NAME

ed, red – text editor

SYNOPSIS

ed [–**s** | –] [–**p** *string*] [–**x**] [–**C**] [*file*]
red [–**s** | –] [–**p** *string*] [–**x**] [–**C**] [*file*]

AVAILABILITY

SUNWcsu

DESCRIPTION

ed is the standard text editor. If the *file* argument is given, **ed** simulates an **e** command (see below) on the named file; that is to say, the file is read into **ed**'s buffer so that it can be edited.

ed operates on a copy of the file it is editing; changes made to the copy have no effect on the file until a **w** (write) command is given. The copy of the text being edited resides in a temporary file called the *buffer*. There is only one buffer.

red is a restricted version of **ed**. It will only allow editing of files in the current directory. It prohibits executing shell commands via **!***shell command*. Attempts to bypass these restrictions result in an error message (*restricted shell*).

Both **ed** and **red** support the **fspec**(4) formatting capability. The default terminal mode is either **stty** –**tabs** or **stty tab3**, where tab stops are set at eight columns (see **stty**(1)). If, however, the first line of *file* contains a format specification, that specification will override the default mode. For example, if the first line of *file* contains

<:t5,10,15 s72:>

tab stops would be set at 5, 10, and 15, and a maximum line length of 72 would be imposed.

Commands to **ed** have a simple and regular structure: zero, one, or two *addresses* followed by a single-character *command*, possibly followed by parameters to that command. These addresses specify one or more lines in the buffer. Every command that requires addresses has default addresses, so that the addresses can very often be omitted.

In general, only one command may appear on a line. Certain commands allow the input of text. This text is placed in the appropriate place in the buffer. While **ed** is accepting text, it is said to be in *input mode*. In this mode, *no* commands are recognized; all input is merely collected. Leave input mode by typing a period (**.**) at the beginning of a line, followed immediately by a carriage return.

Regular Expressions

ed supports a limited form of *regular expression* notation. Regular expressions are used in addresses to specify lines and in some commands (for example, **s**) to specify portions of a line that are to be substituted. To understand addressing in **ed**, it is necessary to know that at any time there is a *current line*. Generally speaking, the current line is the last line affected by a command; the exact effect on the current line is discussed under the description of each command.

Internationalized Regular Expressions are used in the POSIX and "C" locales. In other locales, Internationalized Regular Expressions are used if the following two conditions are met:

- **/usr/lib/locale/***locale***/LC_COLLATE/CollTable** is present
- **/usr/lib/locale/***locale***/LC_COLLATE/coll.so** is not present;

otherwise, Simple Regular Expressions are used.

Internationalized Regular Expressions are explained on **regex**(5).
Simple Regular Expressions are explained on **regexp**(5).

ed Commands

Commands may require zero, one, or two addresses. Commands that require no addresses regard the presence of an address as an error. Commands that accept one or two addresses assume default addresses when an insufficient number of addresses is given; if more addresses are given than such a command requires, the last one(s) are used.

Typically, addresses are separated from each other by a comma (**,**). They may also be separated by a semicolon (**;**). In the latter case, the first address is calculated, the current line (.) is set to that value, and then the second address is calculated. This feature can be used to determine the starting line for forward and backward searches (see Rules 5 and 6, above). The second address of any two-address sequence must correspond to a line in the buffer that follows the line corresponding to the first address.

In the following list of **ed** commands, the parentheses shown prior to the command are *not* part of the address; rather they show the default address(es) for the command.

Each address component can be preceded by zero or more blank characters. The command letter can be preceded by zero or more blank characters. If a suffix letter **(**l**, n** or **p**) is given, it must immediately follow the command.

The **e**, **E**, **f**, **r**, and **w** commands take an optional *file* parameter, separated from the command letter by one or more blank characters.

