

Chapter 3

UNIX Utilities for Power Users

Graham Glass and King Ables,
UNIX for Programmers and Users,
Third Edition, Pearson Prentice Hall, 2003.

Original Notes by Raj Sunderraman
Converted to presentation and updated by
Michael Weeks

Grep

- Filtering patterns: egrep, fgrep, grep
 - `grep -hlnvw pattern {fileName}`*
 - displays lines from files that match the pattern
 - pattern : regular expression
 - -h : do 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

Grep variations

- 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

- `sort -tc -r [+POS1 [-POS2]] {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 (3 letter month abbreviation)
 - -n numeric sort
 - +POS1 [-POS2] key positions start [up to end]`

Sort Examples

```
$ 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 Examples

```
$ 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

- What if you have two files with the same name, but different sizes?
- `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

Comparing Files

- `cmp -ls file1 file2 offset1 offset2`
- 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

Comparing Files

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

Archiving

- `tar -crtuvx tarFileName fileList`
 - creates a “tape archive”-format file from the `fileList`
 - Typically, use either `-c` or `-x`, but not both!
 - `-c` for Create
 - `-x` for Extracts
 - `-t` option generates a table of contents
 - `-r` option unconditionally appends listed files to tar formatted file

Archiving

- `tar -crtuvx tarFileName fileList`
 - `-u` option appends only files that are more recent than those already archived
 - `-f` option enables you to give a tar file name
 - Default name is `/dev/rmt0`
 - `-v` verbose
- If `fileList` contains directory, its contents are appended/extracted recursively.

Archive Example

```
$ tar -cvf mgwin.tar mgwin
```

```
$ ls -al mgwin/
```

```
total 24
```

```
drwxrwxr-x  6 cscqxcx cscqxcx 4096 Jan 23 23:39 .
```

```
drwxrwxr-x  6 cscqxcx cscqxcx 4096 Jan 23 23:49 ..
```

```
drwxr-xr-x  2 cscqxcx cscqxcx 4096 Nov 18 2006 bin
```

```
drwxr-xr-x  3 cscqxcx cscqxcx 4096 Nov 18 2006 doc
```

```
drwxr-xr-x  3 cscqxcx cscqxcx 4096 Nov 18 2006
```

```
include
```

```
drwxr-xr-x  2 cscqxcx cscqxcx 4096 Nov 18 2006 lib
```

```
$ ls -alR mgwin
```

Archive Example

\$ ls mgwin.tar -l

```
-rw-rw-r-- 1 cscqxcx cscqxcx 5396480 Jan 23 23:49 mgwin.tar
```

\$ tar -tvf mgwin.tar

```
drwxrwxr-x cscqxcx/cscqxcx  0 2008-01-23 23:39:36 mgwin/  
drwxr-xr-x cscqxcx/cscqxcx  0 2006-11-18 17:54:39 mgwin/bin/  
-rwxr-xr-x cscqxcx/cscqxcx 11643 2006-11-18 17:54:39 mgwin/bin/mingwm10.dll  
drwxr-xr-x cscqxcx/cscqxcx   0 2006-11-18 17:54:38 mgwin/doc/  
drwxr-xr-x cscqxcx/cscqxcx   0 2006-11-18 17:54:40 mgwin/doc/mingw-runtime/  
-rw-r--r-- cscqxcx/cscqxcx 1112 2006-11-18 17:54:40 mgwin/doc/mingw-  
runtime/CONTRIBUTORS  
-rw-r--r-- cscqxcx/cscqxcx  519 2006-11-18 17:54:40 mgwin/doc/mingw-  
runtime/DISCLAIMER  
-rw-r--r-- cscqxcx/cscqxcx  398 2006-11-18 17:54:40 mgwin/doc/mingw-runtime/README  
drwxr-xr-x cscqxcx/cscqxcx   0 2006-11-18 17:54:47 mgwin/include/  
.....
```

Compress/Uncompress

- compress/uncompress
 - % compress fileName (.Z)
 - % uncompress fileName
- gzip
 - % gzip fileName (.gz)
 - % gunzip fileName
- crypt
 - % crypt key < sample.txt > sample.crypt (to crypt)
 - % crypt key < sample.crypt > sample.txt (to uncrypt)
 - key could be any string

