UNIX Lecture Notes

Professor Glenn Luecke
Spring 2017

References

The material presented has been taken from several sources with the primary source being the unix tutorials from the University of Surrey, http://www.ee.surrey.ac.uk/Teaching/Unix/ and from the book “UNIX in a Nutshell” by Arnold Robbins and http://people.ischool.berkeley.edu/~kevin/unix-tutorial/toc.html.

BACKGROUND

A computer uses a set of programs, generally called the operating system, to manage its hardware resources (memory, disks, displays, input devices, etc.) on behalf of the user. UNIX is an example of such a system. It was originally developed as a research project at AT&T Bell Labs in 1969. Unix is free, portable, multi-user and multi-tasking (multiple programs can run at the same time without problems). There are many variations of unix: e.g. IBM’s AIX, Fedora Linux, Red Hat Linux, MacOS X, etc. Unix is a standard implemented by different vendors whereas Linux is an open source system.

The 3 Parts of UNIX

1. **The kernel:** The kernel allocates time and memory to programs and handles the filestore and communications in response to system calls.
2. **The shell:** The shell acts as an interface between the user and the kernel. The default is the bash-shell. There is also a c-shell. To determine what shell you are in issue “echo $0”. To change to bash, tcsh issue “bash”, “tcsh” respectively. I recommend using the bash shell since it is commonly used for HPC systems.
3. **The programs:** For example, when one issues “ls” then this executes the “ls” program.

PROCESSES, FILES AND DIRECTORIES

1. A process is an executing program identified by a unique PID (process identifier).
2. A file is a collection of data.
3. Files are grouped together in the directory structure.
4. The file-system is arranged in a hierarchical structure, like an inverted tree with the top of the hierarchy usually called root.

Obtaining an Account on the Student Cluster

The instructor requests accounts for the students in his class.

Logging Onto the Student Cluster

To logon to hpc-class, issue: **ssh username@hpc-class.its.iastate.edu** The “username@” can be omitted if the username on the machine from which you are issuing the ssh is the same as that on hpc-class. If you need an X-session, change “ssh” to “ssh -X” when logging on. The vi, emacs and xemacs editors are available on hpc-class.

Copying Data to/from the Student Cluster

Use “scp” or “rsync -ae ssh” to copy files to and from hpc-class. “rsync” should be used for large files and has the advantage that it will only copy updated files. In the following examples one can replace “scp” or “scp -r” by “rsync -ae ssh”.

Example 1: Suppose you are logged on a machine that is not hpc-class and would like to copy the file mydir/prog.f90 from hpc-class to ./mydir/prog.f90 in your current working directory. This can be done by issuing the following in your current working directory on the non-hpc-class machine:

```
scp username@hpc-class.its.iastate.edu:mydir/prog.f90  ./mydir/prog.f90
```

Example 2: To copy the entire “mydir” directory in example 1, issue the following from your current working directory:

```
scp -r username@hpc-class.its.iastate.edu:mydir  ./mydir
```

Unix Commands

Hidden (dot) files and directories

A hidden/dot file or directory is a file or directory that begins with a “.”. They are commonly used for storing user preferences or preserving the state of a utility. They are frequently created implicitly by using various utilities. They are not a security mechanism because access is not restricted. Usually the intent is simply not "clutter" the display of the contents of a directory listing with files the user did not directly create. Issuing **ls -a** will provide a listing of both regular and hidden files.

The * indicates that I find the command to be useful

The man (manual) page *

- % man unix-command [ret] provides detailed information about the specified unix command. For example:
- % man ssh [ret] will bring up the man pages for ssh.
- % man ifort [ret] will bring up the man pages for Intel’s Fortran compiler
- To page down, hit [ret].
- To leave the man page, enter q for “quit”.
- To perform a search, enter “/keyword” [ret]; entering “n” will bring the next listing of the keyword. Entering “u” will move the cursor up.

The ls (list) command *

- % ls [return] lists all files in working directory excluding those starting with a dot (hidden files)
- % ls -a [return] lists all files including those that begin with a dot.
- % ls -l [ret] gives the long information about the files and directories
- % ls ex* [ret] lists all files beginning with ex
- % ls m525/ [ret] lists all files in the directory m525

The cd (change directory) command *

- % cd examples [ret]
- % ls [ret] you will see all the files in the examples directory
- % cd [ret] moves one to the home directory
- % cd .. [ret] moves one up one level

The ~/ reference the home directory *

For example, when issued from the home directory, the following are the same.

