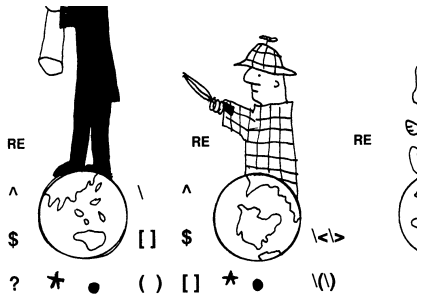
chapter

3

The *Grep* Family



The *grep* family consists of the commands *grep, egrep*, and *fgrep*. The *grep* command globally searches for regular expressions in files and prints all lines that contain the expression. The *egrep* and *fgrep* commands are simply variants of *grep*. The *egrep* command is an extended *grep*, supporting more RE metacharacters. The *fgrep* command, called *fixed grep,* and sometimes *fast grep*, treats all characters as literals; that is, regular expression metacharacters aren't special—they match themselves.

3.1 The *Grep* Command

3.1.1 The Meaning of *Grep*

The name *grep* can be traced back to the *ex* editor. If you invoked that editor and wanted to search for a string, you would type at the *ex* prompt:

```
: /pattern/p
```

The first line containing the string *pattern* would be printed as "*p*" by the *print* command. If you wanted all the lines that contained *pattern* to be printed, you would type:

```
:g/pattern/p
```

When *g* precedes *pattern*, it means "all lines in the file," or "perform a global substitution."

Because the search pattern is called a *regular expression*, we can substitute *RE* for *pattern* and the command reads:

```
: g/RE/p
```

And there you have it. The meaning of *grep* and the origin of its name. It means "globally search for the *r*egular *e*xpression (RE) and *p*rint out the line." The nice part of using *grep* is that you do not have to invoke an editor to perform a search, and you do not need to enclose the regular expression in forward slashes. It is much faster than using *ex* or *vi*.

3.1.2 How *Grep* Works

The *grep* command searches for a pattern of characters in a file or multiple files. If the pattern contains white space, it must be quoted. The pattern is either a quoted string or a single word[1], and all other words following it are treated as filenames. *Grep* sends its output to the screen and does not change or affect the input file in any way.

FORMAT

```
grep word filename filename
```

EXAMPLE 3.1

```
grep Tom /etc/passwd
```

EXPLANATION

Grep will search for the pattern *Tom* in a file called */etc/passwd*. If successful, the line from the file will appear on the screen; if the pattern is not found, there will be no output at all; and if the file is not a legitimate file, an error will be sent to the screen. If the pattern is found, *grep* returns an exit status of 0, indicating success; if the pattern is not found, the exit status returned is 1; and if the file is not found, the exit status is 2.

The *grep* program can get its input from a standard input or a pipe, as well as from files. If you forget to name a file, *grep* will assume it is getting input from standard input, the keyboard, and will stop until you type something. If coming from a pipe, the output of a command will be piped as input to the *grep* command, and if a desired pattern is matched, *grep* will print the output to the screen.

1. A word is also called a token.

EXAMPLE 3.2

```
% ps -ef | grep root
```

EXPLANATION

The output of the *ps* command (ps -ef displays all processes running on this system) is sent to *grep* and all lines containing *root* are printed.

The *grep* command supports a number of regular expression metacharacters (see Table 3.1) to help further define the search pattern. It also provides a number of options (see Table 3.2) to modify the way it does its search or displays lines. For example, you can provide options to turn off case-sensitivity, display line numbers, display errors only, and so on.

EXAMPLE 3.3

```
% grep -n  '^jack:' /etc/passwd
```

EXPLANATION

Grep searches the */etc/passwd* file for *jack*; if *jack* is at the beginning of a line, *grep* prints out the number of the line on which *jack* was found and where in the line *jack* was found.

