Computer Architecture

Lab 4 (Advanced Shell Features and Commands)

Overview

This lesson discusses advanced commands that can be used to perform tasks such as copying, moving, renaming, and deleting files. It will also cover advanced shell features like auto completion, wildcards, pipes, and redirection.

Commands covered in this lesson:

Command	Purpose
`mv`	Move or rename files and directories.
`cp`	Copy files and directories.
`rm`	Remove files.
`mkdir` `rmdir`	Create/remove directories.
`touch`	Update time stamps on a file.
`lsof`	List open files.
`fuser`	Display information about open files.
`cksum`	Display the checksum of a file.
`md5sum`	Display the MD5 hash of a file.
`ln`	Create links (shortcuts) to files or directories.
`alias`	Create command line aliases.
`gzip` `gunzip`	Compress/uncompress files.
`split`	Split large files into multiple pieces.
`shred`	Securely erase files.
`watch`	Periodically execute the specified command.
`env`	Display environment variables.

Glossary of terms used in this lesson:

Alias	A shortcut for a command.
Append	Add data to the end of a file instead of overwriting its contents.
Checksum	A data integrity verification algorithm.
Compression	A process used to reduce the size of files.
Interactive	Display confirmation prompts before executing a task.
Link	A shortcut to a file or directory.
MD5 Sum	An enhanced data integrity verification algorithm.
Parent Directory	Higher level directory that contains the current directory.
Pipe	A command line facility that connects the output of one command to the input of another.
Redirection	Command line facilities used to redirect the input or output of a command.
Variable	Adjustable program/environment settings stored in memory.
Verbose	Extended output from a command.
Wildcards	Symbols used to match text patterns.

Auto-Completion

Most shells support command line completion. Command line completion is used to have the shell automatically complete commands or file paths. Command line completion is activated using the Tab key on most systems and shown in the following example.

```
$ whe<TAB>
$ whereis
```

Using command line completion

In the above example typing whe and pressing the Tab key automatically completes the command whereis without having to type the entire command.

Auto-completion also works on file paths. Typing ls -l /etc/en and pressing the Tab key would auto-complete to the file /etc/environment as shown in the next example.

```
$ ls -l /etc/en<TAB>
$ ls -l /etc/environment
```

Command line completion of file names

When more than one match is found, the shell will display all matching results. In the next example, typing ls -l /etc/host and pressing Tab displays all matching files in the /etc directory.

```
$ ls -l /etc/host<TAB>
host.conf    hostname    hosts       hosts.allow  hosts.deny
```

Displaying multiple matches using file name completion

Wildcards

Wildcards are used to pattern match one against one or more text elements. They are helpful on the command line for performing bulk tasks such as listing or removing groups of files. The table below lists the different types of wildcards that can be used on the command line.

Wildcard	Function
`*`	Matches 0 or more characters
`?`	Matches 1 character
`[abc]`	Matches one of the characters listed
`[a-c]`	Matches one character in the range
`[!abc]`	Matches any character not listed
`[!a-c]`	Matches any character not listed in the range
`{tacos,nachos}`	Matches one word in the list

Types of wildcards

The asterisk (*) is the simplest and most helpful wildcard. The example below demonstrates using the asterisk wildcard to display all files that match a file name.

```
$ ls -l /etc/host*
-rw-r--r-- 1 root root  92 2008-12-23 12:53 /etc/host.conf
-rw-r--r-- 1 root root   6 2009-04-23 15:50 /etc/hostname
-rw-r--r-- 1 root root 251 2009-05-22 14:55 /etc/hosts
-rw-r--r-- 1 root root 579 2009-04-20 09:14 /etc/hosts.allow
-rw-r--r-- 1 root root 878 2009-04-20 09:14 /etc/hosts.deny
```

Listing files using the asterisk wildcard

Typing ls -l /etc/host* lists all the files in the /etc directory that start with the word host. Other examples of wildcards are demonstrated below.

```
$ ls -l /etc/hosts.{allow,deny}
-rw-r--r-- 1 root root 579 2009-04-20 09:14 /etc/hosts.allow
-rw-r--r-- 1 root root 878 2009-04-20 09:14 /etc/hosts.deny
$ ls -l /etc/hosts.[!a]*
-rw-r--r-- 1 root root 878 2009-04-20 09:14 /etc/hosts.deny
$ ls -l /etc/host?
-rw-r--r-- 1 root root 251 2009-05-22 14:55 /etc/hosts
```

Examples of other wildcards

In this example, the first command uses {allow,deny} to display all matches that end with the word allow or deny. The second command uses [!a]* to display matches that do not begin with the letter a (after the period). The third example uses the ? wildcard to match only a single character.

