversion("/dev/hd1", "df");
 *
 * The first argument is the special file for the file system.
 * The second is the program name, which is used in error messages.
 */

#include <sys/types.h>
#include <minix/config.h>
#include <minix/const.h>
#include <minix/minlib.h>
#include <minix/type.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>

#include "../../fs/const.h"
#include "../../fs/type.h"
#include "../../fs/super.h"

static struct super_block super, *sp;

int fsversion(dev, prog)
char *dev, *prog;
{
  int fd;

  if ((fd = open(dev, O_RDONLY)) < 0) {
        std_err(prog);
        std_err(" cannot open ");
        perror(dev);
        return(-1);
  }

  lseek(fd, (off_t) BLOCK_SIZE, SEEK_SET);      /* skip boot block */
  if (read(fd, (char *) &super, (unsigned) SUPER_SIZE) != SUPER_SIZE) {
        std_err(prog);
        std_err(" cannot read super block on ");
        perror(dev);
        close(fd);
        return(-1);
  }
  close(fd);
  sp = &super;
  if (sp->s_magic == SUPER_MAGIC) return(1);
  if (sp->s_magic == SUPER_V2) return(2);
  return(-1);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/getgrent.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*      getgrent(), getgrgid(), getgrnam() - group file routines
 *
 *                                                      Author: Kees J. Bot
 *                                                              31 Jan 1994
 */
#define nil 0
#define open _open
#define fcntl _fcntl
#define read _read
#define close _close
#include <sys/types.h>
#include <grp.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

#define arraysize(a)    (sizeof(a) / sizeof((a)[0]))
#define arraylimit(a)   ((a) + arraysize(a))

static char GROUP[]= "/etc/group";      /* The group file. */
static const char *grfile;              /* Current group file. */

static char buf[1024];                  /* Read buffer. */
static char grline[512];                /* One line from the group file. */
static struct group entry;              /* Entry to fill and return. */
static char *members[64];               /* Group members with the entry. */
static int grfd= -1;                    /* Filedescriptor to the file. */
static char *bufptr;                    /* Place in buf. */
static ssize_t buflen= 0;               /* Remaining characters in buf. */
static char *lineptr;                   /* Place in the line. */

void endgrent(void)
/* Close the group file. */
{
        if (grfd >= 0) {
                (void) close(grfd);
                grfd= -1;
                buflen= 0;
        }
}

int setgrent(void)
/* Open the group file. */
{
        if (grfd >= 0) endgrent();

        if (grfile == nil) grfile= GROUP;

        if ((grfd= open(grfile, O_RDONLY)) < 0) return -1;
        (void) fcntl(grfd, F_SETFD, fcntl(grfd, F_GETFD) | FD_CLOEXEC);
        return 0;
}

void setgrfile(const char *file)
```

```
/* Prepare for reading an alternate group file. */
{
        endgrent();
        grfile= file;
}

static int getline(void)
/* Get one line from the group file, return 0 if bad or EOF. */
{
        lineptr= grline;

        do {
                if (buflen == 0) {
                        if ((buflen= read(grfd, buf, sizeof(buf))) <= 0)
                                return 0;
                        bufptr= buf;
                }

                if (lineptr == arraylimit(grline)) return 0;
                buflen--;
        } while ((*lineptr++ = *bufptr++) != '\n');

        lineptr= grline;
        return 1;
}

static char *scan_punct(int punct)
/* Scan for a field separator in a line, return the start of the field. */
{
        char *field= lineptr;
        char *last;

        for (;;) {
                last= lineptr;
                if (*lineptr == 0) return nil;
                if (*lineptr == '\n') break;
                if (*lineptr++ == punct) break;
                if (lineptr[-1] == ':') return nil;     /* :::,,,:,,,? */
        }
        *last= 0;
        return field;
}

struct group *getgrent(void)
/* Read one entry from the group file. */
{
        char *p;
        char **mem;

        /* Open the file if not yet open. */
        if (grfd < 0 && setgrent() < 0) return nil;

        /* Until a good line is read. */
        for (;;) {
                if (!getline()) return nil;     /* EOF or corrupt. */

                if ((entry.gr_name= scan_punct(':')) == nil) continue;
                if ((entry.gr_passwd= scan_punct(':')) == nil) continue;
                if ((p= scan_punct(':')) == nil) continue;
                entry.gr_gid= strtol(p, nil, 0);
```

```

                entry.gr_mem= mem= members;
                if (*lineptr != '\n') {
                        do {
                                if ((*mem= scan_punct(',')) == nil) goto again;
                                if (mem < arraylimit(members) - 1) mem++;
                        } while (*lineptr != 0);
                }
                *mem= nil;
                return &entry;
        again:;
        }
}

struct group *getgrgid(Gid_t gid)
/* Return the group file entry belonging to the user-id. */
{
        struct group *gr;

        endgrent();
        while ((gr= getgrent()) != nil && gr->gr_gid != gid) {}
        endgrent();
        return gr;
}

struct group *getgrnam(const char *name)
/* Return the group file entry belonging to the user name. */
{
        struct group *gr;

        endgrent();
        while ((gr= getgrent()) != nil && strcmp(gr->gr_name, name) != 0) {}
        endgrent();
        return gr;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/other/getlogin.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*  getlogin(3)
 *
 *  Author: Terrence W. Holm          Aug. 1988
 */

#include <lib.h>
#include <pwd.h>
#include <unistd.h>
#include <string.h>
#include <stdio.h>

#ifndef  L_cuserid
#define  L_cuserid   9
#endif

```

```
char *getlogin()
{
  PRIVATE char userid[L_cuserid];
  struct passwd *pw_entry;

  pw_entry = getpwuid(getuid());

  if (pw_entry == (struct passwd *)NULL) return((char *)NULL);

  strcpy(userid, pw_entry->pw_name);

  return(userid);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/getopt.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * getopt - parse command-line options
 */
/* $Header: getopt.c,v 1.1 89/12/18 14:39:31 eck Exp $ */

#include        <stdlib.h>
#include        <string.h>
#include        <stdio.h>

#define ERR(s, c)       if(opterr){\
        fputs(argv[0], stderr);\
        fputs(s, stderr);\
        fputc(c, stderr);\
        fputc('\n', stderr);}

int     opterr = 1;
int     optind = 1;
int     optopt;
char    *optarg;

int
getopt(argc, argv, opts)
int argc;
char **argv;
char *opts;
{
        static int sp = 1;
        register c;
        register char *cp;

        if (sp == 1)
                if (optind >= argc ||
                   argv[optind][0] != '-' || argv[optind][1] == '\0')
                        return EOF;
                else if (!strcmp(argv[optind], "--")) {
                        optind++;
                        return EOF;
                }
        optopt = c = argv[optind][sp];
        if (c == ':' || (cp=strchr(opts, c)) == NULL) {
```

```
                        ERR (": illegal option -- ", c);
                        if (argv[optind][++sp] == '\0') {
                                optind++;
                                sp = 1;
                        }
                        return '?';
                }
                if (*++cp == ':') {
                        if (argv[optind][sp+1] != '\0')
                                optarg = &argv[optind++][sp+1];
                        else if (++optind >= argc) {
                                ERR (": option requires an argument -- ", c);
                                sp = 1;
                                return '?';
                        } else
                                optarg = argv[optind++];
                        sp = 1;
                } else {
                        if (argv[optind][++sp] == '\0') {
                                sp = 1;
                                optind++;
                        }
                        optarg = NULL;
                }
                return c;
        }
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/other/getpass.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*      getpass() - read a password             Author: Kees J. Bot
 *                                                      Feb 16 1993
 */
#define open _open
#define sigaction _sigaction
#define sigemptyset _sigemptyset
#define tcgetattr _tcgetattr
#define tcsetattr _tcsetattr
#define write _write
#define read _read
#define close _close
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <signal.h>
#include <termios.h>
#include <string.h>

static int intr;

static void catch(int sig)
{
        intr= 1;
}
```

```
char *getpass(const char *prompt)
{
        struct sigaction osa, sa;
        struct termios cooked, raw;
        static char password[32+1];
        int fd, n= 0;

        /* Try to open the controlling terminal. */
        if ((fd= open("/dev/tty", O_RDONLY)) < 0) return NULL;

        /* Trap interrupts unless ignored. */
        intr= 0;
        sigaction(SIGINT, NULL, &osa);
        if (osa.sa_handler != SIG_IGN) {
                sigemptyset(&sa.sa_mask);
                sa.sa_flags= 0;
                sa.sa_handler= catch;
                sigaction(SIGINT, &sa, &osa);
        }

        /* Set the terminal to non-echo mode. */
        tcgetattr(fd, &cooked);
        raw= cooked;
        raw.c_iflag|= ICRNL;
        raw.c_lflag&= ~ECHO;
        raw.c_lflag|= ECHONL;
        raw.c_oflag|= OPOST | ONLCR;
        tcsetattr(fd, TCSANOW, &raw);

        /* Print the prompt.  (After setting non-echo!) */
        write(2, prompt, strlen(prompt));

        /* Read the password, 32 characters max. */
        while (read(fd, password+n, 1) > 0) {
                if (password[n] == '\n') break;
                if (n < 32) n++;
        }
        password[n]= 0;

        /* Terminal back to cooked mode. */
        tcsetattr(fd, TCSANOW, &cooked);

        close(fd);

        /* Interrupt? */
        sigaction(SIGINT, &osa, NULL);
        if (intr) raise(SIGINT);

        return password;
}
```

++
src/lib/other/getpwent.c
++

```
/*      getpwent(), getpwuid(), getpwnam() - password file routines
 *
 *                                                          Author: Kees J. Bot
 *                                                                  31 Jan 1994
 */
#define nil 0
#define open _open
#define fcntl _fcntl
#define read _read
#define close _close
#include <sys/types.h>
#include <pwd.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

#define arraysize(a)    (sizeof(a) / sizeof((a)[0]))
#define arraylimit(a)   ((a) + arraysize(a))

static char PASSWD[]= "/etc/passwd";    /* The password file. */
static const char *pwfile;              /* Current password file. */

static char buf[1024];                  /* Read buffer. */
static char pwline[256];                /* One line from the password file. */
static struct passwd entry;             /* Entry to fill and return. */
static int pwfd= -1;                    /* Filedescriptor to the file. */
static char *bufptr;                    /* Place in buf. */
static ssize_t buflen= 0;               /* Remaining characters in buf. */
static char *lineptr;                   /* Place in the line. */

void endpwent(void)
/* Close the password file. */
{
        if (pwfd >= 0) {
                (void) close(pwfd);
                pwfd= -1;
                buflen= 0;
        }
}

int setpwent(void)
/* Open the password file. */
{
        if (pwfd >= 0) endpwent();

        if (pwfile == nil) pwfile= PASSWD;

        if ((pwfd= open(pwfile, O_RDONLY)) < 0) return -1;
        (void) fcntl(pwfd, F_SETFD, fcntl(pwfd, F_GETFD) | FD_CLOEXEC);
        return 0;
}

void setpwfile(const char *file)
/* Prepare for reading an alternate password file. */
```

```
{
        endpwent();
        pwfile= file;
}

static int getline(void)
/* Get one line from the password file, return 0 if bad or EOF. */
{
        lineptr= pwline;

        do {
                if (buflen == 0) {
                        if ((buflen= read(pwfd, buf, sizeof(buf))) <= 0)
                                return 0;
                        bufptr= buf;
                }

                if (lineptr == arraylimit(pwline)) return 0;
                buflen--;
        } while ((*lineptr++ = *bufptr++) != '\n');

        lineptr= pwline;
        return 1;
}

static char *scan_colon(void)
/* Scan for a field separator in a line, return the start of the field. */
{
        char *field= lineptr;
        char *last;

        for (;;) {
                last= lineptr;
                if (*lineptr == 0) return nil;
                if (*lineptr == '\n') break;
                if (*lineptr++ == ':') break;
        }
        *last= 0;
        return field;
}

struct passwd *getpwent(void)
/* Read one entry from the password file. */
{
        char *p;

        /* Open the file if not yet open. */
        if (pwfd < 0 && setpwent() < 0) return nil;

        /* Until a good line is read. */
        for (;;) {
                if (!getline()) return nil;     /* EOF or corrupt. */

                if ((entry.pw_name= scan_colon()) == nil) continue;
                if ((entry.pw_passwd= scan_colon()) == nil) continue;
                if ((p= scan_colon()) == nil) continue;
                entry.pw_uid= strtol(p, nil, 0);
                if ((p= scan_colon()) == nil) continue;
                entry.pw_gid= strtol(p, nil, 0);
                if ((entry.pw_gecos= scan_colon()) == nil) continue;
```

```c
                if ((entry.pw_dir= scan_colon()) == nil) continue;
                if ((entry.pw_shell= scan_colon()) == nil) continue;

                if (*lineptr == 0) return &entry;
        }
}

struct passwd *getpwuid(Uid_t uid)
/* Return the password file entry belonging to the user-id. */
{
        struct passwd *pw;

        endpwent();
        while ((pw= getpwent()) != nil && pw->pw_uid != uid) {}
        endpwent();
        return pw;
}

struct passwd *getpwnam(const char *name)
/* Return the password file entry belonging to the user name. */
{
        struct passwd *pw;

        endpwent();
        while ((pw= getpwent()) != nil && strcmp(pw->pw_name, name) != 0) {}
        endpwent();
        return pw;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/getttyent.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*      getttyent(3) - get a ttytab entry               Author: Kees J. Bot
 *                                                               28 Oct 1995
 */
#define nil 0
#define open _open
#define close _close
#define fcntl _fcntl
#define read _read
#include <string.h>
#include <sys/types.h>
#include <ttyent.h>
#include <unistd.h>
#include <fcntl.h>

#define arraysize(a)    (sizeof(a) / sizeof((a)[0]))
#define arraylimit(a)   ((a) + arraysize(a))

static char TTYTAB[]= "/etc/ttytab";    /* The table of terminal devices. */

static char buf[512];                   /* Read buffer. */
static char ttline[256];                /* One line from the ttytab file. */
static char *ttargv[32];                /* Compound arguments. */
static struct ttyent entry;             /* Entry to fill and return. */
static int ttfd= -1;                    /* Filedescriptor to the file. */
static char *bufptr;                    /* Place in buf. */
```

```
static ssize_t buflen= 0;               /* Remaining characters in buf. */
static char *lineptr;                   /* Place in the line. */
static char **argvptr;                  /* Place in word lists. */

void endttyent(void)
/* Close the ttytab file. */
{
        if (ttfd >= 0) {
                (void) close(ttfd);
                ttfd= -1;
                buflen= 0;
        }
}

int setttyent(void)
/* Open the ttytab file. */
{
        if (ttfd >= 0) endttyent();

        if ((ttfd= open(TTYTAB, O_RDONLY)) < 0) return -1;
        (void) fcntl(ttfd, F_SETFD, fcntl(ttfd, F_GETFD) | FD_CLOEXEC);
        return 0;
}

static int getline(void)
/* Get one line from the ttytab file, return 0 if bad or EOF. */
{
        lineptr= ttline;
        argvptr= ttargv;

        do {
                if (buflen == 0) {
                        if ((buflen= read(ttfd, buf, sizeof(buf))) <= 0)
                                return 0;
                        bufptr= buf;
                }

                if (lineptr == arraylimit(ttline)) return 0;
                buflen--;
        } while ((*lineptr++ = *bufptr++) != '\n');

        lineptr= ttline;
        return 1;
}

static int white(int c)
/* Whitespace? */
{
        return c == ' ' || c == '\t';
}

static char *scan_white(int quoted)
/* Scan for a field separator in a line, return the start of the field.
 * "quoted" is set if we have to watch out for double quotes.
 */
{
        char *field, *last;

        while (white(*lineptr)) lineptr++;
        if (!quoted && *lineptr == '#') return nil;
```

```
        field= lineptr;
        for (;;) {
                last= lineptr;
                if (*lineptr == 0) return nil;
                if (*lineptr == '\n') break;
                if (quoted && *lineptr == '"') return field;
                if (white(*lineptr++)) break;
        }
        *last= 0;
        return *field == 0 ? nil : field;
}

static char **scan_quoted(void)
/* Read a field that may be a quoted list of words. */
{
        char *p, **field= argvptr;

        while (white(*lineptr)) lineptr++;

        if (*lineptr == '"') {
                /* Quoted list of words. */
                lineptr++;
                while ((p= scan_white(1)) != nil && *p != '"') {
                        if (argvptr == arraylimit(ttargv)) return nil;
                        *argvptr++= p;
                }
                if (*lineptr == '"') *lineptr++= 0;
        } else {
                /* Just one word. */
                if ((p= scan_white(0)) == nil) return nil;
                if (argvptr == arraylimit(ttargv)) return nil;
                *argvptr++= p;
        }
        if (argvptr == arraylimit(ttargv)) return nil;
        *argvptr++= nil;
        return field;
}

struct ttyent *getttyent(void)
/* Read one entry from the ttytab file. */
{
        /* Open the file if not yet open. */
        if (ttfd < 0 && setttyent() < 0) return nil;

        /* Look for a line with something on it. */
        for (;;) {
                if (!getline()) return nil;     /* EOF or corrupt. */

                if ((entry.ty_name= scan_white(0)) == nil) continue;
                entry.ty_type= scan_white(0);
                entry.ty_getty= scan_quoted();
                entry.ty_init= scan_quoted();

                return &entry;
        }
}

struct ttyent *getttynam(const char *name)
/* Return the ttytab file entry for a given tty. */
```

```
{
        struct ttyent *tty;

        endttyent();
        while ((tty= getttyent()) != nil && strcmp(tty->ty_name, name) != 0) {}
        endttyent();
        return tty;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/other/getw.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * getw - read a word from a stream
 */
/* $Header: getw.c,v 1.1 89/12/18 14:39:51 eck Exp $ */

#include        <stdio.h>

_PROTOTYPE(int getw, (FILE *stream ));

int getw(stream)
register FILE *stream;
{
        register int cnt = sizeof(int);
        int w;
        register char *p = (char *) &w;

        while (cnt--) {
                *p++ = getc(stream);
        }
        if (feof(stream) || ferror(stream)) return EOF;
        return w;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/other/hypot.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1988 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 *
 * Author: Ceriel J.H. Jacobs
 */

#include <math.h>

struct complex {
        double r,i;
};

_PROTOTYPE(double hypot, (double x, double y ));
_PROTOTYPE(double cabs, (struct complex p_compl ));
```

```

/* $Header: hypot.c,v 1.1 91/02/26 18:08:08 ceriel Exp $ */

double
hypot(x, y)
double x, y;
{
        /*      Computes sqrt(x*x+y*y), avoiding overflow */

        if (x < 0) x = -x;
        if (y < 0) y = -y;
        if (x > y) {
                double t = y;
                y = x;
                x = t;
        }
        /* sqrt(x*x+y*y) = sqrt(y*y*(x*x/(y*y)+1.0)) = y*sqrt(x*x/(y*y)+1.0) */
        if (y == 0.0) return 0.0;
        x /= y;
        return y*sqrt(x*x+1.0);
}

double
cabs(p_compl)
struct complex p_compl;
{
        return hypot(p_compl.r, p_compl.i);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/index.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
/* index - find first occurrence of a character in a string */

#include <string.h>

char *index(s, charwanted)      /* found char, or NULL if none */
_CONST char *s;
char charwanted;
{
  return(strchr(s, charwanted));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                           src/lib/other/itoa.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
/* Integer to ASCII for signed decimal integers. */

PRIVATE int next;
PRIVATE char qbuf[8];

_PROTOTYPE( char *itoa, (int n));

char *itoa(n)
int n;
{
  register int r, k;
  int flag = 0;

  next = 0;
  if (n < 0) {
        qbuf[next++] = '-';
        n = -n;
  }
  if (n == 0) {
        qbuf[next++] = '0';
  } else {
        k = 10000;
        while (k > 0) {
                r = n / k;
                if (flag || r > 0) {
                        qbuf[next++] = '0' + r;
                        flag = 1;
                }
                n -= r * k;
                k = k / 10;
        }
  }
  qbuf[next] = 0;
  return(qbuf);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                           src/lib/other/loadname.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#include <string.h>

PUBLIC void _loadname(name, msgptr)
_CONST char *name;
message *msgptr;
{
/* This function is used to load a string into a type m3 message. If the
 * string fits in the message, it is copied there.  If not, a pointer to
 * it is passed.
```

```
 */

  register size_t k;

  k = strlen(name) + 1;
  msgptr->m3_i1 = k;
  msgptr->m3_p1 = (char *) name;
  if (k <= sizeof msgptr->m3_ca1) strcpy(msgptr->m3_ca1, name);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/lock.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#include <errno.h>
#include <sys/types.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <stdio.h>
#if _ANSI
#include <stdlib.h>
#endif

typedef enum {
  False, True
} BOOLEAN;

#define LOCKDIR "/tmp/"         /* or /usr/tmp/ as the case may be */
#define MAXTRIES 3
#define NAPTIME (unsigned int)5

PRIVATE _PROTOTYPE( char *lockpath, (char *name));
_PROTOTYPE( void syserr, (char *errstring));
_PROTOTYPE( BOOLEAN lock, (char *name));
_PROTOTYPE( void unlock, (char *name));

void
syserr(errstring)
char *errstring;
{
        fprintf(stderr,"couldn't %s\n", errstring);
        exit(1);
}

BOOLEAN lock(name)              /* acquire lock */
char *name;
{
  char *path;
  int fd, tries;

  path = lockpath(name);
  tries = 0;
  while ((fd = creat(path, 0)) == -1 && errno == EACCES) {
        if (++tries >= MAXTRIES) return(False);
        sleep(NAPTIME);
  }
  if (fd == -1 || close(fd) == -1) syserr("lock");
```

```
  return(True);
}

void unlock(name)               /* free lock */
char *name;
{
  if (unlink(lockpath(name)) == -1) syserr("unlock");
}

PRIVATE char *lockpath(name)    /* generate lock file path */
char *name;
{
  PRIVATE char path[20];

  strcpy(path, LOCKDIR);
  return(strcat(path, name));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/lrand.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*  lrand(3)
 *
 *  Author: Terrence W. Holm          Nov. 1988
 *
 *
 *  A prime modulus multiplicative linear congruential
 *  generator (PMMLCG), or "Lehmer generator".
 *  Implementation directly derived from the article:
 *
 *      S. K. Park and K. W. Miller
 *      Random Number Generators: Good Ones are Hard to Find
 *      CACM vol 31, #10. Oct. 1988. pp 1192-1201.
 *
 *
 *  Using the following multiplier and modulus, we obtain a
 *  generator which:
 *
 *      1)  Has a full period: 1 to 2^31 - 2.
 *      2)  Is testably "random" (see the article).
 *      3)  Has a known implementation by E. L. Schrage.
 */

#include <lib.h>

_PROTOTYPE( long seed, (long lseed));
_PROTOTYPE( long lrand, (void));

#define  A        16807L        /* A "good" multiplier    */
#define  M   2147483647L        /* Modulus: 2^31 - 1      */
#define  Q       127773L        /* M / A                  */
#define  R         2836L        /* M % A                  */

PRIVATE long _lseed = 1L;

long seed(lseed)
```

```
long lseed;
{
  long previous_seed = _lseed;

  _lseed = lseed;

  return(previous_seed);
}


long lrand()
{
  _lseed = A * (_lseed % Q) - R * (_lseed / Q);

  if (_lseed < 0) _lseed += M;

  return(_lseed);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/other/lsearch.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#include <string.h>
/*  lsearch(3)  and  lfind(3)
 *
 *  Author: Terrence W. Holm          Sep. 1988
 */

#include <stddef.h>

_PROTOTYPE( char *lsearch, (char *key, char *base,
                            unsigned *count, unsigned width,
                            int (*keycmp)(const void *, const void *)));
_PROTOTYPE( char *lfind, (char *key, char *base,
                            unsigned *count, unsigned width,
                            int (*keycmp)(const void *, const void *)));

char *lsearch(key, base, count, width, keycmp)
char *key;
char *base;
unsigned *count;
unsigned width;
_PROTOTYPE( int (*keycmp), (const void *, const void *));
{
  char *entry;
  char *last = base + *count * width;

  for (entry = base; entry < last; entry += width)
        if (keycmp(key, entry) == 0) return(entry);

  bcopy(key, last, width);
  *count += 1;
  return(last);
}
```

```
char *lfind(key, base, count, width, keycmp)
char *key;
char *base;
unsigned *count;
unsigned width;
_PROTOTYPE( int (*keycmp), (const void *, const void *));
{
  char *entry;
  char *last = base + *count * width;

  for (entry = base; entry < last; entry += width)
        if (keycmp(key, entry) == 0) return(entry);

  return((char *)NULL);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/other/memccpy.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
/* memccpy - copy bytes up to a certain char
 *
 * CHARBITS should be defined only if the compiler lacks "unsigned char".
 * It should be a mask, e.g. 0377 for an 8-bit machine.
 */

#include <ansi.h>
#include <stddef.h>

_PROTOTYPE( void *memccpy, (void *dst, const void *src,
                            int ucharstop, size_t size));
#ifndef CHARBITS
#       define  UNSCHAR(c)      ((unsigned char)(c))
#else
#       define  UNSCHAR(c)      ((c)&CHARBITS)
#endif

void *memccpy(dst, src, ucharstop, size)
void * dst;
_CONST void * src;
int ucharstop;
_SIZET size;
{
  register char *d;
  register _CONST char *s;
  register _SIZET n;
  register int uc;

  if (size <= 0) return( (void *) NULL);

  s = (char *) src;
  d = (char *) dst;
  uc = UNSCHAR(ucharstop);
  for (n = size; n > 0; n--)
```

```
        if (UNSCHAR(*d++ = *s++) == (char) uc) return( (void *) d);

  return( (void *) NULL);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/other/mtab.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/* This package consists of 4 routines for handling the /etc/mtab file.
 * The /etc/mtab file contains information about the root and mounted file
 * systems as a series of lines, each one with exactly four fields separated
 * by one space as follows:
 *
 *      special mounted_on version rw_flag
 *
 * where
 *      special is the name of the block special file
 *      mounted_on is the directory on which it is mounted
 *      version is either 1 or 2 for MINIX V1 and V2 file systems
 *      rw_flag is rw or ro for read/write or read only
 *
 * An example /etc/mtab:
 *
 *      /dev/ram / 2 rw
 *      /dev/hd1 /usr 2 rw
 *      /dev/fd0 /user 1 ro
 *
 *
 * The four routines for handling /etc/mtab are as follows.  They use two
 * (hidden) internal buffers, mtab_in for input and mtab_out for output.
 *
 *      load_mtab(&prog_name)              - read /etc/mtab into mtab_in
 *      get_mtab_entry(&s1, &s2, &s3, &s4) - arrays that are filled in
 *      put_mtab_entry(&s1, &s2, &s3, &s4) - append a line to mtab_out
 *      rewrite_mtab(&prog_name)           - write mtab_out to /etc/mtab
 *
 * If load_mtab and rewrite_mtab work, they return 0.  If they fail, they
 * print their own error messages on stderr and return -1.  When get_mtab_entry
 * runs out of entries to return, it sets the first pointer to NULL and returns
 * -1 instead of 0.  Also, rewrite_mtab returns -1 if it fails.
 */

#include <sys/types.h>
#include <minix/minlib.h>
#include <ctype.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <stdio.h>

#define BUF_SIZE   512            /* size of the /etc/mtab buffer */

char *etc_mtab = "/etc/mtab";     /* name of the /etc/mtab file */
static char mtab_in[BUF_SIZE+1];  /* holds /etc/mtab when it is read in */
static char mtab_out[BUF_SIZE+1]; /* buf to build /etc/mtab for output later */
static char *iptr = mtab_in;      /* pointer to next line to feed out. */
static char *optr = mtab_out;     /* pointer to place where next line goes */
```