Archive Example

```
$ rm -rf mgwin
```

```
$ tar -rvf mgwin.tar netstat.txt  
netstat.txt
```

```
$ tar -xvf mgwin.tar  
mgwin/  
mgwin/bin/  
mgwin/bin/mingwm10.dll  
mgwin/doc/  
mgwin/doc/mingw-runtime/  
mgwin/doc/mingw-runtime/CONTRIBUTORS  
mgwin/doc/mingw-runtime/DISCLAIMER  
...
```

```
$ tar -xvf mgwin.tar | grep netstat.txt  
netstat.txt
```

Find Utility

- find pathList expression
- Finds files starting at pathList
- Finds files descending from there
- Allows you to perform certain actions
 - e.g. deleting the files

Find Utility

- 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

Find Utility

- -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 argument it is replaced by the file name currently matched

Find Utility

- `-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 Examples

- `$ find . -name assert.h`
 - searches for file `assert.h` 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`

Find Examples

```
$ find mgwin -name 'lib*'
```

```
mgwin/lib
```

```
mgwin/lib/libcoldname.a
```

```
mgwin/lib/libcrt.dll.a
```

```
mgwin/lib/libgmon.a
```

```
mgwin/lib/libm.a
```

```
mgwin/lib/libmingw32.a
```

```
mgwin/lib/libmingwex.a
```

```
mgwin/lib/libmingwthrd.a
```

```
mgwin/lib/libmoldname.a
```

```
mgwin/lib/libmoldnamed.a
```

```
mgwin/lib/libmsvc70.a
```

```
mgwin/lib/libmsvc70d.a
```

```
mgwin/lib/libmsvc71.a
```

```
mgwin/lib/libmsvc71d.a
```

```
mgwin/lib/libmsvc80.a
```

```
mgwin/lib/libmsvc80d.a
```

```
mgwin/lib/libmsvcrt.a
```

```
mgwin/lib/libmsvcrt.d.a
```

Hard Links

- % In original newLink
 - Creates a hard link to original file called newLink
 - Both labels will refer to the same file
 - File will be deleted only when both labels are removed
 - If newLink is a directory then links are made within the directory

Hard Link Example

```
% cat > hold  
This is an example !!!
```

```
% ln hold hold.1  
% cat hold.1  
This is an example !!!
```

```
% vi hold.1 # changed "an" to "a great"  
% cat hold.1  
This is a great example !!!
```

```
% cat hold  
This is a great example !!!
```

```
% ls -l hold*  
-rw-rw-r-- 2 cscqxcx cscqxcx 22 Jan 28 13:28 hold  
-rw-rw-r-- 2 cscqxcx cscqxcx 22 Jan 28 13:28 hold.1
```

Soft (Symbolic) Links

- `% ln -s original newLink`
 - Symbolic/soft links can be created from one file system into another file system
 - Hard links are restricted to one file system
- Use `ls -lL` to view link details
 - which file it refers to
- `ls -l` displays the contents of the symbolic link

```
% ls -l
lrwxrwxrwx 1 root root    23 2006-09-22 20:26 javac -> j2sdk1.4.2_05/bin/javac
% ls -lL
-rwxrwxr-x 1 10016 users 66156 2004-06-04 00:32 javac
% ls -l j2sdk1.4.2_05/bin/javac
-rwxrwxr-x 1 10016 users 66156 2004-06-04 00:32 j2sdk1.4.2_05/bin/javac
```

Substituting User

- `% su [-] userName`
 - If `userName` is not specified, `root` is assumed
 - Need access privileges for this
 - Requires password
- `% sudo command`
 - User can execute command as superuser
 - Requires password
- `% whoami` Display the name of the owner of the shell

Scheduling Command (cron)

- `crontab cronTabName`
- `crontab -ler [userName]`
 - Allows you to schedule a series of jobs
 - Executed on a periodic basis
 - Uses a file with the following line format:
 - Text following a % are copied to a temporary file and used as the command's standard input

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

Scheduling Example

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

```
$ cat crontab.cron
```

```
* * * * * echo One Minute passed
```

```
0 12 29 1 * mail users % Are you ready for EXAM?
```