If changes have been made in the buffer since the last **w** command that wrote the entire buffer, **ed** will warn the user if an attempt is made to destroy the editor buffer via the **e** or **q** commands. The **ed** utility will write the string:

"?\ n"

(followed by an explanatory message if *help mode* has been enabled via the **H** command) to standard output and will continue in command mode with the current line number unchanged. If the **e** or **q** command is repeated with no intervening command, it will take effect.

If an end-of-file is detected on standard input when a command is expected, the **ed** utility acts as if a **q** command had been entered.

It is generally illegal for more than one command to appear on a line. However, any command (except **e**, **f**, **r**, or **w**) may be suffixed by **l**, **n**, or **p** in which case the current line is either listed, numbered or written, respectively, as discussed below under the **l**, **n**, and **p** commands.

(.)a
<*text*>
. The **a**ppend command accepts zero or more lines of text and appends it after the addressed line in the buffer. The current line (**.**) is left at the last inserted

line, or, if there were none, at the addressed line. Address 0 is legal for this command: it causes the ''appended'' text to be placed at the beginning of the buffer. The maximum number of characters that may be entered from a terminal is 256 per line (including the new-line character).

(.)c
<*text*>
. The **c**hange command deletes the addressed lines from the buffer, then accepts zero or more lines of text that replaces these lines in the buffer. The current line (**.**) is left at the last line input, or, if there were none, at the first line that was not deleted; if the lines deleted were originally at the end of the buffer, the current line number will be set to the address of the new last line; if no lines remain in the buffer, the current line number will be set to zero.

C Same as the **X** command, described later, except that **ed** assumes all text read in for the **e** and **r** commands is encrypted unless a null key is typed in.

(. , .)d The **d**elete command deletes the addressed lines from the buffer. The line after the last line deleted becomes the current line; if the lines deleted were originally at the end of the buffer, the new last line becomes the current line. If no lines remain in the buffer, the current line number will be set to zero.

e *file* The **e**dit command deletes the entire contents of the buffer and then reads the contents of *file* into the buffer. The current line (**.**) is set to the last line of the buffer. If *file* is not given, the currently remembered file name, if any, is used (see the **f** command). The number of bytes read will be written to standard output, unless the **–s** option was specified, in the following format:

"%d\ n" <*number of bytes read*>

file is remembered for possible use as a default file name in subsequent **e**, **E**, **r**, and **w** commands. If *file* is replaced by **!**, the rest of the line is taken to be a shell (**sh**(1)) command whose output is to be read. Such a shell command is *not* remembered as the current file name. See also **DIAGNOSTICS** below. All marks will be discarded upon the completion of a successful **e** command. If the buffer has changed since the last time the entire buffer was written, the user will be warned, as described previously.

E *file* The **E**dit command is like **e**, except that the editor does not check to see if any changes have been made to the buffer since the last **w** command.

f *file* If *file* is given, the **f** command will change the currently remembered path name to *file*; whether the name is changed or not, it then will write the (possibly new) currently remembered path name to the standard output in the following format:

"%s\ n" *pathname*

The current line number is unchanged.

(1 , $)g/*RE***/***command list*
In the **g**lobal command, the first step is to mark every line that matches the given RE. Then, for every such line, the given *command list* is executed with

the current line (**.**) initially set to that line. When the **g** command completes, the current line number will have the value assigned by the last command in the command list. If there were no matching lines, the current line number will not be changed. A single command or the first of a list of commands appears on the same line as the global command. All lines of a multi-line list except the last line must be ended with a ****; **a**, **i**, and **c** commands and associated input are permitted. The **.** terminating input mode may be omitted if it would be the last line of the *command list*. An empty *command list* is equivalent to the **p** command. The **g**, **G**, **v**, **V**, and **!** commands are *not* permitted in the *command list*. See also the **NOTES** and the last paragraph before **FILES** below. Any character other than space or newline can be used instead of a slash to delimit the *RE*. Within the *RE*, the *RE* delimiter itself can be used as a literal character if it is preceded by a backslash.