```

_PROTOTYPE(int load_mtab, (char *prog_name ));
_PROTOTYPE(int rewrite_mtab, (char *prog_name ));
_PROTOTYPE(int get_mtab_entry, (char *special, char *mounted_on,
                                        char *version, char *rw_flag));
_PROTOTYPE(int put_mtab_entry, (char *special, char *mounted_on,
                                        char *version, char *rw_flag));
_PROTOTYPE(void err, (char *prog_name, char *str ));


int load_mtab(prog_name)
char *prog_name;
{
/* Read in /etc/mtab and store it in /etc/mtab. */

  int fd, n;
  char *ptr;

  /* Open the file. */
  fd = open(etc_mtab, O_RDONLY);
  if (fd < 0) {
        err(prog_name, ": cannot open ");
        return(-1);
  }

  /* File opened.  Read it in. */
  n = read(fd, mtab_in, BUF_SIZE);
  if (n <= 0) {
        /* Read failed. */
        err(prog_name, ": cannot read ");
        return(-1);
  }
  if (n == BUF_SIZE) {
        /* Some nut has mounted 50 file systems or something like that. */
        std_err(prog_name);
        std_err(": file too large: ");
        std_err(etc_mtab);
        return(-1);
  }

  close(fd);

  /* Replace all the whitespace by '\0'. */
  ptr = mtab_in;
  while (*ptr != '\0') {
        if (isspace(*ptr)) *ptr = '\0';
        ptr++;
  }
  return(0);
}


int rewrite_mtab(prog_name)
char *prog_name;
{
/* Write mtab_out to /etc/mtab. */

  int fd, n;

  /* Do a creat to truncate the file. */
```

```
  fd = creat(etc_mtab, 0777);
  if (fd < 0) {
        err(prog_name, ": cannot overwrite ");
        return(-1);
  }

  /* File created.  Write it. */
  n = write(fd, mtab_out, (unsigned int)(optr - mtab_out));
  if (n <= 0) {
        /* Write failed. */
        err(prog_name, " could not write ");
        return(-1);
  }

  close(fd);
  return(0);
}


int get_mtab_entry(special, mounted_on, version, rw_flag)
char *special;
char *mounted_on;
char *version;
char *rw_flag;
{
/* Return the next entry from mtab_in. */

  if (iptr >= &mtab_in[BUF_SIZE]) {
        special[0] = ’\0’;
        return(-1);
  }

  strcpy(special, iptr);
  while (isprint(*iptr)) iptr++;
  while (*iptr == ’\0’&& iptr < &mtab_in[BUF_SIZE]) iptr++;

  strcpy(mounted_on, iptr);
  while (isprint(*iptr)) iptr++;
  while (*iptr == ’\0’&& iptr < &mtab_in[BUF_SIZE]) iptr++;

  strcpy(version, iptr);
  while (isprint(*iptr)) iptr++;
  while (*iptr == ’\0’&& iptr < &mtab_in[BUF_SIZE]) iptr++;

  strcpy(rw_flag, iptr);
  while (isprint(*iptr)) iptr++;
  while (*iptr == ’\0’&& iptr < &mtab_in[BUF_SIZE]) iptr++;
  return(0);
}


int put_mtab_entry(special, mounted_on, version, rw_flag)
char *special;
char *mounted_on;
char *version;
char *rw_flag;
{
/* Append an entry to the mtab_out buffer. */

  int n1, n2, n3, n4;
```

```

  n1 = strlen(special);
  n2 = strlen(mounted_on);
  n3 = strlen(version);
  n4 = strlen(rw_flag);

  if (optr + n1 + n2 + n3 + n4 + 5 >= &mtab_out[BUF_SIZE]) return(-1);
  strcpy(optr, special);
  optr += n1;
  *optr++ = ' ';

  strcpy(optr, mounted_on);
  optr += n2;
  *optr++ = ' ';

  strcpy(optr, version);
  optr += n3;
  *optr++ = ' ';

  strcpy(optr, rw_flag);
  optr += n4;
  *optr++ = '\n';
  return(0);
}


void
err(prog_name, str)
char *prog_name, *str;
{
  std_err(prog_name);
  std_err(str);
  std_err(etc_mtab);
  perror(" ");
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/nlist.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * "nlist.c", Peter Valkenburg, january 1989.
 */

#include <lib.h>
#include <string.h>
#include <a.out.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>

#define fail(fp)        (fclose(fp), -1)        /* ret. exp. when nlist fails */

_PROTOTYPE( int nlist, (char *file, struct nlist nl[]));
```

```

/*
 * Nlist fills fields n_sclass and n_value of array nl with values found in
 * non-stripped executable file.  Entries that are not found have their
 * n_value/n_sclass fields set to 0.  Nl ends with a 0 or nul string n_name.
 * The return value is -1 on failure, else the number of entries not found.
 */
int nlist(file, nl)
char *file;
struct nlist nl[];
{
        int nents, nsrch, nfound, i;
        struct nlist nlent;
        FILE *fp;
        struct exec hd;

        /* open executable with namelist */
        if ((fp = fopen(file, "r")) == NULL)
                return -1;

        /* get header and seek to start of namelist */
        if (fread((char *) &hd, sizeof(struct exec), 1, fp) != 1 ||
            BADMAG(hd) || fseek(fp, A_SYMPOS(hd), SEEK_SET) != 0)
                return fail(fp);

        /* determine number of entries searched for & reset fields */
        nsrch = 0;
        while (nl[nsrch].n_name != NULL && *(nl[nsrch].n_name) != '\0') {
                nl[nsrch].n_sclass = 0;
                nl[nsrch].n_value = 0;
                nl[nsrch].n_type = 0;           /* for compatability */
                nsrch++;
        }

        /* loop through namelist & fill in user array */
        nfound = 0;
        for (nents = (hd.a_syms & 0xFFFF) / sizeof(struct nlist);
             nents > 0; nents--) {
                if (nsrch == nfound)
                        break;                  /* no need to look further */
                if (fread((char *) &nlent, sizeof(struct nlist), 1, fp) != 1)
                        return fail(fp);
                for (i = 0; i < nsrch; i++)
                        if (nl[i].n_sclass == 0 &&
                            strncmp(nl[i].n_name, nlent.n_name,
                                    sizeof(nlent.n_name)) == 0) {
                                nl[i] = nlent;
                                nfound++;
                                break;
                        }
        }

        (void) fclose(fp);

        return nsrch - nfound;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/peekpoke.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/* Peek and poke using /dev/mem.
 *
 * Callers now ought to check the return values.
 *
 * Calling peek() requires read permission on /dev/mem, and consumes
 * a file descriptor.  Calling poke() requires write permission, and
 * consumes another file descriptor.
 */

#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>

_PROTOTYPE( int peek, (unsigned segment, unsigned offset));
_PROTOTYPE( int poke, (unsigned segment, unsigned offset, unsigned value));

#define SEGSIZE 0x10

int peek(segment, offset)
unsigned segment;
unsigned offset;
{
  unsigned char chvalue;
  static int infd = -1;

  if (infd < 0) infd = open("/dev/mem", O_RDONLY);
  if (infd < 0 ||
      lseek(infd, (unsigned long) segment * SEGSIZE + offset, SEEK_SET) < 0 ||
      read(infd, (char *) &chvalue, (unsigned) 1) != 1)
        return(-1);
  return(chvalue);
}

int poke(segment, offset, value)
unsigned segment;
unsigned offset;
unsigned value;
{
  unsigned char chvalue;
  static int outfd = -1;

  chvalue = value;
  if (outfd < 0) outfd = open("/dev/mem", O_WRONLY);
  if (outfd < 0 ||
      lseek(outfd, (unsigned long) segment * SEGSIZE + offset, SEEK_SET) < 0 ||
      write(outfd, (char *) &chvalue, (unsigned) 1) != 1)
        return(-1);
  return(chvalue);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/popen.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * popen - open a pipe
 */
/* $Header: popen.c,v 1.3 90/08/28 14:53:34 eck Exp $ */

#include        <sys/types.h>
#include        <limits.h>
#include        <errno.h>
#include        <signal.h>
#include        <stdio.h>

#if     defined(__BSD4_2)
union wait {
        int     w_status;
};
typedef union wait wait_arg;
#else
typedef int wait_arg;
#endif  /* __BSD4_2 */

#include        "../stdio/loc_incl.h"

#ifdef _ANSI
int _close(int d);
int _dup2(int oldd, int newd);          /* not present in System 5 */
int _execl(const char *name, const char *_arg, ... );
pid_t _fork(void);
int _pipe(int fildes[2]);
pid_t _wait(wait_arg *status);
void _exit(int status);
#endif

static int pids[OPEN_MAX];

FILE *
popen(command, type)
_CONST char *command;
_CONST char *type;
{
        int piped[2];
        int Xtype = *type == 'r' ? 0 : *type == 'w' ? 1 : 2;
        int pid;

        if (Xtype == 2 ||
            _pipe(piped) < 0 ||
            (pid = _fork()) < 0) return 0;

        if (pid == 0) {
                /* child */
                register int *p;

                for (p = pids; p < &pids[OPEN_MAX]; p++) {
                        if (*p) _close((int)(p - pids));
                }
                _close(piped[Xtype]);
```

```
                        _dup2(piped[!Xtype], !Xtype);
                        _close(piped[!Xtype]);
                        _execl("/bin/sh", "sh", "-c", command, (char *) 0);
                        _exit(127);     /* like system() ??? */
                }

                pids[piped[Xtype]] = pid;
                _close(piped[!Xtype]);
                return fdopen(piped[Xtype], type);
        }

        #if     defined(__BSD4_2)
        #define ret_val status.w_status
        #else
        #define ret_val status
        #endif

        int
        pclose(stream)
        FILE *stream;
        {
                int fd = fileno(stream);
                wait_arg status;
                int wret;

        #ifdef _ANSI
                void (*intsave)(int) = signal(SIGINT, SIG_IGN);
                void (*quitsave)(int) = signal(SIGQUIT, SIG_IGN);
        #else
                void (*intsave)() = signal(SIGINT, SIG_IGN);
                void (*quitsave)() = signal(SIGQUIT, SIG_IGN);
        #endif
                fclose(stream);
                while ((wret = _wait(&status)) != -1) {
                        if (wret == pids[fd]) break;
                }
                if (wret == -1) ret_val = -1;
                signal(SIGINT, intsave);
                signal(SIGQUIT, quitsave);
                pids[fd] = 0;
                return ret_val;
        }

        #if     defined(__USG)
        int _dup(int fildes);

        static int
        _dup2(oldd, newd)
        int oldd, newd;
        {
                int i = 0, fd, tmp;
                int fdbuf[_NFILES];

                /* ignore the error on the close() */
                tmp = errno; (void) _close(newd); errno = tmp;
                while ((fd = _dup(oldd)) != newd) {
                        if (fd == -1) break;
                        fdbuf[i++] = fd;
                }
                tmp = errno;
```

```
        while (--i >= 0) {
                _close(fdbuf[i]);
        }
        errno = tmp;
        return -(fd == -1);
}
#endif  /* __USG */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/other/printk.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*      printk() - kernel printf()                      Author: Kees J. Bot
 *                                                              15 Jan 1994
 */
#define nil 0
#include <stdarg.h>
#include <limits.h>

#define isdigit(c)      ((unsigned) ((c) - '0') <  (unsigned) 10)

#if !__STDC__
/* Classic C stuff, ignore. */
void putk();
void printk(fmt) char *fmt;
#else

/* Printk() uses putk() to print characters. */
void putk(int c);

void printk(const char *fmt, ...)
#endif
{
        int c;
        enum { LEFT, RIGHT } adjust;
        enum { LONG, INT } intsize;
        int fill;
        int width, max, len, base;
        static char X2C_tab[]= "0123456789ABCDEF";
        static char x2c_tab[]= "0123456789abcdef";
        char *x2c;
        char *p;
        long i;
        unsigned long u;
        char temp[8 * sizeof(long) / 3 + 2];

        va_list argp;

        va_start(argp, fmt);

        while ((c= *fmt++) != 0) {
                if (c != '%') {
                        /* Ordinary character. */
                        putk(c);
                        continue;
                }

```

```
                /* Format specifier of the form:
                 *      %[adjust][fill][width][.max]keys
                 */
                c= *fmt++;

                adjust= RIGHT;
                if (c == '-') {
                        adjust= LEFT;
                        c= *fmt++;
                }

                fill= ' ';
                if (c == '0') {
                        fill= '0';
                        c= *fmt++;
                }

                width= 0;
                if (c == '*') {
                        /* Width is specified as an argument, e.g. %*d. */
                        width= va_arg(argp, int);
                        c= *fmt++;
                } else
                if (isdigit(c)) {
                        /* A number tells the width, e.g. %10d. */
                        do {
                                width= width * 10 + (c - '0');
                        } while (isdigit(c= *fmt++));
                }

                max= INT_MAX;
                if (c == '.') {
                        /* Max field length coming up. */
                        if ((c= *fmt++) == '*') {
                                max= va_arg(argp, int);
                                c= *fmt++;
                        } else
                        if (isdigit(c)) {
                                max= 0;
                                do {
                                        max= max * 10 + (c - '0');
                                } while (isdigit(c= *fmt++));
                        }
                }

                /* Set a few flags to the default. */
                x2c= x2c_tab;
                i= 0;
                base= 10;
                intsize= INT;
                if (c == 'l' || c == 'L') {
                        /* "Long" key, e.g. %ld. */
                        intsize= LONG;
                        c= *fmt++;
                }
                if (c == 0) break;

                switch (c) {
                        /* Decimal.  Note that %D is treated as %ld. */
                case 'D':
```

```
                        intsize= LONG;
                case 'd':
                        i= intsize == LONG ? va_arg(argp, long)
                                                : va_arg(argp, int);
                        u= i < 0 ? -i : i;
                        goto int2ascii;

                        /* Octal. */
                case 'O':
                        intsize= LONG;
                case 'o':
                        base= 010;
                        goto getint;

                        /* Hexadecimal.  %X prints upper case A-F, not %lx. */
                case 'X':
                        x2c= X2C_tab;
                case 'x':
                        base= 0x10;
                        goto getint;

                        /* Unsigned decimal. */
                case 'U':
                        intsize= LONG;
                case 'u':
                getint:
                        u= intsize == LONG ? va_arg(argp, unsigned long)
                                                : va_arg(argp, unsigned int);
                int2ascii:
                        p= temp + sizeof(temp)-1;
                        *p= 0;
                        do {
                                *--p= x2c[u % base];
                        } while ((u /= base) > 0);
                        goto string_length;

                        /* A character. */
                case 'c':
                        p= temp;
                        *p= va_arg(argp, int);
                        len= 1;
                        goto string_print;

                        /* Simply a percent. */
                case '%':
                        p= temp;
                        *p= '%';
                        len= 1;
                        goto string_print;

                        /* A string.  The other cases will join in here. */
                case 's':
                        p= va_arg(argp, char *);

                string_length:
                        for (len= 0; p[len] != 0 && len < max; len++) {}

                string_print:
                        width -= len;
                        if (i < 0) width--;
```

```
                                if (fill == '0' && i < 0) putk('-');
                                if (adjust == RIGHT) {
                                        while (width > 0) { putk(fill); width--; }
                                }
                                if (fill == ' ' && i < 0) putk('-');
                                while (len > 0) { putk((unsigned char) *p++); len--; }
                                while (width > 0) { putk(fill); width--; }
                                break;

                                /* Unrecognized format key, echo it back. */
                        default:
                                putk('%');
                                putk(c);
                        }
                }

                /* Mark the end with a null (should be something else, like -1). */
                putk(0);
                va_end(argp);
        }

        /*
         * $PchHeader: /mount/hd2/minix/lib/minix/other/RCS/printk.c,v 1.2 1994/09/07 18:45:05 philip Exp $
         */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/other/putenv.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * (c) copyright 1989 by the Vrije Universiteit, Amsterdam, The Netherlands.
 * See the copyright notice in the ACK home directory, in the file "Copyright".
 */
/* $Header: putenv.c,v 1.3 90/05/14 12:30:18 ceriel Exp $ */

#include        <stdlib.h>
#include        <string.h>

#define ENTRY_INC       10
#define rounded(x)      (((x / ENTRY_INC) + 1) * ENTRY_INC)

extern _CONST char **environ;

int
putenv(name)
_CONST char *name;
{
        register _CONST char **v = environ;
        register char *r;
        static int size = 0;
        /* When size != 0, it contains the number of entries in the
         * table (including the final NULL pointer). This means that the
         * last non-null entry  is environ[size - 2].
         */

        if (!name) return 0;
        if (environ == NULL) return 1;
        if (r = strchr(name, '=')) {
                register _CONST char *p, *q;
```

```

                *r = '\0';

                if (v != NULL) {
                        while ((p = *v) != NULL) {
                                q = name;
                                while (*q && (*q++ == *p++))
                                        /* EMPTY */ ;
                                if (*q || (*p != '=')) {
                                        v++;
                                } else {
                                        /* The name was already in the
                                         * environment.
                                         */
                                        *r = '=';
                                        *v = name;
                                        return 0;
                                }
                        }
                }
                *r = '=';
                v = environ;
        }

        if (!size) {
                register _CONST char **p;
                register int i = 0;

                if (v)
                        do {
                                i++;
                        } while (*v++);
                if (!(v = malloc(rounded(i) * sizeof(char **))))
                        return 1;
                size = i;
                p = environ;
                environ = v;
                while (*v++ = *p++);            /* copy the environment */
                v = environ;
        } else if (!(size % ENTRY_INC)) {
                if (!(v = realloc(environ, rounded(size) * sizeof(char **))))
                        return 1;
                environ = v;
        }
        v[size - 1] = name;
        v[size] = NULL;
        size++;
        return 0;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/putw.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * putw - write an word on a stream
 */
/* $Header: putw.c,v 1.1 89/12/18 14:40:15 eck Exp $ */

#include        <stdio.h>

_PROTOTYPE(int putw, (int w, FILE *stream ));

int
putw(w, stream)
int w;
register FILE *stream;
{
        register int cnt = sizeof(int);
        register char *p = (char *) &w;

        while (cnt--) {
                putc(*p++, stream);
        }
        if (ferror(stream)) return EOF;
        return w;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/regexp.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/* regcomp and regexec -- regsub and regerror are elsewhere
 *
 *      Copyright (c) 1986 by University of Toronto.
 *      Written by Henry Spencer.  Not derived from licensed software.
 *
 *      Permission is granted to anyone to use this software for any
 *      purpose on any computer system, and to redistribute it freely,
 *      subject to the following restrictions:
 *
 *      1. The author is not responsible for the consequences of use of
 *              this software, no matter how awful, even if they arise
 *              from defects in it.
 *
 *      2. The origin of this software must not be misrepresented, either
 *              by explicit claim or by omission.
 *
 *      3. Altered versions must be plainly marked as such, and must not
 *              be misrepresented as being the original software.
 *
 * Beware that some of this code is subtly aware of the way operator
 * precedence is structured in regular expressions.  Serious changes in
 * regular-expression syntax might require a total rethink.
 *
 *      The third parameter to regexec was added by Martin C. Atkins.
 *      Andy Tanenbaum also made some changes.
```

```
 */

#include <minix/config.h>
#include <minix/const.h>
#include <stdlib.h>
#include <string.h>
#include <regexp.h>
#include <stdio.h>

/* The first byte of the regexp internal "program" is actually this magic
 * number; the start node begins in the second byte.
 */
#define MAGIC   0234

/* The "internal use only" fields in regexp.h are present to pass info from
 * compile to execute that permits the execute phase to run lots faster on
 * simple cases.  They are:
 *
 * regstart     char that must begin a match; '\0' if none obvious
 * reganch      is the match anchored (at beginning-of-line only)?
 * regmust      string (pointer into program) that match must include, or NULL
 * regmlen      length of regmust string
 *
 * Regstart and reganch permit very fast decisions on suitable starting points
 * for a match, cutting down the work a lot.  Regmust permits fast rejection
 * of lines that cannot possibly match.  The regmust tests are costly enough
 * that regcomp() supplies a regmust only if the r.e. contains something
 * potentially expensive (at present, the only such thing detected is * or +
 * at the start of the r.e., which can involve a lot of backup).  Regmlen is
 * supplied because the test in regexec() needs it and regcomp() is computing
 * it anyway.
 */

/* Structure for regexp "program".  This is essentially a linear encoding
 * of a nondeterministic finite-state machine (aka syntax charts or
 * "railroad normal form" in parsing technology).  Each node is an opcode
 * plus a "next" pointer, possibly plus an operand.  "Next" pointers of
 * all nodes except BRANCH implement concatenation; a "next" pointer with
 * a BRANCH on both ends of it is connecting two alternatives.  (Here we
 * have one of the subtle syntax dependencies:  an individual BRANCH (as
 * opposed to a collection of them) is never concatenated with anything
 * because of operator precedence.)  The operand of some types of node is
 * a literal string; for others, it is a node leading into a sub-FSM.  In
 * particular, the operand of a BRANCH node is the first node of the branch.
 * (NB this is *not* a tree structure:  the tail of the branch connects
 * to the thing following the set of BRANCHes.)  The opcodes are:
 */

/* Definition   number  opnd?   meaning */
#define END     0               /* no   End of program. */
#define BOL     1               /* no   Match "" at beginning of line. */
#define EOL     2               /* no   Match "" at end of line. */
#define ANY     3               /* no   Match any one character. */
#define ANYOF   4               /* str  Match any character in this string. */
#define ANYBUT  5               /* str  Match any character not in this
                         * string. */
#define BRANCH  6               /* node Match this alternative, or the
                         * next... */
#define BACK    7               /* no   Match "", "next" ptr points backward. */
#define EXACTLY 8               /* str  Match this string. */
```

```
#define NOTHING 9               /* no   Match empty string. */
#define STAR    10              /* node Match this (simple) thing 0 or more
                         * times. */
#define PLUS    11              /* node Match this (simple) thing 1 or more
                         * times. */
#define OPEN    20              /* no   Mark this point in input as start of
                         * #n. */
 /* OPEN+1 is number 1, etc. */
#define CLOSE   30              /* no   Analogous to OPEN. */

/* Opcode notes:
 *
 * BRANCH       The set of branches constituting a single choice are hooked
 *              together with their "next" pointers, since precedence prevents
 *              anything being concatenated to any individual branch.  The
 *              "next" pointer of the last BRANCH in a choice points to the
 *              thing following the whole choice.  This is also where the
 *              final "next" pointer of each individual branch points; each
 *              branch starts with the operand node of a BRANCH node.
 *
 * BACK         Normal "next" pointers all implicitly point forward; BACK
 *              exists to make loop structures possible.
 *
 * STAR,PLUS    '?', and complex '*' and '+', are implemented as circular
 *              BRANCH structures using BACK.  Simple cases (one character
 *              per match) are implemented with STAR and PLUS for speed
 *              and to minimize recursive plunges.
 *
 * OPEN,CLOSE   ...are numbered at compile time.
 */

/* A node is one char of opcode followed by two chars of "next" pointer.
 * "Next" pointers are stored as two 8-bit pieces, high order first.  The
 * value is a positive offset from the opcode of the node containing it.
 * An operand, if any, simply follows the node.  (Note that much of the
 * code generation knows about this implicit relationship.)
 *
 * Using two bytes for the "next" pointer is vast overkill for most things,
 * but allows patterns to get big without disasters.
 */
#define OP(p)   (*(p))
#define NEXT(p) (((*((p)+1)&0377)<<8) + (*((p)+2)&0377))
#define OPERAND(p)      ((p) + 3)

/* Utility definitions.
 */
#ifndef CHARBITS
#define UCHARAT(p)      ((int)*(unsigned char *)(p))
#else
#define UCHARAT(p)      ((int)*(p)&CHARBITS)
#endif

#define CFAIL(m)        { regerror(m); return((char *)NULL); }
#define RFAIL(m)        { regerror(m); return((regexp *)NULL); }
#define ISMULT(c)       ((c) == '*' || (c) == '+' || (c) == '?')
#define META    "^$.[()|?+*\\"

/* Flags to be passed up and down.
 */
#define HASWIDTH        01      /* Known never to match null string. */
```

```
#define SIMPLE          02      /* Simple enough to be STAR/PLUS operand. */
#define SPSTART         04      /* Starts with * or +. */
#define WORST           0       /* Worst case. */

/* Global work variables for regcomp().
 */
PRIVATE char *regparse;         /* Input-scan pointer. */
PRIVATE int regnpar;            /* () count. */
PRIVATE char regdummy;
PRIVATE char *regcode;          /* Code-emit pointer; &regdummy = don't. */
PRIVATE long regsize;           /* Code size. */

/* Forward declarations for regcomp()'s friends.
 */
#ifndef STATIC
#define STATIC  PRIVATE
#endif
STATIC _PROTOTYPE( char *reg, (int paren, int *flagp)                   );
STATIC _PROTOTYPE( char *regbranch, (int *flagp)                        );
STATIC _PROTOTYPE( char *regpiece, (int *flagp)                         );
STATIC _PROTOTYPE( char *regatom, (int *flagp)                          );
STATIC _PROTOTYPE( char *regnode, (int op)                              );
STATIC _PROTOTYPE( char *regnext, (char *p)                             );
STATIC _PROTOTYPE( void regc, (int b)                                   );
STATIC _PROTOTYPE( void reginsert, (int op, char *opnd)                 );
STATIC _PROTOTYPE( void regtail, (char *p, char *val)                   );
STATIC _PROTOTYPE( void regoptail, (char *p, char *val)                 );

/*
 - regcomp - compile a regular expression into internal code
 *
 * We can't allocate space until we know how big the compiled form will be,
 * but we can't compile it (and thus know how big it is) until we've got a
 * place to put the code.  So we cheat:  we compile it twice, once with code
 * generation turned off and size counting turned on, and once "for real".
 * This also means that we don't allocate space until we are sure that the
 * thing really will compile successfully, and we never have to move the
 * code and thus invalidate pointers into it.  (Note that it has to be in
 * one piece because free() must be able to free it all.)
 *
 * Beware that the optimization-preparation code in here knows about some
 * of the structure of the compiled regexp.
 */
regexp *regcomp(exp)
char *exp;
{
  register regexp *r;
  register char *scan;
  register char *longest;
  register int len;
  int flags;

  if (exp == (char *)NULL) RFAIL("NULL argument");

  /* First pass: determine size, legality. */
  regparse = exp;
  regnpar = 1;
  regsize = 0L;
  regcode = &regdummy;
  regc(MAGIC);
```

```c
	if (reg(0, &flags) == (char *)NULL) return((regexp *)NULL);

	/* Small enough for pointer-storage convention? */
	if (regsize >= 32767L)		/* Probably could be 65535L. */
		RFAIL("regexp too big");

	/* Allocate space. */
	r = (regexp *) malloc(sizeof(regexp) + (unsigned) regsize);
	if (r == (regexp *)NULL) RFAIL("out of space");

	/* Second pass: emit code. */
	regparse = exp;
	regnpar = 1;
	regcode = r->program;
	regc(MAGIC);
	if (reg(0, &flags) == (char *)NULL) return((regexp *)NULL);

	/* Dig out information for optimizations. */
	r->regstart = '\0';		/* Worst-case defaults. */
	r->reganch = 0;
	r->regmust = (char *)NULL;
	r->regmlen = 0;
	scan = r->program + 1;		/* First BRANCH. */
	if (OP(regnext(scan)) == END) {	/* Only one top-level choice. */
		scan = OPERAND(scan);

		/* Starting-point info. */
		if (OP(scan) == EXACTLY)
			r->regstart = *OPERAND(scan);
		else if (OP(scan) == BOL)
			r->reganch++;

		/* If there's something expensive in the r.e., find the
		 * longest literal string that must appear and make it the
		 * regmust.  Resolve ties in favor of later strings, since
		 * the regstart check works with the beginning of the r.e.
		 * and avoiding duplication strengthens checking.  Not a
		 * strong reason, but sufficient in the absence of others. */
		if (flags & SPSTART) {
			longest = (char *)NULL;
			len = 0;
			for (; scan != (char *)NULL; scan = regnext(scan))
				if (OP(scan) == EXACTLY && strlen(OPERAND(scan)) >= len) {
					longest = OPERAND(scan);
					len = strlen(OPERAND(scan));
				}
			r->regmust = longest;
			r->regmlen = len;
		}
	}
	return(r);
}