Table 3.1 *Grep*'s Regular Expression Metacharacters

Metacharacter	***Function***	***Example***	***What It Matches***
^	Beginning of line anchor	'^love'	Matches all lines beginning with *love*.
$	End of line anchor	'love$'	Matches all lines ending with *love*.
.	Matches one character	'l..e'	Matches lines containing an *l*, followed by two characters, followed by an *e*.
*	Matches zero or more characters	' *love'	Matches lines with zero or more spaces, of the preceding characters followed by the pattern *love*.
[ ]	Matches one character in the set	'[Ll]ove'	Matches lines containing *love* or *Love*.
[^]	Matches one character not in the set	'[^A–K]ove'	Matches lines not containing *A* through *K* followed by *ove*.
\<	Beginning of word anchor	'\<love'	Matches lines containing a word that begins with *love*.
\>	End of word anchor	'love\>'	Matches lines containing a word that ends with *love*.
\(..\)	Tags matched characters	'\(love\)ing'	Tags marked portion in a register to be remembered later as number 1. To reference later, use *\1* to repeat the pattern. May use up to nine tags, starting with the first tag at the leftmost part of the pattern. For example, the pattern *love* is saved in register 1 to be referenced later as *\1*.
x\{m\} x\{m,\} x\{m,n\}[a]	Repetition of character x, m times, at least m times, or between m and n times	'o\{5\}' 'o\{5,\}' 'o\{5,10\}'	Matches if line has 5 *o*'s, at least 5 *o*'s, or between 5 and 10 *o*'s

a. The \{ \} metacharacters are not supported on all versions of UNIX or all pattern-matching utilities; they usually work with *vi* and *grep*.

Table 3.2 *Grep*'s Options

Option	*What It Does*
–b	Precedes each line by the block number on which it was found. This is sometimes useful in locating disk block numbers by context.
–c	Displays a count of matching lines rather than displaying the lines that match.
–h	Does not display filenames.
–i	Ignores the case of letters in making comparisons (i.e., upper- and lowercase are considered identical).
–l	Lists only the names of files with matching lines (once), separated by newline characters.
–n	Precedes each line by its relative line number in the file.
–s	Works silently, that is, displays nothing except error messages. This is useful for checking the exit status.
–v	Inverts the search to display only lines that do not match.
–w	Searches for the expression as a word, as if surrounded by \< and \>. This applies to *grep* only. (Not all versions of *grep* support this feature; e.g., SCO UNIX does not.)

3.1.3 *Grep* and Exit Status

The *grep* command is very useful in shell scripts, because it always returns an exit status to indicate whether it was able to locate the pattern or the file you were looking for. If the pattern is found, *grep* returns an exit status of 0, indicating success; if *grep* cannot find the pattern, it returns 1 as its exit status; and if the file cannot be found, *grep* returns an exit status of 2. (Other UNIX utilities that search for patterns, such as *sed* and *awk*, do not use the exit status to indicate the success or failure of locating a pattern; they report failure only if there is a syntax error in a command.)

In the following example, *john* is not found in the */etc/passwd* file.

EXAMPLE 3.4

```
1  % grep 'john' /etc/passwd
2  % echo $status (csh)
   1
   or
   $ echo $?  (sh, ksh)
   1
```

EXPLANATION

1 *Grep* searches for *john* in the */etc/passwd* file, and if successful, *grep* exits with a status of 0. If *john* is not found in the file, *grep* exits with 1. If the file is not found, an exit status of 2 is returned.

2 The C shell variable, *status*, and the Bourne/Korn shell variable, ?, are assigned the exit status of the last command that was executed.

3.2 *Grep* Examples with Regular Expressions

The file being used for these examples is called *datafile*.

```
% cat datafile
northwest     NW    Charles Main     3.0    .98    3    34
western       WE    Sharon Gray      5.3    .97    5    23
southwest     SW    Lewis Dalsass    2.7    .8     2    18
southern      SO    Suan Chin        5.1    .95    4    15
southeast     SE    Patricia Hemenway 4.0   .7     4    17
eastern       EA    TB Savage        4.4    .84    5    20
northeast     NE    AM Main Jr.      5.1    .94    3    13
north         NO    Margot Weber     4.5    .89    5     9
central       CT    Ann Stephens     5.7    .94    5    13
```

EXAMPLE 3.5

```
grep NW datafile
northwest    NW    Charles Main    3.0    .98    3    34
```

EXPLANATION

Prints all lines containing the regular expression *NW* in a file called *datafile*.

EXAMPLE 3.6

```
grep NW d*
datafile: northwest     NW          Charles Main  3.0  .98  3    34
db:northwest            NW          Joel Craig    30   40   5    123
```

EXPLANATION

Prints all lines containing the regular expression *NW* in all files starting with a *d*. The shell expands *d** to all files that begin with a *d*, in this case the filenames are *db* and *datafile*.

