COP 3344 Lecture Notes

Supplement

Dr. David A. Gaitros

References

Unix System Administration Handbook, Second Edition, Nemeth, Snyder, Seebass, Hein, Prentice Hall 1995

Linux Clearly Explained, Phaffenberger, Morgan Kaufmann, 1999

Unix for Dummies, Quick Reference 4th Edition, Young and Levine, IDG Books Worldwide, 1998

Unix man pages

echo

Send it's output to the console. Usually used in login or similar types of scripts where you want to send some type of message or status to the user.

Format:

```
echo [-n] text to display

-n  Does not begin a new line after echo.
```

Example:

```
echo –n "Your Report is Now Printing"
```

man

Displays reference manual page for Unix commands. .

Format:

```
ma [-k keywords] topic

-k Specifies one or more keywords to
   search for.
```

Example:

```
man ls
```

alias

Creates an alias for a command or shows which aliases exist for your process. Used mostly in login or setup scripts to set complex variables or to make commands more friendly to use. Used in C, Bash, or Korn shells only.

Format (C shell):

```
alias [name [‘command’]]
```

Example:

```
alias dir ‘ls –al’
```

Format (Korn and Bash shells):

```
alias [name=[‘command’]]
```

Example:

```
alias dir=’ls –al’
```

Aliases created when you are logged in are not saved when you log out. You should add any aliases you want to keep to your start up files.

awk

Awk is a programming language designed to make many common information retrieval and text manipulation tasks easy to state and to perform. The name awk comes from the names of its creators Alfred V. **A**ho, Peter J. **W**einberger, and Brian W. **K**ernighan. A GNU version of awk, called *gawk*, is also in wide use (and is recommended). (Also, for most practical purposes, the awk programming language has been superseded by Larry Wall's perl language.)

The basic operation of awk is to scan a set of input lines in order, searching for lines which match any of a set of patterns which the user has specified. For each pattern, an action can be specified; this action will be performed on each line that matches the pattern.

Format:

```
awk [-f program] file

-f Specifies that a filename contains the
   awk program.  Otherwise you put the
   awk program between single quotes.
```

Example:

```
awk '{print $1,$2,sin($3/$2)}' filename
```

This command will print the first and second fields, and then the sine of the third field divided by the second. So, the second and third field had better be numbers. Awk has other built in math functions like sine; read the Unix man page for others.

cal

Prints a calendar for a month or a year
If you just type cal with no month or year, you get a calendar for the current month. If you provide a year but no month, you get the whole year. You can redirect the output to a file if you wish.

Format:

```
cal [month][year]

[month] Specifies the month 1-12
[year]  specifies year from 1 - 9999

-n  Does not begin a new line after echo.
```

Example:

```
cal 12 1999 > calendar.output.dat
```

Note: Pope Gregory XIII took the advice of his mathematicians and shortened October of A.D. 1582 by ten days to correct problems with the current calendar. Thursday, October 4, 1582 (Julian) was followed immediately by Friday, October 15, 1582 (Gregorian). Getting dates using cal before this date or after 9999 are not advisable.

more

Displays the contents of a file or information to the output one screen at a time so that you can read it easily.

Format:

```
more [-s][-u][+/text][filename]

-s Suppresses extra blank lines
-u Ignores underscores and backspace
+/text Starts to display file two lines
   before the start of the 'text' that is
   found in the file.
```

Example:

```
more +/windows .login

ls | more
```

Note: The last example takes the output of the ls command and uses more to display the contents of the directory command one page at a time.

cat

Displays a file to the screen. Only works with text files and displays the WHOLE file at once without stopping. Difficult to look at large files. This command was used before there was a convenient ftp method to transfer files. Often used to con**cat**enate files. The command "more" is used instead.

Format:

```
cat filename [filename]
```

Example:

```
cat file1 file2 file3 file4 >one.big.file
```

Note: The above example takes four files and places them one after another and puts the results in a file called one.big.file

cd

Literally this stands for change directory. Moves forward or backward. To go to your home directory, just type cd with no other input.

Format:

```
cd [directory]
```

Examples:

```
cd ..
cd ~gaitrosd
cd /home/bin/
cd ../public_html/classes/cop3344
cd ../../MorphBankv2.2/bin/utils
```

Note: The two periods (..) indicate the parent directory of the current directory (so one up). A single period (.) indicates the current directory.

rm

Removes a file or files from a directory. Actually, like the Windows delete command this just removes the entry in the directory table. It does not erase the contents. You cannot "undelete" a file.

Format:

```
rm [-i][-r] filename
[-i] Ask to confirm before delete
 [-r] Performs the delete operation in
     all subdirectories.
```

Examples:

```
rm text.junk
rm –i text.junk
rm *.out
rm *.*
rm –ir *.*
```

mkdir

Creates a new directory. Your account must have write privileges in order to do this. You should only create directories and sub-directories within you own account.