Pipes

Pipes (also referred to as pipelines) can be used to direct the output of one command to the input of another. Pipes are executed using the | key (usually located above the backslash key) on the keyboard.

```
$ ls -l /etc | more
total 968
-rw-r--r-- 1 root root      2975 2008-08-18 13:30 adduser.conf
-rw-r--r-- 1 root root        44 2010-04-06 16:59 adjtime
-rw-r--r-- 1 root root        51 2008-08-18 13:49 aliases
-rw-r--r-- 1 root root     12288 2009-08-28 13:39 aliases.db
drwxr-xr-x 2 root root      4096 2010-04-05 10:59 alternatives
drwxr-xr-x 7 root root      4096 2010-04-05 10:59 apache2
drwxr-xr-x 3 root root      4096 2008-08-18 13:48 apm
drwxr-xr-x 2 root root      4096 2009-08-28 13:39 apparmor
drwxr-xr-x 6 root root      4096 2008-08-18 13:47 apparmor.d
drwxr-xr-x 4 root root      4096 2010-01-25 13:44 apt
-rw-r----- 1 root daemon     144 2007-02-20 07:41 at.deny
-rw-r--r-- 1 root root      1733 2008-05-12 13:33 bash.bashrc
-rw-r--r-- 1 root root    216529 2008-04-14 20:45 bash_completion
drwxr-xr-x 2 root root      4096 2010-04-05 10:59 bash_completion.d
:
```
Using pipes on the command line

Using ls -l on the /etc directory would normally rapidly scroll the contents of the directory across the screen. Piping the output of ls -l to the **more** command displays the contents of the /etc directory one page at a time.

Another command commonly used with pipes is **grep**. The grep utility can be used to filter the output of a command or file and display matching results. The next example demonstrates piping the output of the ls command to grep to filter the results and display matches that contain the word hosts.

```
$ ls -l /etc | grep host
-rw-r--r-- 1 root root     92 2007-10-20 06:51 host.conf
-rw-r--r-- 1 root root      9 2008-08-19 15:29 hostname
-rw-r--r-- 1 root root    300 2009-12-07 09:19 hosts
-rw-r--r-- 1 root root    579 2008-08-18 13:30 hosts.allow
-rw-r--r-- 1 root root    878 2008-08-18 13:30 hosts.deny
```
Using a pipe with the grep command to filter a command's output

Redirection

The output of a command can be redirected to other locations such as a text file. Redirection is initiated by using the > character on the keyboard.

```
$ date > date.txt
$ ls -l date.txt
-rw-r--r-- 1 nick nick 29 2009-06-10 11:37 date.txt
```

Redirecting the output of the date command to a file

In the above example, the date command's output is redirected to a file called date.txt instead of being displayed on the screen. If the specified file does not exist it will automatically be created. If it does exist, it will be overwritten. To prevent overwriting a file you can use >> to append to the file as shown in the next example.

```
$ date >> date.txt
```

Appending the output of a command to a file

mv

Purpose: Move or rename files and directories.

Usage syntax: mv [OPTIONS] [SOURCE] [DESTINATION]

```
$ ls -l
-rw-r--r-- 1 nick nick    55 2009-05-20 15:32 MyFile
$ mv MyFile MyFile.old
$ ls -l
-rw-r--r-- 1 nick nick    55 2009-05-20 15:32 MyFile.old
```

Using the mv command to rename a file

The mv command moves or renames files. In the above example, MyFile file is renamed to MyFile.old using the mv command. In the next example, MyFile.old is moved to the /tmp directory.

```
$ mv MyFile.old /tmp/
$ ls -l /tmp/
-rw-r--r-- 1 nick nick    55 2009-05-20 15:32 MyFile.old
```

Moving a file to a different directory

cp

Purpose: Copy files and directories.