/*
 - reg - regular expression, i.e. main body or parenthesized thing
 *
 * Caller must absorb opening parenthesis.
 *
 * Combining parenthesis handling with the base level of regular expression
 * is a trifle forced, but the need to tie the tails of the branches to what
```

```
 * follows makes it hard to avoid.
 */
PRIVATE char *reg(paren, flagp)
int paren;                      /* Parenthesized? */
int *flagp;
{
  register char *ret;
  register char *br;
  register char *ender;
  register int parno;
  int flags;

  *flagp = HASWIDTH;            /* Tentatively. */

  /* Make an OPEN node, if parenthesized. */
  if (paren) {
        if (regnpar >= NSUBEXP) CFAIL("too many ()");
        parno = regnpar;
        regnpar++;
        ret = regnode(OPEN + parno);
  } else {
        parno = 0;              /* not actually used, keep compiler quiet */
        ret = (char *)NULL;
  }

  /* Pick up the branches, linking them together. */
  br = regbranch(&flags);
  if (br == (char *)NULL) return((char *)NULL);
  if (ret != (char *)NULL)
        regtail(ret, br);       /* OPEN -> first. */
  else
        ret = br;
  if (!(flags & HASWIDTH)) *flagp &= ~HASWIDTH;
  *flagp |= flags & SPSTART;
  while (*regparse == '|') {
        regparse++;
        br = regbranch(&flags);
        if (br == (char *)NULL) return((char *)NULL);
        regtail(ret, br);       /* BRANCH -> BRANCH. */
        if (!(flags & HASWIDTH)) *flagp &= ~HASWIDTH;
        *flagp |= flags & SPSTART;
  }

  /* Make a closing node, and hook it on the end. */
  ender = regnode((paren) ? CLOSE + parno : END);
  regtail(ret, ender);

  /* Hook the tails of the branches to the closing node. */
  for (br = ret; br != (char *)NULL; br = regnext(br)) regoptail(br, ender);

  /* Check for proper termination. */
  if (paren && *regparse++ != ')') {
        CFAIL("unmatched ()");
  } else if (!paren && *regparse != '\0') {
        if (*regparse == ')') {
                CFAIL("unmatched ()");
        } else
                CFAIL("junk on end");   /* "Can't happen". */
        /* NOTREACHED */
  }
```

```
  return(ret);
}

/*
 - regbranch - one alternative of an | operator
 *
 * Implements the concatenation operator.
 */
PRIVATE char *regbranch(flagp)
int *flagp;
{
  register char *ret;
  register char *chain;
  register char *latest;
  int flags;

  *flagp = WORST;               /* Tentatively. */

  ret = regnode(BRANCH);
  chain = (char *)NULL;
  while (*regparse != '\0' && *regparse != '|' && *regparse != ')') {
        latest = regpiece(&flags);
        if (latest == (char *)NULL) return((char *)NULL);
        *flagp |= flags & HASWIDTH;
        if (chain == (char *)NULL)      /* First piece. */
                *flagp |= flags & SPSTART;
        else
                regtail(chain, latest);
        chain = latest;
  }
  if (chain == (char *)NULL)            /* Loop ran zero times. */
        regnode(NOTHING);

  return(ret);
}

/*
 - regpiece - something followed by possible [*+?]
 *
 * Note that the branching code sequences used for ? and the general cases
 * of * and + are somewhat optimized:  they use the same NOTHING node as
 * both the endmarker for their branch list and the body of the last branch.
 * It might seem that this node could be dispensed with entirely, but the
 * endmarker role is not redundant.
 */
PRIVATE char *regpiece(flagp)
int *flagp;
{
  register char *ret;
  register char op;
  register char *next;
  int flags;

  ret = regatom(&flags);
  if (ret == (char *)NULL) return((char *)NULL);

  op = *regparse;
  if (!ISMULT(op)) {
        *flagp = flags;
        return(ret);
```

```
	}
	if (!(flags & HASWIDTH) && op != '?') CFAIL("*+ operand could be empty");
	*flagp = (op != '+') ? (WORST | SPSTART) : (WORST | HASWIDTH);

	if (op == '*' && (flags & SIMPLE))
		reginsert(STAR, ret);
	else if (op == '*') {
		/* Emit x* as (x&|), where & means "self". */
		reginsert(BRANCH, ret); /* Either x */
		regoptail(ret, regnode(BACK));  /* and loop */
		regoptail(ret, ret);    /* back */
		regtail(ret, regnode(BRANCH));  /* or */
		regtail(ret, regnode(NOTHING)); /* null. */
	} else if (op == '+' && (flags & SIMPLE))
		reginsert(PLUS, ret);
	else if (op == '+') {
		/* Emit x+ as x(&|), where & means "self". */
		next = regnode(BRANCH); /* Either */
		regtail(ret, next);
		regtail(regnode(BACK), ret);    /* loop back */
		regtail(next, regnode(BRANCH)); /* or */
		regtail(ret, regnode(NOTHING)); /* null. */
	} else if (op == '?') {
		/* Emit x? as (x|) */
		reginsert(BRANCH, ret); /* Either x */
		regtail(ret, regnode(BRANCH));  /* or */
		next = regnode(NOTHING);/* null. */
		regtail(ret, next);
		regoptail(ret, next);
	}
	regparse++;
	if (ISMULT(*regparse)) CFAIL("nested *?+");

	return(ret);
}

/*
 - regatom - the lowest level
 *
 * Optimization:  gobbles an entire sequence of ordinary characters so that
 * it can turn them into a single node, which is smaller to store and
 * faster to run.  Backslashed characters are exceptions, each becoming a
 * separate node; the code is simpler that way and it's not worth fixing.
 */
PRIVATE char *regatom(flagp)
int *flagp;
{
	register char *ret;
	int flags;

	*flagp = WORST;		/* Tentatively. */

	switch (*regparse++) {
	    case '^':	ret = regnode(BOL);		break;
	    case '$':	ret = regnode(EOL);		break;
	    case '.':
		ret = regnode(ANY);
		*flagp |= HASWIDTH | SIMPLE;
		break;
	    case '[':{
```

```
                register int class;
                register int classend;

                if (*regparse == '^') { /* Complement of range. */
                        ret = regnode(ANYBUT);
                        regparse++;
                } else
                        ret = regnode(ANYOF);
                if (*regparse == ']' || *regparse == '-') regc(*regparse++);
                while (*regparse != '\0' && *regparse != ']') {
                        if (*regparse == '-') {
                                regparse++;
                                if (*regparse == ']' || *regparse == '\0')
                                        regc('-');
                                else {
                                        class = UCHARAT(regparse - 2) + 1;
                                        classend = UCHARAT(regparse);
                                        if (class > classend + 1)
                                                CFAIL("invalid [] range");
                                        for (; class <= classend; class++)
                                                regc(class);
                                        regparse++;
                                }
                        } else
                                regc(*regparse++);
                }
                regc('\0');
                if (*regparse != ']') CFAIL("unmatched []");
                regparse++;
                *flagp |= HASWIDTH | SIMPLE;
          }
          break;
        case '(':
          ret = reg(1, &flags);
          if (ret == (char *)NULL) return((char *)NULL);
          *flagp |= flags & (HASWIDTH | SPSTART);
          break;
        case '\0':
        case '|':
        case ')':
          CFAIL("internal urp");  /* Supposed to be caught earlier. */
          break;
        case '?':
        case '+':
        case '*': CFAIL("?+* follows nothing");           break;
        case '\\':
          if (*regparse == '\0') CFAIL("trailing \\");
          ret = regnode(EXACTLY);
          regc(*regparse++);
          regc('\0');
          *flagp |= HASWIDTH | SIMPLE;
          break;
        default:{
                register int len;
                register char ender;

                regparse--;
                len = strcspn(regparse, META);
                if (len <= 0) CFAIL("internal disaster");
                ender = *(regparse + len);
```

```
                if (len > 1 && ISMULT(ender))
                        len--;  /* Back off clear of ?+* operand. */
                *flagp |= HASWIDTH;
                if (len == 1) *flagp |= SIMPLE;
                ret = regnode(EXACTLY);
                while (len > 0) {
                        regc(*regparse++);
                        len--;
                }
                regc('\0');
        }
        break;
  }

  return(ret);
}

/*
 - regnode - emit a node
 */
PRIVATE char *regnode(op)
char op;
{
  register char *ret;
  register char *ptr;

  ret = regcode;
  if (ret == &regdummy) {
        regsize += 3;
        return(ret);
  }
  ptr = ret;
  *ptr++ = op;
  *ptr++ = '\0';                /* Null "next" pointer. */
  *ptr++ = '\0';
  regcode = ptr;

  return(ret);
}

/*
 - regc - emit (if appropriate) a byte of code
 */
PRIVATE void regc(b)
char b;
{
  if (regcode != &regdummy)
        *regcode++ = b;
  else
        regsize++;
}

/*
 - reginsert - insert an operator in front of already-emitted operand
 *
 * Means relocating the operand.
 */
PRIVATE void reginsert(op, opnd)
char op;
char *opnd;
```

```
{
  register char *src;
  register char *dst;
  register char *place;

  if (regcode == &regdummy) {
        regsize += 3;
        return;
  }
  src = regcode;
  regcode += 3;
  dst = regcode;
  while (src > opnd) *--dst = *--src;

  place = opnd;                 /* Op node, where operand used to be. */
  *place++ = op;
  *place++ = '\0';
  *place++ = '\0';
}

/*
 - regtail - set the next-pointer at the end of a node chain
 */
PRIVATE void regtail(p, val)
char *p;
char *val;
{
  register char *scan;
  register char *temp;
  register int offset;

  if (p == &regdummy) return;

  /* Find last node. */
  scan = p;
  for (;;) {
        temp = regnext(scan);
        if (temp == (char *)NULL) break;
        scan = temp;
  }

  if (OP(scan) == BACK)
        offset = scan - val;
  else
        offset = val - scan;
  *(scan + 1) = (offset >> 8) & 0377;
  *(scan + 2) = offset & 0377;
}

/*
 - regoptail - regtail on operand of first argument; nop if operandless
 */
PRIVATE void regoptail(p, val)
char *p;
char *val;
{
  /* "Operandless" and "op != BRANCH" are synonymous in practice. */
  if (p == (char *)NULL || p == &regdummy || OP(p) != BRANCH) return;
  regtail(OPERAND(p), val);
}
```

```
/* regexec and friends
 */

/* Global work variables for regexec().
 */
PRIVATE char *reginput;         /* String-input pointer. */
PRIVATE char *regbol;           /* Beginning of input, for ^ check. */
PRIVATE char **regstartp;       /* Pointer to startp array. */
PRIVATE char **regendp;         /* Ditto for endp. */

/* Forwards.
 */
STATIC _PROTOTYPE( int regtry, (regexp *prog, char *string)             );
STATIC _PROTOTYPE( int regmatch, (char *prog)                           );
STATIC _PROTOTYPE( int regrepeat, (char *p)                             );

#ifdef DEBUG
int regnarrate = 0;
void regdump();
STATIC _PROTOTYPE( char *regprop, (char *op)                            );
#endif

/*
 - regexec - match a regexp against a string
 */
int regexec(prog, string, bolflag)
register regexp *prog;
register char *string;
int bolflag;
{
  register char *s;

  /* Be paranoid... */
  if (prog == (regexp *)NULL || string == (char *)NULL) {
        regerror("NULL parameter");
        return(0);
  }

  /* Check validity of program. */
  if (UCHARAT(prog->program) != MAGIC) {
        regerror("corrupted program");
        return(0);
  }

  /* If there is a "must appear" string, look for it. */
  if (prog->regmust != (char *)NULL) {
        s = string;
        while ((s = strchr(s, prog->regmust[0])) != (char *)NULL) {
                if (strncmp(s, prog->regmust, prog->regmlen) == 0)
                        break;  /* Found it. */
                s++;
        }
        if (s == (char *)NULL)          /* Not present. */
                return(0);
  }

  /* Mark beginning of line for ^ . */
  if (bolflag)
        regbol = string;
```

```
  else
        regbol = (char *)NULL;

  /* Simplest case:  anchored match need be tried only once. */
  if (prog->reganch) return(regtry(prog, string));

  /* Messy cases:  unanchored match. */
  s = string;
  if (prog->regstart != '\0')   /* We know what char it must start with. */
        while ((s = strchr(s, prog->regstart)) != (char *)NULL) {
                if (regtry(prog, s)) return(1);
                s++;
        }
  else
        /* We don't -- general case. */
        do {
                if (regtry(prog, s)) return(1);
        } while (*s++ != '\0');

  /* Failure. */
  return(0);
}

/*
 - regtry - try match at specific point
 */
PRIVATE int regtry(prog, string)   /* 0 failure, 1 success */
regexp *prog;
char *string;
{
  register int i;
  register char **sp;
  register char **ep;

  reginput = string;
  regstartp = prog->startp;
  regendp = prog->endp;

  sp = prog->startp;
  ep = prog->endp;
  for (i = NSUBEXP; i > 0; i--) {
        *sp++ = (char *)NULL;
        *ep++ = (char *)NULL;
  }
  if (regmatch(prog->program + 1)) {
        prog->startp[0] = string;
        prog->endp[0] = reginput;
        return(1);
  } else
        return(0);
}

/*
 - regmatch - main matching routine
 *
 * Conceptually the strategy is simple:  check to see whether the current
 * node matches, call self recursively to see whether the rest matches,
 * and then act accordingly.  In practice we make some effort to avoid
 * recursion, in particular by going through "ordinary" nodes (that don't
 * need to know whether the rest of the match failed) by a loop instead of
```

```
 * by recursion.
 */
PRIVATE int regmatch(prog)      /* 0 failure, 1 success */
char *prog;
{
  register char *scan;          /* Current node. */
  char *next;                   /* Next node. */

  scan = prog;
#ifdef DEBUG
  if (scan != (char *)NULL && regnarrate) fprintf(stderr, "%s(\n", regprop(scan));
#endif
  while (scan != (char *)NULL) {
#ifdef DEBUG
        if (regnarrate) fprintf(stderr, "%s...\n", regprop(scan));
#endif
        next = regnext(scan);

        switch (OP(scan)) {
            case BOL:
                if (reginput != regbol) return(0);
                break;
            case EOL:
                if (*reginput != '\0') return(0);
                break;
            case ANY:
                if (*reginput == '\0') return(0);
                reginput++;
                break;
            case EXACTLY:{
                        register int len;
                        register char *opnd;

                        opnd = OPERAND(scan);
                        /* Inline the first character, for speed. */
                        if (*opnd != *reginput) return(0);
                        len = strlen(opnd);
                        if (len > 1 && strncmp(opnd, reginput, len) != 0)
                                return(0);
                        reginput += len;
                }
                break;
            case ANYOF:
                if (*reginput == '\0' || strchr(OPERAND(scan), *reginput) == (char *)NULL)
                        return(0);
                reginput++;
                break;
            case ANYBUT:
                if (*reginput == '\0' || strchr(OPERAND(scan), *reginput) != (char *)NULL)
                        return(0);
                reginput++;
                break;
            case NOTHING:
                break;
            case BACK:
                break;
            case OPEN + 1:
            case OPEN + 2:
            case OPEN + 3:
            case OPEN + 4:
```

```
        case OPEN + 5:
        case OPEN + 6:
        case OPEN + 7:
        case OPEN + 8:
        case OPEN + 9:{
                register int no;
                register char *save;

                no = OP(scan) - OPEN;
                save = reginput;

                if (regmatch(next)) {
                        /* Don't set startp if some later
                         * invocation of the same parentheses
                         * already has. */
                        if (regstartp[no] == (char *)NULL)
                                regstartp[no] = save;
                        return(1);
                } else
                        return(0);
            }
            break;
        case CLOSE + 1:
        case CLOSE + 2:
        case CLOSE + 3:
        case CLOSE + 4:
        case CLOSE + 5:
        case CLOSE + 6:
        case CLOSE + 7:
        case CLOSE + 8:
        case CLOSE + 9:{
                register int no;
                register char *save;

                no = OP(scan) - CLOSE;
                save = reginput;

                if (regmatch(next)) {
                        /* Don't set endp if some later
                         * invocation of the same parentheses
                         * already has. */
                        if (regendp[no] == (char *)NULL) regendp[no] = save;
                        return(1);
                } else
                        return(0);
            }
            break;
        case BRANCH:{
                register char *save;

                if (OP(next) != BRANCH) /* No choice. */
                        next = OPERAND(scan);   /* Avoid recursion. */
                else {
                        do {
                                save = reginput;
                                if (regmatch(OPERAND(scan)))
                                        return(1);
                                reginput = save;
                                scan = regnext(scan);
                        } while (scan != (char *)NULL && OP(scan) == BRANCH);
```

```
                                return(0);
                                /* NOTREACHED */
                        }
                }
                break;
            case STAR:
            case PLUS:{
                        register char nextch;
                        register int no;
                        register char *save;
                        register int min;

                        /* Lookahead to avoid useless match attempts
                         * when we know what character comes next. */
                        nextch = '\0';
                        if (OP(next) == EXACTLY) nextch = *OPERAND(next);
                        min = (OP(scan) == STAR) ? 0 : 1;
                        save = reginput;
                        no = regrepeat(OPERAND(scan));
                        while (no >= min) {
                                /* If it could work, try it. */
                                if (nextch == '\0' || *reginput == nextch)
                                        if (regmatch(next)) return(1);
                                /* Couldn't or didn't -- back up. */
                                no--;
                                reginput = save + no;
                        }
                        return(0);
                }
                break;
            case END:
                return(1);      /* Success! */
                break;
            default:
                regerror("memory corruption");
                return(0);
                break;
        }

        scan = next;
  }

  /* We get here only if there's trouble -- normally "case END" is the
   * terminating point. */
  regerror("corrupted pointers");
  return(0);
}

/*
 - regrepeat - repeatedly match something simple, report how many
 */
PRIVATE int regrepeat(p)
char *p;
{
  register int count = 0;
  register char *scan;
  register char *opnd;

  scan = reginput;
  opnd = OPERAND(p);
```

```
  switch (OP(p)) {
      case ANY:
        count = strlen(scan);
        scan += count;
        break;
      case EXACTLY:
        while (*opnd == *scan) {
                count++;
                scan++;
        }
        break;
      case ANYOF:
        while (*scan != '\0' && strchr(opnd, *scan) != (char *)NULL) {
                count++;
                scan++;
        }
        break;
      case ANYBUT:
        while (*scan != '\0' && strchr(opnd, *scan) == (char *)NULL) {
                count++;
                scan++;
        }
        break;
      default:                  /* Oh dear.  Called inappropriately. */
        regerror("internal foulup");
        count = 0;              /* Best compromise. */
        break;
  }
  reginput = scan;

  return(count);
}

/*
 - regnext - dig the "next" pointer out of a node
 */
PRIVATE char *regnext(p)
register char *p;
{
  register int offset;

  if (p == &regdummy) return((char *)NULL);

  offset = NEXT(p);
  if (offset == 0) return((char *)NULL);

  if (OP(p) == BACK)
        return(p - offset);
  else
        return(p + offset);
}

#ifdef DEBUG

STATIC char *regprop();

/*
 - regdump - dump a regexp onto stdout in vaguely comprehensible form
 */
void regdump(r)
```

```
regexp *r;
{
  register char *s;
  register char op = EXACTLY;   /* Arbitrary non-END op. */
  register char *next;

  s = r->program + 1;
  while (op != END) {            /* While that wasn't END last time... */
        op = OP(s);
        printf("%2d%s", (int) (s - r->program), regprop(s));    /* Where, what. */
        next = regnext(s);
        if (next == (char *)NULL)       /* Next ptr. */
                printf("(0)");
        else
                printf("(%d)", (int) (s - r->program) + (int) (next - s));
        s += 3;
        if (op == ANYOF || op == ANYBUT || op == EXACTLY) {
                /* Literal string, where present. */
                while (*s != '\0') {
                        putchar(*s);
                        s++;
                }
                s++;
        }
        putchar('\n');
  }

  /* Header fields of interest. */
  if (r->regstart != '\0') printf("start c' ", r->regstart);
  if (r->reganch) printf("anchored ");
  if (r->regmust != (char *)NULL) printf("must have \"%s\"", r->regmust);
  printf("\n");
}

/*
 - regprop - printable representation of opcode
 */
PRIVATE char *regprop(op)
char *op;
{
  register char *p;
  PRIVATE char buf[50];

  (void) strcpy(buf, ":");

  switch (OP(op)) {
      case BOL: p = "BOL";              break;
      case EOL: p = "EOL";              break;
      case ANY: p = "ANY";              break;
      case ANYOF:       p = "ANYOF";            break;
      case ANYBUT:      p = "ANYBUT";           break;
      case BRANCH:      p = "BRANCH";           break;
      case EXACTLY:     p = "EXACTLY";          break;
      case NOTHING:     p = "NOTHING";          break;
      case BACK:        p = "BACK";             break;
      case END: p = "END";              break;
      case OPEN + 1:
      case OPEN + 2:
      case OPEN + 3:
      case OPEN + 4:
```

```
      case OPEN + 5:
      case OPEN + 6:
      case OPEN + 7:
      case OPEN + 8:
      case OPEN + 9:
        sprintf(buf + strlen(buf), "OPEN%d", OP(op) - OPEN);
        p = (char *)NULL;
        break;
      case CLOSE + 1:
      case CLOSE + 2:
      case CLOSE + 3:
      case CLOSE + 4:
      case CLOSE + 5:
      case CLOSE + 6:
      case CLOSE + 7:
      case CLOSE + 8:
      case CLOSE + 9:
        sprintf(buf + strlen(buf), "CLOSE%d", OP(op) - CLOSE);
        p = (char *)NULL;
        break;
      case STAR:        p = "STAR";             break;
      case PLUS:        p = "PLUS";             break;
      default:  regerror("corrupted opcode"); p = (char *) NULL; break;
  }
  if (p != (char *)NULL) (void) strcat(buf, p);
  return(buf);
}

#endif
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/regsub.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/* regsub
 *
 *      Copyright (c) 1986 by University of Toronto.
 *      Written by Henry Spencer.  Not derived from licensed software.
 *
 *      Permission is granted to anyone to use this software for any
 *      purpose on any computer system, and to redistribute it freely,
 *      subject to the following restrictions:
 *
 *      1. The author is not responsible for the consequences of use of
 *              this software, no matter how awful, even if they arise
 *              from defects in it.
 *
 *      2. The origin of this software must not be misrepresented, either
 *              by explicit claim or by omission.
 *
 *      3. Altered versions must be plainly marked as such, and must not
 *              be misrepresented as being the original software.
 */

#include <lib.h>
#include <string.h>
#include <regexp.h>
#include <stdio.h>

```

```c
/* The first byte of the regexp internal "program" is actually this magic
 * number; the start node begins in the second byte.
 */
#define MAGIC   0234

#define CHARBITS 0377
#ifndef CHARBITS
#define UCHARAT(p)      ((int)*(unsigned char *)(p))
#else
#define UCHARAT(p)      ((int)*(p)&CHARBITS)
#endif

/*
 - regsub - perform substitutions after a regexp match
 */
void regsub(prog, source, dest)
regexp *prog;
char *source;
char *dest;
{
  register char *src;
  register char *dst;
  register char c;
  register int no;
  register int len;

  if (prog == (regexp *)NULL || source == (char *)NULL || dest == (char *)NULL) {
        regerror("NULL parm to regsub");
        return;
  }
  if (UCHARAT(prog->program) != MAGIC) {
        regerror("damaged regexp fed to regsub");
        return;
  }
  src = source;
  dst = dest;
  while ((c = *src++) != '\0') {
        if (c == '&')
                no = 0;
        else if (c == '\\' && '0' <= *src && *src <= '9')
                no = *src++ - '0';
        else
                no = -1;

        if (no < 0) {           /* Ordinary character. */
                if (c == '\\' && (*src == '\\' || *src == '&')) c = *src++;
                *dst++ = c;
        } else
        if (prog->startp[no] != (char *)NULL && prog->endp[no] != (char *)NULL) {
                len = (int) (prog->endp[no] - prog->startp[no]);
                strncpy(dst, prog->startp[no], len);
                dst += len;
                if (len != 0 && *(dst - 1) == '\0') {   /* strncpy hit NUL. */
                        regerror("damaged match string");
                        return;
                }
        }
  }
  *dst++ = '\0';
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/rindex.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
/* rindex - find last occurrence of a character in a string  */

#include <string.h>

char *rindex(s, charwanted)     /* found char, or NULL if none */
_CONST char *s;
char charwanted;
{
  return(strrchr(s, charwanted));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/stderr.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#include <sys/types.h>
#include <unistd.h>

_PROTOTYPE( void std_err, (char *s));

void std_err(s)
char *s;
{
  register char *p = s;

  while (*p != 0) p++;
  write(2, s, (int) (p - s));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/swab.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
/*  swab(3)
 *
 *  Author: Terrence W. Holm          Sep. 1988
 */
_PROTOTYPE( void swab, (char *from, char *to, int count));

void swab(from, to, count)
char *from;
char *to;
int count;
{
  register char temp;

  count >>= 1;
```

```
  while (--count >= 0) {
        temp = *from++;
        *to++ = *from++;
        *to++ = temp;
  }
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/other/syscall.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>

PUBLIC int _syscall(who, syscallnr, msgptr)
int who;
int syscallnr;
register message *msgptr;
{
  int status;

  msgptr->m_type = syscallnr;
  status = _sendrec(who, msgptr);
  if (status != 0) {
        /* 'sendrec' itself failed. */
        /* XXX - strerror doesn't know all the codes */
        msgptr->m_type = status;
  }
  if (msgptr->m_type < 0) {
        errno = -msgptr->m_type;
        return(-1);
  }
  return(msgptr->m_type);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/other/sysconf.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/* sysconf.c                                            POSIX 4.8.1
 *      long int sysconf(int name);
 *
 *      POSIX allows some of the values in <limits.h> to be increased at
 *      run time.  The sysconf() function allows such values to be checked
 *      at run time.  MINIX does not use this facility - the run time
 *      limits are those given in <limits.h>.
 */

#include <lib.h>
#include <unistd.h>
#include <time.h>

PUBLIC long int sysconf(name)
int name;                       /* property being inspected */
```

```
{
  switch(name) {
	case _SC_ARG_MAX:
		return (long) ARG_MAX;

	case _SC_CHILD_MAX:
		return (long) CHILD_MAX;

	case _SC_CLK_TCK:
		return (long) CLOCKS_PER_SEC;

	case _SC_NGROUPS_MAX:
		return (long) NGROUPS_MAX;

	case _SC_OPEN_MAX:
		return (long) OPEN_MAX;

	case _SC_JOB_CONTROL:
		return -1L;			/* no job control */

	case _SC_SAVED_IDS:
		return -1L;			/* no saved uid/gid */

	case _SC_VERSION:
		return (long) _POSIX_VERSION;

	case _SC_STREAM_MAX:
		return (long) STREAM_MAX;

	case _SC_TZNAME_MAX:
		return (long) TZNAME_MAX;

	default:
		errno = EINVAL;
		return -1L;
  }
}
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/telldir.c
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*	telldir()					Author: Kees J. Bot
 *								24 Apr 1989
 */
#define nil 0
#include <lib.h>
#include <sys/types.h>
#include <dirent.h>
#include <errno.h>

off_t telldir(DIR *dp)
/* Return the current read position in a directory. */
{
	if (dp == nil) { errno= EBADF; return -1; }

	return dp->_pos;
```

```
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/termcap.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 *      termcap.c       V1.1    20/7/87         agc     Joypace Ltd
 *
 *      Copyright Joypace Ltd, London, UK, 1987. All rights reserved.
 *      This file may be freely distributed provided that this notice
 *      remains attached.
 *
 *      A public domain implementation of the termcap(3) routines.
 *
 *
 *
 *       Klamer Schutte       V1.2    Nov. 1988
 *
 *   - Can match multiple terminal names                 [tgetent]
 *   - Removal of **area assignments                     [tgetstr]
 *
 *       Terrence W. Holm     V1.3    May, Sep, Oct.  1988
 *
 *   - Correct when TERM != name and TERMCAP is defined  [tgetent]
 *   - Correct the comparison for the terminal name      [tgetent]
 *   - Correct the value of ^x escapes                   [tgetstr]
 *   - Added %r to reverse row/column                    [tgoto]
 *   - Fixed end of definition test                      [tgetnum/flag/str]
 *
 *       Terrence W. Holm     V1.4    Jan. 1989
 *
 *   - Incorporated Klamer's V1.2 fixes into V1.3
 *   - Added %d, (old %d is now %2)                      [tgoto]
 *   - Allow '#' comments in definition file             [tgetent]
 */

