

CHAPTER 24

Advanced UNIX Tools - Regular Expressions, sed, awk, and grep

Three Commands

The three commands covered in this chapter, along with regular expressions (pattern matching), are often grouped together. There are even many books available devoted to **awk** and **sed**. In these books, **grep** usually goes along for the ride because **awk** is derived to some extent from **sed** and **grep**. In addition, the use of regular expressions for pattern matching that are used for **awk**, **sed**, and **grep** are similar.

We'll take a look at regular expressions and pattern matching in general, and then cover the three commands in this chapter:

- Regular expressions
- **sed**, **awk**, and **grep** commands



Regular Expression Words-of-Caution

Regular expressions describe patterns for which you are searching. A regular expression usually defines the pattern for which you are searching using wildcards. Since a regular expression defines a pattern you are searching for, the terms "regular expression" and "pattern matching" are often used interchangeably.

Let's get down to a couple of words-of-caution immediately:



- **Regular expressions are different from file matching patterns used by the shell.** Regular expressions are used by both the shell and many other programs, including those covered in this chapter. The file matching done by the shell and programs such as **find** is different from the regular expressions covered in this chapter.
- **Use single quotes around regular expressions.** The meta-characters used in this chapter must be quoted in order to be passed to the shell as an argument. You will, therefore, see most regular expressions in this chapter quoted.

Expressions Are Strings and Wildcards



When using the programs in this book, such as **grep** and **vi**, you provide a regular expression that the program evaluates. The command will search for the pattern you supply. The pattern could be as simple as a string or it could wildcards. The wildcards used by many programs are called *meta-characters*.

Table 24-1 shows a list of meta-characters and the program(s) to which they apply. Only the programs covered in this book (**awk**, **grep**, **sed**, and **vi**) are shown in Table 24-1. These meta-characters may be used with other programs, such as **ed** and **egrep**, as well, which are not covered in the book. Table 24-1 describes the meta-characters and their use.

TABLE 24-1 Meta-Characters and Programs to Which They Apply

Meta-Character	awk	grep	sed	vi	Use
.	Yes	Yes	Yes	Yes	Match any single character.
*	Yes	Yes	Yes	Yes	Match any number of the single character that precedes *.
[...]	Yes	Yes	Yes	Yes	Match any <i>one</i> of the characters in the set [...].
\$	Yes	Yes	Yes	Yes	Matches the end of the line.
^	Yes	Yes	Yes	Yes	Matches the beginning of the line.
\	Yes	Yes	Yes	Yes	Escape the special character that follows \.
{ <i>n,m</i> }	Yes	Yes	No	No	Match a range of occurrences of a single character between <i>n</i> and <i>m</i> .
+	Yes	No	No	No	Match one or more occurrences of the preceding regular expression.
?	Yes	No	No	No	Match zero or one occurrence of the preceding regular expression.
	Yes	No	No	No	The preceding <u>or</u> following regular expression can be matched.
()	Yes	No	No	No	Groups regular expressions in a typical parenthetical fashion.
{\}	No	No	No	Yes	Match a word's beginning or end.

You may want to refer to this table when regular expressions are used for one of the commands in the table.

sed



Most of the editing performed on UNIX systems is done with **vi**. I have devoted a chapter to **vi** in this book, because of its prominence as a UNIX editor. Many times, we don't have the luxury of invoking **vi** when we need to edit a file. You may be writing a shell program or piping information between processes and need to edit in a non-interactive manner. **sed** can help here. Its name comes from *stream editor*, and it's a tool for filtering text files.

You can specify the name of the file you wish to edit with **sed** or it takes its input from standard input. **sed** reads one line at a time and performs the editing you specify to each line. You can specify line numbers for **sed** to edit as well.

sed uses many of the same commands as **ed**. You can view some of the **ed** commands in the **vi** chapter, and I also supply a summary of these at the end of this **sed** section.

You can invoke **sed** in the following two ways:

```
sed [-n][-e] 'command' filename(s)
sed [-n]-f scriptfile filename(s)
```

The first form of **sed** is for issuing commands on the command line. By default, **sed** will display all lines. The *-n* specifies that you want only to print lines that are specified with the **p** command.

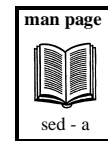
If you supply more than one instruction on the command line, then you *-e* is used to inform **sed** that the next argument is an instruction.

The second form allows you to specify one or more scripts containing editing commands.

The following is a summary of the three options that appear in the two different forms of **sed**:

<i>-n</i>	Print only lines that are specified with the p command.
<i>-e command</i>	The argument following <i>-e</i> is an editing command.
<i>-f filename</i>	The argument following <i>-f</i> is a file containing editing commands.