EXAMPLE 3.7

```
grep '^n' datafile
northwest       NW     Charles Main     3.0  .98   3   34
northeast       NE     AM Main Jr.      5.1  .94   3   13
north           NO     Margot Weber     4.5  .89   5    9
```

EXPLANATION

Prints all lines beginning with an *n*. The caret (^) is the beginning of line anchor.

EXAMPLE 3.8

```
grep '4$' datafile
northwest       NW     Charles Main     3.0  .98   3   34
```

EXPLANATION

Prints all lines ending with a *4*. The dollar sign ($) is the end of line anchor.

EXAMPLE 3.9

```
grep TB Savage datafile
grep: Savage: No such file or directory
datafile:eastern   EA   TB Savage       4.4  .84   5   20
```

EXPLANATION

Since the first argument is the pattern and all of the remaining arguments are filenames, *grep* will search for *TB* in a file called *Savage* and a file called *datafile*. To search for *TB Savage*, see the next example.

EXAMPLE 3.10

```
grep 'TB Savage' datafile
eastern         EA     TB Savage        4.4  .84  5    20
```

EXPLANATION

Prints all lines containing the pattern *TB Savage*. Without quotes (in this example, either single or double quotes will do), the white space between *TB* and *Savage* would cause *grep* to search for *TB* in a file called *Savage* and a file called *datafile*, as in the previous example.

```
% cat datafile
northwest       NW      Charles Main        3.0     .98     3       34
western         WE      Sharon Gray         53      .97     5       23
southwest       SW      Lewis Dalsass       2.7     .8      2       18
southern        SO      Suan Chin           5.1     .95     4       15
southeast       SE      Patricia Hemenway   4.0     .7      4       17
eastern         EA      TB Savage           4.4     .84     5       20
northeast       NE      AM Main Jr.         5.1     .94     3       13
north           NO      Margot Weber        4.5     .89     5       9
central         CT      Ann Stephens        5.7     .94     5       13
```

EXAMPLE 3.11

```
grep '5\..' datafile
western         WE      Sharon Gray     5.3  .97   5   23
southern        SO      Suan Chin       5.1  .95   4   15
northeast       NE      AM Main Jr.     5.1  .94   3   13
central         CT      Ann Stephens    5.7  .94   5   13
```

EXPLANATION

Prints a line containing the number 5, followed by a literal period and any single character. The "dot" metacharacter represents a single character, unless it is escaped with a backslash. When escaped, the character is no longer a special metacharacter, but represents itself, a literal period.

EXAMPLE 3.12

```
grep '\.5' datafile
north           NO      Margot Weber    4.5  .89  5    9
```

EXPLANATION

Prints any line containing the expression *.5*.

EXAMPLE 3.13

```
grep '^[we]' datafile
western         WE      Sharon Gray     5.3  .97   5   23
eastern         EA      TB Savage       4.4  .84   5   20
```

EXPLANATION

Prints lines beginning with either a *w* or an *e*. The caret (^) is the beginning of line anchor, and either one of the characters in the brackets will be matched.

EXAMPLE 3.14

```
grep '[^0-9]' datafile
northwest       NW     Charles Main         3.0  .98  3    34
western         WE     Sharon Gray          5.3  .97  5    23
southwest       SW     Lewis Dalsass        2.7  .8   2    18
southern        SO     Suan Chin            5.1  .95  4    15
southeast       SE     Patricia Hemenway    4.0  .7   4    17
eastern         EA     TB Savage            4.4  .84  5    20
northeast       NE     AM Main Jr.          5.1  .94  3    13
north           NO     Margot Weber         4.5  .89  5    9
central         CT     Ann Stephens         5.7  .94  5    13
```

EXPLANATION

Prints all lines containing one non-digit. Because all lines have at least one non-digit, all lines are printed. (See the *-v* option.)

EXAMPLE 3.15

```
grep '[A-Z][A-Z] [A-Z]' datafile
eastern         EA     TB Savage        4.4  .84   5   20
northeast       NE     AM Main Jr.      5.1  .94   3   13
```

EXPLANATION

Prints all lines containing two capital letters followed by a space and a capital letter, e.g., *TB Savage* and *AM Main*.