Format:

```
mkdir directory
```

Example:

```
mkdir tempdir
```

rmdir

Removes a directory. Your account must have write privileges in order to do this. Normally you are only allowed to remove directories that are empty. Use the rm –r command to delete the contents of subdirectories.

Format:

```
rmdir directory
```

Example (Assume you are in the directory *tempdir*):

```
rm –r *.*
rm –r *
cd ..
rmdir tempdir
```

chgrp

Changes the group that has access to the a file or directory(System V only). You have to own the file or directory to use this command. You can use a wildcard to change a group of files or directories.

Format:

```
chgrp [-r] newgroup filename

[-r] Tells chmod to change permissions on
     files in subdirectories too.
newgroup: Specifies the name of the group
     that takes ownership.
filename: Specifies the file(s) that are
     affected by the change.
```

Example:

```
chgrp userserv thisfile.dat
```

clear

Clears the contents of the screen.

Format:

```
clear
```

Example:

```
clear
```

cp

Copy a file or files. Differs slightly from other operating system copy functions. For instance, to copy a file to the current directory the target directory is represented by a period. Also, if you try to copy a file into a location that already has a file with the same name, the older file is replaced. Use the –i option to have Unix ask you before it replaces the file.

Format:

```
cp [-] [-R] oldfile directory[/newfiles]

[-i] Ask before replacing files.
[-R] When copying a directory, also
copies the contents of all
subdirectories.
oldfile:  Specifies the name of the file
you want to copy.
directory:  The directory path of where
you want the file to go.
newfiles: Specifies the name you want to
give the new file(s). Defaults to the
oldfile name.
```

Example:

```
cp this.out this.dat
cp ~gaitrosd/public_html/index.html index.back
```

mv

There is no rename option if Unix. Therefore, we have a move (mv) command. This moves or renames a file. Actually, very seldom is the file physically moved. The reference is changed in the directory table structure. Moving a file to a name that already exists will have the same affect as the copy function. The older file will be overwritten without warning. Use the –i option to avoid this.

Format:

```
mv [-] oldfile directory[/newfiles]

[-i] Ask before replacing files.
oldfile:  Specifies the name of the file
you want to copy.
directory:  The directory path of where
you want the file to go.
newfiles: Specifies the name you want to
give the new file(s). Defaults to the
oldfile name.
```

Example:

```
mv this.out this.dat
mv ~gaitrosd/public_html/index.html index.html
```

grep

Very nice command to have and very efficient. Stands for global regular expressions. Finds the occurance of a string in one or more files.

Format:

```
grep [-i] [-l] [-n] [-v] text filenames

[-i] Ignores case when searching
[-l] Displays only the names of the files
     that contain the text, not the
     lines.
[-n] Displays the line numbers of the
     lines that contain the text
[-v] Specifies you are looking for text
     that does NOT contain the text.
text The string you are looking for.
filenames The files you are looking in.
```

Special Characters:

.	Matches any single character.
*	Matches any number of characters. Wildcard.
[]	Matches any one of the characters in the braces.
^	Represents the beginning of a line. ^T matches a T at the beginning of a line.
$	Represents the end of a line.
\	Tells grep to take the next character literally, not as a special character. If you want to search for I.B.M. you have to enter I\.B\.M\.

Example:

```
grep “microwave shelf-stable” *.*
grep –I “Hello World”  This.out
grep –I “End of World\.” This.dat
```

mount

The tools that are available to mount devices today are much more sophisticated and easier to use then in days past. However, you should know how to manually mount a device. All files must be mounted and unmounted to maintain a proper state in the operating System.

After installing a new disk or device, you should mount it manually and test it to make sure everything works and the permissions are correct. Like most of this class, this is not a complete description of the command. Only an introduction to the most common parts.

The file systems in a Unix environment are arranged in one bit tree. A device can be mounted as a node virtually anywhere in that tree.

Format:

```
mount [-lhV][-t type]

mount -a [-fFnrsvw] [-t vfstype] [-O optlist]
mount [-fnrsvw] [-o options [,...]] device |
   dir
mount [-fnrsvw] [-t vfstype] [-o options]
   device dir
```

mount [-lhv] does not actually mount anything. The "-l" option lists all mounted file systems of type "type". The "-h" option prints a help message, and the "-V" option prints a version string.