Scheduling Example

```
$ 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 2007)
```

```
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
```

```
* * * * * echo One Minute Passed > /dev/pts/1
```

```
* * * * * pwd > /dev/pts/1
```

```
30 14 29 1 * mail users % Are you ready for exam1?
```

```
$ crontab -r
```

```
unregister crontab file
```

Scheduling Command (at)

- *at* command
 - allows you to schedule one-time commands/scripts
- `at -csm time [date [, year]] [+increment] [script]`
 - `-c` option : use C-shell
 - `-s` option : use Bourne shell
 - `-m` option : send mail
- time is specified as HH or HHMM followed by an optional AM/PM

Scheduling Command (at)

- `at -r [jobId]+`
- `at -l [jobId]+`
 - `-r` option : remove entry from at queue
 - `-l` option : list entries in at queue

Scheduling Command (at)

- `at -csm time [date [, year]] [+increment] [script]`
 - *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
 - keyword `today / tomorrow` can be used in place of date sequence
 - stated time may be augmented by an increment (number followed by `minutes/hours/days/weeks/months/years`)

Scheduling Example

```
$ 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
```

Pattern Scanning and Processing

- *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
 - breaks it into fields separated by tabs/spaces
 - or other separators specified by *-F* option

awk Command

- awk program has one or more commands:
- [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

awk Command

- `awk [condition] [\{ action \}]`
- action is one of the following kinds of C-like statements
 - `if-else; while; for; break; continue`
 - assignment statement (`variable=expression`)
 - `print; printf;`
 - `next` (skip remaining patterns on current line)
 - `exit` (skips the rest of the current line)
 - list of statements

awk Command

- accessing individual fields:
 - \$1, ..., \$n refer to fields 1 thru n
 - \$0 refers to entire line
- built-in variable NF means number of fields
- % `awk -F: '{ print NF, $1 }' /etc/passwd`
 - prints the number of fields and the first field in the `/etc/passwd` file
 - `-F:` means to use `:` as the field separator

awk Command

- BEGIN condition is triggered before first line is read
- END condition is triggered after last line has been read
- FILENAME: built-in variable for name of file being processed
- We will use this data in following examples:

```
% cat /etc/passwd
```

```
nobody:*:-2:-2:Unprivileged User:/:/usr/bin/false
```

```
root:*:0:0:System Administrator:/var/root:/bin/sh
```

```
...
```

```
lp:*:26:26:Printing Services:/var/spool/cups:/usr/bin/false
```

awk Example

```
% cat p2.awk
BEGIN { print "Start of file: "}
{ print $1 " " " $6 " " " $7 }
END { print "End of file", FILENAME }
```

```
% awk -F: -f p2.awk /etc/passwd
```

```
Start of file:
```

```
nobody / /usr/bin/false
```

```
root /var/root /bin/sh
```

```
...
```

```
lp /var/spool/cups /usr/bin/false
```

```
End of file /etc/passwd
```

awk Operators

- built-in variable NR contains current line #
- remember, “-F:” uses colon as separator

```
% cat p3.awk
```

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

```
% awk -F: -f p3.awk /etc/passwd
```

```
2 root /var/root /bin/sh 7
```

```
3 daemon /var/root /usr/bin/false 7
```

awk Variables

```
cat awk4
```

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

```
awk -f awk4 float
```

```
Scanning file ....
```

```
line 1: Wish I was floating in blue across the sky,
```

```
line 2: My imagination is strong,
```

```
line 3: And I often visit the days
```

```
line 4: When everything seemed so clear
```

```
line 5: Now I wonder what I'm doing here at all...
```

```
lines = 5, workds = 33
```

awk Control Structures

```
% cat awk5
```

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

```
% awk -f awk5 float
```

```
sky, the across blue in floating was I Wish  
strong, is imagination My  
days the visit often I And  
clear so seemed everything When  
all... at here doing I'm what wonder I Now
```

awk Condition Ranges

- 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: ' /nobody/,/root/ {print $0}' /etc/passwd
nobody:*:-2:-2:Unprivileged User:/:usr/bin/false
root:*:0:0:System Administrator:/var/root:/bin/sh
```

```
% awk '/strong/ , /clear/ {print $0}' float
My imagination is strong,
And I often visit the days
When everything seemed so clear
```

awk Built-in Functions

- Built-in functions:

- `exp()`

- `log()`

- `sqrt()`

- `substr()` etc.

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

