Contents

Name

sed - stream editor

Description

The sed utility is a stream editor that reads one or more text input files, makes editing changes under the control of an editing script, and writes the results to a single named output file or back to the original input files.

Limits

Operands

Input File(s):

One or more filenames (possibly containing DOS wild-card characters) naming the input file or files. Multiple filenames appearing in this field should be separated by space characters (e.g., "foo.bar *.txt x*).

Note. When multiple input files are specified, line numbering restarts at 1 for each file. This is different to the UNIX sed(1) command, which treats multiple files as a single input stream and logically concatenates them together (such that there is only one line number 1). If this effect is required with the WinXs tools, first concatenate the input files using the cat(1) filter and then process the output file as a single entity.

Output file (Optional):

A filename identifying the output file. The input files will be appended to this file (if specified), preserving its original contents. If this field is blank, the processed output will be written back to the original input file(s).

Current Directory:

Contains the name of the current working directory, which can be changed by double-clicking a directory name in the associated listbox. Drives can be changed by double-clicking a drive number; double-clicking a filename causes that name to be added to the "Input File(s)" field described above.

Suppress Default Output:

Normally each input line is written to the output file after all editing commands have been executed for that line. Checking this option suppresses that behaviour , such that the only lines sent to the output file will be those written explicitly by commands like 'p', 'P', 'l' and '='. This is equivalent to the '-n' command line option in the UNIX version of sed.

Development Mode:

After sed has successfully applied the editing script to all input lines, it will terminate and return control to the calling program (normally the Windows Program Manager or File Manager). Checking this option overrides that behaviour such that control is returned to the sed dialog, allowing the various dialog fields to be modified and the script to be re-run. When this mode is applied, 'cancel' must be selected to close the dialog.

Paste From...

Brings up the Windows Open dialog, which allows a text file containing a sed script to be opened and read into the "Script" field described below.

Copy To...

Copies the current contents of the "Script" field to a named output file.   The contents of this file can be reloaded at any time using the "Paste from..." facility described above.

Script:

This is a multi-line edit control that can be used to enter and modify the editing script.   Normal Windows key sequences apply in this control, e.g.,

```
Ctrl+C     Copy currently selected text (to the clipboard).
Ctrl+V     Paste text from the clipboard.
Ctrl+X     Cut currently selected text.
```

The script consists of editing commands, one per line, of the following form:

```
[addr[,addr]] command [arguments]
```

Zero or more blank characters are accepted before the first address and before the command.

In default operation, sed cyclically copies a line of text, less its terminating newline character, into a pattern space, applies in sequence all commands whose addresses select that pattern space, and at the end of the script copies the pattern space to the named output file and deletes the pattern space. Whenever the pattern space is written to the output file, sed will immediately follow it with a newline character.

Some of the commands also use a hold space to save all or part of the pattern space for subsequent retrieval.

Addresses
Regular Expression
Commands

Limits

The following numerical limits are imposed by the implementation:

Maximum size of pattern space	8192 bytes
Maximum size of hold space	8192 bytes
Maximum size of input line	512 bytes
Maximum size of file for 'r' command	32768 bytes
Maximum size of editing script	32768 bytes
Maximum no of lines in script	1024
Maximum no of input files	500
Maximum size of input file(s)	Unlimited
Maximum size of output file	Unlimited

Addresses

An address is either empty, a decimal number that counts input lines within each input file, a $ character that addresses the last line of an input file, or a context address (which consists of a regular expression preceded and followed by a delimiter, usually the slash character). A command line with no addresses selects every pattern space; a command line with one address selects each pattern space that matches the address.

A command line with two addresses selects the inclusive range from the first pattern space that matches the first address, to the next pattern space that matches the second address. Starting at the first line following such a range, sed looks again for the first address. Thereafter the process is repeated.

Editing commands can be applied only to non-selected pattern space by use of the negation command ! (see editing commands for more detaills). For example:

```
20          selects input line number 20
200,$       selects lines 200 to end-of-file
/^$/        selects all blank lines
/^$/!       selects all non-blank lines
/{/,/}/     select ranges starting with '{' and ending
            with '}'.
```

Sed supports standard regular expression syntax with the following additions:

- In a context address, the construction '\cREc', where 'c' is any character other than a backslash or newline character, is identical to '/RE/'. For example '\%foo%' is equivalent to '/foo/'.
- The escape sequences '\n' and '\t' match a newline character and a tab character respectively.

Regular Expression

Regular Expression

A regular expression is made up of ordinary characters and metacharacters. An ordinary character matches itself; for example, the re "aardvark" matches the character sequence <a><a><r><d><v><a><r><k> anywhere in an input line. A metacharacter or metacharacter sequence has special meaning as described below.

Metacharacters

. A dot (.) matches any character. For example, the re "ab." matches the character sequence <a><b><any-character>.

* An asterisk (*) matches the previous ordinary character, metacharacter or metacharacter sequence any number of times. For example, the re "ab.*" matches the character sequence <a><b><any-character...>.

\ A backslash (\) escapes the following character and is commonly used to escape the meaning of a metacharacter. For example, the re "ab\." matches the character sequence <a><b><.>.

\n This expression matches the same string of characters as was matched by the nth expression enclosed between the metacharacter sequence \(and \) (see below for details), appearing earlier in the same regular expression. For example, the re "^\(abc\)\1$" matches a line consisting of two repeated appearances of the string <a><b><c>.

^ A caret (^) anchors a regular expression at the start of an input line. For example, the re "^abc" will only match input lines that start with the character sequence <a><b><c>

$ A dollar ($) anchors a regular expression at the end of an input line. For example, the re "abc$" will only match input lines that end with the character sequence <a><b><c>.