Notes:

- Since Linux 2.4.0 it is possible to remount part of a file.
- fstab - The file fstab contains descriptive information about the various file systems. fstab is only read by programs, and not

written; it is the duty of the system administrator to properly create and maintain this file. Each filesystem is described on a separate line; fields on each line are separated by tabs or spaces. Lines starting with '#' are comments. The order of records in fstab is important because fsck, mount, and umount sequentially iterate through fstab doing their thing. The file is used during boot time to mount files.

```
-V     Output version.

-h     Print a help message.

-v     Verbose mode.
-p num If the mount requires a passphrase to be entered, read it
       from file descriptor num instead of from the terminal.
-a     Mount all file systems (of the given types) mentioned in
       fstab.

-f     Causes everything to be done except for the actual system
       call;if it's not obvious, this ``fakes'' mounting the
       file system. This option is useful in conjunction with the
       -v flag to determine what the mount command is trying to
       do. It can also be used to add entries for devices that
       were mounted earlier with the -n option.

-I     Don't call the /sbin/mount.<filesystem> helper even
       if it exists.

-n     Mount without writing in /etc/mtab. This is necessary for
       example when /etc is on a read-only file system.

-p num In case of a loop mount with encryption, read the
       passphrase from file descriptor num instead of from the
       terminal.
-s     Tolerate sloppy mount options rather than failing. This
       will ignore mount options not supported by a filesystem
       type. Not all filesystems support this option. This option
       exists for support of the Linux autofs-based automounter.

-r     Mount the file system read-only. A synonym is -o ro.

-w     Mount the file system read/write. This is the default. A
       synonym is -o rw.

-L label Mount the partition that has the specified label.

-U uuid  Mount the partition that has the specified uuid.
       These two options require the file /proc/partitions
       (present since Linux 2.1.116) to exist.
```

```
-t vfstype  The  argument following the -t is used to indicate
      the file system type.
```

Examples:

```
mount –l
mount –V
mount /dev/cdrom   or mount /cd
```

passwd

This allows the user to change their own password at the command line level. The command asks the user to type their old password and then asks the user to type the new password twice. Many versions of passwd try to enforce rules that require passwords to meet certain criteria such as length and variation of letters, characters, and numbers.

Format:

```
passwd
```

Example:

```
passwd
```

ps

Displays information about your processes or jobs.

Format:

```
ps [-a] [-l] [-ttty] [-u] [-x]

-a  Displays info about all processes
-l  Displays more information
-ttty Display information about process
     started by terminal tty.
-u  Displays a user oriented report
-x  Displays all background jobs
```

Example:

```
ps -al
```

pwd

Print working directory. OK, so your lost inside Unix and you don't know where your are. This command will display the working directory of your current location. Many times this is used in scripts to get a complete path to your current location.

Format:

```
pwd
```

Example:

```
pwd

/home/gaitrosd/public_html
```

set

Very handy command. Sets a shell variable to the value you specify or displays the value of the shell variable. Works slightly different in the C Shell. Just using the `set` command by itself will display the value of all shell variables. To see the value of just one variable use the `echo` command. Remember that all shell variables must begin with a dollar ($) sign.

Format (Bourne, BASH, and Korn shells):

```
set
```

Format (C shell):

```
set [variable = value]
```

Example (Bourne, BASH, and Korn shells):

```
WORKDIR = /home/gaitrosd/public_html

echo $WORKDIR
cd $workdir
```

Example (C shell):

```
set workdir=/home/gaitrosd/public_html

echo  $workdir
cd $workdir
```

uuencode

There are times when you want to send an executable program or binary file using a communication protocol that only works with text. The problem is that there are plenty of bit strings that do not translate to text fields. uuencode takes a binary file and transforms it into text allowing you to send the file via email or other mediums that only use text. Very handy.

Format:

```
uuencode existingfile decodedname

existinfile:  The binary file you want
               to disguise

decodedname:   The name of the
               transformed file
```

Example:

```
uuencode a.out newprogram.txt
```

uudecode

Converts a uuencoded file back to the original format. Very handy.

Format:

```
uudecode [filename]

filename:   This is the name of the file you want to decode.  You don't give the parameter to rename the file. The original name is embedded in the encoded file.
```

Example:

```
uudecode uu.incoming
```

Other commands worth looking at:

lpr	print to a printer
cancel	cancel a job
diff	displays differences in files
date	display date
finger	show information about a user
sftp	secure file transfer protocol
gunzip	Unix zip program
jobs	list jobs that are running
kill	Cancel a job
history	lists the last 20 or so commands
ksh	runs the Korn shell
lpq	list the status of printers
lprm	removes a print job
mesg	turns messaging of/off
pack	packs a file, see also unpack

sed	lets you prerecord commands so you make makes changes
sh	runs the Bourne shell
sort	predefined Unix sort command, very handy
spell	Looks through a text file and picks out words that are in in the UNIX dictionary
stty	sets UNIX terminal options
tail	displays the last few lines of a file (see also head)
tar	copies a file to/from an archive backup. Very handy and if you use UNIX very much you will want to get familiar with this command
unalias	removes alias reference. Used when names conflict.

unpack	restores a packed file to its original size.
vacation	automatically responds to incoming email messages by telling people you are on vacation.
wc	counts the number of lines, words, and characters in a file
who	who is using this computer. Who is logged in.