Let's view a couple of simple examples of what you can do with **sed**. These examples use some of the **sed** commands that appear at the end of this section. We'll use a file called **passwd.test**. We'll view this file with **cat** and then view only lines *16*, *17*, and *18* using the *p* option to **sed**, indicating that we want only the specified lines printed:



```
# cat passwd.test
root:PgYQckVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
#
# sed 16,18p passwd.test
root:PgYQckVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
```



```

gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
#
# sed -n 16,18p passwd.test
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash

```

The first attempt to print only lines 16, 17, and 18 results in all of the lines in the file being printed and lines 16, 17, and 18 being printed twice. The reason is that **sed** reads each line of input and acts on each line. In order to specify the lines on which to act, we used the **-n** switch to suppress all lines from going to standard output. We then specify the lines that we want to print and these will indeed go to standard output.

Now that we know how to view lines 16, 17, and 18 of the file, let's again view **passwd.test** and delete those same three lines with **d**:

```

root:PgYQckVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
#
# sed 16,18d passwd.test
root:PgYQckVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:

```

As with our earlier **grep** example, we enclose any special characters in single quotes to make sure that they are not interfered with and are passed directly to **sed** unmodified and uninterpreted by the shell. In this example, we specify the range of lines to delete, *16* through *18*, and the *d* for delete. We could specify just one line to delete, such as *16*, and not specify an entire range. Because we did not redirect the output as part of the **sed** command line, the result is sent to standard output. The original file remains intact.

We could search for a pattern in a file and delete only those lines containing the pattern. The following example shows searching for *bash* and deleting the lines that contain *bash*:

```
# cat passwd.test
root:PgYQckVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
#
# sed '/bash/ d' passwd.test
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
```

Both lines containing *bash* were deleted from **passwd.test** (the *root* line and the *col* line).



As I had mentioned earlier, it is a good idea to use single quotes around all regular expressions. In this example, I enclosed in single quotes the pattern for which I was searching and the command to execute.

What if you wanted to delete all lines except those that contain *bash*? You would insert an exclamation mark before the *d* to delete all lines except those that contain *bash*, as shown in the following example:

```
# cat passwd.test
root:PgYQcKvH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yZfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
#
# sed '/bash/ !d' passwd.test
root:PgYQcKvH65hyQ:0:0:root:/root:/bin/bash
col:Wh0yZfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
```

This resulted in all but the two lines containing *bash* to be deleted from *passwd.test*.

Now that we have seen how to display and delete specific lines of the file, let's see how to add three lines to the end of the file:

```
# sed '$a\
> This is a backup of passwd file\
> for viewing purposes only\
> so do not modify' passwd.test
root:PgYQcKvH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yZfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
This is a backup of passwd file
```

```
for viewing purposes only
so do not modify
```

The backslashes (\) are used liberally in this example. Each backslash represents a new line. We go to the end of the file, as designated by the \$, then we add a new line with the backslash, and then add the text we wish and a new line after the text. These lines are great to add to the end of the file, but we really should add them to the beginning of the file. The following example shows this approach:

```
# sed 'li\
> This is a backup passwd file\
> for viewing purposes only\
> so do not modify\
> ' passwd.test
This is a backup passwd file
for viewing purposes only
so do not modify
root:PgYQckVH65hyQ:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:11:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
man:*:15:15:Manuals Owner:/:
nobody:*:65534:65534:Nobody:/:/bin/false
col:Wh0yzfAV2qm2Y:100:100:Caldera OpenLinux User:/home/col:/bin/bash
```

First, we run **sed**, specifying that on the first line one we are going to begin inserting the text shown. We use the single quote immediately following **sed** and use another single quote on the last line when we are done specifying all the information, except for the input file, which is **passwd.test**.

We have only scratched the surface of commands you can use with **sed**. The following **sed** summary includes the commands we have used (*p* for print; *d* for delete; and *a* for add), as well as others that were not part of the examples.





sed - Stream editor.

Commands

- a Append text.
- b Branch to a label.
- c Replace lines with text.
- d Delete the current text buffer.
- D Delete the first line of the current text buffer.
- g Paste overwriting contents of the hold space.
- G Paste the hold space below the address rather than overwriting it.
- h Copy the pattern space into hold space.
- H Append the contents of pattern space into hold space.
- i Insert text.
- l List the contents of the pattern space.
- n Read the next line of input into the pattern space.
- N Append next line of input to pattern space.
- p Print the pattern space.
- P Print from the start of the pattern space up to and including new line.
- q Quit when address is encountered.
- r Read in a file.
- s Substitute patterns.
- t Branch if substitution has been made to the current pattern space.
- w Append the contents of the pattern space to the specified *file*.
- x Interchange the contents of the holding area and pattern space.
- y Translate characters.