Shell Tools

awk, *sed*, and *egrep* are a related set of Unix shell tools for text processing. *awk* uses a DFA match engine, *egrep* switches between a DFA and NFA match engine, depending on which features are being used, and *sed* uses an NFA engine. For an explanation of the rules behind these engines, see "Introduction to Regexes and Pattern Matching."

This reference covers GNU *egrep* 2.4.2, a program for searching lines of text; GNU *sed* 3.02, a tool for scripting editing commands; and GNU *awk* 3.1, a programming language for text processing.

Supported Metacharacters

awk, *egrep*, and *sed* support the metacharacters and metasequences listed in Table 61 through Table 65. For expanded definitions of each metacharacter, see "Regex Metacharacters, Modes, and Constructs."

Table 61. Shell character representations

Sequence	Meaning	Tool
`\a`	Alert (bell).	*awk, sed*
`\b`	Backspace; supported only in character class.	*awk*
`\f`	Form feed.	*awk, sed*
`\n`	Newline (line feed).	*awk, sed*
`\r`	Carriage return.	*awk, sed*
`\t`	Horizontal tab.	*awk, sed*
`\v`	Vertical tab.	*awk, sed*
`\o`*octal*	A character specified by a one-, two-, or three-digit octal code.	*sed*

Table 61. Shell character representations (continued)

Sequence	Meaning	Tool
octal	A character specified by a one-, two-, or three-digit octal code.	*awk*
\x*hex*	A character specified by a two-digit hexadecimal code.	*awk, sed*
\d*decimal*	A character specified by a one, two, or three decimal code.	*awk, sed*
\c*char*	A named control character (e.g., \cC is Control-C).	*awk, sed*
\b	Backspace.	*awk*
metacharacter	Escape the metacharacter, so that it literally represents itself.	*awk, sed, egrep*

Table 62. Shell character classes and class-like constructs

Class	Meaning	Tool
[...]	Matches any single character listed, or contained within a listed range.	*awk, sed, egrep*
[^...]	Matches any single character that is not listed, or contained within a listed range.	*awk, sed, egrep*
.	Matches any single character, except newline.	*awk, sed, egrep*
\w	Matches an ASCII word character, [a-zA-Z0-9_].	*egrep, sed*

Table 62. Shell character classes and class-like constructs (continued)

Class	Meaning	Tool
`\W`	Matches a character that is not an ASCII word character, `[^a-zA-Z0-9_]`.	*egrep, sed*
`[:prop:]`	Matches any character in the POSIX character class.	*awk, sed*
`[^[:prop:]]`	Matches any character not in the POSIX character class.	*awk, sed*

Table 63. Shell anchors and other zero-width testshell tools

Sequence	Meaning	Tool
`^`	Matches only start of string, even if newlines are embedded.	*awk, sed, egrep*
`$`	Matches only end of search string, even if newlines are embedded.	*awk, sed, egrep*
`\<`	Matches beginning of word boundary.	*egrep*
`\>`	Matches end of word boundary.	*egrep*

Table 64. Shell comments and mode modifiers

Modifier	Meaning	Tool
flag: `i` or `I`	Case-insensitive matching for ASCII characters.	*sed*
command-line option: `-i`	Case-insensitive matching for ASCII characters.	*egrep*
set `IGNORECASE` to *`non-zero`*	Case-insensitive matching for Unicode characters.	*awk*

Table 65. Shell grouping, capturing, conditional, and control

Sequence	Meaning	Tool
(*PATTERN*)	Grouping.	*awk*
\(*PATTERN*\)	Group and capture submatches, filling \1, \2,...,\9.	*sed*
n	Contains the *n*th earlier submatch.	*sed*
...|...	Alternation; match one or the other.	*egrep, awk, sed*
Greedy quantifiers		
*	Match 0 or more times.	*awk, sed, egrep*
+	Match 1 or more times.	*awk, sed, egrep*
?	Match 1 or 0 times.	*awk, sed, egrep*
\{*n*\}	Match exactly *n* times.	*sed, egrep*
\{*n*,\}	Match at least *n* times.	*sed, egrep*
\{*x*,*y*\}	Match at least *x* times, but no more than *y* times.	*sed, egrep*

egrep

egrep [*options*] *pattern files*

egrep searches *files* for occurrences of *pattern*, and prints out each matching line.

Example

```
$ echo 'Spiderman Menaces City!' > dailybugle.txt
$ egrep -i 'spider[- ]?man' dailybugle.txt
Spiderman Menaces City!
```

sed

sed '[*address1*][,*address2*]s/*pattern*/*replacement*/[*flags*]' *files*

sed -f *script files*

By default, *sed* applies the substitution to every line in *files*. Each address can be either a line number, or a regular expression pattern. A supplied regular expression must be defined within the forward slash delimiters (/.../).

If *address1* is supplied, substitution will begin on that line number, or the first matching line, and continue until either the end of the file, or the line indicated or matched by *address2*. Two subsequences, & and *n*, will be interpreted in *replacement* based on the match results.

The sequence & is replaced with the text matched by *pattern*. The sequence *n* corresponds to a capture group (1…9) in the current match. Here are the available flags:

n
: Substitute the *n*th match in a line, where *n* is between 1 and 512.

g
: Substitute all occurrences of *pattern* in a line.

p
: Print lines with successful substitutions.

w *file*
: Write lines with successful substitutions to *file*.

Example

Change date formats from *MM/DD/YYYY* to *DD.MM.YYYY*.

```
$ echo 12/30/1969' |
 sed 's!\([0-9][0-9]\)/\([0-9][0-9]\)/\([0-9]\{2,4\}\)!
\2.\1.\3!g'
```

awk

```
awk 'instructions' files
awk -f script files
```

The *awk* script contained in *instructions* or *script* should be a series of */pattern/* {*action*} pairs. The *action* code is applied to each line matched by *pattern*. *awk* also supplies several functions for pattern matching.

Functions

match(*text*, *pattern*)
: If *pattern* matches in *text*, return the position in *text* where the match starts. A failed match returns zero. A successful match also sets the variable RSTART to the position where the match started, and the variable RLENGTH to the number of characters in the match.

`gsub(`*`pattern`*`, `*`replacement`*`, `*`text`*`)`
: Substitute each match of *`pattern`* in *`text`* with *`replacement`*, and return the number of substitutions. Defaults to `$0` if *`text`* is not supplied.

`sub(`*`pattern`*`, `*`replacement`*`, `*`text`*`)`
: Substitute first match of *`pattern`* in *`text`* with *`replacement`*. A successful substitution returns 1, and an unsuccessful substitution returns 0. Defaults to `$0` if *`text`* is not supplied.

Example

Create an *awk* file and then run it from the command line.

```
$ cat sub.awk
{
    gsub(/https?:\/\/[a-z_.\\w\/\\#~:?+=&;%@!-]*/,
                   "<a href=\"\&\">\&</a>");

    print
}

$ echo "Check the web site, http://www.oreilly.com/
catalog/repr" | awk -f sub.awk
```

Other Resources

- *sed and awk*, by Dale Dougherty and Arnold Robbins (O'Reilly), is an introduction and reference to both tools.