EXAMPLE 3.16

```
grep 'ss* ' datafile
northwest       NW     Charles Main       3.0 .98  3  34
southwest       SW     Lewis Dalsass      2.7 .8   2  18
```

EXPLANATION

Prints all lines containing an *s* followed by zero or more consecutive *s*'s and a space. Finds *Charles* and *Dalsass*.

```
% cat datafile
northwest       NW     Charles Main       3.0  .98  3  34
western         WE     Sharon Gray        53   .97  5  23
southwest       SW     Lewis Dalsass      2.7  .8   2  18
southern        SO     Suan Chin          5.1  .95  4  15
southeast       SE     Patricia Hemenway  4.0  .7   4  17
eastern         EA     TB Savage          4.4  .84  5  20
northeast       NE     AM Main Jr.        5.1  .94  3  13
north           NO     Margot Weber       4.5  .89  5  9
central         CT     Ann Stephens       5.7  .94  5  13
```

EXAMPLE 3.17

```
grep '[a-z]\{9\}' datafile
northwest       NW     Charles Main       3.0  .98  3  34
southwest       SW     Lewis Dalsass      2.7  .8   2  18
southeast       SE     Patricia Hemenway  4.0  .7   4  17
northeast       NE     AM Main Jr.        5.1  .94  3  13
```

EXPLANATION

Prints all lines where there are at least nine consecutive lowercase letters, for example, *northwest, southwest, southeast*, and *northeast*.

EXAMPLE 3.18

```
grep '\(3\)\.[0-9].*\1    *\1' datafile
northwest       NW     Charles Main       3.0 .98   3 34
```

EXPLANATION

Prints the line if it contains a 3 followed by a period and another number, followed by any number of characters (.*), another 3 (originally tagged), any number of tabs, and another 3. Since the 3 was enclosed in parentheses, \(3\), it can be later referenced with *\1*. *\1* means that this was the first expression to be tagged with the \(\) pair.

EXAMPLE 3.19

```
grep '\<north' datafile
northwest       NW     Charles Main         3.0  .98   3   34
northeast       NE     AM Main Jr.          5.1  .94   3   13
north           NO     Margot Weber         4.5  .89   5    9
```

EXPLANATION

Prints all lines containing a word starting with *north*. The \< is the beginning of word anchor.

EXAMPLE 3.20

```
grep '\<north\>' datafile
north           NO     Margot Weber         4.5  .89  5    9
```

EXPLANATION

Prints the line if it contains the word *north*. The \< is the beginning of word anchor, and the \> is the end of word anchor.

EXAMPLE 3.21

```
grep '\<[a-z].*n\>' datafile
northwest       NW     Charles Main     3.0  .98   3   34
western         WE     Sharon Gray      5.3  .97   5   23
southern        SO     Suan Chin        5.1  .95   4   15
eastern         EA     TB Savage        4.4  .84   5   20
northeast       NE     AM Main Jr.      5.1  .94   3   13
central         CT     Ann Stephens     5.7  .94   5   13
```

EXPLANATION

Prints all lines containing a word starting with a lowercase letter, followed by any number of characters, and a word ending in *n*. Watch the .* symbol. It means any character, including white space.

3.3 *Grep* with Pipes

Instead of taking its input from a file, *grep* often gets its input from a pipe.

EXAMPLE 3.22

```
%  ls -l
drwxrwxrwx  2  ellie    2441 Jan 6 12:34  dir1
-rw-r--r--  1  ellie    1538 Jan 2 15:50  file1
-rw-r--r--  1  ellie    1539 Jan 3 13:36  file2
drwxrwxrwx  2  ellie    2341 Jan 6 12:34  grades

% ls -l | grep '^d'
drwxrwxrwx  2  ellie    2441 Jan 6 12:34  dir1
drwxrwxrwx  2  ellie    2341 Jan 6 12:34  grades
```

EXPLANATION

The output of the *ls* command is piped to *grep*. All lines of output that begin with a *d* are printed; that is, all directories are printed.

3.4 *Grep* with Options

The *grep* command has a number of options that control its behavior. Not all versions of UNIX support exactly the same options, so be sure to check your man pages for a complete list.