#include <lib.h>
#include <termcap.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

char *capab = (char *)NULL;             /* the capability itself */

#if 0
/*  The following are not yet used.  */
extern short ospeed;            /* output speed */
extern char PC;                 /* padding character */
extern char *BC;                /* back cursor movement */
extern char *UP;                /* up cursor movement */
#endif

/*
 *      tgetent - get the termcap entry for terminal name, and put it
```

```
 *      in bp (which must be an array of 1024 chars). Returns 1 if
 *      termcap entry found, 0 if not found, and -1 if file not found.
 */

int tgetent(bp, name)
char *bp;
char *name;
{
  FILE *fp;
  char *file;
  char *term;
  short len = strlen(name);

  capab = bp;

  /* If TERMCAP begins with a '/' then use TERMCAP as the path   */
  /* Name of the termcap definitions file. If TERMCAP is a       */
  /* Definition and TERM equals "name" then use TERMCAP as the   */
  /* Definition. Otherwise use "/etc/termcap" as the path name.  */

  if ((file = getenv("TERMCAP")) == (char *)NULL)
        file = "/etc/termcap";
  else if (*file != '/')
        if ((term = getenv("TERM")) != (char *)NULL && strcmp(term, name) == 0) {
                *bp = '\0';
                strncat(bp, file, 1023);
                return(1);
        } else
                file = "/etc/termcap";

  if ((fp = fopen(file, "r")) == (FILE *) NULL) {
        capab = (char *)NULL;           /* no valid termcap  */
        return(-1);
  }
  for (;;) {
        /* Read in each definition */
        int def_len = 0;
        char *cp = bp;

        do {
                if (fgets(&bp[def_len], (unsigned int)(1024 - def_len), fp) == (char *)NULL) {
                        fclose(fp);
                        capab = (char *)NULL;   /* no valid termcap */
                        return(0);
                }
                def_len = strlen(bp) - 2;
        } while (bp[def_len] == '\\');

        while (isspace(*cp)) cp++;

        /* Comment lines start with a '#'  */
        if (*cp == '#') continue;

        /* See if any of the terminal names in this definition */
        /* Match "name".                                                         */

        do {
                if (strncmp(name, cp, len) == 0 &&
                    (cp[len] == '|' || cp[len] == ':')) {
                        fclose(fp);
```

```
                        return(1);
                }
                while ((*cp) && (*cp != '|') && (*cp != ':')) cp++;
        } while (*cp++ == '|');
  }
}


/*
 *      tgetnum - get the numeric terminal capability corresponding
 *      to id. Returns the value, -1 if invalid.
 */

int tgetnum(id)
char *id;
{
  register char *cp = capab;

  if (cp == (char *)NULL || id == (char *)NULL) return(-1);

  for (;;) {
        while (*cp++ != ':')
                if (cp[-1] == '\0') return(-1);

        while (isspace(*cp)) cp++;

        if (strncmp(cp, id, 2) == 0 && cp[2] == '#') return(atoi(cp + 3));
  }
}


/*
 *      tgetflag - get the boolean flag corresponding to id. Returns -1
 *      if invalid, 0 if the flag is not in termcap entry, or 1 if it is
 *      present.
 */

int tgetflag(id)
char *id;
{
  register char *cp = capab;

  if (cp == (char *)NULL || id == (char *)NULL) return(-1);

  for (;;) {
        while (*cp++ != ':')
                if (cp[-1] == '\0') return(0);

        while (isspace(*cp)) cp++;

        if (strncmp(cp, id, 2) == 0) return(1);
  }
}


/*
 *      tgetstr - get the string capability corresponding to id and place
 *      it in area (advancing area at same time). Expand escape sequences
 *      etc. Returns the string, or NULL if it can't do it.
 */
```

```
char *tgetstr(id, area)
char *id;
char **area;
{
  register char *cp = capab;
  register char *wsp = *area;   /* workspace pointer  */

  if (cp == (char *)NULL || id == (char *)NULL) return((char *)NULL);

  for (;;) {
        while (*cp++ != ':')
                if (cp[-1] == '\0') return((char *)NULL);

        while (isspace(*cp)) cp++;

        if (strncmp(cp, id, 2) == 0 && cp[2] == '=') {
                for (cp += 3; *cp && *cp != ':'; wsp++, cp++) switch (*cp) {
                            case '^':
                                *wsp = *++cp - '@';
                                break;

                            case '\\':
                                switch (*++cp) {
                                    case 'E':
                                        *wsp = '\033';
                                        break;
                                    case 'n':
                                        *wsp = '\n';
                                        break;
                                    case 'r':
                                        *wsp = '\r';
                                        break;
                                    case 't':
                                        *wsp = '\t';
                                        break;
                                    case 'b':
                                        *wsp = '\b';
                                        break;
                                    case 'f':
                                        *wsp = '\f';
                                        break;
                                    case '0':
                                    case '1':
                                    case '2':
                                    case '3':
                                        {
                                                int i;
                                                int t = 0;
                                                for (i = 0; i < 3 &&
                                                     isdigit(*cp); ++i, ++cp)
                                                        t = t * 8 + *cp - '0';
                                                *wsp = t;
                                                cp--;
                                                break;
                                        }
                                    default:
                                        *wsp = *cp;
                                }
                                break;
```

```
                                default:    *wsp = *cp;
                                }

                        *wsp++ = '\0';

                        {
                                char *ret = *area;
                                *area = wsp;
                                return(ret);
                        }
                }
  }                             /* end for(;;) */
}



/*
 *      tgoto - given the cursor motion string cm, make up the string
 *      for the cursor to go to (destcol, destline), and return the string.
 *      Returns "OOPS" if something's gone wrong, or the string otherwise.
 */

char *tgoto(cm, destcol, destline)
char *cm;
int destcol;
int destline;
{
  PRIVATE char ret[24];
  char *rp = ret;
  int incr = 0;
  int argno = 0;
  int numval;

  for (; *cm; cm++) {
        if (*cm == '%') {
                switch (*++cm) {
                    case 'i':   incr = 1;                               break;

                    case 'r':   argno = 1;                              break;

                    case '+':
                        numval = (argno == 0 ? destline : destcol);
                        *rp++ = numval + incr + *++cm;
                        argno = 1 - argno;
                        break;

                    case '2':
                        numval = (argno == 0 ? destline : destcol);
                        numval = (numval + incr) % 100;
                        *rp++ = '0' + (numval / 10);
                        *rp++ = '0' + (numval % 10);
                        argno = 1 - argno;
                        break;

                    case 'd':
                        numval = (argno == 0 ? destline : destcol);
                        numval = (numval + incr) % 1000;
                        if (numval > 99) *rp++ = '0' + (numval / 100);
                        if (numval > 9) *rp++ = '0' + (numval / 10) % 10;
```

```
                        *rp++ = '0' + (numval % 10);
                        argno = 1 - argno;
                        break;

                    case '%':   *rp++ = '%';                                break;

                    default:    return("OOPS");
                }

        } else
                *rp++ = *cm;
  }

  *rp = '\0';
  return(ret);
}
```

```
/*
 *      tputs - put the string cp out onto the terminal, using the function
 *      outc. This should do padding for the terminal, but I can't find a
 *      terminal that needs padding at the moment...
 */

int tputs(cp, affcnt, outc)
register char *cp;
int affcnt;
_PROTOTYPE( void (*outc), (int ch));
{
  if (cp == (char *)NULL) return(1);
  /* Do any padding interpretation - left null for MINIX just now */
  while (*cp) (*outc) (*cp++);
  return(1);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/other/ttyname.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/* ttyname.c                                            POSIX 4.7.2
 *      char *ttyname(int fildes);
 *
 *      Determines name of a terminal device.
 */

#include <lib.h>
#include <sys/stat.h>
#include <dirent.h>
#include <fcntl.h>
#include <stddef.h>
#include <string.h>
#include <unistd.h>

PRIVATE char base[] = "/dev";
```

```
PRIVATE char path[sizeof(base) + 1 + NAME_MAX]; /* extra 1 for '/' */

PUBLIC char *ttyname(fildes)
int fildes;
{
  DIR *devices;
  struct dirent *entry;
  struct stat tty_stat;
  struct stat dev_stat;

  /* Simple first test: file descriptor must be a character device */
  if (fstat(fildes, &tty_stat) < 0 || !S_ISCHR(tty_stat.st_mode))
        return (char *) NULL;

  /* Open device directory for reading  */
  if ((devices = opendir(base)) == (DIR *) NULL)
        return (char *) NULL;

  /* Scan the entries for one that matches perfectly */
  while ((entry = readdir(devices)) != (struct dirent *) NULL) {
        if (tty_stat.st_ino != entry->d_ino)
                continue;
        strcpy(path, base);
        strcat(path, "/");
        strcat(path, entry->d_name);
        if (stat(path, &dev_stat) < 0 || !S_ISCHR(dev_stat.st_mode))
                continue;
        if (tty_stat.st_ino == dev_stat.st_ino &&
            tty_stat.st_dev == dev_stat.st_dev &&
            tty_stat.st_rdev == dev_stat.st_rdev) {
                closedir(devices);
                return path;
        }
  }

  closedir(devices);
  return (char *) NULL;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/other/ttyslot.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
ttyslot.c

Return the index in the utmp file for the current user's terminal. The
current user's terminal is the first file descriptor in the range 0..2
for which ttyname() returns a name. The index is the line number in the
/etc/ttytab file. 0 will be returned in case of an error.

Created:        Oct 11, 1992 by Philip Homburg
*/

#define _MINIX_SOURCE

#include <sys/types.h>
#include <ttyent.h>
```

```
#include <string.h>
#include <unistd.h>

int ttyslot()
{
        int slot;

        slot= fttyslot(0);
        if (slot == 0) slot= fttyslot(1);
        if (slot == 0) slot= fttyslot(2);
        return slot;
}

int fttyslot(fd)
int fd;
{
        char *tname;
        int lineno;
        struct ttyent *ttyp;

        tname= ttyname(fd);
        if (tname == NULL) return 0;

        /* Assume that tty devices are in /dev */
        if (strncmp(tname, "/dev/", 5) != 0)
                return 0;       /* Malformed tty name. */
        tname += 5;

        /* Scan /etc/ttytab. */
        lineno= 1;
        while ((ttyp= getttyent()) != NULL)
        {
                if (strcmp(tname, ttyp->ty_name) == 0)
                {
                        endttyent();
                        return lineno;
                }
                lineno++;
        }
        /* No match */
        endttyent();
        return 0;
}

/*
 * $PchHeader: /mount/hd2/minix/lib/misc/RCS/ttyslot.c,v 1.3 1994/12/22 13:49:12 philip Exp $
 */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/__exit.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#define _exit   __exit
#include <lib.h>
#include <unistd.h>

PUBLIC void _exit(status)
int status;
{
  message m;

  m.m1_i1 = status;
  _syscall(MM, EXIT, &m);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_access.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define access  _access
#include <unistd.h>

PUBLIC int access(name, mode)
_CONST char *name;
int mode;
{
  message m;

  m.m3_i2 = mode;
  _loadname(name, &m);
  return(_syscall(FS, ACCESS, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_alarm.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define alarm   _alarm
#include <unistd.h>

PUBLIC unsigned int alarm(sec)
unsigned int sec;
{
  message m;

  m.m1_i1 = (int) sec;
  return( (unsigned) _syscall(MM, ALARM, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_cfgetispeed.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
posix/_cfgetispeed

Created:        June 11, 1993 by Philip Homburg
*/

#include <termios.h>

speed_t _cfgetispeed(const struct termios *termios_p)
{
  return termios_p->c_ispeed;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_cfgetospeed.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
posix/_cfgetospeed

Created:        June 11, 1993 by Philip Homburg
*/

#include <termios.h>

speed_t _cfgetospeed(const struct termios *termios_p)
{
  return termios_p->c_ospeed;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_cfsetispeed.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
posix/_cfsetispeed

Created:        June 11, 1993 by Philip Homburg
*/

#include <termios.h>

int _cfsetispeed(struct termios *termios_p, speed_t speed)
{
  termios_p->c_ispeed= speed;
  return 0;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_cfsetospeed.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
posix/_cfsetospeed

Created:        June 11, 1993 by Philip Homburg
*/

#include <termios.h>

int _cfsetospeed(struct termios *termios_p, speed_t speed)
{
  termios_p->c_ospeed= speed;
  return 0;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_chdir.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define chdir   _chdir
#include <unistd.h>

PUBLIC int chdir(name)
_CONST char *name;
{
  message m;

  _loadname(name, &m);
  return(_syscall(FS, CHDIR, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_chmod.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define chmod   _chmod
#include <sys/stat.h>

PUBLIC int chmod(name, mode)
_CONST char *name;
Mode_t mode;
{
  message m;

  m.m3_i2 = mode;
  _loadname(name, &m);
  return(_syscall(FS, CHMOD, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/posix/_chown.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define chown   _chown
#include <string.h>
#include <unistd.h>

PUBLIC int chown(name, owner, grp)
_CONST char *name;
Uid_t owner;
Gid_t grp;
{
  message m;

  m.m1_i1 = strlen(name) + 1;
  m.m1_i2 = owner;
  m.m1_i3 = grp;
  m.m1_p1 = (char *) name;
  return(_syscall(FS, CHOWN, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/posix/_chroot.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define chroot  _chroot
#include <unistd.h>

PUBLIC int chroot(name)
_CONST char *name;
{
  message m;

  _loadname(name, &m);
  return(_syscall(FS, CHROOT, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/posix/_close.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define close   _close
#include <unistd.h>

PUBLIC int close(fd)
int fd;
{
  message m;

  m.m1_i1 = fd;
```

```
  return(_syscall(FS, CLOSE, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_closedir.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*      closedir()                                      Author: Kees J. Bot
 *                                                              24 Apr 1989
 */
#define nil 0
#include <lib.h>
#define close   _close
#define closedir _closedir
#include <sys/types.h>
#include <dirent.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>

int closedir(DIR *dp)
/* Finish reading a directory. */
{
        int d;

        if (dp == nil) { errno= EBADF; return -1; }

        d= dp->_fd;
        free((void *) dp);
        return close(d);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_creat.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define creat   _creat
#include <fcntl.h>

PUBLIC int creat(name, mode)
_CONST char *name;
Mode_t mode;
{
  message m;

  m.m3_i2 = mode;
  _loadname(name, &m);
  return(_syscall(FS, CREAT, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/posix/_dup.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define dup     _dup
#define fcntl   _fcntl
#include <fcntl.h>
#include <unistd.h>

PUBLIC int dup(fd)
int fd;
{
  return(fcntl(fd, F_DUPFD, 0));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/posix/_dup2.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define close   _close
#define dup2    _dup2
#define fcntl   _fcntl
#include <fcntl.h>
#include <unistd.h>

PUBLIC int dup2(fd, fd2)
int fd, fd2;
{
/* The behavior of dup2 is defined by POSIX in 6.2.1.2 as almost, but not
 * quite the same as fcntl.
 */

  if (fd2 < 0 || fd2 > OPEN_MAX) {
        errno = EBADF;
        return(-1);
  }

  /* Check to see if fildes is valid. */
  if (fcntl(fd, F_GETFL) < 0) {
        /* 'fd' is not valid. */
        return(-1);
  } else {
        /* 'fd' is valid. */
        if (fd == fd2) return(fd2);
        close(fd2);
        return(fcntl(fd, F_DUPFD, fd2));
  }
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_exec.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define execl   _execl
#define execle  _execle
#define execv   _execv
#define execve  _execve
#define sbrk    _sbrk
#include <minix/minlib.h>
#include <stdarg.h>
#include <string.h>
#include <unistd.h>

extern char **environ;

#define PTRSIZE (sizeof(char *))

#ifdef _ANSI
PUBLIC int execl(const char *name, const char *arg, ...)
#else
PUBLIC int execl(name)
char *name;
#endif
{
  va_list argp;
  int result;

  va_start(argp, name);

  /* The following cast of argp is not portable.  Doing it right by copying
   * the args to a true array will cost as much as ARG_MAX bytes of space.
   */
  result = execve(name, (char **) argp, environ);
  va_end(argp);
  return(result);
}


#ifdef _ANSI
PUBLIC int execle(const char *name, const char *arg, ...)
#else
PUBLIC int execle(name)
char *name;
#endif
{
  va_list argp;
  char **p;
  int result;

  va_start(argp, name);

  /* The following cast of argp is not portable, as for execl(). */
  p = (char **) argp;
  while (*p++ != NIL_PTR)
        ;                       /* null statement */
  result = execve(name, (char **) argp, (char **) *p);
  va_end(argp);
```

```
  return(result);
}


PUBLIC int execv(name, argv)
_CONST char *name;
char * _CONST argv[];
{
  return(execve(name, argv, environ));
}


PUBLIC int execve(path, argv, envp)
_CONST char *path;              /* pointer to name of file to be executed */
char * _CONST argv[];           /* pointer to argument array */
char * _CONST envp[];           /* pointer to environment */
{
  int i, j;

  /* Count the argument pointers and environment pointers. */
  i = 0;
  if (argv != NULL)
  {
        while (argv[i] != NULL) i++;
  }
  j = 0;
  if (envp != NULL)
  {
        while (envp[j] != NULL) j++;
  }

  return(__execve(path, argv, envp, i, j));
}


PUBLIC int __execve(path, argv, envp, nargs, nenvps)
_CONST char *path;              /* pointer to name of file to be executed */
char * _CONST argv[];           /* pointer to argument array */
char * _CONST envp[];           /* pointer to environment */
int nargs;                      /* number of args */
int nenvps;                     /* number of environment strings */
{
/* This is split off from execve to be called from execvp, so execvp does not
 * have to allocate up to ARG_MAX bytes just to prepend "sh" to the arg array.
 */

  char *hp, **ap, *p;
  int i, stackbytes, npointers, overflow, temp;
  char *stack;
  message m;

  /* Decide how big a stack is needed. Be paranoid about overflow. */
  overflow = FALSE;
  npointers = 1 + nargs + 1 + nenvps + 1;       /* 1's for argc and NULLs */
  stackbytes = nargs + nenvps;          /* 1 byte for each null in strings */
  if (nargs < 0 || nenvps < 0 || stackbytes < nargs || npointers < stackbytes)
        overflow = TRUE;
  for (i = PTRSIZE; i != 0; i--) {
        temp = stackbytes + npointers;
        if (temp < stackbytes) overflow = TRUE;
```

```
                stackbytes = temp;
        }
        ap = (char **) argv;
        for (i = 0; i < nargs; i++) {
                temp = stackbytes + strlen(*ap++);
                if (temp < stackbytes) overflow = TRUE;
                stackbytes = temp;
        }
        ap = (char **) envp;
        for (i = 0; i < nenvps; i++) {
                temp = stackbytes + strlen(*ap++);
                if (temp < stackbytes) overflow = TRUE;
                stackbytes = temp;
        }
        temp = stackbytes + PTRSIZE - 1;
        if (temp < stackbytes) overflow = TRUE;
        stackbytes = (temp / PTRSIZE) * PTRSIZE;

        /* Check for overflow before committing sbrk. */
        if (overflow) {
                errno = E2BIG;
                return(-1);
        }

        /* Allocate the stack. */
        stack = sbrk(stackbytes);
        if (stack == (char *) -1) {
                errno = E2BIG;
                return(-1);
        }

        /* Prepare the stack vector and argc. */
        ap = (char **) stack;
        hp = &stack[npointers * PTRSIZE];
        *ap++ = (char *) nargs;

        /* Prepare the argument pointers and strings. */
        for (i = 0; i < nargs; i++) {
                *ap++ = (char *) (hp - stack);
                p = *argv++;
                while ( (*hp++ = *p++) != 0)
                        ;
        }
        *ap++ = (char *) NULL;

        /* Prepare the environment pointers and strings. */
        for (i = 0; i < nenvps; i++) {
                *ap++ = (char *) (hp - stack);
                p = *envp++;
                while ( (*hp++ = *p++) != 0)
                        ;
        }
        *ap++ = (char *) NULL;

        /* Do the real work. */
        m.m1_i1 = strlen(path) + 1;
        m.m1_i2 = stackbytes;
        m.m1_p1 = (char *) path;
        m.m1_p2 = stack;
        (void) _syscall(MM, EXEC, &m);
```

```

  /* The exec failed. */
  sbrk(-stackbytes);
  return(m.m_type);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_execn.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#include <string.h>

#define PTRSIZE sizeof(char *)
_PROTOTYPE( int _execn, (char * name));

PUBLIC int _execn(name)
char *name;                     /* pointer to file to be exec'd */
{
/* Special version used when there are no args and no environment.  This call
 * is principally used by INIT, to avoid having to allocate ARG_MAX.
 */

  PRIVATE char stack[3 * PTRSIZE];
  message m;

  m.m1_i1 = strlen(name) + 1;
  m.m1_i2 = sizeof(stack);
  m.m1_p1 = name;
  m.m1_p2 = stack;
  (void) _syscall(MM, EXEC, &m);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_fcntl.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define fcntl _fcntl
#include <fcntl.h>
#include <stdarg.h>

#if _ANSI
PUBLIC int fcntl(int fd, int cmd, ...)
#else
PUBLIC int fcntl(fd, cmd)
int fd;
int cmd;
#endif
{
  va_list argp;
  message m;
```

```
  va_start(argp, cmd);

  /* Set up for the sensible case where there is no variable parameter.  This
   * covers F_GETFD, F_GETFL and invalid commands.
   */
  m.m1_i3 = 0;
  m.m1_p1 = NIL_PTR;

  /* Adjust for the stupid cases. */
  switch(cmd) {
     case F_DUPFD:
     case F_SETFD:
     case F_SETFL:
        m.m1_i3 = va_arg(argp, int);
        break;
     case F_GETLK:
     case F_SETLK:
     case F_SETLKW:
        m.m1_p1 = (char *) va_arg(argp, struct flock *);
        break;
  }

  /* Clean up and make the system call. */
  va_end(argp);
  m.m1_i1 = fd;
  m.m1_i2 = cmd;
  return(_syscall(FS, FCNTL, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_fork.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define fork    _fork
#include <unistd.h>

PUBLIC pid_t fork()
{
  message m;

  return(_syscall(MM, FORK, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_fpathconf.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/* POSIX fpathconf (Sec. 5.7.1)                 Author: Andy Tanenbaum */

#include <lib.h>
#define fstat           _fstat
#define fpathconf       _fpathconf
#include <sys/stat.h>
#include <errno.h>
#include <limits.h>
#include <unistd.h>
#include <termios.h>

PUBLIC long fpathconf(fd, name)
int fd;                         /* file descriptor being interrogated */
int name;                       /* property being inspected */
{
/* POSIX allows some of the values in <limits.h> to be increased at
 * run time.  The pathconf and fpathconf functions allow these values
 * to be checked at run time.  MINIX does not use this facility.
 * The run-time limits are those given in <limits.h>.
 */

  struct stat stbuf;

  switch(name) {
        case _PC_LINK_MAX:
                /* Fstat the file.  If that fails, return -1. */
                if (fstat(fd, &stbuf) != 0) return(-1);
                if (S_ISDIR(stbuf.st_mode))
                        return(1L);     /* no links to directories */
                else
                        return( (long) LINK_MAX);

        case _PC_MAX_CANON:
                return( (long) MAX_CANON);

        case _PC_MAX_INPUT:
                return( (long) MAX_INPUT);

        case _PC_NAME_MAX:
                return( (long) NAME_MAX);

        case _PC_PATH_MAX:
                return( (long) PATH_MAX);

        case _PC_PIPE_BUF:
                return( (long) PIPE_BUF);

        case _PC_CHOWN_RESTRICTED:
                return( (long) _POSIX_CHOWN_RESTRICTED);

        case _PC_NO_TRUNC:
                return( (long) _POSIX_NO_TRUNC);

        case _PC_VDISABLE:
                return( (long) _POSIX_VDISABLE);
```

```

        default:
                errno = EINVAL;
                return(-1);
  }
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_fstat.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define fstat   _fstat
#include <sys/stat.h>

PUBLIC int fstat(fd, buffer)
int fd;
struct stat *buffer;
{
  message m;

  m.m1_i1 = fd;
  m.m1_p1 = (char *) buffer;
  return(_syscall(FS, FSTAT, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_getcwd.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*      getcwd() - get the name of the current working directory.
 *                                                      Author: Kees J. Bot
 *                                                              30 Apr 1989
 */
#define nil 0
#define chdir _chdir
#define closedir _closedir
#define getcwd _getcwd
#define opendir _opendir
#define readdir _readdir
#define rewinddir _rewinddir
#define stat _stat
#include <sys/types.h>
#include <sys/stat.h>
#include <errno.h>
#include <unistd.h>
#include <dirent.h>
#include <limits.h>
#include <string.h>

static int addpath(const char *path, char **ap, const char *entry)
/* Add the name of a directory entry at the front of the path being built.
 * Note that the result always starts with a slash.
 */
{
```

```
        const char *e= entry;
        char *p= *ap;

        while (*e != 0) e++;

        while (e > entry && p > path) *--p = *--e;

        if (p == path) return -1;
        *--p = '/';
        *ap= p;
        return 0;
}

static int recover(char *p)
/* Undo all those chdir("..")'s that have been recorded by addpath.  This
 * has to be done entry by entry, because the whole pathname may be too long.
 */
{
        int e= errno, slash;
        char *p0;

        while (*p != 0) {
                p0= ++p;

                do p++; while (*p != 0 && *p != '/');
                slash= *p; *p= 0;

                if (chdir(p0) < 0) return -1;
                *p= slash;
        }
        errno= e;
        return 0;
}

char *getcwd(char *path, size_t size)
{
        struct stat above, current, tmp;
        struct dirent *entry;
        DIR *d;
        char *p, *up, *dotdot;
        int cycle;

        if (path == nil || size <= 2) { errno= EINVAL; return nil; }

        p= path + size;
        *--p = 0;

        if (stat(".", &current) < 0) return nil;

        while (1) {
                dotdot= "..";
                if (stat(dotdot, &above) < 0) { recover(p); return nil; }

                if (above.st_dev == current.st_dev
                                        && above.st_ino == current.st_ino)
                        break;  /* Root dir found */

                if ((d= opendir(dotdot)) == nil) { recover(p); return nil; }

                /* Cycle is 0 for a simple inode nr search, or 1 for a search
```

```
		 * for inode *and* device nr.
		 */
		cycle= above.st_dev == current.st_dev ? 0 : 1;

		do {
			char name[3 + NAME_MAX + 1];

			tmp.st_ino= 0;
			if ((entry= readdir(d)) == nil) {
				switch (++cycle) {
				case 1:
					rewinddir(d);
					continue;
				case 2:
					closedir(d);
					errno= ENOENT;
					recover(p);
					return nil;
				}
			}
			if (strcmp(entry->d_name, ".") == 0) continue;
			if (strcmp(entry->d_name, "..") == 0) continue;

			switch (cycle) {
			case 0:
				/* Simple test on inode nr. */
				if (entry->d_ino != current.st_ino) continue;
				/*FALL THROUGH*/

			case 1:
				/* Current is mounted. */
				strcpy(name, "../");
				strcpy(name+3, entry->d_name);
				if (stat(name, &tmp) < 0) continue;
				break;
			}
		} while (tmp.st_ino != current.st_ino
					|| tmp.st_dev != current.st_dev);

		up= p;
		if (addpath(path, &up, entry->d_name) < 0) {
			closedir(d);
			errno = ERANGE;
			recover(p);
			return nil;
		}
		closedir(d);

		if (chdir(dotdot) < 0) { recover(p); return nil; }
		p= up;

		current= above;
	}
	if (recover(p) < 0) return nil;	/* Undo all those chdir("..")'s. */
	if (*p == 0) *--p = '/';	/* Cwd is "/" if nothing added */
	if (p > path) strcpy(path, p);	/* Move string to start of path. */
	return path;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_getegid.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define getegid _getegid
#include <unistd.h>

