NAME

edit – text editor (variant of ex for casual users)

SYNOPSIS

/usr/bin/edit [– | –**s**] [–**l**] [–**L**] [–**R**] [–**r** [*filename*]] [–**t** *tag*] [–**v**] [–**V**] [–**x**] [–**w***n*] [–**C**] [+*command* | –**c** *command*] *filename*. . .

/usr/xpg4/bin/edit [– | –**s**] [–**l**] [–**L**] [–**R**] [–**r** [*filename*]] [–**t** *tag*] [–**v**] [–**V**] [–**x**] [–**w***n*] [–**C**] [+*command* | –**c** *command*] *filename*. . .

AVAILABILITY

/usr/bin/edit SUNWcsu

/usr/xpg4/bin/edit SUNWxcu4

DESCRIPTION

edit is a variant of the text editor **ex** recommended for new or casual users who wish to use a command-oriented editor. It operates precisely as **ex** with the following options automatically set:

novice	**ON**
report	**ON**
showmode	**ON**
magic	**OFF**

The following brief introduction should help you get started with **edit**. If you are using a CRT terminal you may want to learn about the display editor **vi**.

To edit the contents of an existing file you begin with the command **edit** *name* to the shell. **edit** makes a copy of the file that you can then edit, and tells you how many lines and characters are in the file. To create a new file, you also begin with the command **edit** with a filename: **edit** *name*; the editor will tell you it is a **[New File]**.

The **edit** command prompt is the colon (**:**), which you should see after starting the editor. If you are editing an existing file, then you will have some lines in **edit**'s buffer (its name for the copy of the file you are editing). When you start editing, **edit** makes the last line of the file the current line. Most commands to **edit** use the current line if you do not tell them which line to use. Thus if you say **print** (which can be abbreviated **p**) and type carriage return (as you should after all **edit** commands), the current line will be printed. If you **delete** (**d**) the current line, **edit** will print the new current line, which is usually the next line in the file. If you **delete** the last line, then the new last line becomes the current one.

If you start with an empty file or wish to add some new lines, then the **append** (**a**) command can be used. After you execute this command (typing a carriage return after the word **append**), **edit** will read lines from your terminal until you type a line consisting of just a dot (**.**); it places these lines after the current line. The last line you type then becomes the current line. The **insert** (**i**) command is like **append**, but places the lines you type before, rather than after, the current line.

edit numbers the lines in the buffer, with the first line having number 1. If you execute the command **1**, then **edit** will type the first line of the buffer. If you then execute the command **d**, **edit** will delete the first line, line 2 will become line 1, and **edit** will print the current line (the new line 1) so you can see where you are. In general, the current line will always be the last line affected by a command.

You can make a change to some text within the current line by using the **substitute** (**s**) command: **s**/*old* /*new*/ where *old* is the string of characters you want to replace and *new* is the string of characters you want to replace *old* with.

The **filename** (**f**) command will tell you how many lines there are in the buffer you are editing and will say **[Modified]** if you have changed the buffer. After modifying a file, you can save the contents of the file by executing a **write** (**w**) command. You can leave the editor by issuing a **quit** (**q**) command. If you run **edit** on a file, but do not change it, it is not necessary (but does no harm) to **write** the file back. If you try to **quit** from **edit** after modifying the buffer without writing it out, you will receive the message **No write since last change (:quit! overrides)**, and **edit** will wait for another command. If you do not want to write the buffer out, issue the **quit** command followed by an exclamation point (**q!**). The buffer is then irretrievably discarded and you return to the shell.

By using the **d** and **a** commands and giving line numbers to see lines in the file, you can make any changes you want. You should learn at least a few more things, however, if you will use **edit** more than a few times.

The **change** (**c**) command changes the current line to a sequence of lines you supply (as in **append**, you type lines up to a line consisting of only a dot (**.**). You can tell **change** to change more than one line by giving the line numbers of the lines you want to change, that is, **3,5c**. You can print lines this way too: **1,23p** prints the first 23 lines of the file.

The **undo** (**u**) command reverses the effect of the last command you executed that changed the buffer. Thus if you execute a **substitute** command that does not do what you want, type **u** and the old contents of the line will be restored. You can also **undo** an **undo** command. **edit** will give you a warning message when a command affects more than one line of the buffer. Note that commands such as **write** and **quit** cannot be undone.