Usage syntax: cp [OPTIONS] [SOURCE] [DESTINATION]

```
$ cp MyFile MyFile.copy
$ ls -l
-rw-r--r-- 1 nick nick    55 2009-05-20 15:32 MyFile
-rw-r--r-- 1 nick nick    55 2009-05-20 15:32 MyFile.copy
```

Creating a copy of a file

The cp command copies files and directories. In the above example, MyFile is copied to create the MyFile.copy file.

The next example demonstrates using cp -r to recursively copy the contents of a directory.

```
$ ls -l
drwxr-xr-x 2 root root      4096 Jul  1 14:06 MyDocuments
$ cp -r MyDocuments/ MyDocuments2/
$ ls -l
drwxr-xr-x 2 root root      4096 Jul  1 14:06 MyDocuments
drwxr-xr-x 2 root root      4096 Jul  1 14:06 MyDocuments2
```

Using the -r option with cp to recursively copy a directory

After executing the cp -r command an exact copy of the specified directory is created.

rm

Purpose: Remove files.

Usage syntax: rm [OPTIONS] [FILE]

```
$ rm MyFile
$ ls -l MyFile
ls: cannot access MyFile: No such file or directory
```

Using the rm command to remove a file

The rm command removes files. In the above example, the rm command is used to remove MyFile. After executing the rm command, MyFile is deleted from the disk and no longer accessible.

Notice that no warning is given when the file is removed. This is the default behavior of rm on most systems. To change this, use the -i option as demonstrated in the next example. This will instruct the system to prompt you to verify you want to remove the file.

```
$ rm -i MyFile
rm: remove regular file 'MyFile'? y
```

Using the -i option with the rm command for interactive prompts

mkdir / rmdir

Purpose: Create/remove directories.

Usage syntax: mkdir [OPTIONS] [DIRECTORY]

```
# mkdir test
# ls -ld test/
drwxr-xr-x 2 root root       4096 Jun  4 09:00 test
```

Creating a directory with mkdir

The mkdir command creates directories. The above example demonstrates creating a directory called test. Notice the permissions section of the ls output contains a d prefix. This indicates that the item is a directory.

The rmdir command removes directories. In the next example, the rmdir command is used to remove the previously created test directory.

Usage syntax: `rmdir [DIRECTORY]`

```
$ rmdir test/
$ ls -ld test/
ls: cannot access test/: No such file or directory
```

Removing a directory using rmdir

Note | **`rmdir`** *will only remove empty directories. To remove a non-empty directory, use* **`rm -r [DIRECTORY]`** *in place of the* **`rmdir`** *command.*

touch

Purpose: Update time stamps on a file.

Usage syntax: touch [OPTIONS] [FILE]

```
$ ls -l testfile
-rw-r--r-- 1 root root 251 2009-04-21 15:50 testfile
$ touch testfile
$ ls -l testfile
-rw-r--r-- 1 root root 251 2009-05-23 14:54 testfile
$ date
Sat May 23 14:54:35 CDT 2009
```

Using the touch command to update the time stamp on a file

The touch command updates the time stamps on the specified file(s). Notice the timestamp on the file in the above example is updated to match the current time and date after executing the touch command.

If the file does not exist, the touch command will create an empty file with the specified file name, as demonstrated in the next example.

```
$ ls -l MyFile
ls: cannot access MyFile: No such file or directory
$ touch MyFile
$ ls -l MyFile
-rw-r--r-- 1 nick nick 0 2009-05-23 14:54 MyFile
```

Creating a new empty file with the the touch command

lsof

Purpose: List open files.

Usage syntax: lsof [OPTIONS] [NAME]

```
# lsof /etc/hosts
COMMAND    PID   USER   FD   TYPE DEVICE SIZE    NODE NAME
tail       12793 nick    3r   REG    8,1  256 1777676 /etc/hosts
```

Using the lsof command to display information about an open file

The lsof command displays information about open files. Executing the lsof command with no arguments will display all open files on the system. Specifying the name of an open file will display information about who is using the file. In the example above, lsof displays which user is using the /etc/hosts file along with other helpful information such as the command name and PID number.

fuser

Purpose: Display information about open files.