PUBLIC gid_t getegid()
{
  message m;

  /* POSIX says that this function is always successful and that no
   * return value is reserved to indicate an error.  Minix syscalls
   * are not always successful and Minix returns the unreserved value
   * (gid_t) -1 when there is an error.
   */
  if (_syscall(MM, GETGID, &m) < 0) return ( (gid_t) -1);
  return( (gid_t) m.m2_i1);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_geteuid.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define geteuid _geteuid
#include <unistd.h>

PUBLIC uid_t geteuid()
{
  message m;

  /* POSIX says that this function is always successful and that no
   * return value is reserved to indicate an error.  Minix syscalls
   * are not always successful and Minix returns the unreserved value
   * (uid_t) -1 when there is an error.
   */
  if (_syscall(MM, GETUID, &m) < 0) return ( (uid_t) -1);
  return( (uid_t) m.m2_i1);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_getgid.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define getgid  _getgid
#include <unistd.h>

PUBLIC gid_t getgid()
{
  message m;

  return( (gid_t) _syscall(MM, GETGID, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_getgroups.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/* getgroups.c                                          POSIX 4.2.3
 *      int getgroups(gidsetsize, grouplist);
 *
 *      This call relates to suplementary group ids, which are not
 *      supported in MINIX.
 */

#include <lib.h>
#define getgroups _getgroups
#include <unistd.h>
#include <time.h>

PUBLIC int getgroups(gidsetsize, grouplist)
int gidsetsize;
gid_t grouplist[];
{
  return(0);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_getpgrp.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define getpgrp _getpgrp
#include <unistd.h>

PUBLIC pid_t getpgrp()
{
  message m;

  return(_syscall(MM, GETPGRP, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_getpid.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define getpid  _getpid
#include <unistd.h>

PUBLIC pid_t getpid()
{
  message m;

  return(_syscall(MM, GETPID, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_getppid.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define getppid _getppid
#include <unistd.h>

PUBLIC pid_t getppid()
{
  message m;

  /* POSIX says that this function is always successful and that no
   * return value is reserved to indicate an error.  Minix syscalls
   * are not always successful and Minix returns the reserved value
   * (pid_t) -1 when there is an error.
   */
  if (_syscall(MM, GETPID, &m) < 0) return ( (pid_t) -1);
  return( (pid_t) m.m2_i1);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_getuid.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define getuid  _getuid
#include <unistd.h>

PUBLIC uid_t getuid()
{
  message m;

  return( (uid_t) _syscall(MM, GETUID, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/posix/_ioctl.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define ioctl   _ioctl
#include <minix/com.h>
#include <sys/ioctl.h>

PUBLIC int ioctl(fd, request, data)
int fd;
int request;
void *data;
{
  message m;

  m.TTY_LINE = fd;
  m.TTY_REQUEST = request;
  m.ADDRESS = (char *) data;
  return(_syscall(FS, IOCTL, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/posix/_isatty.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define isatty _isatty
#define tcgetattr _tcgetattr
#include <termios.h>
#include <unistd.h>

PUBLIC int isatty(fd)
int fd;
{
  struct termios dummy;

  return(tcgetattr(fd, &dummy) == 0);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/posix/_kill.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define kill    _kill
#include <signal.h>

PUBLIC int kill(proc, sig)
int proc;                       /* which process is to be sent the signal */
int sig;                        /* signal number */
{
  message m;

```

```
  m.m1_i1 = proc;
  m.m1_i2 = sig;
  return(_syscall(MM, KILL, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_link.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define link    _link
#include <string.h>
#include <unistd.h>

PUBLIC int link(name, name2)
_CONST char *name, *name2;
{
  message m;

  m.m1_i1 = strlen(name) + 1;
  m.m1_i2 = strlen(name2) + 1;
  m.m1_p1 = (char *) name;
  m.m1_p2 = (char *) name2;
  return(_syscall(FS, LINK, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_lseek.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define lseek   _lseek
#include <unistd.h>

PUBLIC off_t lseek(fd, offset, whence)
int fd;
off_t offset;
int whence;
{
  message m;

  m.m2_i1 = fd;
  m.m2_l1 = offset;
  m.m2_i2 = whence;
  if (_syscall(FS, LSEEK, &m) < 0) return( (off_t) -1);
  return( (off_t) m.m2_l1);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/posix/_mkdir.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define mkdir   _mkdir
#include <sys/stat.h>
#include <string.h>

PUBLIC int mkdir(name, mode)
_CONST char *name;
Mode_t mode;
{
  message m;

  m.m1_i1 = strlen(name) + 1;
  m.m1_i2 = mode;
  m.m1_p1 = (char *) name;
  return(_syscall(FS, MKDIR, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/posix/_mkfifo.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define mkfifo  _mkfifo
#define mknod   _mknod
#include <sys/stat.h>
#include <unistd.h>

PUBLIC int mkfifo(name, mode)
_CONST char *name;
Mode_t mode;
{
  return mknod(name, mode | S_IFIFO, (Dev_t) 0);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/posix/_mknod.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define mknod   _mknod
#include <string.h>
#include <stdlib.h>
#include <unistd.h>

PUBLIC int mknod(name, mode, dev)
_CONST char *name;
Mode_t mode;
Dev_t dev;
```

```
{
  message m;

  m.m1_i1 = strlen(name) + 1;
  m.m1_i2 = mode;
  m.m1_i3 = dev;
  m.m1_p1 = (char *) name;
  m.m1_p2 = (char *) ((int) 0);         /* obsolete size field */
  return(_syscall(FS, MKNOD, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/posix/_mktemp.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/* mktemp - make a name for a temporary file */

#include <lib.h>
#define access  _access
#define getpid  _getpid
#define mktemp  _mktemp
#include <unistd.h>

PUBLIC char *mktemp(template)
char *template;
{
  register int k;
  register char *p;
  register pid_t pid;

  pid = getpid();               /* get process id as semi-unique number */
  p = template;
  while (*p != 0) p++;          /* find end of string */

  /* Replace XXXXXX at end of template with a letter, then as many of the
   * trailing digits of the pid as fit.
   */
  while (*--p == 'X') {
        *p = '0' + (pid % 10);
        pid /= 10;
  }
  if (*++p != 0) {
        for (k = 'a'; k <= 'z'; k++) {
                *p = k;
                if (access(template, F_OK) < 0) return(template);
        }
  }
  return("/");
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_mount.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define mount   _mount
#include <string.h>
#include <unistd.h>

PUBLIC int mount(special, name, rwflag)
char *name, *special;
int rwflag;
{
  message m;

  m.m1_i1 = strlen(special) + 1;
  m.m1_i2 = strlen(name) + 1;
  m.m1_i3 = rwflag;
  m.m1_p1 = special;
  m.m1_p2 = name;
  return(_syscall(FS, MOUNT, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_open.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define open    _open
#include <fcntl.h>
#include <stdarg.h>
#include <string.h>

#if _ANSI
PUBLIC int open(const char *name, int flags, ...)
#else
PUBLIC int open(name, flags)
_CONST char *name;
int flags;
#endif
{
  va_list argp;
  message m;

  va_start(argp, flags);
  if (flags & O_CREAT) {
        m.m1_i1 = strlen(name) + 1;
        m.m1_i2 = flags;
        m.m1_i3 = va_arg(argp, Mode_t);
        m.m1_p1 = (char *) name;
  } else {
        _loadname(name, &m);
        m.m3_i2 = flags;
  }
  va_end(argp);
  return (_syscall(FS, OPEN, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_opendir.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*      opendir()                                       Author: Kees J. Bot
 *                                                              24 Apr 1989
 */
#define nil 0
#include <lib.h>
#define close   _close
#define fcntl   _fcntl
#define fstat   _fstat
#define open    _open
#define opendir _opendir
#define stat    _stat
#include <sys/types.h>
#include <sys/stat.h>
#include <dirent.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>

DIR *opendir(const char *name)
/* Open a directory for reading. */
{
        int d, f;
        DIR *dp;
        struct stat st;

        /* Only read directories. */
        if (stat(name, &st) < 0) return nil;
        if (!S_ISDIR(st.st_mode)) { errno= ENOTDIR; return nil; }

        if ((d= open(name, O_RDONLY | O_NONBLOCK)) < 0) return nil;

        /* Check the type again, mark close-on-exec, get a buffer. */
        if (fstat(d, &st) < 0
                || (errno= ENOTDIR, !S_ISDIR(st.st_mode))
                || (f= fcntl(d, F_GETFD)) < 0
                || fcntl(d, F_SETFD, f | FD_CLOEXEC) < 0
                || (dp= (DIR *) malloc(sizeof(*dp))) == nil
        ) {
                int err= errno;
                (void) close(d);
                errno= err;
                return nil;
        }

        dp->_fd= d;
        dp->_v7= -1;
        dp->_count= 0;
        dp->_pos= 0;

        return dp;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_pathconf.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/* POSIX pathconf (Sec. 5.7.1)          Author: Andy Tanenbaum */

#include <lib.h>
#define close           _close
#define open            _open
#define pathconf        _pathconf
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>

PUBLIC long pathconf(path, name)
_CONST char *path;              /* name of file being interrogated */
int name;                       /* property being inspected */
{
/* POSIX allows some of the values in <limits.h> to be increased at
 * run time.  The pathconf and fpathconf functions allow these values
 * to be checked at run time.  MINIX does not use this facility.
 * The run-time limits are those given in <limits.h>.
 */

  int fd;
  long val;

  if ( (fd = open(path, O_RDONLY)) < 0) return(-1L);
  val = fpathconf(fd, name);
  close(fd);
  return(val);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_pause.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define pause   _pause
#include <unistd.h>

PUBLIC int pause()
{
  message m;

  return(_syscall(MM, PAUSE, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_pipe.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define pipe    _pipe
#include <unistd.h>

PUBLIC int pipe(fild)
int fild[2];
{
  message m;

  if (_syscall(FS, PIPE, &m) < 0) return(-1);
  fild[0] = m.m1_i1;
  fild[1] = m.m1_i2;
  return(0);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_ptrace.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define ptrace  _ptrace
#include <unistd.h>

PUBLIC long ptrace(req, pid, addr, data)
int req;
pid_t pid;
long addr;
long data;
{
  message m;

  m.m2_i1 = pid;
  m.m2_i2 = req;
  m.m2_l1 = addr;
  m.m2_l2 = data;
  if (_syscall(MM, PTRACE, &m) < 0) return(-1);

  /* There was no error, but -1 is a legal return value.  Clear errno if
   * necessary to distinguish this case.  _syscall has set errno to nonzero
   * for the error case.
   */
  if (m.m2_l2 == -1) errno = 0;
  return(m.m2_l2);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_read.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
#include <lib.h>
#define read    _read
#include <unistd.h>

PUBLIC ssize_t read(fd, buffer, nbytes)
int fd;
void *buffer;
size_t nbytes;
{
  message m;

  m.m1_i1 = fd;
  m.m1_i2 = nbytes;
  m.m1_p1 = (char *) buffer;
  return(_syscall(FS, READ, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_readdir.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*      readdir()                                       Author: Kees J. Bot
 *                                                              24 Apr 1989
 */
#define nil 0
#include <lib.h>
#define read    _read
#define readdir _readdir
#include <sys/types.h>
#include <sys/stat.h>
#include <dirent.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <limits.h>
#include <errno.h>
#include <string.h>

#define v7ent(p)        ((struct _v7_direct *) (p))

struct dirent *readdir(DIR *dp)
/* Return the next entry in a directory.  Handle V7 and FLEX format dirs. */
{
        struct dirent *e;

        if (dp == nil) { errno= EBADF; return nil; }

        do {
                if (dp->_count <= 0) {
                        /* Read the next directory block. */
                        dp->_count= read(dp->_fd, dp->_buf, sizeof(dp->_buf));
```

```
                        if (dp->_count <= 0) return nil;

                        dp->_count/= sizeof(dp->_buf[0]);
                        dp->_ptr= dp->_buf;

                        /* Extent is zero of the first flex entry. */
                        if (dp->_v7 == (char)-1) dp->_v7= dp->_buf[0].d_extent;
                }

                if (!dp->_v7) {
                        /* FLEX. */
                        e= (struct dirent *) dp->_ptr;
                } else {
                        /* V7: transform to FLEX. */
                        e= (struct dirent *) dp->_v7f;
                        e->d_ino= v7ent(dp->_ptr)->d_ino;
                        e->d_extent= 1;
                        memcpy(e->d_name, v7ent(dp->_ptr)->d_name, 14);
                        e->d_name[14]= 0;
                }

                dp->_ptr+= 1 + e->d_extent;
                dp->_count-= 1 + e->d_extent;
                dp->_pos+= (1 + e->d_extent) * sizeof(*dp->_ptr);

        } while (e->d_ino == 0);
        return e;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_rename.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define rename  _rename
#include <string.h>
#include <stdio.h>

PUBLIC int rename(name, name2)
_CONST char *name, *name2;
{
  message m;

  m.m1_i1 = strlen(name) + 1;
  m.m1_i2 = strlen(name2) + 1;
  m.m1_p1 = (char *) name;
  m.m1_p2 = (char *) name2;
  return(_syscall(FS, RENAME, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_rewinddir.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*      rewinddir()                                     Author: Kees J. Bot
 *                                                               24 Apr 1989
 */
#define nil 0
#include <lib.h>
#define rewinddir _rewinddir
#define seekdir _seekdir
#include <sys/types.h>
#include <dirent.h>

void rewinddir(DIR *dp)
{
        (void) seekdir(dp, 0);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_rmdir.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define rmdir   _rmdir
#include <unistd.h>

PUBLIC int rmdir(name)
_CONST char *name;
{
  message m;

  _loadname(name, &m);
  return(_syscall(FS, RMDIR, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_setgid.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define setgid  _setgid
#include <unistd.h>

PUBLIC int setgid(grp)
gid_t grp;
{
  message m;

  m.m1_i1 = (int) grp;
  return(_syscall(MM, SETGID, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_setsid.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define setsid  _setsid
#include <unistd.h>

PUBLIC pid_t setsid()
{
  message m;

  return(_syscall(MM, SETSID, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_setuid.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define setuid  _setuid
#include <unistd.h>

PUBLIC int setuid(usr)
Uid_t usr;
{
  message m;

  m.m1_i1 = usr;
  return(_syscall(MM, SETUID, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_sigaction.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define sigaction _sigaction
#include <sys/sigcontext.h>
#include <signal.h>

_PROTOTYPE(int __sigreturn, (void));

PUBLIC int sigaction(sig, act, oact)
int sig;
_CONST struct sigaction *act;
struct sigaction *oact;
{
  message m;

  m.m1_i2 = sig;
```

```

  /* XXX - yet more type puns because message struct is short of types. */
  m.m1_p1 = (char *) act;
  m.m1_p2 = (char *) oact;
  m.m1_p3 = (char *) __sigreturn;

  return(_syscall(MM, SIGACTION, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/posix/_sigpending.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define sigpending _sigpending
#include <signal.h>

PUBLIC int sigpending(set)
sigset_t *set;
{
  message m;

  if (_syscall(MM, SIGPENDING, &m) < 0) return(-1);
  *set = (sigset_t) m.m2_l1;
  return(m.m_type);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/posix/_sigprocmask.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define sigprocmask _sigprocmask
#include <signal.h>

PUBLIC int sigprocmask(how, set, oset)
int how;
_CONST sigset_t *set;
sigset_t *oset;
{
  message m;

  if (set == (sigset_t *) NULL) {
        m.m2_i1 = SIG_INQUIRE;
        m.m2_l1 = 0;
  } else {
        m.m2_i1 = how;
        m.m2_l1 = (long) *set;
  }
  if (_syscall(MM, SIGPROCMASK, &m) < 0) return(-1);
  if (oset != (sigset_t *) NULL) *oset = (sigset_t) (m.m2_l1);
  return(m.m_type);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_sigreturn.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define sigfillset      _sigfillset
#define sigjmp          _sigjmp
#define sigprocmask     _sigprocmask
#define sigreturn       _sigreturn
#include <sys/sigcontext.h>
#include <setjmp.h>
#include <signal.h>

_PROTOTYPE( int sigjmp, (jmp_buf jb, int retval));

#if (_SETJMP_SAVES_REGS == 0)
/* 'sigreturn' using a short format jmp_buf (no registers saved). */
PUBLIC int sigjmp(jb, retval)
jmp_buf jb;
int retval;
{
  struct sigcontext sc;

  sc.sc_flags = jb[0].__flags;
  sc.sc_mask = jb[0].__mask;

#if (CHIP == INTEL)
  sc.sc_pc = (int) jb[0].__pc;
  sc.sc_sp = (int) jb[0].__sp;
  sc.sc_fp = (int) jb[0].__lb;
#endif

#if (CHIP == M68000)
  sc.sc_pc = (long) jb[0].__pc;
  sc.sc_sp = (long) jb[0].__sp;
  sc.sc_fp = (long) jb[0].__lb;
#endif

  sc.sc_retreg = retval;
  return sigreturn(&sc);
}
#endif

PUBLIC int sigreturn(scp)
register struct sigcontext *scp;
{
  sigset_t set;

  /* The message can't be on the stack, because the stack will vanish out
   * from under us.  The send part of sendrec will succeed, but when
   * a message is sent to restart the current process, who knows what will
   * be in the place formerly occupied by the message?
   */
  static message m;

  /* Protect against race conditions by blocking all interrupts. */
  sigfillset(&set);             /* splhi */
  sigprocmask(SIG_SETMASK, &set, (sigset_t *) NULL);

```

```
  m.m2_l1 = scp->sc_mask;
  m.m2_i2 = scp->sc_flags;
  m.m2_p1 = (char *) scp;
  return(_syscall(MM, SIGRETURN, &m));  /* normally this doesn't return */
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_sigset.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
/* XXX - these have to be hidden because signal() uses them and signal() is
 * ANSI and not POSIX.  It would be surely be better to use macros for the
 * library and system uses, and perhaps macros as well as functions for the
 * POSIX user interface.  The macros would not need underlines.  It may be
 * inconvenient to match the exact semantics of the current functions
 * because the interface is bloated by reporting errors.  For library and
 * system uses, the signal number is mostly already known to be valid
 * before the sigset-changing routines are called.
 */
#define sigaddset       _sigaddset
#define sigdelset       _sigdelset
#define sigemptyset     _sigemptyset
#define sigfillset      _sigfillset
#define sigismember     _sigismember
#include <signal.h>

/* Low bit of signal masks. */
#define SIGBIT_0        ((sigset_t) 1)

/* Mask of valid signals (0 - _NSIG).  Assume the shift doesn't overflow. */
#define SIGMASK         ((SIGBIT_0 << (_NSIG + 1)) - 1)

#define sigisvalid(signo) ((unsigned) (signo) <= _NSIG)

PUBLIC int sigaddset(set, signo)
sigset_t *set;
int signo;
{
  if (!sigisvalid(signo)) {
        errno = EINVAL;
        return -1;
  }
  *set |= SIGBIT_0 << signo;
  return 0;
}

PUBLIC int sigdelset(set, signo)
sigset_t *set;
int signo;
{
  if (!sigisvalid(signo)) {
        errno = EINVAL;
        return -1;
  }
```

```
  *set &= ~(SIGBIT_0 << signo);
  return 0;
}

PUBLIC int sigemptyset(set)
sigset_t *set;
{
  *set = 0;
  return 0;
}

PUBLIC int sigfillset(set)
sigset_t *set;
{
  *set = SIGMASK;
  return 0;
}

PUBLIC int sigismember(set, signo)
sigset_t *set;
int signo;
{
  if (!sigisvalid(signo)) {
        errno = EINVAL;
        return -1;
  }
  if (*set & (SIGBIT_0 << signo))
        return 1;
  return 0;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_sigsetjmp.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#include <sys/sigcontext.h>
#include <setjmp.h>

PUBLIC void siglongjmp(env, val)
sigjmp_buf env;
int val;
{
  if (env[0].__flags & SC_SIGCONTEXT)
        longjmp(env, val);
  else
        _longjmp(env, val);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/posix/_sigsuspend.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define sigsuspend _sigsuspend
#include <signal.h>

PUBLIC int sigsuspend(set)
_CONST sigset_t *set;
{
  message m;

  m.m2_l1 = (long) *set;
  return(_syscall(MM, SIGSUSPEND, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/posix/_sleep.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*  sleep(3)
 *
 *  Sleep(n) pauses for 'n' seconds by scheduling an alarm interrupt.
 *
 *  Changed to conform with POSIX      Terrence W. Holm      Oct. 1988
 */

#include <lib.h>
#define sleep _sleep
#include <signal.h>
#include <unistd.h>

FORWARD _PROTOTYPE( void _alfun, (int signo) );

PRIVATE void _alfun(signo)
int signo;
{
/* Dummy signal handler used with sleep() below. */
}

PUBLIC unsigned sleep(secs)
unsigned secs;
{
  unsigned current_secs;
  unsigned remaining_secs;
  struct sigaction act, oact;
  sigset_t ss;

  if (secs == 0) return(0);

  current_secs = alarm(0);      /* is there currently an alarm?  */

  if (current_secs == 0 || current_secs > secs) {
        act.sa_flags = 0;
        act.sa_mask = 0;
```

```
        act.sa_handler = _alfun;
        sigaction(SIGALRM, &act, &oact);

        alarm(secs);
        sigemptyset(&ss);
        sigsuspend(&ss);
        remaining_secs = alarm(0);

        sigaction(SIGALRM, &oact, (struct sigaction *) NULL);

        if (current_secs > secs)
                alarm(current_secs - (secs - remaining_secs));

        return(remaining_secs);
  }

  /* Current_secs <= secs,  ie. alarm should occur before secs.  */

  alarm(current_secs);
  pause();
  remaining_secs = alarm(0);

  alarm(remaining_secs);

  return(secs - (current_secs - remaining_secs));
}
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/posix/_stat.c
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define stat    _stat
#include <sys/stat.h>
#include <string.h>

PUBLIC int stat(name, buffer)
_CONST char *name;
struct stat *buffer;
{
  message m;

  m.m1_i1 = strlen(name) + 1;
  m.m1_p1 = (char *) name;
  m.m1_p2 = (char *) buffer;
  return(_syscall(FS, STAT, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_stime.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define stime   _stime
#include <minix/minlib.h>
#include <time.h>

PUBLIC int stime(top)
long *top;
{
  message m;

  m.m2_l1 = *top;
  return(_syscall(FS, STIME, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_sync.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define sync    _sync
#include <unistd.h>

PUBLIC int sync()
{
  message m;

  return(_syscall(FS, SYNC, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_tcdrain.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
posix/_tcdrain.c

Created:        July 26, 1994 by Philip Homburg
*/

#define tcdrain _tcdrain
#define ioctl _ioctl
#include <termios.h>
#include <sys/ioctl.h>

int tcdrain(fd)
int fd;
{
  return(ioctl(fd, TCDRAIN, (void *)0));
```

```
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/posix/_tcflow.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
posix/_tcflow.c

Created:        June 8, 1993 by Philip Homburg
*/

#define tcflow _tcflow
#define ioctl _ioctl
#include <termios.h>
#include <sys/ioctl.h>

int tcflow(fd, action)
int fd;
int action;
{
  return(ioctl(fd, TCFLOW, &action));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/posix/_tcflush.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*      tcflush() - flush buffered characters           Author: Kees J. Bot
 *                                                              13 Jan 1994
 */
#define tcflush _tcflush
#define ioctl _ioctl
#include <termios.h>
#include <sys/ioctl.h>

int tcflush(int fd, int queue_selector)
{
  return(ioctl(fd, TCFLSH, &queue_selector));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_tcgetattr.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#define tcgetattr _tcgetattr
#define ioctl _ioctl
#include <sys/ioctl.h>
#include <errno.h>
#include <termios.h>

int tcgetattr(fd, termios_p)
int fd;
struct termios *termios_p;
{
  return(ioctl(fd, TCGETS, termios_p));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_tcsendbreak.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*      tcsendbreak() - send a break                    Author: Kees J. Bot
 *                                                              13 Jan 1994
 */
#define tcsendbreak _tcsendbreak
#define ioctl _ioctl
#include <termios.h>
#include <sys/ioctl.h>

int tcsendbreak(int fd, int duration)
{
  return(ioctl(fd, TCSBRK, &duration));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_tcsetattr.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
posix/_tcsetattr.c

Created:        June 11, 1993 by Philip Homburg
*/

#define tcsetattr _tcsetattr
#define ioctl _ioctl
#include <errno.h>
#include <termios.h>
#include <sys/ioctl.h>
#include <sys/types.h>

int tcsetattr(fd, opt_actions, termios_p)
int fd;
```

```
int opt_actions;
_CONST struct termios *termios_p;
{
  int request;

  switch(opt_actions)
  {
    case TCSANOW:       request = TCSETS;       break;
    case TCSADRAIN:     request = TCSETSW;      break;
    case TCSAFLUSH:     request = TCSETSF;      break;
    default:            errno = EINVAL;         return(-1);
  };
  return(ioctl(fd, request, (void *) termios_p));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_time.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define time    _time
#include <time.h>

PUBLIC time_t time(tp)
time_t *tp;
{
  message m;

  if (_syscall(FS, TIME, &m) < 0) return( (time_t) -1);
  if (tp != (time_t *) 0) *tp = m.m2_l1;
  return(m.m2_l1);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_times.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define times   _times
#include <sys/times.h>
#include <time.h>

PUBLIC clock_t times(buf)
struct tms *buf;
{
  message m;

  m.m4_l5 = 0;                  /* return this if system is pre-1.6 */
  if (_syscall(FS, TIMES, &m) < 0) return( (clock_t) -1);
  buf->tms_utime = m.m4_l1;
  buf->tms_stime = m.m4_l2;
  buf->tms_cutime = m.m4_l3;
  buf->tms_cstime = m.m4_l4;
  return(m.m4_l5);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_umask.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define umask   _umask
#include <sys/stat.h>

PUBLIC mode_t umask(complmode)
Mode_t complmode;
{
  message m;

  m.m1_i1 = complmode;
  return( (mode_t) _syscall(FS, UMASK, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_umount.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define umount  _umount
#include <unistd.h>

PUBLIC int umount(name)
_CONST char *name;
{
  message m;

  _loadname(name, &m);
  return(_syscall(FS, UMOUNT, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_uname.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*      uname() - get system info                       Author: Kees J. Bot
 *                                                              7 Nov 1994
 * Returns information about the Minix system.  Alas most
 * of it is gathered at compile time, so machine is wrong, and
 * release and version become wrong if not recompiled.
 * More chip types and Minix versions need to be added.
 */
#define uname   _uname
#define open    _open
#define read    _read
#define close   _close
#include <sys/types.h>
#include <sys/utsname.h>
#include <unistd.h>
#include <fcntl.h>
```

```
#include <string.h>
#include <errno.h>
#include <minix/config.h>
#include <minix/minlib.h>

int uname(name) struct utsname *name;
{
  int hf, n, err;
  char *nl;