(1,$)G/*RE***/**
In the interactive **G**lobal command, the first step is to mark every line that matches the given RE. Then, for every such line, that line is written to standard output, the current line (**.**) is changed to that line, and any *one* command (other than one of the **a**, **c**, **i**, **g**, **G**, **v**, and **V** commands) may be input and is executed. After the execution of that command, the next marked line is written, and so on; a new-line acts as a null command; an **&** causes the re-execution of the most recent non-null command executed within the current invocation of **G**. Note: The commands input as part of the execution of the **G** command may address and affect *any* lines in the buffer. The final value of the current line number will be the value set by the last command successfully executed. (Note that the last command successfully executed will be the **G** command itself if a command fails or the null command is specified.) If there were no matching lines, the current line number will not be changed. The **G** command can be terminated by a SIGINT signal. The **G** command can be terminated by an interrupt signal (ASCII DEL or BREAK). Any character other than space or newline can be used instead of a slash to delimit the *RE*. Within the *RE*, the *RE* delimiter itself can be used as a literal character if it is preceded by a backslash.

h The **h**elp command gives a short error message that explains the reason for the most recent **?** diagnostic. The current line number is unchanged.

H The **H**elp command causes **ed** to enter a mode in which error messages are written for all subsequent **?** diagnostics. It will also explain the previous **?** if there was one. The **H** command alternately turns this mode on and off; it is initially off. The current line number is unchanged.

(.)i
<*text*>
. The **i**nsert command accepts zero or more lines of text and inserts it before the addressed line in the buffer. The current line (**.**) is left at the last inserted line, or, if there were none, at the addressed line. This command differs from the **a** command only in the placement of the input text. Address 0 is not legal for

this command. The maximum number of characters that may be entered from a terminal is 256 per line (including the new-line character).

(. , .+1)j The **j**oin command joins contiguous lines by removing the appropriate new-line characters. If exactly one address is given, this command does nothing. If lines are joined, the current line number will be set to the address of the joined line; otherwise, the current line number is unchanged.

(.)k*x* The mar**k** command marks the addressed line with name *x*, which must be an ASCII lower-case letter (**a**–**z**). The address ′*x* then addresses this line; the current line (**.**) is unchanged.

(. , .)l The **l** command writes to standard output the addressed lines in a visually unambiguous form. The characters (**** , **\ a**, **\ b**, **\ f**, **\ r**, **\ t**, **\v**) will be written as the corresponding escape sequence; the **\ n** in that table is not applicable. Non-printable characters not in the table will be written as one three-digit octal number (with a preceding backslash character) for each byte in the character (most significant byte first).

Long lines will be folded, with the point of folding indicated by writing backslash/newline character; the length at which folding occurs is unspecified, but should be appropriate for the output device. The end of each line will be marked with a **$**. An **l** command can be appended to any other command other than **e**, **E**, **f**, **q**, **Q**, **r**, **w**, or **!**. The current line number will be set to the address of the last line written.

(. , .)m*a* The **m**ove command repositions the addressed line(s) after the line addressed by *a*. Address **0** is legal for *a* and causes the addressed line(s) to be moved to the beginning of the file. It is an error if address *a* falls within the range of moved lines; the current line (**.**) is left at the last line moved.

(. , .)n The **n**umber command writes the addressed lines, preceding each line by its line number and a tab character; the current line (**.**) is left at the last line written. The **n** command may be appended to any command other than **e**, **E**, **f**, **q**, **Q**, **r**, **w**, or **!**.

(. , .)p The **p**rint command writes the addressed lines to standard output; the current line (**.**) is left at the last line written. The **p** command may be appended to any command other than **e**, **E**, **f**, **q**, **Q**, **r**, **w**, or **!**. For example, **dp** deletes the current line and writes the new current line.

