Command: **de – disk editor**
Syntax: **de** [**–w**] *block_device*
de –r *file*
Flags: **–r** Recover a file that has been removed
–w Enable writing, so device can be modified
Examples: de –r /usr/ast/prog.c # Undo the effects of: *rm /usr/ast/prog.c*
de –w /dev/fd0 # Edit */dev/fd0* for writing

The *de* program allows a system administrator to examine and modify a MINIX file system device. Commands are available to move to any address on the disk and display the disk block contents. This information may be presented in one of three visual modes: as two-byte words, as ASCII characters or as a bit map. The disk may be searched for a string of characters. If the **–w** option is given, *de* will open the device for writing and words may be modified. Without this flag, writing is prohibited. Lost blocks and files can be recovered using a variety of commands. The **–r** option supports automated recovery of files removed by *unlink*.

Positioning

Disks are divided into blocks (also called ‘‘zones’’) of 1024 bytes. *De* keeps a current address on the disk as a block number and a byte offset within the block. In some visual modes the offset is rounded off, for example, in ‘‘word’’ mode the offset must be even.

There are different types of blocks on a file system device, including a super block, bit maps, i-nodes and data blocks. *De* knows the type of the current block, but will allow most positioning commands and visual modes to function anywhere on the disk.

The *f* command (or PGDN on the keypad) moves forward to the next block, similarly *b* (PGUP) moves backwards one block. *F* (END) moves to the last block and *B* (HOME) moves to the first block.

The arrow keys (or *u*, *d*, *l*, and *r*) change the current address by small increments. The size of the increment depends on the current display mode, as shown below. The various sizes suit each display and pointers move on the screen to follow each press of an arrow key.

Mode	Up	Down	Left	Right
Word	–2	+2	–32	+32
Block	–64	+64	–1	+1
Map	–256	+256	–4	+4

The *g* command allows movement to any specified block. Like all commands that take arguments, a prompt and subsequent input are written to the bottom line of the screen. Numerical entry may be decimal, octal or hexadecimal, for example 234, –1, 070, 0xf3, –X3C.

While checking an i-node one may want to move to a block listed as a zone of the file. The *G* command takes the contents at the current address in the device as a block number and indirectly jumps to that block.

The address may be set to the start of any i-node using the command and supplying an i-node number. The *I* command maps a given file name into an i-node address. The file must exist on the current device and this device must be mounted.

The Display

The first line of the display contains the device name, the name of the current output file (if one is open) and the current search string. If *de* is being run with the **–w** option then the device name is flagged with ‘‘(w).’’ If a string is too long to fit on the line it is marked with ‘‘...’’.

The second line contains the current block number, the total number of blocks, and the type of the current block. The types are: boot, super, i-node bit map, zone bit map, i-nodes and data block. If the current address is within a data block then the string ‘‘in use’’ is displayed if the block corresponds to a set in the zone bit map.

The third line shows the offset in the current block. If the current address is within either the i-node or zone bit maps then the i-node or block number corresponding to the current bit is shown. If the current address is within an i-node then the i-node number and ‘‘in use’’ status is displayed. If the address is within a bit map or i-node block, but past the last usable entry, then the string ‘‘padding’’ is shown.

The rest of the screen is used to display data from the current block. There are three visual display modes: ‘‘word,’’ ‘‘block,’’ and ‘‘map.’’ The *v* command followed by *w*, *b*, or *m* sets the current display mode.

In ‘‘word’’ mode 16 words, of two bytes each, are shown in either base 2, 8, 10 or 16. The current base is displayed to the far right of the screen. It can be changed using the *o* command followed by either an *h* (hexadecimal), *d* (decimal), *o* (octal) or *b* (binary).

De knows where i-nodes are, and will display the contents in a readable format, including the *rwx* bits, the user name and the time field. If the current page is at the beginning of the super block, or an executable file or an *ar* archive, then *de* will also inform the user. In all other cases the contents of the 16 words are shown to the right as equivalent ASCII characters.

In ‘‘block’’ mode a whole block of 1024 bytes is displayed as ASCII characters, 64 columns by 16 lines. Control codes are shown as highlighted characters. If the high order bit is set in any of the 1024 bytes then an ‘‘MSB’’ flag is shown on the far right of the screen, but these bytes are not individually marked.

In ‘‘map’’ mode 2048 bits (256 bytes) are displayed from the top to the bottom (32 bits) and from the left to the right of the screen. Bit zero of a byte is towards the top of the screen. This visual mode is generally used to observe the bit map blocks. The number of set bits displayed is written on the far right of the screen.

Searching

A search for an ASCII string is initiated by the / command. Control characters not used for other purposes may be entered in the search string, for example CTRL-J is an end-of-line character. The search is from the current position to the end of the current device.

Once a search string has been defined by a use of /, the next search may be initiated with the *n* command, (a / followed immediately by an ENTER is equivalent to an *n*).

Whenever a search is in progress *de* will append one . to the prompt line for every 500 blocks searched. If the string is found between the end of the file system and the actual end of the device, then the current address is set to the end of the file system.