  /* Read the node name from /etc/hostname.file. */
  if ((hf = open("/etc/hostname.file", O_RDONLY)) < 0) {
        if (errno != ENOENT) return(-1);
        strcpy(name->nodename, "noname");
  } else {
        n = read(hf, name->nodename, sizeof(name->nodename) - 1);
        err = errno;
        close(hf);
        errno = err;
        if (n < 0) return(-1);
        name->nodename[n] = 0;
        if ((nl = strchr(name->nodename, '\n')) != NULL) {
                memset(nl, 0, (name->nodename + sizeof(name->nodename)) - nl);
        }
  }

  strcpy(name->sysname, "Minix");
  strcpy(name->release, OS_RELEASE);
  strcpy(name->version, OS_VERSION);
#if (CHIP == INTEL)
  name->machine[0] = 'i';
  strcpy(name->machine + 1, itoa(getprocessor()));
#if _WORD_SIZE == 4
  strcpy(name->arch, "i386");
#else
  strcpy(name->arch, "i86");
#endif
#endif
  return(0);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_unlink.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define unlink  _unlink
#include <unistd.h>

PUBLIC int unlink(name)
_CONST char *name;
{
  message m;

  _loadname(name, &m);
```

```
  return(_syscall(FS, UNLINK, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_utime.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/* utime(2) for POSIX           Authors: Terrence W. Holm & Edwin L. Froese */

#include <lib.h>
#define utime   _utime
#include <string.h>
#include <utime.h>

PUBLIC int utime(name, timp)
_CONST char *name;
_CONST struct utimbuf *timp;
{
  message m;

  if (timp == NULL) {
        m.m2_i1 = 0;            /* name size 0 means NULL imp' */
        m.m2_i2 = strlen(name) + 1;     /* actual size here */
  } else {
        m.m2_l1 = timp->actime;
        m.m2_l2 = timp->modtime;
        m.m2_i1 = strlen(name) + 1;
  }
  m.m2_p1 = (char *) name;
  return(_syscall(FS, UTIME, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_wait.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include <lib.h>
#define wait    _wait
#include <sys/wait.h>

PUBLIC pid_t wait(status)
int *status;
{
  message m;

  if (_syscall(MM, WAIT, &m) < 0) return(-1);
  if (status != 0) *status = m.m2_i1;
  return(m.m_type);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_waitpid.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define waitpid _waitpid
#include <sys/wait.h>

PUBLIC pid_t waitpid(pid, status, options)
pid_t pid;
int *status;
int options;
{
  message m;

  m.m1_i1 = pid;
  m.m1_i2 = options;
  if (_syscall(MM, WAITPID, &m) < 0) return(-1);
  if (status != 0) *status = m.m2_i1;
  return m.m_type;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/posix/_write.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#include <lib.h>
#define write   _write
#include <unistd.h>

PUBLIC ssize_t write(fd, buffer, nbytes)
int fd;
_CONST void *buffer;
size_t nbytes;
{
  message m;

  m.m1_i1 = fd;
  m.m1_i2 = nbytes;
  m.m1_p1 = (char *) buffer;
  return(_syscall(FS, WRITE, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/loc_incl.h
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * loc_incl.h - local include file for stdio library
 */
/* $Header: loc_incl.h,v 1.5 91/06/10 17:07:18 ceriel Exp $ */

#include        <stdio.h>

#define io_testflag(p,x)        ((p)->_flags & (x))

#include        <stdarg.h>

#ifdef _ANSI
int _doprnt(const char *format, va_list ap, FILE *stream);
int _doscan(FILE * stream, const char *format, va_list ap);
char *_i_compute(unsigned long val, int base, char *s, int nrdigits);
char *_f_print(va_list *ap, int flags, char *s, char c, int precision);
void __cleanup(void);

FILE *popen(const char *command, const char *type);
FILE *fdopen(int fd, const char *mode);

#ifndef NOFLOAT
char *_ecvt(long double value, int ndigit, int *decpt, int *sign);
char *_fcvt(long double value, int ndigit, int *decpt, int *sign);
#endif  /* NOFLOAT */
#endif

#define FL_LJUST        0x0001          /* left-justify field */
#define FL_SIGN         0x0002          /* sign in signed conversions */
#define FL_SPACE        0x0004          /* space in signed conversions */
#define FL_ALT          0x0008          /* alternate form */
#define FL_ZEROFILL     0x0010          /* fill with zero's */
#define FL_SHORT        0x0020          /* optional h */
#define FL_LONG         0x0040          /* optional l */
#define FL_LONGDOUBLE   0x0080          /* optional L */
#define FL_WIDTHSPEC    0x0100          /* field width is specified */
#define FL_PRECSPEC     0x0200          /* precision is specified */
#define FL_SIGNEDCONV   0x0400          /* may contain a sign */
#define FL_NOASSIGN     0x0800          /* do not assign (in scanf) */
#define FL_NOMORE       0x1000          /* all flags collected */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/stdio/clearerr.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * clearerr.c - clear error and end-of-file indicators of a stream
 */
/* $Header: clearerr.c,v 1.2 91/01/03 14:23:54 ceriel Exp $ */

#include        <stdio.h>

void
(clearerr)(FILE *stream)
{
        clearerr(stream);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/stdio/data.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * data.c - this is the initialization for the standard streams
 */
/* $Header: data.c,v 1.3 90/04/09 15:35:09 eck Exp $ */

#include        <stdio.h>

struct __iobuf __stdin = {
        0, 0, _IOREAD, 0,
        (unsigned char *)NULL, (unsigned char *)NULL,
};

struct __iobuf __stdout = {
        0, 1, _IOWRITE, 0,
        (unsigned char *)NULL, (unsigned char *)NULL,
};

struct __iobuf __stderr = {
        0, 2, _IOWRITE | _IOLBF, 0,
        (unsigned char *)NULL, (unsigned char *)NULL,
};

FILE *__iotab[FOPEN_MAX] = {
        &__stdin,
        &__stdout,
        &__stderr,
        0
};
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/doprnt.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * doprnt.c - print formatted output
 */
/* $Header: doprnt.c,v 1.8 91/06/10 17:06:53 ceriel Exp $ */

#include        <ctype.h>
#include        <stdio.h>
#include        <stdarg.h>
#include        <string.h>
#include        "loc_incl.h"

/* gnum() is used to get the width and precision fields of a format. */
static const char *
gnum(register const char *f, int *ip, va_list *app)
{
        register int    i, c;

        if (*f == '*') {
                *ip = va_arg((*app), int);
                f++;
        } else {
                i = 0;
                while ((c = *f - '0') >= 0 && c <= 9) {
                        i = i*10 + c;
                        f++;
                }
                *ip = i;
        }
        return f;
}

#if     _EM_WSIZE == _EM_PSIZE
#define set_pointer(flags)                              /* nothing */
#elif   _EM_LSIZE == _EM_PSIZE
#define set_pointer(flags)      (flags |= FL_LONG)
#else
#error garbage pointer size
#define set_pointer(flags)              /* compilation might continue */
#endif

/* print an ordinal number */
static char *
o_print(va_list *ap, int flags, char *s, char c, int precision, int is_signed)
{
        long signed_val;
        unsigned long unsigned_val;
        char *old_s = s;
        int base;

        switch (flags & (FL_SHORT | FL_LONG)) {
        case FL_SHORT:
                if (is_signed) {
                        signed_val = (short) va_arg(*ap, int);
                } else {
                        unsigned_val = (unsigned short) va_arg(*ap, unsigned);
```

```
                        }
                        break;
                case FL_LONG:
                        if (is_signed) {
                                signed_val = va_arg(*ap, long);
                        } else {
                                unsigned_val = va_arg(*ap, unsigned long);
                        }
                        break;
                default:
                        if (is_signed) {
                                signed_val = va_arg(*ap, int);
                        } else {
                                unsigned_val = va_arg(*ap, unsigned int);
                        }
                        break;
                }

                if (is_signed) {
                        if (signed_val < 0) {
                                *s++ = '-';
                                signed_val = -signed_val;
                        } else if (flags & FL_SIGN) *s++ = '+';
                        else if (flags & FL_SPACE) *s++ = ' ';
                        unsigned_val = signed_val;
                }
                if ((flags & FL_ALT) && (c == 'o')) *s++ = '0';
                if (!unsigned_val) {
                         if (!precision)
                                return s;
                } else if (((flags & FL_ALT) && (c == 'x' || c == 'X'))
                            || c == 'p') {
                        *s++ = '0';
                        *s++ = (c == 'X' ? 'X' : 'x');
                }

                switch (c) {
                case 'b':       base = 2;       break;
                case 'o':       base = 8;       break;
                case 'd':
                case 'i':
                case 'u':       base = 10;      break;
                case 'x':
                case 'X':
                case 'p':       base = 16;      break;
                }

                s = _i_compute(unsigned_val, base, s, precision);

                if (c == 'X')
                        while (old_s != s) {
                                *old_s = toupper(*old_s);
                                old_s++;
                        }

                return s;
        }

        int
        _doprnt(register const char *fmt, va_list ap, FILE *stream)
```

```
{
        register char   *s;
        register int    j;
        int             i, c, width, precision, zfill, flags, between_fill;
        int             nrchars=0;
        const char      *oldfmt;
        char            *s1, buf[1025];

        while (c = *fmt++) {
                if (c != '%') {
#ifdef  CPM
                        if (c == '\n') {
                                if (putc('\r', stream) == EOF)
                                        return nrchars ? -nrchars : -1;
                                nrchars++;
                        }
#endif
                        if (putc(c, stream) == EOF)
                                return nrchars ? -nrchars : -1;
                        nrchars++;
                        continue;
                }
                flags = 0;
                do {
                        switch(*fmt) {
                        case '-':       flags |= FL_LJUST;      break;
                        case '+':       flags |= FL_SIGN;       break;
                        case ' ':       flags |= FL_SPACE;      break;
                        case '#':       flags |= FL_ALT;        break;
                        case '0':       flags |= FL_ZEROFILL;   break;
                        default:        flags |= FL_NOMORE;     continue;
                        }
                        fmt++;
                } while(!(flags & FL_NOMORE));

                oldfmt = fmt;
                fmt = gnum(fmt, &width, &ap);
                if (fmt != oldfmt) flags |= FL_WIDTHSPEC;

                if (*fmt == '.') {
                        fmt++; oldfmt = fmt;
                        fmt = gnum(fmt, &precision, &ap);
                        if (precision >= 0) flags |= FL_PRECSPEC;
                }

                if ((flags & FL_WIDTHSPEC) && width < 0) {
                        width = -width;
                        flags |= FL_LJUST;
                }
                if (!(flags & FL_WIDTHSPEC)) width = 0;

                if (flags & FL_SIGN) flags &= ~FL_SPACE;

                if (flags & FL_LJUST) flags &= ~FL_ZEROFILL;


                s = s1 = buf;

                switch (*fmt) {
                case 'h':       flags |= FL_SHORT; fmt++; break;
```

```
                case 'l':       flags |= FL_LONG; fmt++; break;
                case 'L':       flags |= FL_LONGDOUBLE; fmt++; break;
                }

                switch (c = *fmt++) {
                default:
#ifdef  CPM
                        if (c == '\n') {
                                if (putc('\r', stream) == EOF)
                                        return nrchars ? -nrchars : -1;
                                nrchars++;
                        }
#endif
                        if (putc(c, stream) == EOF)
                                return nrchars ? -nrchars : -1;
                        nrchars++;
                        continue;
                case 'n':
                        if (flags & FL_SHORT)
                                *va_arg(ap, short *) = (short) nrchars;
                        else if (flags & FL_LONG)
                                *va_arg(ap, long *) = (long) nrchars;
                        else
                                *va_arg(ap, int *) = (int) nrchars;
                        continue;
                case 's':
                        s1 = va_arg(ap, char *);
                        if (s1 == NULL)
                                s1 = "(null)";
                        s = s1;
                        while (precision || !(flags & FL_PRECSPEC)) {
                                if (*s == '\0')
                                        break;
                                s++;
                                precision--;
                        }
                        break;
                case 'p':
                        set_pointer(flags);
                        /* fallthrough */
                case 'b':
                case 'o':
                case 'u':
                case 'x':
                case 'X':
                        if (!(flags & FL_PRECSPEC)) precision = 1;
                        else if (c != 'p') flags &= ~FL_ZEROFILL;
                        s = o_print(&ap, flags, s, c, precision, 0);
                        break;
                case 'd':
                case 'i':
                        flags |= FL_SIGNEDCONV;
                        if (!(flags & FL_PRECSPEC)) precision = 1;
                        else flags &= ~FL_ZEROFILL;
                        s = o_print(&ap, flags, s, c, precision, 1);
                        break;
                case 'c':
                        *s++ = va_arg(ap, int);
                        break;
#ifndef NOFLOAT
```

```
                case 'G':
                case 'g':
                        if ((flags & FL_PRECSPEC) && (precision == 0))
                                precision = 1;
                case 'f':
                case 'E':
                case 'e':
                        if (!(flags & FL_PRECSPEC))
                                precision = 6;

                        if (precision >= sizeof(buf))
                                precision = sizeof(buf) - 1;

                        flags |= FL_SIGNEDCONV;
                        s = _f_print(&ap, flags, s, c, precision);
                        break;
#endif  /* NOFLOAT */
                case 'r':
                        ap = va_arg(ap, va_list);
                        fmt = va_arg(ap, char *);
                        continue;
                }
                zfill = ' ';
                if (flags & FL_ZEROFILL) zfill = '0';
                j = s - s1;

                /* between_fill is true under the following conditions:
                 * 1- the fill character is '0'
                 * and
                 * 2a- the number is of the form 0x... or 0X...
                 * or
                 * 2b- the number contains a sign or space
                 */
                between_fill = 0;
                if ((flags & FL_ZEROFILL)
                    && (((c == 'x' || c == 'X') && (flags & FL_ALT))
                        || (c == 'p')
                        || ((flags & FL_SIGNEDCONV)
                            && ( *s1 == '+' || *s1 == '-' || *s1 == ' '))))
                        between_fill++;

                if ((i = width - j) > 0)
                        if (!(flags & FL_LJUST)) {      /* right justify */
                                nrchars += i;
                                if (between_fill) {
                                    if (flags & FL_SIGNEDCONV) {
                                        j--; nrchars++;
                                        if (putc(*s1++, stream) == EOF)
                                                return nrchars ? -nrchars : -1;
                                    } else {
                                        j -= 2; nrchars += 2;
                                        if ((putc(*s1++, stream) == EOF)
                                            || (putc(*s1++, stream) == EOF))
                                                return nrchars ? -nrchars : -1;
                                    }
                                }
                                do {
                                        if (putc(zfill, stream) == EOF)
                                                return nrchars ? -nrchars : -1;
                                } while (--i);
```

```
                        }

                nrchars += j;
                while (--j >= 0) {
                        if (putc(*s1++, stream) == EOF)
                                return nrchars ? -nrchars : -1;
                }

                if (i > 0) nrchars += i;
                while (--i >= 0)
                        if (putc(zfill, stream) == EOF)
                                return nrchars ? -nrchars : -1;
        }
        return nrchars;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/stdio/doscan.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * doscan.c - scan formatted input
 */
/* $Header: doscan.c,v 1.9 91/03/26 18:41:47 ceriel Exp $ */

#include        <stdio.h>
#include        <stdlib.h>
#include        <ctype.h>
#include        <stdarg.h>
#include        "loc_incl.h"

#if     _EM_WSIZE == _EM_PSIZE
#define set_pointer(flags)                              /* nothing */
#elif   _EM_LSIZE == _EM_PSIZE
#define set_pointer(flags)      (flags |= FL_LONG)
#else
#error garbage pointer size
#define set_pointer(flags)              /* compilation might continue */
#endif

#define NUMLEN  512
#define NR_CHARS        256

static char     Xtable[NR_CHARS];
static char     inp_buf[NUMLEN];

/* Collect a number of characters which constitite an ordinal number.
 * When the type is 'i', the base can be 8, 10, or 16, depending on the
 * first 1 or 2 characters. This means that the base must be adjusted
 * according to the format of the number. At the end of the function, base
 * is then set to 0, so strtol() will get the right argument.
 */
static char *
o_collect(register int c, register FILE *stream, char type,
                        unsigned int width, int *basep)
```

```
{
        register char *bufp = inp_buf;
        register int base;

        switch (type) {
        case 'i':       /* i means octal, decimal or hexadecimal */
        case 'p':
        case 'x':
        case 'X':       base = 16;      break;
        case 'd':
        case 'u':       base = 10;      break;
        case 'o':       base = 8;       break;
        case 'b':       base = 2;       break;
        }

        if (c == '-' || c == '+') {
                *bufp++ = c;
                if (--width)
                    c = getc(stream);
        }

        if (width && c == '0' && base == 16) {
                *bufp++ = c;
                if (--width)
                        c = getc(stream);
                if (c != 'x' && c != 'X') {
                        if (type == 'i') base = 8;
                }
                else if (width) {
                        *bufp++ = c;
                        if (--width)
                                c = getc(stream);
                }
        }
        else if (type == 'i') base = 10;

        while (width) {
                if (((base == 10) && isdigit(c))
                    || ((base == 16) && isxdigit(c))
                    || ((base == 8) && isdigit(c) && (c < '8'))
                    || ((base == 2) && isdigit(c) && (c < '2'))) {
                        *bufp++ = c;
                        if (--width)
                                c = getc(stream);
                }
                else break;
        }

        if (width && c != EOF) ungetc(c, stream);
        if (type == 'i') base = 0;
        *basep = base;
        *bufp = '\0';
        return bufp - 1;
}

#ifndef NOFLOAT
/* The function f_collect() reads a string that has the format of a
 * floating-point number. The function returns as soon as a format-error
 * is encountered, leaving the offending character in the input. This means
 * that 1.el leaves the 'l' in the input queue. Since all detection of
```

```
 * format errors is done here, _doscan() doesn't call strtod() when it's
 * not necessary, although the use of the width field can cause incomplete
 * numbers to be passed to strtod(). (e.g. 1.3e+)
 */
static char *
f_collect(register int c, register FILE *stream, register unsigned int width)
{
        register char *bufp = inp_buf;
        int digit_seen = 0;

        if (c == '-' || c == '+') {
                *bufp++ = c;
                if (--width)
                        c = getc(stream);
        }

        while (width && isdigit(c)) {
                digit_seen++;
                *bufp++ = c;
                if (--width)
                        c = getc(stream);
        }
        if (width && c == '.') {
                *bufp++ = c;
                if(--width)
                        c = getc(stream);
                while (width && isdigit(c)) {
                        digit_seen++;
                        *bufp++ = c;
                        if (--width)
                                c = getc(stream);
                }
        }

        if (!digit_seen) {
                if (width && c != EOF) ungetc(c, stream);
                return inp_buf - 1;
        }
        else digit_seen = 0;

        if (width && (c == 'e' || c == 'E')) {
                *bufp++ = c;
                if (--width)
                        c = getc(stream);
                if (width && (c == '+' || c == '-')) {
                        *bufp++ = c;
                        if (--width)
                                c = getc(stream);
                }
                while (width && isdigit(c)) {
                        digit_seen++;
                        *bufp++ = c;
                        if (--width)
                                c = getc(stream);
                }
                if (!digit_seen) {
                        if (width && c != EOF) ungetc(c,stream);
                        return inp_buf - 1;
                }
        }
```

```
        if (width && c != EOF) ungetc(c, stream);
        *bufp = '\0';
        return bufp - 1;
}
#endif  /* NOFLOAT */


/*
 * the routine that does the scanning
 */

int
_doscan(register FILE *stream, const char *format, va_list ap)
{
        int             done = 0;       /* number of items done */
        int             nrchars = 0;    /* number of characters read */
        int             conv = 0;       /* # of conversions */
        int             base;           /* conversion base */
        unsigned long   val;            /* an integer value */
        register char   *str;           /* temporary pointer */
        char            *tmp_string;    /* ditto */
        unsigned        width = 0;      /* width of field */
        int             flags;          /* some flags */
        int             reverse;        /* reverse the checking in [...] */
        int             kind;
        register int    ic = EOF;       /* the input character */
#ifndef NOFLOAT
        long double     ld_val;
#endif

        if (!*format) return 0;

        while (1) {
                if (isspace(*format)) {
                        while (isspace(*format))
                                format++;       /* skip whitespace */
                        ic = getc(stream);
                        nrchars++;
                        while (isspace (ic)) {
                                ic = getc(stream);
                                nrchars++;
                        }
                        if (ic != EOF) ungetc(ic,stream);
                        nrchars--;
                }
                if (!*format) break;    /* end of format */

                if (*format != '%') {
                        ic = getc(stream);
                        nrchars++;
                        if (ic != *format++) break;     /* error */
                        continue;
                }
                format++;
                if (*format == '%') {
                        ic = getc(stream);
                        nrchars++;
                        if (ic == '%') {
                                format++;
```

```
                                continue;
                        }
                        else break;
                }
                flags = 0;
                if (*format == '*') {
                        format++;
                        flags |= FL_NOASSIGN;
                }
                if (isdigit (*format)) {
                        flags |= FL_WIDTHSPEC;
                        for (width = 0; isdigit (*format);)
                                width = width * 10 + *format++ - '0';
                }

                switch (*format) {
                case 'h': flags |= FL_SHORT; format++; break;
                case 'l': flags |= FL_LONG; format++; break;
                case 'L': flags |= FL_LONGDOUBLE; format++; break;
                }
                kind = *format;
                if ((kind != 'c') && (kind != '[') && (kind != 'n')) {
                        do {
                                ic = getc(stream);
                                nrchars++;
                        } while (isspace(ic));
                        if (ic == EOF) break;           /* outer while */
                } else if (kind != 'n') {               /* %c or %[ */
                        ic = getc(stream);
                        if (ic == EOF) break;           /* outer while */
                        nrchars++;
                }
                switch (kind) {
                default:
                        /* not recognized, like %q */
                        return conv || (ic != EOF) ? done : EOF;
                        break;
                case 'n':
                        if (!(flags & FL_NOASSIGN)) {   /* silly, though */
                                if (flags & FL_SHORT)
                                        *va_arg(ap, short *) = (short) nrchars;
                                else if (flags & FL_LONG)
                                        *va_arg(ap, long *) = (long) nrchars;
                                else
                                        *va_arg(ap, int *) = (int) nrchars;
                        }
                        break;
                case 'p':               /* pointer */
                        set_pointer(flags);
                        /* fallthrough */
                case 'b':               /* binary */
                case 'd':               /* decimal */
                case 'i':               /* general integer */
                case 'o':               /* octal */
                case 'u':               /* unsigned */
                case 'x':               /* hexadecimal */
                case 'X':               /* ditto */
                        if (!(flags & FL_WIDTHSPEC) || width > NUMLEN)
                                width = NUMLEN;
                        if (!width) return done;
```

```
				str = o_collect(ic, stream, kind, width, &base);
				if (str < inp_buf
				    || (str == inp_buf
					    && (*str == '-'
						|| *str == '+'))) return done;

				/*
				 * Although the length of the number is str-inp_buf+1
				 * we don't add the 1 since we counted it already
				 */
				nrchars += str - inp_buf;

				if (!(flags & FL_NOASSIGN)) {
					if (kind == 'd' || kind == 'i')
					    val = strtol(inp_buf, &tmp_string, base);
					else
					    val = strtoul(inp_buf, &tmp_string, base);
					if (flags & FL_LONG)
						*va_arg(ap, unsigned long *) = (unsigned long) val;
					else if (flags & FL_SHORT)
						*va_arg(ap, unsigned short *) = (unsigned short) val;
					else
						*va_arg(ap, unsigned *) = (unsigned) val;
				}
				break;
			case 'c':
				if (!(flags & FL_WIDTHSPEC))
					width = 1;
				if (!(flags & FL_NOASSIGN))
					str = va_arg(ap, char *);
				if (!width) return done;

				while (width && ic != EOF) {
					if (!(flags & FL_NOASSIGN))
						*str++ = (char) ic;
					if (--width) {
						ic = getc(stream);
						nrchars++;
					}
				}

				if (width) {
					if (ic != EOF) ungetc(ic,stream);
					nrchars--;
				}
				break;
			case 's':
				if (!(flags & FL_WIDTHSPEC))
					width = 0xffff;
				if (!(flags & FL_NOASSIGN))
					str = va_arg(ap, char *);
				if (!width) return done;

				while (width && ic != EOF && !isspace(ic)) {
					if (!(flags & FL_NOASSIGN))
						*str++ = (char) ic;
					if (--width) {
						ic = getc(stream);
						nrchars++;
```

```
                                }
                        }
                        /* terminate the string */
                        if (!(flags & FL_NOASSIGN))
                                *str = '\0';
                        if (width) {
                                if (ic != EOF) ungetc(ic,stream);
                                nrchars--;
                        }
                        break;
                case '[':
                        if (!(flags & FL_WIDTHSPEC))
                                width = 0xffff;
                        if (!width) return done;

                        if ( *++format == '^' ) {
                                reverse = 1;
                                format++;
                        } else
                                reverse = 0;

                        for (str = Xtable; str < &Xtable[NR_CHARS]
                                                        ; str++)
                                *str = 0;

                        if (*format == ']') Xtable[*format++] = 1;

                        while (*format && *format != ']') {
                                Xtable[*format++] = 1;
                                if (*format == '-') {
                                        format++;
                                        if (*format
                                            && *format != ']'
                                            && *(format) >= *(format -2)) {
                                                int c;

                                                for( c = *(format -2) + 1
                                                    ; c <= *format ; c++)
                                                        Xtable[c] = 1;
                                                format++;
                                        }
                                        else Xtable['-'] = 1;
                                }
                        }
                        if (!*format) return done;

                        if (!(Xtable[ic] ^ reverse)) return done;

                        if (!(flags & FL_NOASSIGN))
                                str = va_arg(ap, char *);

                        do {
                                if (!(flags & FL_NOASSIGN))
                                        *str++ = (char) ic;
                                if (--width) {
                                        ic = getc(stream);
                                        nrchars++;
                                }
                        } while (width && ic != EOF && (Xtable[ic] ^ reverse));

```

```
				if (width) {
					if (ic != EOF) ungetc(ic, stream);
					nrchars--;
				}
				if (!(flags & FL_NOASSIGN)) {	/* terminate string */
					*str = '\0';
				}
				break;
#ifndef NOFLOAT
			case 'e':
			case 'E':
			case 'f':
			case 'g':
			case 'G':
				if (!(flags & FL_WIDTHSPEC) || width > NUMLEN)
					width = NUMLEN;

				if (!width) return done;
				str = f_collect(ic, stream, width);

				if (str < inp_buf
				    || (str == inp_buf
					&& (*str == '-'
					    || *str == '+'))) return done;

				/*
				 * Although the length of the number is str-inp_buf+1
				 * we don't add the 1 since we counted it already
				 */
				nrchars += str - inp_buf;

				if (!(flags & FL_NOASSIGN)) {
					ld_val = strtod(inp_buf, &tmp_string);
					if (flags & FL_LONGDOUBLE)
						*va_arg(ap, long double *) = (long double) ld_val;
					else
					    if (flags & FL_LONG)
						*va_arg(ap, double *) = (double) ld_val;
					else
						*va_arg(ap, float *) = (float) ld_val;
				}
				break;
#endif
			}		/* end switch */
			conv++;
			if (!(flags & FL_NOASSIGN) && kind != 'n') done++;
			format++;
		}
		return conv || (ic != EOF) ? done : EOF;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/ecvt.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/* $Header: ecvt.c,v 1.4 90/02/27 16:47:28 eck Exp $ */

#ifndef NOFLOAT

#include        "../ansi/ext_fmt.h"
void _dbl_ext_cvt(double value, struct EXTEND *e);
char *_ext_str_cvt(struct EXTEND *e, int ndigit, int *decpt, int * sign, int ecvtflag);

static char *
cvt(long double value, int ndigit, int *decpt, int *sign, int ecvtflag)
{
        struct EXTEND e;

