NAME

int64, add64, add64u, add64ul, sub64, sub64u, sub64ul, diff64, cvu64, cvul64, cv64u, cv64ul, div64u, rem64u, mul64u, cmp64, cmp64u, cmp64ul, ex64lo, ex64hi, make64 − 64 bit disk offset computations

SYNOPSIS

#include <minix/u64.h>

u64_t add64(u64_t *i*, **u64_t** *j***)**
u64_t add64u(u64_t *i*, **unsigned** *j***)**
u64_t add64ul(u64_t *i*, **unsigned long** *j***)**
u64_t sub64(u64_t *i*, **u64_t** *j***)**
u64_t sub64u(u64_t *i*, **unsigned** *j***)**
u64_t sub64ul(u64_t *i*, **unsigned long** *j***)**
unsigned diff64(u64_t *i*, **u64_t** *j***)**
u64_t cvu64(unsigned *i***)**
u64_t cvul64(unsigned long *i***)**
unsigned cv64u(u64_t *i***)**
unsigned long cv64ul(u64_t *i***)**
unsigned long div64u(u64_t *i*, **unsigned** *j***)**
unsigned rem64u(u64_t *i*, **unsigned** *j***)**
u64_t mul64u(unsigned long *i*, **unsigned** *j***)**
int cmp64(u64_t *i*, **u64_t** *j***)**
int cmp64u(u64_t *i*, **unsigned** *j***)**
int cmp64ul(u64_t *i*, **unsigned long** *j***)**
unsigned long ex64lo(u64_t *i***)**
unsigned long ex64hi(u64_t *i***)**
u64_t make64(unsigned long *lo*, **unsigned long** *hi***)**

DESCRIPTION

The **int64** family of functions allow Minix to handle disks of up to 4 terabytes using 32 bit sector numbers and 64 bit byte offsets on a machine where the C type **long** is 32 bits. The <minix/u64.h> include file defines a 64 bit data type, **u64_t**, and a number of functions to operate on them. Note that these functions are geared towards common disk offset and block computations, and do not provide a full set of 64 bit operations. They are:

u64_t add64(u64_t *i*, **u64_t** *j***)**
: Add the 64 bit numbers *i* and *j* forming a 64 bit result.

u64_t add64u(u64_t *i*, **unsigned** *j***)**
: Add an unsigned *j* to a 64 bit number *i* forming a 64 bit result.

u64_t add64ul(u64_t *i*, **unsigned long** *j***)**
: Add an unsigned long *j* to a 64 bit number *i* forming a 64 bit result.

u64_t sub64(u64_t *i*, **u64_t** *j***)**
: Subtract the 64 bit number *j* from the 64 bit number *i* forming a 64 bit result.

u64_t sub64u(u64_t *i*, **unsigned** *j***)**
: Subtract the unsigned *j* from the 64 bit number *i* forming a 64 bit result.

u64_t sub64ul(u64_t *i*, **unsigned long** *j***)**
: Subtract the unsigned long *j* from the 64 bit number *i* forming a 64 bit result.

unsigned diff64(u64_t *i*, **u64_t** *j***)**
: Subtract the 64 bit number *j* from the 64 bit number *i* forming an unsigned. Overflow is not checked.

u64_t cvu64(unsigned *i***)**
: Convert an unsigned to a 64 bit number.

u64_t cvul64(unsigned long *i***)**
: Convert an unsigned long to a 64 bit number.

unsigned cv64u(u64_t *i***)**
Convert a 64 bit number to an unsigned if it fits, otherwise return **UINT_MAX**.

unsigned long cv64ul(u64_t *i***)**
Convert a 64 bit number to an unsigned long if it fits, otherwise return **ULONG_MAX**.

unsigned long div64u(u64_t *i***, unsigned** *j***)**
Divide the 64 bit number *i* by the unsigned *j* giving an unsigned long. Overflow is not checked. (Typical "byte offset to block number" conversion.)

unsigned rem64u(u64_t *i***, unsigned** *j***)**
Compute the remainder of the division of the 64 bit number *i* by the unsigned *j* as an unsigned. (Typical "byte offset within a block" computation.)

u64_t mul64u(unsigned long *i***, unsigned** *j***)**
Multiply the unsigned long *i* by the unsigned *j* giving a 64 bit number. (Typical "block number to byte offset" conversion.)

int cmp64(u64_t *i***, u64_t** *j***)**
Compare two 64 bit numbers. Returns **-1** if $i < j$, **0** if $i == j$, and **1** if $i > j$.

int cmp64u(u64_t *i***, unsigned** *j***)**
Likewise compare a 64 bit number with an unsigned.

int cmp64ul(u64_t *i***, unsigned long** *j***)**
Likewise compare a 64 bit number with an unsigned long.

unsigned long ex64lo(u64_t *i***)**
Extract the low 32 bits of a 64 bit number.

unsigned long ex64hi(u64_t *i***)**
Extract the high 32 bits of a 64 bit number.

u64_t make64(unsigned long *lo***, unsigned long** *hi***)**
Combine the low and high parts of a 64 bit number to a 64 bit number. (The last three functions are used to pass 64 bit numbers in messages within the kernel. They should not be used for anything else.)

SEE ALSO

fcntl(2), **controller**(4).

NOTES

With the usual disk block size of 512 bytes the maximum disk size is 512 ∗ 4 gigabytes = 2 terabytes.

Standard Minix only uses 64 bit computations withing the disk drivers, so individual partitions are still limited to 4 gigabytes. Minix-vmd has 64 bit computations also in the file system code.

Special care must be taken when accessing disk devices. For Minix one may have to temporarily change the start of the partition to go beyond 4 G. Minix-vmd can go beyond 4 G, but the **lseek** system call is still limited to a 32 bit offset. One needs to use

fcntl(*fd***, F_SEEK, u64_t** *offset***)**

to seek to a 64 bit position.

AUTHOR

Kees J. Bot (kjb@cs.vu.nl)