Ch. 7 Utilities

Filtering patterns: egrep, fgrep, grep

```
grep -hilnvw pattern {fileName}*
```

displays lines from files that match the pattern

pattern : regular expression

```
-h : do not list file names if many files are specified
-i : ignore case
-l : displays list of files containing pattern
-n : display line numbers
-v : displays lines that do not match the pattern
-w : matches only whole words only
```

```
fgrep : pattern must be fixed string
egrep : pattern can be extended regular expression
-x option in fgrep: displays only lines that are exactly
                    equal to string

extended regular expressions:
  + matches one or more of the single preceding character
  ? matches zero or one of the single preceding character
  | either or (ex. a* | b*)
 () *, +, ? operate on entire subexpression not just on
                 preceding character; ex.  (ab | ba)*
```

```
sort -tc -r {sortField -bfMn}* {fileName}*
-tc separator is c instead of blank
-r descending instead of ascending
-b ignore leading blanks, -f ignore case, -M month sort, -n numeric sort

$ cat sort.dat
John Smith 1222 20 Apr 1956
Tony Jones 1012 20 Mar 1950
John Duncan 1111 20 Jan 1966
Larry Jones 1223 20 Dec 1946

$ sort +0 -2 sort.dat
John Duncan 1111 20 Jan 1966
John Smith 1222 20 Apr 1956
Larry Jones 1223 20 Dec 1946
Tony Jones 1012 20 Mar 1950

$ sort +4 -5 -M  sort.dat
John Duncan 1111 20 Jan 1966
Tony Jones 1012 20 Mar 1950
John Smith 1222 20 Apr 1956
Larry Jones 1223 20 Dec 1946

$ sort +4 -5  sort.dat
John Smith 1222 20 Apr 1956
Larry Jones 1223 20 Dec 1946
John Duncan 1111 20 Jan 1966
Tony Jones 1012 20 Mar 1950
```

```
- Comparing files
cmp -ls file1 file2 offset1 offset2

compares two files for equality; reports the first byte where
 there is a mismatch;
if one file is a prefix of the other, EOF message is displayed;

Optional values offset1 and offset2 are the offsets into the files
 where comparison begins.

-l option displays the line number and byte offset of all
   mismatched bytes
-s option suppresses all output

diff -i file1 file2
compares two files and outputs a description of their differences;
-i flag ignores case
```

Archiving:

```
tar -cfrtuvx tarFileName fileList
```

creates a tar-format (tape archive) file from the fileList

-c option creates the tar-format file

-x option extracts files from the tar-format file

-t option generates a table of contents

-r option unconditionally appends listed files to tar formatted file

-u option appends only files that are more recent than those already archived

-f option enables you to give a tar file name (default is /dev/rmt0)