P The **P** command causes **ed** to prompt with an asterisk (*) (or *string*, if **−p** is specified) for all subsequent commands. The **P** command alternatively turns this mode on and off; it is initially on if the **−p** option is specified, otherwise off. The current line is unchanged.

q The **q**uit command causes **ed** to exit. If the buffer has changed since the last time the entire buffer was written, the user will be warned; see **DIAGNOSTICS**.

Q The editor exits without checking if changes have been made in the buffer since the last **w** command.

($)r *file* The **r**ead command reads the contents of *file* into the buffer. If *file* is not given, the currently remembered file name, if any, is used (see the **e** and **f** commands). The currently remembered file name is *not* changed unless *file* is the very first file name mentioned since **ed** was invoked. Address 0 is legal for **r** and causes the file to be read in at the beginning of the buffer. If the read is successful and the **–s** option was not specified, the number of characters read is written to standard output in the following format:

"%d\ n", <*number of bytes read*>

The current line (**.**) is set to the last line read. If *file* is replaced by **!**, the rest of the line is taken to be a shell (see **sh**(1)) command whose output is to be read. For example, **$r !ls** appends current directory to the end of the file being edited. Such a shell command is *not* remembered as the current file name.

(. , .)s/*RE***/***replacement***/**
(. , .)s/*RE***/***replacement***/***count*, *count*=[**1-512**]
(. , .)s/*RE***/***replacement***/g**
(. , .)s/*RE***/***replacement***/l**
(. , .)s/*RE***/***replacement***/n**
(. , .)s/*RE***/***replacement***/p**

The **s**ubstitute command searches each addressed line for an occurrence of the specified RE. Zero or more substitution commands can be specified. In each line in which a match is found, all (non-overlapped) matched strings are replaced by the *replacement* if the global replacement indicator **g** appears after the command. If the global indicator does not appear, only the first occurrence of the matched string is replaced. If a number *count* appears after the command, only the *count*-th occurrence of the matched string on each addressed line is replaced. It is an error if the substitution fails on *all* addressed lines. Any character other than space or new-line may be used instead of **/** to delimit the RE and the *replacement*; the current line (**.**) is left at the last line on which a substitution occurred. Within the *RE*, the *RE* delimiter itself can be used as a literal character if it is preceded by a backslash. See also the last paragraph before **FILES** below.

An ampersand (**&**) appearing in the *replacement* is replaced by the string matching the RE on the current line. The special meaning of **&** in this context may be suppressed by preceding it by **** . As a more general feature, the characters *****n*, where *n* is a digit, are replaced by the text matched by the *n*-th regular subexpression of the specified RE enclosed between **\ (** and **\ )**. When nested parenthesized subexpressions are present, *n* is determined by counting occurrences of **\ (** starting from the left. When the character **%** is the only character in the *replacement*, the *replacement* used in the most recent substitute command is used as the *replacement* in the current substitute command; if there was no previous substitute command, the use of **%** in this manner is an error. The **%** loses its special meaning when it is in a replacement string of more than one character or is preceded by a **** . For each backslash (\) encountered in scanning *replacement* from beginning to end, the following character loses its special meaning (if any). It is unspecified what special meaning is given to any character other than **& , \ , %** or digits.

A line may be split by substituting a new-line character into it. The new-line in the *replacement* must be escaped by preceding it by **** . Such substitution cannot be done as part of a **g** or **v** command list. The current line number will be set to the address of the last line on which a substitution is performed. If no substitution is performed, the current line number is unchanged. If a line is split, a substitution is considered to have been performed on each of the new lines for the purpose of determining the new current line number. A substitution is considered to have been performed even if the replacement string is identical to the string that it replaces.