% **cat datafile**

northwest	*NW*	*Charles Main*	*3.0*	*.98*	*3*	*34*
western	*WE*	*Sharon Gray*	*53*	*.97*	*5*	*23*
southwest	*SW*	*Lewis Dalsass*	*2.7*	*.8*	*2*	*18*
southern	*SO*	*Suan Chin*	*5.1*	*.95*	*4*	*15*
southeast	*SE*	*Patricia Hemenway*	*4.0*	*.7*	*4*	*17*
eastern	*EA*	*TB Savage*	*4.4*	*.84*	*5*	*20*
northeast	*NE*	*AM Main Jr.*	*5.1*	*.94*	*3*	*13*
north	*NO*	*Margot Weber*	*4.5*	*.89*	*5*	*9*
central	*CT*	*Ann Stephens*	*5.7*	*.94*	*5*	*13*

EXAMPLE 3.23

```
grep –n '^south' datafile
3:southwest     SW     Lewis Dalsass        2.7  .8   2    18
4:southern      SO     Suan Chin            5.1  .95  4    15
5:southeast     SE     Patricia Hemenway    4.0  .7   4    17
```

EXPLANATION

The *-n* option precedes each line with the number of the line where the pattern was found, followed by the line.

EXAMPLE 3.24

```
grep –i 'pat' datafile
southeast       SE     Patricia Hemenway    4.0    .7  4   17
```

EXPLANATION

The *-i* option turns off case-sensitivity. It does not matter if the expression *pat* contains any combination of upper- or lowercase letters.

EXAMPLE 3.25

```
grep –v 'Suan Chin' datafile
northwest       NW     Charles Main         3.0  .98  3    34
western         WE     Sharon Gray          5.3  .97  5    23
southwest       SW     Lewis Dalsass        2.7  .8   2    18
southeast       SE     Patricia Hemenway    4.0  .7   4    17
eastern         EA     TB Savage            4.4  .84  5    20
northeast       NE     AM Main Jr.          5.1  .94  3    13
north           NO     Margot Weber         4.5  .89  5     9
central         CT     Ann Stephens         5.7  .94  5    13
```

EXPLANATION

Prints all lines *not* containing the pattern *Suan Chin*. This option is used when deleting a specific entry from the input file. To really remove the entry, you would redirect the output of *grep* to a temporary file, and then change the name of the temporary file back to the name of the original file as shown here:

```
grep -v 'Suan Chin' datafile > temp
mv temp datafile
```

Remember that you must use a temporary file when redirecting the output from *datafile*. If you redirect from *datafile* to *datafile*, the shell will "clobber" the *datafile*. (See "Redirection" on page 16.)

```
% cat datafile
northwest       NW      Charles Main    3.0     .98     3       34
western         WE      Sharon Gray     53      .97     5       23
southwest       SW      Lewis Dalsass   2.7     .8      2       18
southern        SO      Suan Chin       5.1     .95     4       15
southeast       SE      Patricia Hemenway 4.0   .7      4       17
eastern         EA      TB Savage       4.4     .84     5       20
northeast       NE      AM Main Jr.     5.1     .94     3       13
north           NO      Margot Weber    4.5     .89     5       9
central         CT      Ann Stephens    5.7     .94     5       13
```

EXAMPLE 3.26

```
grep –l 'SE'  *
datafile
datebook
```

EXPLANATION

The *-l* option causes *grep* to print out only the filenames where the pattern is found instead of the line of text.

EXAMPLE 3.27

```
grep –c 'west' datafile
3
```

EXPLANATION

The *-c* option causes *grep* to print the number of lines where the pattern was found. This does not mean the number of occurrences of the pattern. For example, if *west* is found three times on a line, it only counts the line once.

EXAMPLE 3.28

```
grep  –w 'north' datafile
north           NO      Margot Weber    4.5  .89   5   9
```

EXPLANATION

The *-w* option causes *grep* to find the pattern only if it is a word,[2] not part of a word. Only the line containing the word *north* is printed, not *northwest*, *northeast*, and so forth.

EXAMPLE 3.29

```
echo $LOGNAME
lewis
grep -i "$LOGNAME" datafile
southwest       SW     Lewis Dalsass    2.7 .8   2    18
```

EXPLANATION

The value of the shell ENV variable, *LOGNAME*, is printed. It contains the user's login name. If the variable is enclosed in double quotes, it will still be expanded by the shell, and in case there is more than one word assigned to the variable, white space is shielded from shell interpretation. If single quotes are used, variable substitution does not take place; that is, *$LOGNAME* is printed.

