CSE 314
Shell Commands

Introduction

- A good command line interface is a marvelously expressive way of communicating with a computer in much the same way the written word is for human beings.
- Graphical user interfaces make easy tasks easy, while command line interfaces make difficult tasks possible
- The *shell* is a program that takes keyboard commands and passes them to the operating system to carry out.
- Almost all Linux distributions supply a shell program from the GNU Project called bash.

Starting up

Launch terminal. You'll see something like:

username@machinename~$

Here ~ indicates the current working directory and *$* indicates a normal user. If the last character is # then it means a superuser.

(These will be clear to you very soon!)

Hello World

echo Hello World

echo ‘Hello World’

echo “Hello World”

It is not essential to surround the string with quotes. But doing so also doesn’t repeat them on the screen.

Actually *echo* can do lots of other powerful things.

They will come later.

Exploring

pwd

prints the current directory

cd <directoryname>

- moves to *directoryname*
- Path can be absolute or relative (be careful!!)
- . means the current directory
- .. means the parent of the current directory
- Default directory is HOME
- HOME also represented by ~
- Can return HOME from anywhere by entering

cd

Exploring

ls [OPTION] [FILE]

- Lists information about directory or file
- In Linux hidden files have names starting with . and they are not shown by *ls*
- *ls –a* lists the hidden files also
- *ls –l* lists in details
- *ls –R* recursively lists subdirectories
- *ls –S* sorts files by size
- There are lots of others options and it is impossible to exhaustively list them all. We'll learn about an easier technique in the next slide

My friend *man*

man command

- Shows the manual for *command*
- You should frequently use it besides Google.
- Now you can see all the options associated with *ls* by

  man ls

- You can learn more from

  man man

Files and Directories

mkdir [OPTION] <directory1> <directory2> ….

- Makes each directory if it already doesn't exist
- *-p* Overwrite directory even if it exists.
- *-v* Shows a verbose description

Files and Directories

cp

- Copy files and directories
- *cp SOURCE DEST* copy SOURCE to DEST (Remember the order!)
- *cp SOURCE… DIRECTORY* copy multiple SOURCEs to DIRECTORY
- *-r* copy directories recursively
- *-i* interactive i.e. prompts before overwriting

Files and Directories

rm [OPTION]... FILE...

- Remove each of FILEs if exists
- -f ignores non-existent files, never prompts
- -i interactive i.e. prompts before deleting
- -r remove contents recursively (Can be used to remove a directory)
- -v shows verbose description

Files and Directories

mv

- *mv SOURCE DEST* moves file from SOURCE to DEST
- *mv SOURCE… DIRECTORY* moves files to DIRECTORY
- *-i* interactive i.e. prompts before overwriting

pushd

popd

Permissions

- Linux is a multiuser system.
- A user may *own* files and directories. When a user owns a file or directory, the user has control over its access.
- Users can, in turn, belong to a *group* consisting of one or more users who are given access to files and directories by their owners.
- In addition to granting access to a group, an owner may also grant some set of access rights to everybody, which in Unix terms is referred to as the *world*.
- To find out information about your identity, use the *id* command:

Permissions

- Access rights to files and directories are defined in terms of *read access*, *write access*, and *execution access*.
- If you run the *ls –l* command, in the first column you will see 10 characters which might look something like this:

drw-rw-r–

Permissions

drw-rw-r–

- These first 10 characters of the listing are the *file attributes*
- The first of these characters is the *file type*.
- The remaining nine characters of the file attributes, called the *file mode,* represent the read, write, and execute permissions for the file's owner, the file's group owner, and everybody else.

Permissions

Example:

-rw-r--r-- means it is a file which has read and write permissions for the owner, read permission for group and read permission for others.

Permissions

chmod

- change the mode (permissions) of a file or directory
- only the file's owner or the superuser can change the mode of a file or directory.
- *chmod* supports two distinct ways of specifying mode changes: octal number representation and symbolic representation

Permissions

chmod

- Symbolic notation is divided into three parts: whom the change will affect, which operation will be performed, and which permission will be set.
- To specify who is affected, a combination of the characters *u*, *g*, *o*, and *a* is used
- The operation may be
  a + indicating that a permission is to be added, a - indicating that a permission is to be taken away
- Permissions are specified with the r, w, and x characters

chmod go+rw filename

adds read and write permissions to group and others.

Permissions

chmod

- With octal notation we use octal numbers to set the pattern of desired permissions.
- Since each digit in an octal number represents three binary digits, this maps nicely to the scheme used to store the file mode

chmod 755 filename

755 => 111 101 101 => rwx r-x r-x

Permissions

chown username <file or directory>

- Changes the owner of a file or directory

chgrp groupname <file or directory>

- Changes the group of a file or directory

Permissions

su [user]

switch user

sudo command

execute command as a superuser

File Viewing

more

less

head

tail

cat [FILE…]

concatenate contents of files

wc [FILE…]

count the line, word and bytes

grep PATTERN [FILE…]

searches FILE for a match with PATTERN

I/O Redirection

- By default, output goes to the screen and input comes from the keyboard (we'll deal with inputs later), but with I/O redirection we can change that.
- I/O redirection allows us to change where output goes and where input comes from.

I/O Ridirection

< get input from a file other than keyboard

> output to a file other than the screen

>> append output to a file

ls –l > out.txt

Pipelines

- Input of a command may come from output of another command
- Can be extended to several stages

ls /bin /usr/bin | sort | uniq | less

Finish

exit

Thank you