The substitute command supports the following indicators:

count Substitute for the *count*th occurrence only of the RE found on each addressed line. *count* must be between **1**-**512**.

g Globally substitute for all non-overlapping instances of the RE rather than just the first one. If both **g** and *count* are specified, the results are unspecified.

l Write to standard output the final line in which a substitution was made. The line will be written in the format specified for the **l** command.

n Write to standard output the final line in which a substitution was made. The line will be written in the format specified for the **n** command.

p Write to standard output the final line in which a substitution was made. The line will be written in the format specified for the **p** command.

(. , .)t*a* This command acts just like the **m** command, except that a *copy* of the addressed lines is placed after address **a** (which may be 0); the current line (**.**) is left at the last line copied.

u The **u**ndo command nullifies the effect of the most recent command that modified anything in the buffer, namely the most recent **a**, **c**, **d**, **g**, **i**, **j**, **m**, **r**, **s**, **t**, **u**, **v**, **G**, or **V** command. All changes made to the buffer by a **g**, **G**, **v** or **V** global command will be undone as a single change; if no changes were made by the global command (such as with **g/** *RE* **/p**), the **u** command will have no effect. The current line number will be set to the value it had immediately before the command being undone started.

(1 , $)v/*RE***/***command list*

This command is the same as the global command **g**, except that the lines marked during the first step are those that do *not* match the RE.

(1 , $)V/*RE***/**
This command is the same as the interactive global command **G**, except that the lines that are marked during the first step are those that do *not* match the RE.

(1 , $)w *file*
The **w**rite command writes the addressed lines into *file*. If *file* does not exist, it is created with mode **666** (readable and writable by everyone), unless your file creation mask dictates otherwise; see the description of the **umask** special command on **sh**(1). The currently remembered file name is *not* changed unless *file* is the very first file name mentioned since **ed** was invoked. If no file name is given, the currently remembered file name, if any, is used (see the **e** and **f** commands); the current line (**.**) is unchanged. If the command is successful, the number of characters written is printed, unless the –**s** option is specified in the following format:

"%d\ n", <*number of bytes written***>**

If *file* is replaced by **!**, the rest of the line is taken to be a shell (see **sh**(1)) command whose standard input is the addressed lines. Such a shell command is *not* remembered as the current path name. This usage of the write command with **!** is be considered as a ''last **w** command that wrote the entire buffer''.

(1 , $)W *file*
This command is the same as the **w**rite command above, except that it appends the addressed lines to the end of *file* if it exists. If *file* does not exist, it is created as described above for the *w* command.

X An educated guess is made to determine whether text read for the **e** and **r** commands is encrypted. A null key turns off encryption. Subsequent **e**, **r**, and **w** commands will use this key to encrypt or decrypt the text. An explicitly empty key turns off encryption. Also, see the –**x** option of **ed**.

($)= The line number of the addressed line will be written to standard output in the following format:

"%d\ n" <*line number*>

The current line number is unchanged by this command.

!*shell command*
The remainder of the line after the **!** is sent to the UNIX system shell (see **sh**(1)) to be interpreted as a command. Within the text of that command, the unescaped character **%** is replaced with the remembered file name; if a **!** appears as the first character of the shell command, it is replaced with the text of the previous shell command. Thus, **!!** will repeat the last shell command. If any replacements of **%** or **!** are performed, the modified line will be written to the standard output before *command* is executed. The **!** command will write:

"!\ n"

to standard output upon completion, unless the –**s** option is specified. The current line number is unchanged.

(.+1)<new-line**>**
An address alone on a line causes the addressed line to be written. A new-line alone is equivalent to **.+1p**; it is useful for stepping forward through the buffer. The current line number will be set to the address of the written line.

If an interrupt signal (ASCII DEL or BREAK) is sent, **ed** writes a "**?\ n**" and returns to *its* command level.

The **ed** utility will take the standard action for all signals with the following exceptions:

SIGINT The **ed** utility will interrupt its current activity, write the string "**?\ n**" to standard output, and return to command mode.