        _dbl_ext_cvt(value, &e);
        return _ext_str_cvt(&e, ndigit, decpt, sign, ecvtflag);
}

char *
_ecvt(long double value, int ndigit, int *decpt, int *sign)
{

        return cvt(value, ndigit, decpt, sign, 1);
}

char *
_fcvt(long double value, int ndigit, int *decpt, int *sign)
{
        return cvt(value, ndigit, decpt, sign, 0);
}

#endif  /* NOFLOAT */
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/fclose.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * fclose.c - flush a stream and close the file
 */
/* $Header: fclose.c,v 1.4 90/01/22 11:10:54 eck Exp $ */

#include        <stdio.h>
#include        <stdlib.h>
#include        "loc_incl.h"

int _close(int d);

int
fclose(FILE *fp)
{
        register int i, retval = 0;
```

```

        for (i=0; i<FOPEN_MAX; i++)
                if (fp == __iotab[i]) {
                        __iotab[i] = 0;
                        break;
                }
        if (i >= FOPEN_MAX)
                return EOF;
        if (fflush(fp)) retval = EOF;
        if (_close(fileno(fp))) retval = EOF;
        if ( io_testflag(fp,_IOMYBUF) && fp->_buf )
                free((void *)fp->_buf);
        if (fp != stdin && fp != stdout && fp != stderr)
                free((void *)fp);
        return retval;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/stdio/feof.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * feof.c - test if eof on a stream occurred
 */
/* $Header: feof.c,v 1.2 89/12/18 15:00:39 eck Exp $ */

#include        <stdio.h>

int
(feof)(FILE *stream)
{
        return feof(stream);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/stdio/ferror.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * ferror .c - test if an error on a stream occurred
 */
/* $Header: ferror.c,v 1.2 89/12/18 15:00:47 eck Exp $ */

#include        <stdio.h>

int
(ferror)(FILE *stream)
{
        return ferror(stream);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/fflush.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * fflush.c - flush stream(s)
 */
/* $Header: fflush.c,v 1.6 90/04/04 15:52:01 eck Exp $ */

#include        <sys/types.h>
#include        <stdio.h>
#include        "loc_incl.h"

ssize_t _write(int d, const char *buf, size_t nbytes);
off_t _lseek(int fildes, off_t offset, int whence);

int
fflush(FILE *stream)
{
        int count, c1, i, retval = 0;

        if (!stream) {
            for(i= 0; i < FOPEN_MAX; i++)
                if (__iotab[i] && fflush(__iotab[i]))
                        retval = EOF;
            return retval;
        }

        if (!stream->_buf
            || (!io_testflag(stream, _IOREADING)
                && !io_testflag(stream, _IOWRITING)))
                return 0;
        if (io_testflag(stream, _IOREADING)) {
                /* (void) fseek(stream, 0L, SEEK_CUR); */
                int adjust = 0;
                if (stream->_buf && !io_testflag(stream,_IONBF))
                        adjust = stream->_count;
                stream->_count = 0;
                _lseek(fileno(stream), (off_t) adjust, SEEK_CUR);
                if (io_testflag(stream, _IOWRITE))
                        stream->_flags &= ~(_IOREADING | _IOWRITING);
                stream->_ptr = stream->_buf;
                return 0;
        } else if (io_testflag(stream, _IONBF)) return 0;

        if (io_testflag(stream, _IOREAD))               /* "a" or "+" mode */
                stream->_flags &= ~_IOWRITING;

        count = stream->_ptr - stream->_buf;
        stream->_ptr = stream->_buf;

        if ( count <= 0 )
                return 0;

        if (io_testflag(stream, _IOAPPEND)) {
                if (_lseek(fileno(stream), 0L, SEEK_END) == -1) {
                        stream->_flags |= _IOERR;
                        return EOF;
                }
```

```
        }
        c1 = _write(stream->_fd, (char *)stream->_buf, count);

        stream->_count = 0;

        if ( count == c1 )
                return 0;

        stream->_flags |= _IOERR;
        return EOF;
}

void
__cleanup(void)
{
        register int i;

        for(i= 0; i < FOPEN_MAX; i++)
                if (__iotab[i] && io_testflag(__iotab[i], _IOWRITING))
                        (void) fflush(__iotab[i]);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/fgetc.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * fgetc - get an unsigned character and return it as an int
 */
/* $Header: fgetc.c,v 1.1 89/05/30 13:27:35 eck Exp $ */

#include        <stdio.h>

int
fgetc(FILE *stream)
{
        return getc(stream);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/fgetpos.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * fgetpos.c - get the position in the file
 */
/* $Header: fgetpos.c,v 1.2 89/12/18 15:01:03 eck Exp $ */

#include        <stdio.h>

int
fgetpos(FILE *stream, fpos_t *pos)
{
```

```
        *pos = ftell(stream);
        if (*pos == -1) return -1;
        return 0;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/stdio/fgets.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * fgets.c - get a string from a file
 */
/* $Header: fgets.c,v 1.3 89/12/18 15:01:11 eck Exp $ */

#include        <stdio.h>

char *
fgets(char *s, register int n, register FILE *stream)
{
        register int ch;
        register char *ptr;

        ptr = s;
        while (--n > 0 && (ch = getc(stream)) != EOF) {
                *ptr++ = ch;
                if ( ch == '\n')
                        break;
        }
        if (ch == EOF) {
                if (feof(stream)) {
                        if (ptr == s) return NULL;
                } else return NULL;
        }
        *ptr = '\0';
        return s;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/stdio/fileno.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * fileno .c - map a stream to a file descriptor
 */
/* $Header: fileno.c,v 1.1 89/12/18 14:59:31 eck Exp $ */

#include        <stdio.h>

int
(fileno)(FILE *stream)
{
        return stream->_fd;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/stdio/fillbuf.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * fillbuf.c - fill a buffer
 */
/* $Header: fillbuf.c,v 1.5 90/06/21 11:13:23 eck Exp $ */

#if     defined(_POSIX_SOURCE)
#include        <sys/types.h>
#endif
#include        <stdio.h>
#include        <stdlib.h>
#include        "loc_incl.h"

ssize_t _read(ssize_t d, char *buf, size_t nbytes);

int
__fillbuf(register FILE *stream)
{
        static unsigned char ch[FOPEN_MAX];
        register int i;

        stream->_count = 0;
        if (fileno(stream) < 0) return EOF;
        if (io_testflag(stream, (_IOEOF | _IOERR ))) return EOF;
        if (!io_testflag(stream, _IOREAD))
             { stream->_flags |= _IOERR; return EOF; }
        if (io_testflag(stream, _IOWRITING))
             { stream->_flags |= _IOERR; return EOF; }

        if (!io_testflag(stream, _IOREADING))
                stream->_flags |= _IOREADING;

        if (!io_testflag(stream, _IONBF) && !stream->_buf) {
                stream->_buf = (unsigned char *) malloc(BUFSIZ);
                if (!stream->_buf) {
                        stream->_flags |= _IONBF;
                }
                else {
                        stream->_flags |= _IOMYBUF;
                        stream->_bufsiz = BUFSIZ;
                }
        }

        /* flush line-buffered output when filling an input buffer */
        for (i = 0; i < FOPEN_MAX; i++) {
                if (__iotab[i] && io_testflag(__iotab[i], _IOLBF))
                        if (io_testflag(__iotab[i], _IOWRITING))
                                (void) fflush(__iotab[i]);
        }

        if (!stream->_buf) {
                stream->_buf = &ch[fileno(stream)];
                stream->_bufsiz = 1;
        }
        stream->_ptr = stream->_buf;
        stream->_count = _read(stream->_fd, (char *)stream->_buf, stream->_bufsiz);
```

```

        if (stream->_count <= 0){
                if (stream->_count == 0) {
                        stream->_flags |= _IOEOF;
                }
                else
                        stream->_flags |= _IOERR;

                return EOF;
        }
        stream->_count--;

        return *stream->_ptr++;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/flushbuf.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * flushbuf.c - flush a buffer
 */
/* $Header: flushbuf.c,v 1.6 91/06/10 17:07:10 ceriel Exp $ */

#include        <stdio.h>
#include        <stdlib.h>
#include        "loc_incl.h"

#include        <sys/types.h>

off_t _lseek(int fildes, off_t offset, int whence);
ssize_t _write(int d, const char *buf, size_t nbytes);
int _isatty(int d);
extern void (*_clean)(void);

static int
do_write(int d, char *buf, int nbytes)
{
        int c;

        /* POSIX actually allows write() to return a positive value less
           than nbytes, so loop ...
        */
        while ((c = _write(d, buf, nbytes)) > 0 && c < nbytes) {
                nbytes -= c;
                buf += c;
        }
        return c > 0;
}

int
__flushbuf(int c, FILE * stream)
{
        _clean = __cleanup;
        if (fileno(stream) < 0) return EOF;
        if (!io_testflag(stream, _IOWRITE)) return EOF;
        if (io_testflag(stream, _IOREADING) && !feof(stream)) return EOF;

        stream->_flags &= ~_IOREADING;
```

```c
	stream->_flags |= _IOWRITING;
	if (!io_testflag(stream, _IONBF)) {
		if (!stream->_buf) {
			if (stream == stdout && _isatty(fileno(stdout))) {
				if (!(stream->_buf =
					    (unsigned char *) malloc(BUFSIZ))) {
					stream->_flags |= _IONBF;
				} else {
					stream->_flags |= _IOLBF|_IOMYBUF;
					stream->_bufsiz = BUFSIZ;
					stream->_count = -1;
				}
			} else {
				if (!(stream->_buf =
					    (unsigned char *) malloc(BUFSIZ))) {
					stream->_flags |= _IONBF;
				} else {
					stream->_flags |= _IOMYBUF;
					stream->_bufsiz = BUFSIZ;
					if (!io_testflag(stream, _IOLBF))
						stream->_count = BUFSIZ - 1;
					else	stream->_count = -1;
				}
			}
			stream->_ptr = stream->_buf;
		}
	}

	if (io_testflag(stream, _IONBF)) {
		char c1 = c;

		stream->_count = 0;
		if (io_testflag(stream, _IOAPPEND)) {
			if (_lseek(fileno(stream), 0L, SEEK_END) == -1) {
				stream->_flags |= _IOERR;
				return EOF;
			}
		}
		if (_write(fileno(stream), &c1, 1) != 1) {
			stream->_flags |= _IOERR;
			return EOF;
		}
		return c;
	} else if (io_testflag(stream, _IOLBF)) {
		*stream->_ptr++ = c;
		if (c == '\n' || stream->_count == -stream->_bufsiz) {
			if (io_testflag(stream, _IOAPPEND)) {
				if (_lseek(fileno(stream), 0L, SEEK_END) == -1) {
					stream->_flags |= _IOERR;
					return EOF;
				}
			}
			if (! do_write(fileno(stream), (char *)stream->_buf,
					-stream->_count)) {
				stream->_flags |= _IOERR;
				return EOF;
			} else {
				stream->_ptr  = stream->_buf;
				stream->_count = 0;
			}
```

```
                }
        } else {
                int count = stream->_ptr - stream->_buf;

                stream->_count = stream->_bufsiz - 1;
                stream->_ptr = stream->_buf + 1;

                if (count > 0) {
                        if (io_testflag(stream, _IOAPPEND)) {
                                if (_lseek(fileno(stream), 0L, SEEK_END) == -1) {
                                        stream->_flags |= _IOERR;
                                        return EOF;
                                }
                        }
                        if (! do_write(fileno(stream), (char *)stream->_buf, count)) {
                                *(stream->_buf) = c;
                                stream->_flags |= _IOERR;
                                return EOF;
                        }
                }
                *(stream->_buf) = c;
        }
        return c;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/stdio/fopen.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * fopen.c - open a stream
 */
/* $Header: fopen.c,v 1.8 91/02/22 16:29:46 ceriel Exp $ */

#if     defined(_POSIX_SOURCE)
#include        <sys/types.h>
#endif
#include        <stdio.h>
#include        <stdlib.h>
#include        "loc_incl.h"

#define PMODE           0666

/* The next 3 defines are true in all UNIX systems known to me.
 */
#define O_RDONLY        0
#define O_WRONLY        1
#define O_RDWR          2

/* Since the O_CREAT flag is not available on all systems, we can't get it
 * from the standard library. Furthermore, even if we know that <fcntl.h>
 * contains such a flag, it's not sure whether it can be used, since we
 * might be cross-compiling for another system, which may use an entirely
 * different value for O_CREAT (or not support such a mode). The safest
 * thing is to just use the Version 7 semantics for open, and use creat()
 * whenever necessary.
 *
 * Another problem is O_APPEND, for which the same holds. When "a"
 * open-mode is used, an lseek() to the end is done before every write()
```

```
 * system-call.
 *
 * The O_CREAT, O_TRUNC and O_APPEND given here, are only for convenience.
 * They are not passed to open(), so the values don't have to match a value
 * from the real world. It is enough when they are unique.
 */
#define O_CREAT         0x010
#define O_TRUNC         0x020
#define O_APPEND        0x040

int _open(const char *path, int flags);
int _creat(const char *path, Mode_t mode);
int _close(int d);

FILE *
fopen(const char *name, const char *mode)
{
        register int i;
        int rwmode = 0, rwflags = 0;
        FILE *stream;
        int fd, flags = 0;

        for (i = 0; __iotab[i] != 0 ; i++)
                if ( i >= FOPEN_MAX-1 )
                        return (FILE *)NULL;

        switch(*mode++) {
        case 'r':
                flags |= _IOREAD | _IOREADING;
                rwmode = O_RDONLY;
                break;
        case 'w':
                flags |= _IOWRITE | _IOWRITING;
                rwmode = O_WRONLY;
                rwflags = O_CREAT | O_TRUNC;
                break;
        case 'a':
                flags |= _IOWRITE | _IOWRITING | _IOAPPEND;
                rwmode = O_WRONLY;
                rwflags |= O_APPEND | O_CREAT;
                break;
        default:
                return (FILE *)NULL;
        }

        while (*mode) {
                switch(*mode++) {
                case 'b':
                        continue;
                case '+':
                        rwmode = O_RDWR;
                        flags |= _IOREAD | _IOWRITE;
                        continue;
                /* The sequence may be followed by additional characters */
                default:
                        break;
                }
                break;
        }

```

```
        /* Perform a creat() when the file should be truncated or when
         * the file is opened for writing and the open() failed.
         */
        if ((rwflags & O_TRUNC)
            || (((fd = _open(name, rwmode)) < 0)
                    && (rwflags & O_CREAT))) {
                if (((fd = _creat(name, PMODE)) > 0) && flags  | _IOREAD) {
                        (void) _close(fd);
                        fd = _open(name, rwmode);
                }

        }

        if (fd < 0) return (FILE *)NULL;

        if (( stream = (FILE *) malloc(sizeof(FILE))) == NULL ) {
                _close(fd);
                return (FILE *)NULL;
        }

        if ((flags & (_IOREAD | _IOWRITE))  == (_IOREAD | _IOWRITE))
                flags &= ~(_IOREADING | _IOWRITING);

        stream->_count = 0;
        stream->_fd = fd;
        stream->_flags = flags;
        stream->_buf = NULL;
        __iotab[i] = stream;
        return stream;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/fprintf.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * fprintf - write output on a stream
 */
/* $Header: fprintf.c,v 1.3 89/12/18 15:01:54 eck Exp $ */

#include        <stdio.h>
#include        <stdarg.h>
#include        "loc_incl.h"

int
fprintf(FILE *stream, const char *format, ...)
{
        va_list ap;
        int retval;

        va_start(ap, format);

        retval = _doprnt (format, ap, stream);

        va_end(ap);
```

```
        return retval;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/stdio/fputc.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * fputc.c - print an unsigned character
 */
/* $Header: fputc.c,v 1.1 89/05/30 13:28:45 eck Exp $ */

#include        <stdio.h>

int
fputc(int c, FILE *stream)
{
        return putc(c, stream);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/stdio/fputs.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * fputs - print a string
 */
/* $Header: fputs.c,v 1.2 89/12/18 15:02:01 eck Exp $ */

#include        <stdio.h>

int
fputs(register const char *s, register FILE *stream)
{
        register int i = 0;

        while (*s)
                if (putc(*s++, stream) == EOF) return EOF;
                else i++;

        return i;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/stdio/fread.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * fread.c - read a number of members into an array
 */
/* $Header: fread.c,v 1.2 89/12/18 15:02:09 eck Exp $ */

#include        <stdio.h>

size_t
fread(void *ptr, size_t size, size_t nmemb, register FILE *stream)
{
        register char *cp = ptr;
        register int c;
        size_t ndone = 0;
        register size_t s;

        if (size)
                while ( ndone < nmemb ) {
                        s = size;
                        do {
                                if ((c = getc(stream)) != EOF)
                                        *cp++ = c;
                                else
                                        return ndone;
                        } while (--s);
                        ndone++;
                }

        return ndone;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/stdio/freopen.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * freopen.c - open a file and associate a stream with it
 */
/* $Header: freopen.c,v 1.8 90/08/28 13:44:48 eck Exp $ */

#if     defined(_POSIX_SOURCE)
#include        <sys/types.h>
#endif
#include        <stdio.h>
#include        <stdlib.h>
#include        "loc_incl.h"

#define PMODE           0666

/* Do not "optimize" this file to use the open with O_CREAT if the file
 * does not exist. The reason is given in fopen.c.
 */
#define O_RDONLY        0
#define O_WRONLY        1
#define O_RDWR          2
```

```c

#define O_CREAT         0x010
#define O_TRUNC         0x020
#define O_APPEND        0x040

int _open(const char *path, int flags);
int _creat(const char *path, Mode_t mode);
int _close(int d);

FILE *
freopen(const char *name, const char *mode, FILE *stream)
{
        register int i;
        int rwmode = 0, rwflags = 0;
        int fd, flags = stream->_flags & (_IONBF | _IOFBF | _IOLBF | _IOMYBUF);

        (void) fflush(stream);                          /* ignore errors */
        (void) _close(fileno(stream));

        switch(*mode++) {
        case 'r':
                flags |= _IOREAD;
                rwmode = O_RDONLY;
                break;
        case 'w':
                flags |= _IOWRITE;
                rwmode = O_WRONLY;
                rwflags = O_CREAT | O_TRUNC;
                break;
        case 'a':
                flags |= _IOWRITE | _IOAPPEND;
                rwmode = O_WRONLY;
                rwflags |= O_APPEND | O_CREAT;
                break;
        default:
                return (FILE *)NULL;
        }

        while (*mode) {
                switch(*mode++) {
                case 'b':
                        continue;
                case '+':
                        rwmode = O_RDWR;
                        flags |= _IOREAD | _IOWRITE;
                        continue;
                /* The sequence may be followed by aditional characters */
                default:
                        break;
                }
                break;
        }

        if ((rwflags & O_TRUNC)
            || (((fd = _open(name, rwmode)) < 0)
                    && (rwflags & O_CREAT))) {
                if (((fd = _creat(name, PMODE)) < 0) && flags | _IOREAD) {
                        (void) _close(fd);
                        fd = _open(name, rwmode);
                }
```

```
        }

        if (fd < 0) {
                for( i = 0; i < FOPEN_MAX; i++) {
                        if (stream == __iotab[i]) {
                                __iotab[i] = 0;
                                break;
                        }
                }
                if (stream != stdin && stream != stdout && stream != stderr)
                        free((void *)stream);
                return (FILE *)NULL;
        }

        stream->_count = 0;
        stream->_fd = fd;
        stream->_flags = flags;
        return stream;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/stdio/fscanf.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * fscanf.c - read formatted input from stream
 */
/* $Header: fscanf.c,v 1.1 89/05/30 13:29:17 eck Exp $ */

#include        <stdio.h>
#include        <stdarg.h>
#include        "loc_incl.h"

int
fscanf(FILE *stream, const char *format, ...)
{
        va_list ap;
        int retval;

        va_start(ap, format);

        retval = _doscan(stream, format, ap);

        va_end(ap);

        return retval;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/fseek.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * fseek.c - perform an fseek
 */
/* $Header: fseek.c,v 1.4 90/01/22 11:12:00 eck Exp $ */

#include        <stdio.h>

#if     (SEEK_CUR != 1) || (SEEK_END != 2) || (SEEK_SET != 0)
#error SEEK_* values are wrong
#endif

#include        "loc_incl.h"

#include        <sys/types.h>

off_t _lseek(int fildes, off_t offset, int whence);

int
fseek(FILE *stream, long int offset, int whence)
{
        int adjust = 0;
        long pos;

        stream->_flags &= ~(_IOEOF | _IOERR);
        /* Clear both the end of file and error flags */

        if (io_testflag(stream, _IOREADING)) {
                if (whence == SEEK_CUR
                    && stream->_buf
                    && !io_testflag(stream,_IONBF))
                        adjust = stream->_count;
                stream->_count = 0;
        } else if (io_testflag(stream,_IOWRITING)) {
                fflush(stream);
        } else  /* neither reading nor writing. The buffer must be empty */
                /* EMPTY */ ;

        pos = _lseek(fileno(stream), offset - adjust, whence);
        if (io_testflag(stream, _IOREAD) && io_testflag(stream, _IOWRITE))
                stream->_flags &= ~(_IOREADING | _IOWRITING);

        stream->_ptr = stream->_buf;
        return ((pos == -1) ? -1 : 0);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/fsetpos.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * fsetpos.c - set the position in the file
 */
/* $Header: fsetpos.c,v 1.1 89/05/30 13:29:34 eck Exp $ */

#include        <stdio.h>

int
fsetpos(FILE *stream, fpos_t *pos)
{
        return fseek(stream, *pos, SEEK_SET);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/ftell.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * ftell.c - obtain the value of the file-position indicator of a stream
 */
/* $Header: ftell.c,v 1.4 90/01/22 11:12:12 eck Exp $ */

#include        <stdio.h>

#if     (SEEK_CUR != 1) || (SEEK_SET != 0) || (SEEK_END != 2)
#error SEEK_* values are wrong
#endif

#include        "loc_incl.h"

#include        <sys/types.h>

off_t _lseek(int fildes, off_t offset, int whence);

long ftell(FILE *stream)
{
        long result;
        int adjust = 0;

        if (io_testflag(stream,_IOREADING))
                adjust = -stream->_count;
        else if (io_testflag(stream,_IOWRITING)
                    && stream->_buf
                    && !io_testflag(stream,_IONBF))
                adjust = stream->_ptr - stream->_buf;
        else adjust = 0;

        result = _lseek(fileno(stream), (off_t)0, SEEK_CUR);

        if ( result == -1 )
                return result;

```

```
        result += (long) adjust;
        return result;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/fwrite.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * fwrite.c - write a number of array elements on a file
 */
/* $Header: fwrite.c,v 1.3 89/12/18 15:02:39 eck Exp $ */

#include        <stdio.h>

size_t
fwrite(const void *ptr, size_t size, size_t nmemb,
            register FILE *stream)
{
        register const unsigned char *cp = ptr;
        register size_t s;
        size_t ndone = 0;

        if (size)
                while ( ndone < nmemb ) {
                        s = size;
                        do {
                                if (putc((int)*cp, stream)
                                        == EOF)
                                        return ndone;
                                cp++;
                        }
                        while (--s);
                        ndone++;
                }
        return ndone;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/getc.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * getc.c - read an unsigned character
 */
/* $Header: getc.c,v 1.2 89/12/18 15:02:45 eck Exp $ */

#include        <stdio.h>

int
(getc)(FILE *stream)
{
        return getc(stream);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/stdio/getchar.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * getchar.c - read a character from the standard input stream
 */
/* $Header: getchar.c,v 1.2 89/12/18 15:02:53 eck Exp $ */

#include        <stdio.h>

int
(getchar)(void)
{
        return getchar();
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/stdio/gets.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * gets.c - read a line from a stream
 */
/* $Header: gets.c,v 1.2 89/12/18 15:03:00 eck Exp $ */

#include        <stdio.h>

char *
gets(char *s)
{
        register FILE *stream = stdin;
        register int ch;
        register char *ptr;

        ptr = s;
        while ((ch = getc(stream)) != EOF && ch != '\n')
                *ptr++ = ch;

        if (ch == EOF) {
                if (feof(stream)) {
                        if (ptr == s) return NULL;
                } else return NULL;
        }

        *ptr = '\0';
        return s;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/stdio/icompute.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * icompute.c - compute an integer
 */
/* $Header: icompute.c,v 1.1 89/12/18 14:59:38 eck Exp $ */

#include        "loc_incl.h"

/* This routine is used in doprnt.c as well as in tmpfile.c and tmpnam.c. */

char *
_i_compute(unsigned long val, int base, char *s, int nrdigits)
{
        int c;

        c= val % base ;
        val /= base ;
        if (val || nrdigits > 1)
                s = _i_compute(val, base, s, nrdigits - 1);
        *s++ = (c>9 ? c-10+'a' : c+'0');
        return s;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/stdio/perror.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * perror.c - print an error message on the standard error output
 */
/* $Header: perror.c,v 1.1 89/05/30 13:31:30 eck Exp $ */

#if     defined(_POSIX_SOURCE)
#include        <sys/types.h>
#endif
#include        <stdio.h>
#include        <errno.h>
#include        <stdio.h>
#include        <string.h>
#include        "loc_incl.h"

ssize_t _write(int d, const char *buf, size_t nbytes);

void
perror(const char *s)
{
        char *p;
        int fd;

        p = strerror(errno);
        fd = fileno(stderr);
        fflush(stdout);
```

```
        fflush(stderr);
        if (s && *s) {
                _write(fd, s, strlen(s));
                _write(fd, ": ", 2);
        }
        _write(fd, p, strlen(p));
        _write(fd, "\n", 1);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/stdio/printf.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * printf - write on the standard output stream
 */
/* $Header: printf.c,v 1.3 89/12/18 15:03:08 eck Exp $ */

#include        <stdio.h>
#include        <stdarg.h>
#include        "loc_incl.h"

int
printf(const char *format, ...)
{
        va_list ap;
        int retval;

        va_start(ap, format);

        retval = _doprnt(format, ap, stdout);

        va_end(ap);

        return retval;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/stdio/putc.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * putc.c - print (or buffer) one character
 */
/* $Header: putc.c,v 1.2 89/12/18 15:03:15 eck Exp $ */

#include        <stdio.h>

int
(putc)(int c, FILE *stream)
{
        return putc(c, stream);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/putchar.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * putchar.c - print (or buffer) a character on the standard output stream
 */
/* $Header: putchar.c,v 1.2 89/12/18 15:03:23 eck Exp $ */

#include        <stdio.h>

int
(putchar)(int c)
{
        return putchar(c);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/puts.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * puts.c - print a string onto the standard output stream
 */
/* $Header: puts.c,v 1.2 89/12/18 15:03:30 eck Exp $ */

#include        <stdio.h>

int
puts(register const char *s)
{
        register FILE *file = stdout;
        register int i = 0;

        while (*s) {
                if (putc(*s++, file) == EOF) return EOF;
                else i++;
        }
        if (putc('\n', file) == EOF) return EOF;
        return i + 1;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/remove.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * remove.c - remove a file
 */
/* $Header: remove.c,v 1.2 90/01/22 11:12:44 eck Exp $ */

#include        <stdio.h>

int _unlink(const char *path);

int
remove(const char *filename) {
        return _unlink(filename);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/rewind.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * rewind.c - set the file position indicator of a stream to the start
 */
/* $Header: rewind.c,v 1.1 89/05/30 13:32:52 eck Exp $ */

#include        <stdio.h>
#include        "loc_incl.h"

void
rewind(FILE *stream)
{
        (void) fseek(stream, 0L, SEEK_SET);
        clearerr(stream);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/scanf.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * scanf.c - read formatted input from the standard input stream
 */
/* $Header: scanf.c,v 1.1 89/05/30 13:33:03 eck Exp $ */

#include        <stdio.h>
#include        <stdarg.h>
#include        "loc_incl.h"

int
scanf(const char *format, ...)
{
        va_list ap;
        int retval;

```

