NAME

sed – stream editor

SYNOPSIS

sed [**−g**] [**−n**] [**−f** commandfile] ... [[**−e**] command] ... [file] ...

DESCRIPTION

Sed copies the input *file*s (default is standard input) to the standard output, perhaps performing one or more editor commands (see *ed*(I)) on each line.

The **−g** flag indicates that all *s* commands should be executed as though followed by a *g*. If only some substitutions are to be done globally, leave out the **−g** flag and put the *g*'s at the end of the appropriate command lines.

The **−n** flag indicates that only lines that are explicitly printed by *p* commands are to be copied to the standard output. In order to avoid getting double copies of some lines in the standard output, the *p* command is ignored unless the **−n** flag is set.

The −**e** flag indicates that the next argument is an editor command.

The −**f** flag indicates that the next argument is a file name; the file contains editor commands, one to a line. Commands that are inherently multi-line, like *a* or *c*, should have the interior newlines escaped by '\'. Append, insert, and change modes are terminated by an non-escaped newline.

The −**e** and −**f** flags may be intermixed in any order.

If no −**e** or −**f** flags are given, the first argument is taken by default to be an editor command.

Addresses are allowed. The meaning of *two* addresses is: "Attempt this command on the first line that matches the first address, and on all subsequent lines up to and including the first subsequent line that matches the second address; then search for a match of the first address and iterate." *One* address means: "Attempt this command on all lines that match the address." Either line-numbers or regular expressions are allowed as addresses. Line numbers increase monotonically throughout *all* the input files, so that, if *n* is the number of the last line of the first input file, then *n+1* is the number of the first line of the second file, etc. A '$' as an address matches the *last* line of the *last* input file.

The intention is to simulate the editor as exactly as possible, but the line-at-a-time operation makes certain differences unavoidable or desirable:

1. There is no notion of '.' and no relative addressing.
2. Commands with no addresses are defaulted to *1,$* rather than to dot.
3. Addresses specified as regular expressions must be delimited by '/'; '?' is an error.
4. Expressions in addresses are not allowed (i.e., '+', '−').
5. Commands may have only as many addresses as they can use. That is, no command may have more than two addresses; the *a, i,* and *r* commands may have only one address.
6. A *p* at the end of a command only works with the *s* command. For other commands, or if the **−n** flag is not in effect, a *p* at the end of a command line is ignored.
7. A *w* may appear at the end of a *s* command. It should be followed by a single space and a file name. If the *s* command succeeds, the modified line is appended to the file. All files are opened when the commands are being compiled, and closed when the program terminates. Only ten distinct file names may appear in *w* commands in a sin-

gle execution of *sed.* Unlike *p, w* takes effect regardless of the **−n** flag. If both *p* and *w* are appended to the same substitute command, they must be in the order *pw.*

8. The only editor commands available are *a, c, d, i, s, p, q, r, w, g, v,* and =. A successful execution of a *q* command causes the current line to be written out if it should be, and execution terminated. When a line is deleted by a *d* or *c* command, no further commands are attempted on its corpse, but another line is immediately read from the input (but see item 10. below).

9. The *n*ext line command, *n,* replaces the current line by the next line from the input file. The list of editing commands is continued after the *n* command is executed.

10. If an *a, i,* or *r* command is successfully executed, the text is inserted into the standard output whether or not the line on which the match was made is later deleted or not. Thus the commands:

    ```
    /b/a\
    XXX
    /b/,/c/d
    ```

    applied to the file

    ```
    a
    b
    c
    d
    ```

    will produce

    ```
    a
    XXX
    d
    ```

    on the output.

11. Text inserted in the output stream by the *a, i, c,* or *r* commands is not scanned for any pattern matches, nor are any editor commands applied to it.

Sed supports three commands to control the flow of processing. These commands do no editing on the input line, but serve to control the order in which multiple editing commands are applied to an input line.

12. The label command, *: label,* marks a place in the list of editing commands which may be referred to by *j* and *t* commands (see 13. and 14. below); the *label* may be any sequence of eight or fewer characters; if two different colon commands have identical labels, a compile-time diagnostic will be generated and no execution attempted.

13. The *j*ump command, *j label,* causes the sequence of editing commands being applied to the current input line to be restarted immediately after the place where a colon command with the same *label* was encountered. If no colon command with the same label can be found after all editing commands have been compiled, a compile-time diagnostic is produced and no execution is attempted. A *j* command with no *label* is taken to be a jump to the end of the list of editing commands; whatever should be done with the current input line is done, and another input line is read; the list of editing commands is restarted from the beginning of that line.

14. The *t*est command, *t label,* tests whether *any* successful substitutions have been made on the current input line; if so, it jumps to *label;* if not, it does nothing. The flag that indicates that a successful substitution has occurred on the current input line is reset by either reading a new line or by executing the *t* command.

Sed also supports command grouping and several operations that can build lines into a pattern space to be operated upon by other commands.

15. Commands may be grouped by curly braces. The opening brace must appear in the place where a command would ordinarily appear; the closing brace must appear on a line by itself (except for leading blanks or tabs). If the first line of a command file has *#n* as its first two characters, the no-copy flag is set, as though the **−n** option had been given on the command line. The remainder of this first line is ignored and may be used for a title or a comment. As an example:

```
#n  Print first non-blank line after a blank line, and first line, if non-blank.
1{
        /./p
}
/^$/{
: loop
        n
        /./{
                p
                j
        }
        j loop
}
```

16. The *N*ext command, *N,* appends the next input line to the current line; the two lines are separated by a new-line character, that may be matched by '\n'.

17. The *D*elete command, *D,* deletes up to and including the first (leftmost) new-line in the current pattern space. If the pattern space becomes empty (the only new-line is at the end of the space), *D*elete reads another line from the input. The list of editing commands is restarted from the beginning.

18. The *P*rint command, *P,* prints on standard output up to and including the first new-line in the pattern space.

SEE ALSO
ed(I)

BUGS
Lines are silently truncated to a maximum length of 512 characters. The "plus", "range", and "through" regular expression operators ("+", "\{\}", "[−]") of *ed*(I) are not implemented in *sed.*