3.4.1 *Grep* Review

Table 3.3 contains examples of *grep* commands and what they do.

Table 3.3 Review of *Grep*

Grep Command	***What It Does***
grep '\<Tom\>' file	Prints lines containing the word *Tom*.
grep 'Tom Savage' file	Prints lines containing *Tom Savage*.
grep '^Tommy' file	Prints lines if *Tommy* is at the beginning of the line.
grep '\.bak$' file	Prints lines ending in *.bak*. Single quotes protect the dollar sign ($) from interpretation.
grep '[Pp]yramid' *	Prints lines from all files containing *pyramid* or *Pyramid* in the current working directory.
grep '[A–Z]' file	Prints lines containing at least one capital letter.
grep '[0–9]' file	Prints lines containing at least one number.
grep '[A–Z]...[0–9]' file	Prints lines containing five-character patterns starting with a capital letter and ending with a number.
grep –w '[tT]est' files	Prints lines with the word *Test* and/or *test*.
grep –s "Mark Todd" file	Finds lines containing *Mark Todd*, but does not print the line. Can be used when checking *grep*'s exit status.

2. A word is a sequence of alphanumeric characters starting at the beginning of a line or preceded by white space and ending in white space, punctuation, or a newline.

Table 3.3 Review of *Grep (Continued)*

Grep Command	***What It Does***
grep –v 'Mary' file	Prints all lines NOT containing *Mary*.
grep –i 'sam' file	Prints all lines containing *sam*, regardless of case (e.g., *SAM, sam, SaM, sAm*).
grep –l 'Dear Boss' *	Lists all filenames containing *Dear Boss*.
grep –n 'Tom' file	Precedes matching lines with line numbers.
grep "$name" file	Expands the value of variable *name* and prints lines containing that value. Must use double quotes.
grep '$5' file	Prints lines containing literal $5. Must use single quotes.
ps –ef| grep "^ *user1"	Pipes output of *ps –ef* to *grep*, searching for *user1* at the beginning of a line, even if it is preceded by zero or more spaces.

3.5 *Egrep* (Extended *Grep*)

The main advantage of using *egrep* is that additional regular expression metacharacters (see Table 3.4) have been added to the set provided by *grep*. The \(\) and \{ \}, however, are not allowed.

Table 3.4 *Egrep*'s Regular Expression Metacharacters

Metacharacter	***Function***	***Example***	***What It Matches***
^	Beginning of line anchor	'^love'	Matches all lines beginning with *love*.
$	End of line anchor	'love$'	Matches all lines ending with *love*.
.	Matches one character	'l..e'	Matches lines containing an *l*, followed by two characters, followed by an *e*.

Table 3.4 *Egrep*'s Regular Expression Metacharacters (Continued)

Metacharacter	***Function***	***Example***	***What It Matches***
*	Matches zero or more characters	'*love'	Matches lines with zero or more spaces, of the preceding characters followed by the pattern *love*.
[ ]	Matches one character in the set	'[Ll]ove'	Matches lines containing *love* or *Love*.
[^]	Matches one character not in the set	'[^A–KM–Z]ove'	Matches lines not containing *A* through *K* or *M* through *Z*, followed by *ove*.
***New with* Egrep**			
+	Matches one or more of the preceding characters	'[a–z]+ove'	Matches one or more lowercase letters, followed by *ove*. Would find *move*, *approve*, *love*, *behoove*, etc.
?	Matches zero or one of the preceding characters	'lo?ve'	Matches for an *l* followed by either one or not any *o*'s at all. Would find *love* or *lve*.
a|b	Matches either a or b	'love|hate'	Matches for either expression, *love* or *hate*.
()	Groups characters	'love(able|ly) (ov)+'	Matches for *lovable* or *lovely*. Matches for one or more occurrences of *ov*.