Some of the positioning commands push the current address and visual mode in a stack before going to a new address. These commands are *B, F, g, G, i, I, n, x* and /. The *p* (previous) command pops the last address and visual mode from the stack. This stack is eight entries deep.

Modifying the File System

The *s* command will prompt for a data word and store it at the current address on the disk. This is used to change information that can not be easily changed by any other means.

The data word is 16 bits wide, it may be entered in decimal, octal or hexadecimal. Remember that the **–w** option must be specified for the *s* command to operate. Be careful when modifying a mounted file system.

Recovering Files

Any block on the disk may be written to an output file. This is used to recover blocks marked as free on the disk. A write command will request a file name the first time it is used, on subsequent writes the data is appended to the current output file.

The name of the current output file is changed using the *c* command. This file should be on a different file system, to avoid overwriting an i-node or block before it is recovered.

An ASCII block is usually recovered using the *w* command. All bytes will have their most significant bit cleared before being written to the output file. Bytes containing ’\0’ or ’\177’ are not copied. The *W* command writes the current block (1024 bytes exactly) to the output file.

When a file is deleted using *unlink* the i-node number in the directory is zeroed, but before its removal, it is copied into the end of the file name field. This allows the i-node of a deleted file to be found by searching through a directory. The *x* command asks for the path name of a lost file, extracts the old i-node number and changes the current disk address to the start of the i-node.

Once an i-node is found, all of the freed blocks may be recovered by checking the i-node zone fields, using 'G' to go to a block, writing it back out using 'w', going back to the i-node with *p* and advancing to the next block. This file extraction process is automated by using the *X* command, which goes through the i-node, indirect and double indirect blocks finding all the block pointers and recovering all the blocks of the file.

The *X* command closes the current output file and asks for the name of a new output file. All of the disk blocks must be marked as free, if they are not the command stops and the file must be recovered manually.

When extracting lost blocks *de* will maintain ''holes'' in the file. Thus, a recovered sparse file does not allocate unused blocks and will keep its efficient storage scheme. This property of the *X* command may be used to move a sparse file from one device to another.

Automatic recovery may be initiated by the **–r** option on the command line. Also specified is the path name of a file just removed by *unlink*. *De* determines which mounted file system device held the file and opens it for reading. The lost i-node is found and the file extracted by automatically performing an *x* and an *X* command.

The recovered file will be written to */tmp*. *De* will refuse to automatically recover a file on the same file system as */tmp*. The lost file must have belonged to the user. If automatic recovery will not complete, then manual recovery may be performed.

Miscellaneous

The user can terminate a session with *de* by typing *q*, CTRL-D, or the key associated with SIGQUIT.

The *m* command invokes the MINIX *sh* shell as a subprocess.

For help while using *de* use *h*.

Command Summary

PGUP	b	Back one block
PGDN	f	Forward one block

HOME	B	Goto first block
END	F	Goto last block
UP	u	Move back 2/64/256 bytes
DOWN	d	Move forward 2/64/256 bytes
LEFT	l	Move back 32/1/4 bytes
RIGHT	r	Move forward 32/1/4 bytes
	g	Goto specified block
	G	Goto block indirectly
	i	Goto specified i-node
	I	Filename to i-node
	/	Search
	n	Next occurrence
	p	Previous address
	h	Help
EOF	q	Quit
	m	MINIX shell
	v	Visual mode (w b m)
	o	Output base (h d o b)
	c	Change file name
	w	Write ASCII block
	W	Write block exactly
	x	Extract lost directory entry
	X	Extract lost file blocks
	s	Store word

NOTES: When entering a line in response to a prompt from *de* there are a couple of editing characters available. The previous character may be erased by typing CTRL-H and the whole line may be erased by typing CTRL-U. ENTER terminates the input. If DELETE or a non-ASCII character is typed then the command requesting the input is aborted.

The commands *G*, *s* and *X* will only function if the current visual display mode is ‘‘word.’’ The commands *i*, *I* and *x* change the mode to ‘‘word’’ on completion. The commands *G* and */* change the mode to ‘‘block’’. These restrictions and automatic mode conversions are intended to aid the user.

The ‘‘map’’ mode uses special graphic characters, and only functions if the user is at the console.

De generates warnings for illegal user input or if erroneous data is found on the disk, for example a corrupted magic number. Warnings appear in the middle of the screen for two seconds, then the current page is redrawn. Some minor errors, for example, setting an unknown visual mode, simply ring the bell. Major errors, for example I/O problems on the file system device cause an immediate exit from *de*.

The i-node and zone bit maps are read from the device when *de* starts up. These determine whether ‘‘in use’’ or ‘‘not in use’’ is displayed in the status field at the top of the screen. The bit maps are not re-read while using *de* and will become out-of-date if observing a mounted file system.

De requires termcap definitions for ‘‘cm’’ and ‘‘cl’’. Furthermore, ‘‘so’’ and ‘‘se’’ will also be used if available. The ANSI strings generated by the keypad arrows are recognized, as well as any single character codes defined by ‘‘ku’’, ‘‘kd’’, ‘‘kl’’ and ‘‘kr’’.

Author

The *de* program was written by Terrence Holm.