Usage syntax: fuser [OPTIONS] [DIRECTORY/FILE]

```
$ fuser -v /home/nick/ShoppingList.txt
28528c(nick)
                     USER        PID  ACCESS  COMMAND
ShoppingList.txt:    nick      14044  ..c..   tail
```

Displaying information about open files with fuser

fuser is a helpful program for identifying the person or program that is using a file. In the example above the fuser command displays the user, process id, and command currently using the ShoppingList.txt file.

cksum

Purpose: Display the checksum of a file.

Usage syntax: cksum [OPTIONS] [FILE]

```
$ cksum ubuntu.iso
3212199805 730554368 ubuntu.iso
```

Displaying the checksum of a large file

The cksum command displays the checksum of the specified file. It is typically used to verify the integrity of files transferred across a network connection. In the above example, the checksum for a downloaded Ubuntu Linux CD image is displayed. The resulting checksum can be compared to the checksum from the original file to ensure it arrived without errors. The table below describes the output fields of the cksum command.

Checksum	File Size	File Name
3212199805	730554368	ubuntu.iso

Output fields of the cksum command

md5sum

Purpose: Display the MD5 hash of a file.

Usage syntax: md5sum [OPTIONS] [FILE]

```
$ md5sum ubuntu.iso
cace6ea9dde8dc158174e345aabe3fae  ubuntu.iso
```

Displaying the md5 hash of a large file using the md5sum command

The md5sum command computes the MD5 sum (sometimes referred to as the hash) of the specified file. It is similar to the previously discussed cksum command except more intensive. MD5 hashes are the equivalent of a digital fingerprint and are not likely to be duplicated or padded in the same way that cksum hashes (in some rare instances) can.

ln

Purpose: Create links (shortcuts) to files or directories.

Usage syntax: ln [OPTIONS] [TARGET] [LINK]

```
$ ln -s TheSourceFile ThisIsTheLink
$ ls -l
-rw-r--r-- 1 nick nick 14 2009-05-23 10:16 TheSourceFile
lrwxrwxrwx 1 nick nick 13 2009-05-23 10:18 ThisIsTheLink -> TheSourceFile
```

Creating a link to a file using the ln command

The ln command creates links to files or directories. A link is the command line equivalent of a shortcut. In the above example, a link to a file called TheSourceFile is created. Notice the link in the above example has an l prefix in the permissions section. This indicates that the file is a link.

The default operation of the ln command on most systems creates what is known as a hard link. Hard links have two major limitations:

1. Hard links cannot refer to directories

2. Hard links cannot span multiple file systems/disks

Symbolic links are more commonly used today to overcome the shortfalls of hard links. They are created when using the -s option with the ln command. This is the recommended way to create a link as it will not suffer from the same limitations of a hard link.

Note	*Editing a symbolic link file is the same as editing the source file, but deleting the symbolic link does not delete the source file.*

Common usage examples:

`ln [SOURCE] [TARGET]`	Create a hard link to the specified target
`ln -s [SOURCE] [TARGET]`	Create a symbolic link to the specified target

alias

Purpose: Create command line aliases.

Usage syntax: alias [OPTIONS] [COMMAND]

```
$ alias rm="rm -i"
$ rm TestFile
rm: remove regular empty file 'TestFile'? y
```

Creating a command alias

The alias command creates command line aliases. This allows you to abbreviate a long command string to something simple. In the above example, the rm command is aliased to be rm -i so that every time the rm command is executed the -i option is automatically included (without having to type it).

Executing alias with no arguments will display all currently defined aliases, as demonstrated in the next example.

```
$ alias
alias cp='cp -i'
alias l='ls -l'
alias rm='rm -i'
```

Displaying all defined aliases

Tip	*Aliases are lost when you log out. The* **`unalias`** *command can be used to delete aliases without having to logoff. To make an alias permanent you must add it to* `/etc/profile` *or* `.*profile` *file in the user's home directory.*

gzip / gunzip

Purpose: Compress/uncompress files.

Usage syntax: gzip [OPTIONS] [FILE]

```
# ls -lh BigFile
-rw-r--r-- 1 root root 3.0M 2010-05-20 14:02 BigFile
# gzip BigFile
# ls -lh BigFile.gz
-rw-r--r-- 1 root root 433K 2010-05-20 14:02 BigFile.gz
```

Using gzip to compress a file

gzip is a simple compression utility found on most Linux and BSD systems. In the above example gzip is used to reduce the size of the BigFile file by compressing it into a .gz archive.