```
        va_start(ap, format);

        retval = _doscan(stdin, format, ap);

        va_end(ap);

        return retval;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/setbuf.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * setbuf.c - control buffering of a stream
 */
/* $Header: setbuf.c,v 1.2 89/06/26 10:36:22 eck Exp $ */

#include        <stdio.h>
#include        "loc_incl.h"

void
setbuf(register FILE *stream, char *buf)
{
        (void) setvbuf(stream, buf, (buf ? _IOFBF : _IONBF), (size_t) BUFSIZ);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/setvbuf.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * setbuf.c - control buffering of a stream
 */
/* $Id: setvbuf.c,v 1.8 1995/12/18 11:02:18 ceriel Exp $ */

#include        <stdio.h>
#include        <stdlib.h>
#include        "loc_incl.h"

extern void (*_clean)(void);

int
setvbuf(register FILE *stream, char *buf, int mode, size_t size)
{
        int retval = 0;

        _clean = __cleanup;
        if (mode != _IOFBF && mode != _IOLBF && mode != _IONBF)
                return EOF;

```

```
        if (stream->_buf && io_testflag(stream,_IOMYBUF) )
                free((void *)stream->_buf);

        stream->_flags &= ~(_IOMYBUF | _IONBF | _IOLBF);

        if (buf && size <= 0) retval = EOF;
        if (!buf && (mode != _IONBF)) {
                if (size <= 0 || (buf = (char *) malloc(size)) == NULL) {
                        retval = EOF;
                } else {
                        stream->_flags |= _IOMYBUF;
                }
        }

        stream->_buf = (unsigned char *) buf;

        stream->_count = 0;
        stream->_flags |= mode;
        stream->_ptr = stream->_buf;

        if (!buf) {
                stream->_bufsiz = 1;
        } else {
                stream->_bufsiz = size;
        }

        return retval;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/stdio/sprintf.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * sprintf - print formatted output on an array
 */
/* $Header: sprintf.c,v 1.2 89/12/18 15:03:52 eck Exp $ */

#include        <stdio.h>
#include        <stdarg.h>
#include        "loc_incl.h"

int
sprintf(char * s, const char *format, ...)
{
        va_list ap;
        int retval;
        FILE tmp_stream;

        va_start(ap, format);

        tmp_stream._fd     = -1;
        tmp_stream._flags  = _IOWRITE + _IONBF + _IOWRITING;
        tmp_stream._buf    = (unsigned char *) s;
        tmp_stream._ptr    = (unsigned char *) s;
        tmp_stream._count  = 32767;

        retval = _doprnt(format, ap, &tmp_stream);
```

```
        putc('\0',&tmp_stream);

        va_end(ap);

        return retval;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/stdio/sscanf.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * sscanf - read formatted output from a string
 */
/* $Header: sscanf.c,v 1.3 90/09/26 13:17:39 eck Exp $ */

#include        <stdio.h>
#include        <stdarg.h>
#include        <string.h>
#include        "loc_incl.h"

int sscanf(const char *s, const char *format, ...)
{
        va_list ap;
        int retval;
        FILE tmp_stream;

        va_start(ap, format);

        tmp_stream._fd     = -1;
        tmp_stream._flags  = _IOREAD + _IONBF + _IOREADING;
        tmp_stream._buf    = (unsigned char *) s;
        tmp_stream._ptr    = (unsigned char *) s;
        tmp_stream._count  = strlen(s);

        retval = _doscan(&tmp_stream, format, ap);

        va_end(ap);

        return retval;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/tmpfile.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * tmpfile.c - create and open a temporary file
 */
/* $Header: tmpfile.c,v 1.3 90/01/22 11:13:15 eck Exp $ */

#if     defined(_POSIX_SOURCE)
#include        <sys/types.h>
#endif
#include        <stdio.h>
#include        <string.h>
#include        "loc_incl.h"

pid_t _getpid(void);

FILE *
tmpfile(void) {
        static char name_buffer[L_tmpnam] = "/tmp/tmp." ;
        static char *name = NULL;
        FILE *file;

        if (!name) {
                name = name_buffer + strlen(name_buffer);
                name = _i_compute(_getpid(), 10, name, 5);
                *name = '\0';
        }

        file = fopen(name_buffer,"wb+");
        if (!file) return (FILE *)NULL;
        (void) remove(name_buffer);
        return file;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/tmpnam.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * tmpnam.c - create a unique filename
 */
/* $Header: tmpnam.c,v 1.4 91/02/26 09:28:39 ceriel Exp $ */

#if     defined(_POSIX_SOURCE)
#include        <sys/types.h>
#endif
#include        <stdio.h>
#include        <string.h>
#include        "loc_incl.h"

pid_t _getpid(void);

char *
```

```
tmpnam(char *s) {
        static char name_buffer[L_tmpnam] = "/tmp/tmp.";
        static unsigned long count = 0;
        static char *name = NULL;

        if (!name) {
                name = name_buffer + strlen(name_buffer);
                name = _i_compute((unsigned long)_getpid(), 10, name, 5);
                *name++ = '.';
                *name = '\0';
        }
        if (++count > TMP_MAX) count = 1;       /* wrap-around */
        *_i_compute(count, 10, name, 3) = '\0';
        if (s) return strcpy(s, name_buffer);
        else return name_buffer;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/ungetc.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*
 * ungetc.c - push a character back onto an input stream
 */
/* $Header: ungetc.c,v 1.3 90/03/28 16:33:05 eck Exp $ */

#include        <stdio.h>
#include        "loc_incl.h"

int
ungetc(int ch, FILE *stream)
{
        unsigned char *p;

        if (ch == EOF  || !io_testflag(stream,_IOREADING))
                return EOF;
        if (stream->_ptr == stream->_buf) {
                if (stream->_count != 0) return EOF;
                stream->_ptr++;
        }
        stream->_count++;
        p = --(stream->_ptr);           /* ??? Bloody vax assembler !!! */
        /* ungetc() in sscanf() shouldn't write in rom */
        if (*p != (unsigned char) ch)
                *p = (unsigned char) ch;
        return ch;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/stdio/vfprintf.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * vfprintf - formatted output without ellipsis
 */
/* $Header: vfprintf.c,v 1.2 89/12/18 15:04:07 eck Exp $ */

#include        <stdio.h>
#include        <stdarg.h>
#include        "loc_incl.h"

int
vfprintf(FILE *stream, const char *format, va_list arg)
{
        return _doprnt (format, arg, stream);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/stdio/vprintf.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * vprintf - formatted output without ellipsis to the standard output stream
 */
/* $Header: vprintf.c,v 1.3 89/12/18 15:04:14 eck Exp $ */

#include        <stdio.h>
#include        <stdarg.h>
#include        "loc_incl.h"

int
vprintf(const char *format, va_list arg)
{
        return _doprnt(format, arg, stdout);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/stdio/vsprintf.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
/*
 * vsprintf - print formatted output without ellipsis on an array
 */
/* $Header: vsprintf.c,v 1.2 90/11/13 10:56:53 eck Exp $ */

#include        <stdio.h>
#include        <stdarg.h>
#include        "loc_incl.h"

int
vsprintf(char *s, const char *format, va_list arg)
{
        int retval;
        FILE tmp_stream;

```

```
        tmp_stream._fd     = -1;
        tmp_stream._flags  = _IOWRITE + _IONBF + _IOWRITING;
        tmp_stream._buf    = (unsigned char *) s;
        tmp_stream._ptr    = (unsigned char *) s;
        tmp_stream._count  = 32767;

        retval = _doprnt(format, arg, &tmp_stream);
        putc('\0',&tmp_stream);

        return retval;
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/vscanf.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * vscanf.c - read formatted input from the standard input stream
 */

#include        <stdio.h>
#include        <stdarg.h>
#include        "loc_incl.h"

int
vscanf(const char *format, va_list ap)
{
        return _doscan(stdin, format, ap);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/stdio/vsscanf.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

/*
 * vsscanf - read formatted output from a string
 */

#include        <stdio.h>
#include        <stdarg.h>
#include        <string.h>
#include        "loc_incl.h"

int vsscanf(const char *s, const char *format, va_list ap)
{
        FILE tmp_stream;

        tmp_stream._fd     = -1;
        tmp_stream._flags  = _IOREAD + _IONBF + _IOREADING;
        tmp_stream._buf    = (unsigned char *) s;
        tmp_stream._ptr    = (unsigned char *) s;
        tmp_stream._count  = strlen(s);

        return _doscan(&tmp_stream, format, ap);
```

```
}
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/_exit.s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern ___exit
.define __exit
.align 2

__exit:
        jmp     ___exit
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/access.s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __access
.define _access
.align 2

_access:
        jmp     __access
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/alarm.s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __alarm
.define _alarm
.align 2

_alarm:
        jmp     __alarm
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/syscall/brk.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __brk
.define _brk
.align 2

_brk:
        jmp     __brk
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/syscall/cfgetispeed.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __cfgetispeed
.define _cfgetispeed
.align 2

_cfgetispeed:
        jmp     __cfgetispeed
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/syscall/cfgetospeed.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __cfgetospeed
.define _cfgetospeed
.align 2

_cfgetospeed:
        jmp     __cfgetospeed
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/syscall/cfsetispeed.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __cfsetispeed
.define _cfsetispeed
.align 2

_cfsetispeed:
        jmp     __cfsetispeed
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/cfsetospeed.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
.sect .text
.extern __cfsetospeed
.define _cfsetospeed
.align 2

_cfsetospeed:
        jmp     __cfsetospeed
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/chdir.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
.sect .text
.extern __chdir
.define _chdir
.align 2

_chdir:
        jmp     __chdir
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/chmod.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
.sect .text
.extern __chmod
.define _chmod
.align 2

_chmod:
        jmp     __chmod
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/chown.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
.sect .text
.extern __chown
.define _chown
.align 2

_chown:
        jmp     __chown
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/chroot.s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __chroot
.define _chroot
.align 2

_chroot:
        jmp     __chroot
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/close.s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __close
.define _close
.align 2

_close:
        jmp     __close
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/closedir.s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __closedir
.define _closedir
.align 2

_closedir:
        jmp     __closedir
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/creat.s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __creat
.define _creat
.align 2

_creat:
        jmp     __creat
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/dup.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __dup
.define _dup
.align 2

_dup:
        jmp     __dup
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/dup2.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __dup2
.define _dup2
.align 2

_dup2:
        jmp     __dup2
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/execl.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __execl
.define _execl
.align 2

_execl:
        jmp     __execl
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/execle.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __execle
.define _execle
.align 2

_execle:
        jmp     __execle
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/execv.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __execv
.define _execv
.align 2

_execv:
        jmp     __execv
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/execve.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __execve
.define _execve
.align 2

_execve:
        jmp     __execve
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/fcntl.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __fcntl
.define _fcntl
.align 2

_fcntl:
        jmp     __fcntl
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/fork.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __fork
.define _fork
.align 2

_fork:
        jmp     __fork
```

src/lib/syscall/fpathconf.s

```
.sect .text
.extern __fpathconf
.define _fpathconf
.align 2

_fpathconf:
        jmp     __fpathconf
```

src/lib/syscall/fstat.s

```
.sect .text
.extern __fstat
.define _fstat
.align 2

_fstat:
        jmp     __fstat
```

src/lib/syscall/getcwd.s

```
.sect .text
.extern __getcwd
.define _getcwd
.align 2

_getcwd:
        jmp     __getcwd
```

src/lib/syscall/getegid.s

```
.sect .text
.extern __getegid
.define _getegid
.align 2

_getegid:
        jmp     __getegid
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/geteuid.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __geteuid
.define _geteuid
.align 2

_geteuid:
        jmp     __geteuid
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/getgid.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __getgid
.define _getgid
.align 2

_getgid:
        jmp     __getgid
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/getgroups.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __getgroups
.define _getgroups
.align 2

_getgroups:
        jmp     __getgroups
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/getpgrp.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __getpgrp
.define _getpgrp
.align 2

_getpgrp:
        jmp     __getpgrp
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/getpid.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __getpid
.define _getpid
.align 2

_getpid:
        jmp     __getpid
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/getppid.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __getppid
.define _getppid
.align 2

_getppid:
        jmp     __getppid
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/getuid.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __getuid
.define _getuid
.align 2

_getuid:
        jmp     __getuid
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/ioctl.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __ioctl
.define _ioctl
.align 2

_ioctl:
        jmp     __ioctl
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/isatty.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __isatty
.define _isatty
.align 2

_isatty:
        jmp     __isatty
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/kill.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __kill
.define _kill
.align 2

_kill:
        jmp     __kill
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/link.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __link
.define _link
.align 2

_link:
        jmp     __link
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/lseek.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __lseek
.define _lseek
.align 2

_lseek:
        jmp     __lseek
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/mkdir.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __mkdir
.define _mkdir
.align 2

_mkdir:
        jmp     __mkdir
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/mkfifo.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __mkfifo
.define _mkfifo
.align 2

_mkfifo:
        jmp     __mkfifo
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/mknod.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __mknod
.define _mknod
.align 2

_mknod:
        jmp     __mknod
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/mktemp.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __mktemp
.define _mktemp
.align 2

_mktemp:
        jmp     __mktemp
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/syscall/mount.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __mount
.define _mount
.align 2

_mount:
        jmp     __mount
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/syscall/open.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __open
.define _open
.align 2

_open:
        jmp     __open
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/syscall/opendir.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __opendir
.define _opendir
.align 2

_opendir:
        jmp     __opendir
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/syscall/pathconf.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __pathconf
.define _pathconf
.align 2

_pathconf:
        jmp     __pathconf
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/pause.s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __pause
.define _pause
.align 2

_pause:
        jmp     __pause
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/pipe.s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __pipe
.define _pipe
.align 2

_pipe:
        jmp     __pipe
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/ptrace.s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __ptrace
.define _ptrace
.align 2

_ptrace:
        jmp     __ptrace
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/read.s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __read
.define _read
.align 2

_read:
        jmp     __read
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/syscall/readdir.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __readdir
.define _readdir
.align 2

_readdir:
        jmp     __readdir
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/syscall/reboot.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __reboot
.define _reboot
.align 2

_reboot:
        jmp     __reboot
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/syscall/rename.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __rename
.define _rename
.align 2

_rename:
        jmp     __rename
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/syscall/rewinddir.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __rewinddir
.define _rewinddir
.align 2

_rewinddir:
        jmp     __rewinddir
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/rmdir.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __rmdir
.define _rmdir
.align 2

_rmdir:
        jmp     __rmdir
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/sbrk.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __sbrk
.define _sbrk
.align 2

_sbrk:
        jmp     __sbrk
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/seekdir.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __seekdir
.define _seekdir
.align 2

_seekdir:
        jmp     __seekdir
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/setgid.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __setgid
.define _setgid
.align 2

_setgid:
        jmp     __setgid
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/setsid.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __setsid
.define _setsid
.align 2

_setsid:
        jmp     __setsid
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/setuid.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __setuid
.define _setuid
.align 2

_setuid:
        jmp     __setuid
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/sigaction.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __sigaction
.define _sigaction
.align 2

_sigaction:
        jmp     __sigaction
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/sigaddset.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __sigaddset
.define _sigaddset
.align 2

_sigaddset:
        jmp     __sigaddset
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/sigdelset.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __sigdelset
.define _sigdelset
.align 2

_sigdelset:
        jmp     __sigdelset
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/sigemptyset.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __sigemptyset
.define _sigemptyset
.align 2

_sigemptyset:
        jmp     __sigemptyset
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/sigfillset.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __sigfillset
.define _sigfillset
.align 2

_sigfillset:
        jmp     __sigfillset
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/sigismember.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __sigismember
.define _sigismember
.align 2

_sigismember:
        jmp     __sigismember
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/syscall/sigpending.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __sigpending
.define _sigpending
.align 2

_sigpending:
        jmp     __sigpending
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/syscall/sigprocmask.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __sigprocmask
.define _sigprocmask
.align 2

_sigprocmask:
        jmp     __sigprocmask
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/syscall/sigreturn.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __sigreturn
.define _sigreturn
.align 2

_sigreturn:
        jmp     __sigreturn
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/syscall/sigsuspend.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __sigsuspend
.define _sigsuspend
.align 2

_sigsuspend:
        jmp     __sigsuspend
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/syscall/sleep.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __sleep
.define _sleep
.align 2

_sleep:
        jmp     __sleep
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/syscall/stat.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __stat
.define _stat
.align 2

_stat:
        jmp     __stat
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/syscall/stime.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __stime
.define _stime
.align 2

_stime:
        jmp     __stime
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        src/lib/syscall/sync.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __sync
.define _sync
.align 2

_sync:
        jmp     __sync
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/tcdrain.s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
.sect .text
.extern __tcdrain
.define _tcdrain
.align 2

_tcdrain:
        jmp     __tcdrain
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/tcflow.s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
.sect .text
.extern __tcflow
.define _tcflow
.align 2

_tcflow:
        jmp     __tcflow
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/tcflush.s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
.sect .text
.extern __tcflush
.define _tcflush
.align 2

_tcflush:
        jmp     __tcflush
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/tcgetattr.s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
.sect .text
.extern __tcgetattr
.define _tcgetattr
.align 2

_tcgetattr:
        jmp     __tcgetattr
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/tcsendbreak.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __tcsendbreak
.define _tcsendbreak
.align 2

_tcsendbreak:
        jmp     __tcsendbreak
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/tcsetattr.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __tcsetattr
.define _tcsetattr
.align 2

_tcsetattr:
        jmp     __tcsetattr
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/time.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __time
.define _time
.align 2

_time:
        jmp     __time
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/times.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __times
.define _times
.align 2

_times:
        jmp     __times
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/syscall/umask.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __umask
.define _umask
.align 2

_umask:
        jmp     __umask
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/syscall/umount.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __umount
.define _umount
.align 2

_umount:
        jmp     __umount
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/syscall/uname.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __uname
.define _uname
.align 2

_uname:
        jmp     __uname
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/syscall/unlink.s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __unlink
.define _unlink
.align 2

_unlink:
        jmp     __unlink
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/utime.s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __utime
.define _utime
.align 2

_utime:
        jmp     __utime
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/wait.s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __wait
.define _wait
.align 2

_wait:
        jmp     __wait
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/waitpid.s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __waitpid
.define _waitpid
.align 2

_waitpid:
        jmp     __waitpid
```

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syscall/write.s
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

.sect .text
.extern __write
.define _write
.align 2

_write:
        jmp     __write
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/syslib/syslib.h
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/*      syslib.h - System library common definitions.   */

#define _SYSTEM

#include <lib.h>
#include <minix/com.h>
#include <minix/syslib.h>
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/syslib/sys_abort.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include "syslib.h"
#include <stdarg.h>
#include <unistd.h>

PUBLIC int sys_abort(int how, ...)
{
/* Something awful has happened.  Abandon ship. */

  message m;
  va_list ap;

  va_start(ap, how);
  if ((m.m1_i1 = how) == RBT_MONITOR) m.m1_p1 = va_arg(ap, char *);
  va_end(ap);

  return(_taskcall(SYSTASK, SYS_ABORT, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                          src/lib/syslib/sys_copy.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include "syslib.h"

PUBLIC int sys_copy(src_proc, src_seg, src_vir,
                                        dst_proc, dst_seg, dst_vir, bytes)
int src_proc;                   /* source process */
int src_seg;                    /* source segment: T, D, or S */
phys_bytes src_vir;             /* source virtual address (phys addr for ABS)*/
int dst_proc;                   /* dest process */
int dst_seg;                    /* dest segment: T, D, or S */
phys_bytes dst_vir;             /* dest virtual address (phys addr for ABS) */
phys_bytes bytes;               /* how many bytes */
{
/* Transfer a block of data.  The source and destination can each either be a
 * process (including MM) or absolute memory, indicate by setting 'src_proc'
 * or 'dst_proc' to ABS.
```

```
 */

  message copy_mess;

  if (bytes == 0L) return(OK);
  copy_mess.SRC_SPACE = src_seg;
  copy_mess.SRC_PROC_NR = src_proc;
  copy_mess.SRC_BUFFER = (long) src_vir;

  copy_mess.DST_SPACE = dst_seg;
  copy_mess.DST_PROC_NR = dst_proc;
  copy_mess.DST_BUFFER = (long) dst_vir;

  copy_mess.COPY_BYTES = (long) bytes;
  return(_taskcall(SYSTASK, SYS_COPY, &copy_mess));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syslib/sys_endsig.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include "syslib.h"

PUBLIC int sys_endsig(proc)
int proc;
{
  message m;

  m.m1_i1 = proc;
  return(_taskcall(SYSTASK, SYS_ENDSIG, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syslib/sys_exec.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include "syslib.h"

PUBLIC int sys_exec(proc, ptr, traced, prog_name, initpc)
int proc;                       /* process that did exec */
char *ptr;                      /* new stack pointer */
int traced;                     /* is tracing enabled? */
char *prog_name;                /* name of the new program */
vir_bytes initpc;
{
/* A process has exec'd.  Tell the kernel. */

  message m;

  m.m1_i1 = proc;
  m.m1_i2 = traced;
```

```
  m.m1_p1 = ptr;
  m.m1_p2 = prog_name;
  m.m1_p3 = (char *)initpc;
  return(_taskcall(SYSTASK, SYS_EXEC, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syslib/sys_fork.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include "syslib.h"

PUBLIC int sys_fork(parent, child, pid, child_base_or_shadow)
int parent;                     /* process doing the fork */
int child;                      /* which proc has been created by the fork */
int pid;                        /* process id assigned by MM */
phys_clicks child_base_or_shadow;       /* position for child [VM386];
                                 * memory allocated for shadow [68000] */
{
/* A process has forked.  Tell the kernel. */

  message m;

  m.m1_i1 = parent;
  m.m1_i2 = child;
  m.m1_i3 = pid;
  m.m1_p1 = (char *) child_base_or_shadow;
  return(_taskcall(SYSTASK, SYS_FORK, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syslib/sys_fresh.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include "syslib.h"

PUBLIC int sys_fresh(proc, ptr, dc, basep, sizep)
int proc;                       /* process whose map is to be changed */
struct mem_map *ptr;            /* pointer to new map */
phys_clicks dc;                 /* size of initialized data */
phys_clicks *basep, *sizep;     /* base and size for free_mem() */
{
/* Create a fresh process image for exec().  Tell the kernel. */

  message m;
  int r;

  m.m1_i1 = proc;
  m.m1_i2 = (int) dc;
  m.m1_p1 = (char *) ptr;
  r = _taskcall(SYSTASK, SYS_FRESH, &m);
  *basep = (phys_clicks) m.m1_i1;
  *sizep = (phys_clicks) m.m1_i2;
  return(r);
```

```
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syslib/sys_getmap.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include "syslib.h"

PUBLIC int sys_getmap(proc, ptr)
int proc;                       /* process whose map is to be fetched */
struct mem_map *ptr;            /* pointer to new map */
{
/* Want to know map of a process, ask the kernel. */

  message m;

  m.m1_i1 = proc;
  m.m1_p1 = (char *) ptr;
  return(_taskcall(SYSTASK, SYS_GETMAP, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syslib/sys_getsp.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include "syslib.h"

PUBLIC int sys_getsp(proc, newsp)
int proc;                       /* process whose sp is wanted */
vir_bytes *newsp;               /* place to put sp read from kernel */
{
/* Ask the kernel what the sp is. */

  message m;
  int r;

  m.m1_i1 = proc;
  r = _taskcall(SYSTASK, SYS_GETSP, &m);
  *newsp = (vir_bytes) m.STACK_PTR;
  return(r);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syslib/sys_kill.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
#include "syslib.h"

PUBLIC int sys_kill(proc, signr)
int proc;                       /* which proc has exited */
int signr;                      /* signal number: 1 - 16 */
{
/* A proc has to be signaled via MM.  Tell the kernel. */
  message m;

  m.m6_i1 = proc;
  m.m6_i2 = signr;
  return(_taskcall(SYSTASK, SYS_KILL, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syslib/sys_newmap.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
#include "syslib.h"

PUBLIC int sys_newmap(proc, ptr)
int proc;                       /* process whose map is to be changed */
struct mem_map *ptr;            /* pointer to new map */
{
/* A process has been assigned a new memory map.  Tell the kernel. */

  message m;

  m.m1_i1 = proc;
  m.m1_p1 = (char *) ptr;
  return(_taskcall(SYSTASK, SYS_NEWMAP, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syslib/sys_oldsig.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```c
#include "syslib.h"

PUBLIC int sys_oldsig(proc, sig, sighandler)
int proc;                       /* process to be signaled  */
int sig;                        /* signal number: 1 to _NSIG */
sighandler_t sighandler;        /* pointer to signal handler in user space */
{
/* A proc has to be signaled.  Tell the kernel. This function is obsolete. */

  message m;

  m.m6_i1 = proc;
  m.m6_i2 = sig;
  m.m6_f1 = sighandler;
  return(_taskcall(SYSTASK, SYS_OLDSIG, &m));
```

```
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syslib/sys_sendsig.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include "syslib.h"

PUBLIC int sys_sendsig(proc, smp)
int proc;
struct sigmsg *smp;
{
  message m;

  m.m1_i1 = proc;
  m.m1_p1 = (char *) smp;
  return(_taskcall(SYSTASK, SYS_SENDSIG, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syslib/sys_sigret.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include "syslib.h"

PUBLIC int sys_sigreturn(proc, scp, flags)
int proc;
vir_bytes scp;
int flags;
{
  message m;

  m.m1_i1 = proc;
  m.m1_i2 = flags;
  m.m1_p1 = (char *) scp;
  return(_taskcall(SYSTASK, SYS_SIGRETURN, &m));
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syslib/sys_times.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include "syslib.h"

PUBLIC int sys_times(proc, ptr)
int proc;                       /* proc whose times are needed */
clock_t ptr[5];                 /* pointer to time buffer */
{
/* Fetch the accounting info for a proc. */
  message m;
  int r;

```

```
  m.m1_i1 = proc;
  m.m1_p1 = (char *)ptr;
  r = _taskcall(SYSTASK, SYS_TIMES, &m);
  ptr[0] = m.USER_TIME;
  ptr[1] = m.SYSTEM_TIME;
  ptr[2] = m.CHILD_UTIME;
  ptr[3] = m.CHILD_STIME;
  ptr[4] = m.BOOT_TICKS;
  return(r);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                           src/lib/syslib/sys_trace.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include "syslib.h"

PUBLIC int sys_trace(req, procnr, addr, data_p)
int req, procnr;
long addr, *data_p;
{
  message m;
  int r;

  m.m2_i1 = procnr;
  m.m2_i2 = req;
  m.m2_l1 = addr;
  if (data_p) m.m2_l2 = *data_p;
  r = _taskcall(SYSTASK, SYS_TRACE, &m);
  if (data_p) *data_p = m.m2_l2;
  return(r);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                           src/lib/syslib/sys_xit.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#include "syslib.h"

PUBLIC int sys_xit(parent, proc, basep, sizep)
int parent;                       /* parent of exiting process */
int proc;                         /* which process has exited */
phys_clicks *basep;               /* where to return base of shadow [68000] */
phys_clicks *sizep;               /* where to return size of shadow [68000] */
{
/* A process has exited.  Tell the kernel. */

  message m;
  int r;

  m.m1_i1 = parent;
  m.m1_i2 = proc;
```

```
  r = _taskcall(SYSTASK, SYS_XIT, &m);
  *basep = (phys_clicks) m.m1_i1;
  *sizep = (phys_clicks) m.m1_i2;
  return(r);
}
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                         src/lib/syslib/taskcall.c
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
/* _taskcall() is the same as _syscall() except it returns negative error
 * codes directly and not in errno.  This is a better interface for MM and
 * FS.
 */

#include <lib.h>
#include <minix/syslib.h>

PUBLIC int _taskcall(who, syscallnr, msgptr)
int who;
int syscallnr;
register message *msgptr;
{
  int status;

  msgptr->m_type = syscallnr;
  status = _sendrec(who, msgptr);
  if (status != 0) return(status);
  return(msgptr->m_type);
}
```