```
no
```

```
ro
```

```
...
```

```
lp
```

```
% cat awk8
```

```
% awk -f awk8 test
```

Stream Editor (sed)

- sed
 - scans one or more text files
 - performs an edit on all lines that match a condition
 - actions and conditions may be stored in a file
 - may be specified at command line in single quotes
 - commands begin with an address or an addressRange or a Regular expression
 - does not modify the input file
 - writes modified file to standard output

Things You Can Do with sed

- *a* append lines to output until one not ending in \
- *c* change lines to following text, as in *a*
- *d* delete lines
- *i* insert following text before next output
- */* list line, making all non-printing characters visible
 - (tabs appear as >; lines broken with \)
- *P* print line
- *q* quit (for scripts)
- *r file* read *file*, copy contents to stdout
- *b label* branch to command : label

Things You Can Do with sed

- *s/pat1/pat2/f* substitute *pat2* for *pat1*
 - *f = g*, replace all occurrences
 - *f = p*, print
 - *f = w file*, write to *file*
- *t label* test: branch to *label* if substitution made to current line
- *w file* write line(s) to *file*
- *y/str1/str2/* replace each character from *str1* with corresponding character from *str2*, no ranges allowed
- *=* print current input line number
- *!cmd* do *sed cmd* if line is not selected
- *: label* set label for *b* and *t* commands

Substituting Text

- `% sed 's/^/ /' file > file.new`
 - indents each line in the file by 2 spaces
- `% sed 's/^ *//' file > file.new`
 - removes all leading spaces from each line of the file
- `% sed '/a/d' file > file.new`
 - deletes all lines containing 'a'
- Example: add two lines in the beginning of the file

```
% cat sed1
```

```
1i\  
abcd\  
efg
```

```
% sed -f sed1 file > file.new
```

Example Using sed

- Replace lines 1-3 by “Lines 1-3 are censored”

```
% cat sed2
```

```
1,3c\
```

```
Lines 1-3 are censored
```

```
% cat dummy2
```

```
one
```

```
two
```

```
three
```

```
four
```

```
five
```

```
six
```

```
% sed -f sed2 dummy2
```

```
Lines 1-3 are censored
```

```
four
```

```
five
```

```
six
```

Example Using sed

- Multiple commands; replaces individual lines

```
% cat sed3
```

```
1c\
```

```
Line 1 is censored
```

```
2c\
```

```
Line 2 is censored
```

```
3c\
```

```
Line 3 is censored
```

```
% sed -f sed3 dummy2
```

```
Line 1 is censored
```

```
Line 2 is censored
```

```
Line 3 is censored
```

```
four
```

```
five
```

```
six
```

Example Using sed

- `% sed '$r file' f > g`
 - appends file at end of file *f*
- `% sed -e 's/^/ << /' -e 's/$/ >>/' file`
 - multiple commands
 - `-e` option is optional! means script on command line

```
% sed -e 's/^/ << /' -e 's/$/ >>/' dummy2
```

```
<< one >>
```

```
<< two >>
```

```
<< three >>
```

```
<< four >>
```

```
<< five >>
```

```
<< six >>
```

Translate Utility (tr)

- `% tr -cds string1 string2`
- maps all characters in std. input from character set `string1` to the corresponding character in `string2`
- If length of `string2` is less, the last character is repeated.

Translate Utility (tr)

- `% tr -cds string1 string2`
 - `-c` option causes `string1` to be complemented (every character not in `string1` now is in `string1`!)
 - `-d` option causes every character in `string1` to be deleted from std. input
 - `-s` option causes every repeated character in `string1` to be condensed to one occurrence

Examples of tr

- `% tr a-z A-Z < file1 > file2`
 - causes lower-case to upper-case conversion
- `% tr -c a X < file1 > file2`
 - causes every non-*a* character to be replaced by *X* (including newline)
- `% tr -c a-z '\012' < file1 > file2`
 - causes all non alphabetic characters to be replaced by ASCII 12 (newline)
- `% tr -d a-c < file1 > file2`
 - causes all *a*, *b*, *c* to be deleted from *file 1*

Review

- Advanced commands
- Pattern matching (grep)
- Sort
- Comparing (cmp, diff)
- Archiving (tar)
- Scheduling (cron, at)
- Pattern matching and processing (awk, sed)