The gunzip (or gzip -d) command uncompresses gzip archives as demonstrated in the next example.

Usage syntax: `gunzip [OPTIONS] [FILE]`

```
$ gunzip BigFile.gz
# ls -lh BigFile
-rw-r--r-- 1 root root 3.0M 2010-05-20 14:02 BigFile
```

Uncompressing a file with gunzip

split

Purpose: Split large files into multiple pieces.

Usage syntax: split [OPTIONS] [FILE] [OUTPUT]

The split command splits large files into multiple pieces. The next example demonstrates splitting the large ubuntu.iso file into several 100MB pieces (as specified by the -b 100M parameter). In this example, the split command will create the required number of 100MB files with an incrementing extension.

```
$ ls -l ubuntu.iso
-rw-r--r-- 1 nick nick 671686656 2009-10-27 12:07 ubuntu-9.10.iso
$ split -d -b 100M ubuntu.iso ubuntu.iso.
$ ls -lh ubuntu*
-rw-r--r-- 1 nick nick 641M 2009-10-27 12:07 ubuntu-9.10.iso
-rw-r--r-- 1 nick nick 100M 2010-04-11 11:44 ubuntu.iso.00
-rw-r--r-- 1 nick nick 100M 2010-04-11 11:44 ubuntu.iso.01
-rw-r--r-- 1 nick nick 100M 2010-04-11 11:44 ubuntu.iso.02
-rw-r--r-- 1 nick nick 100M 2010-04-11 11:44 ubuntu.iso.03
-rw-r--r-- 1 nick nick 100M 2010-04-11 11:44 ubuntu.iso.04
-rw-r--r-- 1 nick nick 100M 2010-04-11 11:44 ubuntu.iso.05
-rw-r--r-- 1 nick nick  41M 2010-04-11 11:44 ubuntu.iso.06
```

Using the split command to split a large file into multiple pieces

The cat command can be used to rejoin the split files as demonstrated in the next example.

```
$ cat ubuntu.iso.* > ubuntu-joined.iso
$ ls -lh *.iso
-rw-r--r-- 1 nick nick 641M 2009-10-27 12:07 ubuntu-9.10.iso
-rw-r--r-- 1 nick nick 641M 2010-04-11 11:47 ubuntu-joined.iso
```

Combining split files using the cat command

shred

Purpose: Securely erase files.

Usage syntax: shred [OPTIONS] [DIRECTORY/FILE]

```
$ shred -u SecretPlans.txt
$ ls -l SecretPlans.txt
ls: cannot access SecretPlans.txt: No such file or directory
```

Using the shred command to securely overwrite a file

The shred command securely overwrites (and optionally deletes) files. In the above example, executing shred -u securely overwrites and removes the SecretPlans.txt file from the disk.

watch

Purpose: Periodically execute the specified command.

Usage syntax: watch [OPTIONS] [COMMAND]

```
$ watch -n 10 who
Every 10.0s: who                                     Sat May 23 11:00:19 2009
steve      tty1        2009-05-21 10:24
root       tty2        2009-05-21 10:44
nick       tty3        2009-05-23 12:20
```

Executing the who command every 10 seconds using watch

watch periodically runs the specified command. It is a helpful program for monitoring the output of a command over a period of time. In the above example, watch -n 10 is used to execute the who command every 10 seconds.

env

Purpose: Display environment variables.

Usage syntax: env [OPTIONS]

```
$ env
TERM=xterm
SHELL=/bin/bash
USER=nick
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:
LANG=en_US.UTF-8
HOME=/home/nick
...
```

Output of the env command

The env command displays your defined environment variables. These variables hold information for common account settings like the location of a user's home directory and the type of shell they use by default.

The following table describes the most common environment variables used on Unix, Linux, and BSD systems.

Variable	Function
EDITOR	Specifies the user's preferred text editor
HISTFILE	Location of the user's command line history file
HISTFILESIZE	Specifies the number of commands to save in HISTFILE
HOME	Path to the user's home directory
LANG	Specifies the user's language locale settings
MAIL	Path to the user's mail file
PATH	Path to search for binary programs
PS1	Customized shell prompt settings
SHELL	Location of the user's shell
TERM	Specifies the type of terminal being used
USER	User's username

Common environment variables used on Unix, Linux, and BSD systems