To look at the next line in the buffer, type carriage return. To look at a number of lines, type **^D** (while holding down the control key, press **d**) rather than carriage return. This will show you a half-screen of lines on a CRT or 12 lines on a hardcopy terminal. You can look at nearby text by executing the **z** command. The current line will appear in the middle of the text displayed, and the last line displayed will become the current line; you can get back to the line where you were before you executed the **z** command by typing ´´. The **z** command has other options: **z**– prints a screen of text (or 24 lines) ending where you are; **z+** prints the next screenful. If you want less than a screenful of lines, type **z.11** to display five lines before and five lines after the current line. (Typing **z.***n*, when *n* is an odd number, displays a total of *n* lines, centered about the current line; when *n* is an even number, it displays *n*–1 lines, so that the lines displayed are centered around the current line.) You can give counts after other commands; for example, you can delete 5 lines starting with the current line with the command **d5**.

To find things in the file, you can use line numbers if you happen to know them; since the line numbers change when you insert and delete lines this is somewhat unreliable. You can search backwards and forwards in the file for strings by giving commands of the form /*text*/ to search forward for *text* or ?*text*? to search backward for *text* . If a search reaches the end of the file without finding *text*, it wraps around and continues to search back to the line where you are. A useful feature here is a search of the form /^*text*/ which searches for *text* at the beginning of a line. Similarly /*text*$/ searches for *text* at the end of a line. You can leave off the trailing **/** or **?** in these commands.

The current line has the symbolic name dot (**.**); this is most useful in a range of lines as in **.,$p** which prints the current line plus the rest of the lines in the file. To move to the last line in the file, you can refer to it by its symbolic name $. Thus the command **$d** deletes the last line in the file, no matter what the current line is. Arithmetic with line references is also possible. Thus the line **$**–**5** is the fifth before the last and **.+20** is 20 lines after the current line.

You can find out the current line by typing **'.='** . This is useful if you wish to move or copy a section of text within a file or between files. Find the first and last line numbers you wish to copy or move. To move lines 10 through 20, type **10,20d a** to delete these lines from the file and place them in a buffer named **a**. **edit** has 26 such buffers named **a** through **z**. To put the contents of buffer **a** after the current line, type **put a**. If you want to move or copy these lines to another file, execute an **edit** (**e**) command after copying the lines; following the **e** command with the name of the other file you wish to edit, that is, **edit chapter2**. To copy lines without deleting them, use **yank** (**y**) in place of **d**. If the text you wish to move or copy is all within one file, it is not necessary to use named buffers. For example, to move lines 10 through 20 to the end of the file, type **10,20m $**.

OPTIONS

These options can be turned on or off using the **set** command in **ex**(1).

– | –**s** Suppress all interactive user feedback. This is useful when processing editor scripts.

–**l** Set up for editing LISP programs.

–**L** List the name of all files saved as the result of an editor or system crash.

–**R** **Readonly** mode; the **readonly** flag is set, preventing accidental overwriting of the file.

–**r** *filename* Edit *filename* after an editor or system crash. (Recovers the version of *filename* that was in the buffer when the crash occurred.)

–**t** *tag* Edit the file containing the *tag* and position the editor at its definition.

–**v** Start up in display editing state using **vi**. You can achieve the same effect by simply typing the **vi** command itself.

–**V** Verbose. Any non-tty input will be echoed on standard error. This may be useful when processing editor commands within shell scripts.

–**x** Encryption option; when used, **edit** simulates the **X** command of **ex** and prompts the user for a key. This key is used to encrypt and decrypt text using the algorithm of the **crypt** command. The **X** command makes an educated guess to determine whether text read in is encrypted or not. The temporary buffer file is encrypted also, using a transformed version of the key typed in for the –**x** option.

–**w***n* Set the default window size to *n*. This is useful when using the editor over a slow speed line.

–**C** Encryption option; same as the –**x** option, except that **vi** simulates the **C** command of **ex**. The **C** command is like the **X** command of **ex**, except that all text read in is assumed to have been encrypted.

+*command* | –**c** *command*
Begin editing by executing the specified editor *command* (usually a search or positioning command).

The *filename* argument indicates one or more files to be edited.

SEE ALSO

ed(1), **ex**(1), **vi**(1)

NOTES

The encryption options are provided with the Security Administration Utilities package, which is available only in the United States.

/usr/xpg4/bin/edit is identical to **/usr/bin/edit**.