3.5.1 *Egrep* Examples

The following example illustrates only the way the new extended set of regular expression metacharacters is used with *egrep*. The *grep* examples presented earlier illustrate the use of the standard metacharacters, which behave the same way with *egrep*. *Egrep* also uses the same options at the command line as *grep*.

```
% cat datafile
northwest       NW     Charles Main     3.0  .98   3   34
western         WE     Sharon Gray      53   .97   5   23
southwest       SW     Lewis Dalsass    2.7  .8    2   18
southern        SO     Suan Chin        5.1  .95   4   15
southeast       SE     Patricia Hemenway 4.0 .7    4   17
eastern         EA     TB Savage        4.4  .84   5   20
northeast       NE     AM Main Jr.      5.1  .94   3   13
north           NO     Margot Weber     4.5  .89   5    9
central         CT     Ann Stephens     5.7  .94   5   13
```

EXAMPLE 3.30

```
egrep 'NW|EA' datafile
northwest       NW     Charles Main     3.0  .98   3   34
eastern         EA     TB Savage        4.4  .84   5   20
```

EXPLANATION

Prints the line if it contains either the expression *NW* or the expression *EA*.

EXAMPLE 3.31

```
egrep '3+' datafile
northwest       NW     Charles Main     3.0  .98  3    34
western         WE     Sharon Gray      5.3  .97  5    23
northeast       NE     AM Main          5.1  .94  3    13
central         CT     Ann Stephens     5.7  .94  5    13
```

EXPLANATION

Prints all lines containing one or more *3*'s.

EXAMPLE 3.32

```
egrep '2\.?[0-9]' datafile
western         WE     Sharon Gray          5.3  .97   5   23
southwest       SW     Lewis Dalsass        2.7  .8    2   18
eastern         EA     TB Savage            4.4  .84   5   20
```

EXPLANATION

Prints all lines containing a 2, followed by zero or one period, followed by a number.

EXAMPLE 3.33

```
egrep '(no)+' datafile
northwest       NW     Charles Main     3.0  .98  3    34
northeast       NE     AM Main          5.1  .94  3    13
north           NO     Margot Weber     4.5  .89  5     9
```

EXPLANATION

Prints lines containing one or more consecutive occurrences of the pattern group *no*.

EXAMPLE 3.34

```
egrep 'S(h|u)' datafile
western         WE     Sharon Gray      5.3  .97   5   23
southern        SO     Suan Chin        5.1  .95   4   15
```

EXPLANATION

Prints all lines containing *S*, followed by either *h* or *u*.

EXAMPLE 3.35

```
egrep 'Sh|u' datafile
western         WE     Sharon Gray          5.3  .97  5    23
southern        SO     Suan Chin            5.1  .95  4    15
southwest       SW     Lewis Dalsass        2.7  .8   2    18
southeast       SE     Patricia Hemenway    4.0  .7   4    17
```

EXPLANATION

Prints all lines containing the expression *Sh* or *u*.

3.5.2 *Egrep* Review

Table 3.5 contains examples of *egrep* commands and what they do.

Table 3.5 Review of *Egrep*[a]

Egrep *Command*	***What It Does***
egrep '^ +' file	Prints lines beginning with one or more spaces.
* egrep '^ *' file	Prints lines beginning with zero or more spaces.
egrep '(Tom | Dan) Savage' file	Prints lines containing *Tom Savage* or *Dan Savage*.

Table 3.5 Review of *Egrep*[a] *(Continued)*

Egrep *Command*	***What It Does***
egrep '(ab)+' file	Prints lines with one or more *ab*'s.
egrep '^X[0–9]?' file	Prints lines beginning with *X* followed by zero or one single digit.
* egrep 'fun\.$' *	Prints lines ending in *fun.* from all files.
egrep '[A–Z]+' file	Prints lines containing one or more capital letters.
* egrep '[0–9]' file	Prints lines containing a number.
* egrep '[A–Z]...[0–9]' file	Prints lines containing five-character patterns starting with a capital letter, followed by three of any character, and ending with a number.
* egrep '[tT]est' files	Prints lines with *Test* and/or *test*.
* egrep "Susan Jean" file	Prints lines containing *Susan Jean*.
* egrep –v 'Mary' file	Prints all lines NOT containing *Mary.*
* egrep –i 'sam' file	Prints all lines containing *sam*, regardless of case (e.g., *SAM, sam, SaM, sAm,* etc.).
* egrep –l 'Dear Boss' *	Lists all filenames containing *Dear Boss*.
* egrep –n 'Tom' file	Precedes matching lines with line numbers.
* egrep –s "$name" file	Expands variable name, finds it, but prints nothing. Can be used to check the exit status of *egrep*.

a. The asterisk preceding the command indicates that both *egrep* and *grep* handle the pattern in the same way.