-v verbose

If fileList contains directory, its contents are appended/extracted recursively.

```
$ tar cvf ch6.tar ch6
ch6/
ch6/menu.csh
ch6/junk/
ch6/junk/junk.csh
ch6/junk.csh
ch6/menu2.csh
ch6/multi.csh
ch6/expr1.csh
ch6/expr3.csh
ch6/expr4.csh
ch6/if.csh
ch6/menu3.csh
$ ls -l ch6.tar
-rw-rw-r--   1 raj      raj         20480 Jun 26 20:08 ch6.tar
$ tar -tvf ch6.tar
drwxr-xr-x raj/raj           0 1999-06-03 09:57 ch6/
-rwxr-xr-x raj/raj         403 1999-06-02 14:50 ch6/menu.csh
drwxr-xr-x raj/raj           0 1999-06-03 09:57 ch6/junk/
-rwxr-xr-x raj/raj        1475 1999-06-03 09:57 ch6/junk/junk.csh
-rwxr-xr-x raj/raj        1475 1999-06-03 09:56 ch6/junk.csh
-rw-r--r-- raj/raj         744 1999-06-02 15:59 ch6/menu2.csh
-rwxr-xr-x raj/raj         445 1999-06-02 15:26 ch6/multi.csh
-rwxr-xr-x raj/raj         279 1999-06-02 15:18 ch6/expr1.csh
-rwxr-xr-x raj/raj          98 1999-06-02 15:20 ch6/expr3.csh
-rwxr-xr-x raj/raj         262 1999-06-02 15:21 ch6/expr4.csh
-rwxr-xr-x raj/raj         204 1999-06-02 15:22 ch6/if.csh
-rw-r--r-- raj/raj         744 1999-06-02 16:01 ch6/menu3.csh
```

```
$ rm -fr ch6
$ tar -rvf ch6.tar date.txt
date.txt
$ tar tvf ch6.tar
drwxr-xr-x raj/raj           0 1999-06-03 09:57 ch6/
-rwxr-xr-x raj/raj         403 1999-06-02 14:50 ch6/menu.csh
drwxr-xr-x raj/raj           0 1999-06-03 09:57 ch6/junk/
-rwxr-xr-x raj/raj        1475 1999-06-03 09:57 ch6/junk/junk.csh
-rwxr-xr-x raj/raj        1475 1999-06-03 09:56 ch6/junk.csh
-rw-r--r-- raj/raj         744 1999-06-02 15:59 ch6/menu2.csh
-rwxr-xr-x raj/raj         445 1999-06-02 15:26 ch6/multi.csh
-rwxr-xr-x raj/raj         279 1999-06-02 15:18 ch6/expr1.csh
-rwxr-xr-x raj/raj          98 1999-06-02 15:20 ch6/expr3.csh
-rwxr-xr-x raj/raj         262 1999-06-02 15:21 ch6/expr4.csh
-rwxr-xr-x raj/raj         204 1999-06-02 15:22 ch6/if.csh
-rw-r--r-- raj/raj         744 1999-06-02 16:01 ch6/menu3.csh
-rw-rw-r-- raj/raj          29 1999-06-21 11:06 date.txt
```

```
$ tar xvf ch6.tar
ch6/
ch6/menu.csh
ch6/junk/
ch6/junk/junk.csh
ch6/junk.csh
ch6/menu2.csh
ch6/multi.csh
ch6/expr1.csh
ch6/expr3.csh
ch6/expr4.csh
ch6/if.csh
ch6/menu3.csh
date.txt
```

```
 - find pathList expression

   The find utility not only allows you to find files starting at
     pathList and descending from there on, but also
     allows you to perform certain actions such as deleting
     the files etc.

Expression          Value/Action
----------          ------------


-name pattern       true if the file name matches pattern
-perm oct           true if the octal description of file's permission
                    equals oct
-type ch            true if the type of the file is ch
                           (b=block, c=char ..)
-user userId        true if the owner of the file is userId
-group groupId      true if the group of the file is groupId
```

```
Expression          Value/Action
----------          ------------

-atime count   true if the file has been accessed within
               count days
-ctime count   true if the contents of the file have been
               modified within count days or any of its
               file attributes have been modified
-exec command  true if the exit code = 0 from executing the command.
               command must be terminated by \;
               If {} is specified as a command line argumentm it is
               replaced by the file name currently matched
-print         prints out the name of the current file and returns true
-ls            displays the current file's attributes and returns true
!expression      negation of expression
expr1 [-a] expr2 short circuit and
expr1 -o expr2   short circuit or
```

```
$ find ch6 -name j*
ch6/junk
ch6/junk/junk.csh
ch6/junk.csh

$ find / -name x.c
searches for file x.c in the entire file system

$ find . -mtime 14 -ls
lists files modified in the last 14 days

$ find . -name '*.bak' -ls -exec rm {} \;
ls and then remove all files that end with .bak
```

- Scheduling commands: crontab and at

```
crontab cronTabName
crontab -ler [useName]
```