SIGHUP If the buffer is not empty and has changed since the last write, the **ed** utility will attempt to write a copy of the buffer in a file. First, the file named **ed.hup** in the current directory will be used; if that fails, the file named **ed.hup** in the directory named by the **HOME** environment variable will be used. In any case, the **ed** utility will exit without returning to command mode.

Some size limitation are in effect: 512 characters in a line, 256 characters in a global command list, and 255 characters in the path name of a file (counting slashes). The limit on the number of lines depends on the amount of user memory; each line takes 1 word.

When reading a file, **ed** discards ASCII and NUL characters.

If a file is not terminated by a new-line character, **ed** adds one and puts out a message explaining what it did.

If the closing delimiter of a RE or of a replacement string (for example, **/**) would be the last character before a new-line, that delimiter may be omitted, in which case the addressed line is written. The foll owing pairs of commands are equivalent:

s/s1/s2	**s/s1/s2/p**
g/s1	**g/s1/p**
?s1	**?s1?**

If an invalid command is entered, **ed** will write the string:

"?\ n"

(followed by an explanatory message if *help mode* has been enabled by the **H** command) to standard output and will continue in command mode with the current line number unchanged.

OPTIONS

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

–p *string* Allows the user to specify a prompt string. By default, there is no prompt string.

–s | **–** Suppresses the writing of character counts by **e**, **r**, and **w** commands, of diagnostics from **e** and **q** commands, and of the **!** prompt after a **!***shell command*.

–x Encryption option; when used, **ed** simulates an **X** command and prompts the user for a key. 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. See **NOTES**.

OPERANDS

The following operand is supported:

file If the *file* argument is given, **ed** will simulate an **e** command on the file named by the path name, *file*, before accepting commands from the standard input.

ENVIRONMENT

See **environ**(5) for descriptions of the following environment variables that affect the execution of **ed**: **HOME**, **LC_CTYPE**, **LC_COLLATE**, **LC_MESSAGES**, and **NLSPATH**.

EXIT STATUS

The following exit values are returned:

0 Successful completion without any file or command errors.

>0 An error occurred.

FILES

$TMPDIR If this environment variable is not **NULL**, its value is used in place of **/var/tmp** as the directory name for the temporary work file.

/var/tmp If **/var/tmp** exists, it is used as the directory name for the temporary work file.

/tmp If the environment variable **TMPDIR** does not exist or is **NULL**, and if **/var/tmp** does not exist, then **/tmp** is used as the directory name for the temporary work file.

ed.hup Work is saved here if the terminal is hung up.

/usr/lib/locale/*locale***/LC_COLLATE/CollTable**
collation table generated by **localedef**

/usr/lib/locale/*locale***/LC_COLLATE/coll.so**
shared object containing string transformation library routines

SEE ALSO

edit(1), **ex**(1), **grep**(1), **sed**(1), **sh**(1), **stty**(1), **umask**(1), **vi**(1), **fspec**(4), **environ**(5), **regex**(5), **regexp**(5)

DIAGNOSTICS

? for command errors.

?*file* for an inaccessible file.
(use the **h**elp and **H**elp commands for detailed explanations).

If changes have been made in the buffer since the last **w** command that wrote the entire buffer, **ed** warns the user if an attempt is made to destroy **ed**'s buffer via the **e** or **q** commands. It writes **?** and allows one to continue editing. A second **e** or **q** command at this point will take effect. The −**s** command-line option inhibits this feature.

NOTES

The − option, although it continues to be supported, has been replaced in the documentation by the −**s** option that follows the **Command Syntax Standard** (see **intro**(1)).

A **!** command cannot be subject to a **g** or a **v** command.

The **!** command and the **!** escape from the **e**, **r**, and **w** commands cannot be used if the editor is invoked from a restricted shell (see **sh**(1)).

The sequence **\ n** in a RE does not match a new-line character.

If the editor input is coming from a command file (for example, **ed** *file* **<** *ed_cmd_file*), the editor exits at the first failure.