3.6 Fixed *Grep* or Fast *Grep*

The *fgrep* command behaves like *grep*, but does not recognize any regular expression metacharacters as being special. All characters represent only themselves. A caret is simply a caret, a dollar sign is a dollar sign, and so forth.

EXAMPLE 3.36

```
% fgrep '[A-Z]****[0-9]..$5.00'  file
```

EXPLANATION

Finds all lines in the file containing the literal string *[A-Z]****[0-9]..$5.00*. All characters are treated as themselves. There are no special characters.

UNIX TOOLS LAB 1

Grep Exercise

Steve Blenheim:238-923-7366:95 Latham Lane, Easton, PA 83755:11/12/56:20300
Betty Boop:245-836-8357:635 Cutesy Lane, Hollywood, CA 91464:6/23/23:14500
Igor Chevsky:385-375-8395:3567 Populus Place, Caldwell, NJ 23875:6/18/68:23400
Norma Corder:397-857-2735:74 Pine Street, Dearborn, MI 23874:3/28/45:245700
Jennifer Cowan:548-834-2348:583 Laurel Ave., Kingsville, TX 83745:10/1/35:58900
Jon DeLoach:408-253-3122:123 Park St., San Jose, CA 04086:7/25/53:85100
Karen Evich:284-758-2857:23 Edgecliff Place, Lincoln, NB 92743:7/25/53:85100
Karen Evich:284-758-2867:23 Edgecliff Place, Lincoln, NB 92743:11/3/35:58200
Karen Evich:284-758-2867:23 Edgecliff Place, Lincoln, NB 92743:11/3/35:58200
Fred Fardbarkle:674-843-1385:20 Parak Lane, DeLuth, MN 23850:4/12/23:780900
Fred Fardbarkle:674-843-1385:20 Parak Lane, DeLuth, MN 23850:4/12/23:780900
Lori Gortz:327-832-5728:3465 Mirlo Street, Peabody, MA 34756:10/2/65:35200
Paco Gutierrez:835-365-1284:454 Easy Street, Decatur, IL 75732:2/28/53:123500
Ephram Hardy:293-259-5395:235 CarltonLane, Joliet, IL 73858:8/12/20:56700
James Ikeda:834-938-8376:23445 Aster Ave., Allentown, NJ 83745:12/1/38:45000
Barbara Kertz:385-573-8326:832 Ponce Drive, Gary, IN 83756:12/1/46:268500
Lesley Kirstin:408-456-1234:4 Harvard Square, Boston, MA 02133:4/22/62:52600
William Kopf:846-836-2837:6937 Ware Road, Milton, PA 93756:9/21/46:43500
Sir Lancelot:837-835-8257:474 Camelot Boulevard, Bath, WY 28356:5/13/69:24500
Jesse Neal:408-233-8971:45 Rose Terrace, San Francisco, CA 92303:2/3/36:25000
Zippy Pinhead:834-823-8319:2356 Bizarro Ave., Farmount, IL 84357:1/1/67:89500
Arthur Putie:923-835-8745:23 Wimp Lane, Kensington, DL 38758:8/31/69:126000
Popeye Sailor:156-454-3322:945 Bluto Street, Anywhere, USA 29358:3/19/35:22350
Jose Santiago:385-898-8357:38 Fife Way, Abilene, TX 39673:1/5/58:95600
Tommy Savage:408-724-0140:1222 Oxbow Court, Sunnyvale, CA 94087:5/19/66:34200
Yukio Takeshida:387-827-1095:13 Uno Lane, Ashville, NC 23556:7/1/29:57000
Vinh Tranh:438-910-7449:8235 Maple Street, Wilmington, VM 29085:9/23/63:68900

(*Refer to the database called* datebook *on the CD.*)

1. Print all lines containing the string *San*.
2. Print all lines where the person's first name starts with *J*.
3. Print all lines ending in *700*.
4. Print all lines that don't contain *834*.
5. Print all lines where birthdays are in *December*.
6. Print all lines where the phone number is in the *408* area code.
7. Print all lines containing an uppercase letter, followed by four lowercase letters, a comma, a space, and one uppercase letter.
8. Print lines where the last name begins with *K* or *k*.
9. Print lines preceded by a line number where the salary is a six-figure digit.
10. Print lines containing *Lincoln* or *lincoln* and *grep* is insensitive to case.