The crontab utility allows you to schedule a series of jobs to be executed on a periodic basis. A file with the following line format must be created:

```
minute hour   day    month   weekday          command
(0-59) (0-23) (1-31) (1-12) (1-7; 1=Monday)  (any Unix command)
```

ex.

```
$ cat crontab.cron
0 8 * * 1 echo Happy Monday Morning
* * * * * echo One Minute passed
30 15 1 * 1 mail users % Jan meeting at 3pm
```

```
$ crontab crontab.cron

$ crontab -l
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (crontab.cron installed on Sat Jun 26 23:33:35 1999)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
0 8 * * 1 echo Happy Monday Morning
* * * * * echo One Minute passed
30 15 1 * 1 mail users % Jan meeting at 3pm

$ crontab -r
unregister crontab file
```

```
- at command allows you to schedule one-time commands/scripts

at -csm time [date [, year]] [+increment] [script]
at -r [jobId]+
at -l [jobId]+

-c option : use C-shell
-s option : use Bourne shell
-m option : send mail
-r option : remove entry from at queue
-l option : list entries in at queue

time is specified as HH or HHMM followed by an optional AM/PM
date is spelled out using first 3 letters of day and/or month
keyword now can be used in place of time sequence
stated time may be augmented by an increment (number followed
  by minutes/hours/days/weeks/months/years)
```

```
$ cat at.csh
#!/bin/csh
echo at done > /dev/tty1

$ at now + 2 minutes at.csh

$ at -l
 lists the job here
```

You may program the script to reschedule itself as follows:

```
#!/bin/csh
date > /dev/tty1
at now + 2 minutes at.csh
```

```
- awk: utility that scans one or more files and performs an
       action on all lines that match a particular condition.
       The conditions and actions are specified in an awk program.

- awk reads a line; it breaks it into fields separated by
    tabs/spaces (could have other separators specified by -F option)
```

```
- awk programs has one or more commands of the form:

    [condition] [ \{ action \} ]

  where condition is one of the following:
    - special tokens BEGIN or END
    - an expression involving logical operators, relational operators,
      and/or regular expressions
  and action is one of the following kinds of C-like statements
    - if-else; while; for; break; continue
    - assignment statement
    - print; printf;
    - next (skip remaining patterns on current line of input)
    - exit (skips the rest of the current line)
    - list of statements
```

```
- accessing individual fields: $1, ..., $n refer to fields 1 thru n
                               $0 refers to entire line
- built-in variable NF referers to number of fields

% awk -F: '{ print NF, $1 }' /etc/passwd
prints the number of fields and the first field in the /etc/passwd file

- BEGIN condition is triggered before the first line is read and
  the END condition is triggered after the last line is read;
p2.awk
   BEGIN { print "Start of file: "}
   { print $1 "  " $6 "  " $7 }
   END { print "End of file", FILENAME }

FILENAME: built-in variable for name of file being processed
To execute the above program, use
%  awk -F: -f p2.awk /etc/passwd
```

- Operators

p3.awk

```
 NR > 1 && NR < 4 { print NR, $1, $6, $NF }
```

built-in variable NR contains the current line number

- Variables

  p4.awk

```
  BEGIN {print "Scanning file"}
  { printf "line %d: %s\n", NR, $0
    lineCount++;
    wordCount += NF;
  }
  END { printf "lines =  %d, words = %d\n", lineCount, wordCount }
```

- Control structures

p5.awk

```
{
  for (i = NF; i >= 1; i--)
    printf "%s ", $i;
  printf "\n";
}
```

- extended regular expressions

```
% awk -F: ' /t.*e/ { print $0} ' /etc/passwd
```

- Condition ranges: two expressions separated by comma;
  awk performs action on every line from the first line that
  matches first expression until line that matches second condition

```
awk -F: ' /root/,/nobody/ {print $0}' /etc/passwd
```

- Built-in functions: exp(), log(), sqrt(), substr() etc.

```
$ awk -F: ' {print substr($1,1,2)}' /etc/passwd
```