Metacharacter Sequences

[...] A bracketed re is a metacharacter sequence that defines a scanset. A scanset defines single characters or character ranges that are matched against the next character in the input line. For example, the re "^[abc]" will match any input line that begins with the characters <a>, <b> or <c>. A character range is denoted by dash (-) separated character sequences; that is, the re "^[a-z]" matches any line beginning with a lower-case letter, the re "^[a-zA-Z]" matches any line beginning with a lower-case or an upper-case letter. Within scansets, metacharacters, other than a dash, lose their special meaning. A dash character appearing as the first or last character in a scanset loses its special meaning and matches itself. A caret (^) appearing as the first character in a scanset causes the scanset to be complemented; that is, the re will match characters NOT specified in the scanset.

\{m,n\} Is a range expression that matches the previous ordinary character or metacharacter a specified number of times. The values of m and n must be integers in the range 1-255. The expression \{m\} matches exactly m occurrences; \{m,\} matches at least m occurrences; \{m,n\} matches any number of occurrences between m and n inclusive. Wherever a choice exists, the re matches as many occurrences as possible.

\(re\) An re enclosed between \(and \) matches whatever the unadorned re matches. However, the enclosed re can also be used for repeating a match (see \n above).

Commands

In the following list of commands, the maximum number of permissible addresses for each command is indicated by [0addr], [1addr] or [2addr], representing zero, one or two addresses respectively. The argument 'text' consists of one or more lines, where each embedded newline character in the text must be preceded by a backslash; other backslashes in the input are ignored.

[2addr] { command-list }

Execute each command in the command-list when pattern space is matched by [2addr]. command-list is a list of sed commands seperated by newline characters as follows:

```
{
command
command
...
}
```

Each command may be preceded by blank characters (space or tab) and can be followed with white space. The terminating '}' character must be preceded by a newline character and (optionally) zero or more blank characters.

[1addr] a\
text

Write 'text' to the output file just before each attempt to fetch a line of input, whether by executing the 'N' command or by beginning a new cycle.

[2addr] b [label]

Branch to the ':' command bearing the 'label'. If 'label' is not specified, branch to the end of the script.

[2addr] c \
text

Delete the pattern space. With 0 or 1 address, or at the end of a 2 address range, write 'text' to the output file.

[2addr] d

Delete the pattern space and start the next cycle.

[2addr] D

Delete the initial segment of pattern space, up to and including the first embedded newline character, and restart the script from the beginning. This command behaves the same as 'd' if there is no embedded newline character.

[2addr] g

Replace the contents of pattern space by the contents of hold space.

[2addr] G

Append a newline character to pattern space, followed by the contents of hold space.

[2addr] h

Replace the contents of hold space with the contents of pattern space.

[2addr] H

Append a newline character to hold space, followed by the contents of pattern space.

[1addr] i \
text

Write 'text' to the output file immediately.

[2addr] l

The letter ell. Write pattern space to the output file in a visually unambiguous form. The characters \\, \a, \b, \f, \r, \t and \v will be written as the corresponding escape sequence (\n is not applicable). Other non-printable characters will be written as a 3-digit octal number with a preceding backslash.

[2addr] n

Write pattern space to the output file if 'suppress default output' is not selected, and replace pattern space with the next line of input.

[2addr] N

Append a newline character to pattern space followed by the next line of input. Note that the current line number changes.

[2addr] p

Write the contents of pattern space to the output file.

[2addr] P

Write the contents of pattern space, up to and including the first embedded newline character, to the output file. This command behaves like 'p' if there is no embedded newline character in pattern space.

[1addr] q

Branch to the end of the script and quit sed without starting a new cycle.

[1addr] r rfile

Copy the contents of 'rfile' to the output file just before attempting to read the next line of input. If 'rfile' does not exist or cannot be read, it is treated as if it were an empty file and no error occurs.

[2addr] s/regular-expression/replacement/flags

Substitute the 'replacement' string for instances of 'regular-expression' in pattern space. Any character other than backslash or '&' can be used instead of slash to delimit the various fields.

An ampersand (&) appearing in the 'replacement' string causes substitution of the string matching the 'regular-expression'. This special meaning of '&' can be suppressed by preceding it with a backslash. Newline and tab characters in 'replacement' can be specified by the escape sequences

'\n' and '\t' respectively; an occurrence of backslash followed by any other character is treated literally and the backslash itself is substituted.

The value of 'flags' can be zero or more of:

n Substitute the 'n'th occurrence of the regular expression found within pattern space, where 1 is the first such occurrence, 2 the second , and so on. (default 1).

g Globally substitute for all non-overlapping instances of the regular expression rather than just the first. If both 'g' and 'n' are specified, 'g' is ignored.

p Write the contents of pattern space to the output file if a replacement was made.

w wfile Append the contents of pattern space to 'wfile' if a replacement was made.

[2addr] t label

Test. Branch to the ':' command bearing 'label' if any substitutions have been made since the most recent reading of an input line or execution of a 't' command. Branch to the end of the script if 'label' is not specified.

[2addr] w wfile

Append pattern space to 'wfile'.

[2addr] x

Exchange the contents of pattern space and hold space.

[2addr] y/string1/string2/

Replace all occurrences of characters in 'string1' with the corresponding characters from 'string2'. Any character other than backslash or newline can be used instead of slash as a field delimiter.

[2addr] ! command

[2addr] ! { command-list }

Apply the command or command-list only to lines that are not selected by the addresses.

[0addr] : label

This command does nothing, other than bear a label for the 'b' and 't' commands.

[1addr] =

Write the number of the current line, followed by a newline character, to the output file.

[0addr]

An empty line is ignored.

[0addr]

Comment. The remainder of the line is ignored, with the single exception that if the first line of the script contains '#n', default output is suppressed (i.e., this is equivalent to checking the 'suppress default output' option).