- % cd examples [ret]
- % cd home/grl/examples [ret]
- % cd ~/examples [ret]

The pwd (print working directory) command *

- % pwd [ret]

The cp (copy) command *

General advice is to us the cp command when copying within a since machine and use the scp command when copying files between different machines.

- % cp file1 file2 [ret] copies file1 in working directory to file2 in working dir.
- % cp ~/temp1 . [ret] copies temp1 from the home directory to the working directory.
- % cp ~/temp1 temp2 [ret] copies temp1 in home dir to temp2 in working directory.
- % cp directory1/* directory2 [ret] copies everything from directory 1 to directory2.

The scp (secure copy) command *

Used for copying between machines. See above for examples.

The mv (move) command *

Behaves like the cp command above but moves the file instead of copying it.

- % mv file1 file2 [ret] moves file1 to file2 overwriting original file2.

The rm (remove) command *

- % rm file1 [ret] removes/deletes file1 in working directory.
- % rm file* [ret] removes all files in working directory starting with file
- % rm -r mydir [ret] removes the directory “mydir” and all files in it.

The mkdir (make directory) command *

- % mkdir math525 [ret]

The rmdir (remove directory) command *

- % rmdir math [ret] removes/deletes directory math only if it is empty.
- % rm -r math [ret] removes/delete directory math and everything in it.

The clear (clear screen) command *

- % clear [ret] clears screen of all text and put prompt in the upper left corner of the screen.

The cat (concatenate) command *

One can use cat to create files (or one can use the vi editor). For example:
% cat > file1 [ret]
% red rot
% blue blau
% green gruen
% and now push down the control key and enter d: ^d If one now issues:
% cat file1 [ret] then you will see this file.
% cat file1 file2 [ret] concatenates file1 and file2 and writes to standard out (one's screen).
% cat file1 file2 > file3 [ret] concatenates file1 and file2 and writes to file3.
% cat >> file1 [ret] allows one to add to file1.
% cat file1 file2 file3 [ret] concatenates the 3 files and writes to standard out.

The head (header) command

- % head file1 [ret] lists the first 10 lines of file1
- % head -15 file1 [ret] lists the first 15 lines of file1

The tail command

- % tail file1 [ret] lists the last 10 lines of file1
- % tail -15 file1 [ret] lists the last 15 lines of file1

The | (pipe) command - Some find this very useful, but I don't use it

The symbol | is the unix pipe symbol that is used on the command line. What it means is that the standard output of the command to the left of the pipe gets sent as standard input of the command to the right of the pipe. Note that this functions a lot like the > symbol used to redirect the standard output of a command to a file. However, the pipe is different because it is used to pass the output of a command to another command, not a file.

The less command

- % less file [ret] for viewing the contents of file but allows backward (hit the b key) and forward movement (hit the n key) through the file. Hit the q key to quit the session. Hitting the return key causes scrolling one line at a time. The spacebar scrolls one page at a time. For a string search during a less session enter /keyword [ret].

The more command

- % more file [ret] for viewing the contents of file but using the less command is better since the less command has more functionality.

The grep command *

is used for searching

- % grep ‘ = sqrt’ file.f90 [ret] lists lines that have the listed string
- % grep ‘ the ‘ file1 file2 [ret] searches file1 and file2
- % grep ‘ the ‘ tmp* [ret] searches all files in working directory beginning with tmp
- % grep ‘ the ‘ rmp* | less [ret] to view a large number of matches.
- % grep ‘A(i) = ‘ file.f90 [ret] case sensitive
- % grep –i ‘A(i) = ‘ file.f90 [ret] not case sensitive
  - -v display those lines that do NOT match
  - -n precede each matching line with the line number
  - -c print only the total count of matched lines
- % grep –ivc ‘A(i) =’ file.f90 [ret] showing how to combine options

The > command *

Directs output from standard out to a specified file

% wc –w file > file1 writes the number of words in file to file1

The date command

- % date [ret] lists the date and time

The wc (word count) command

- % wc –w file [ret] returns the number of words in file
- % wc –l file [ret] returns the number of (new)lines in file
- % wc file [ret] returns the number of (new)lines, words, and bytes in file.

UNIX FILE PERMISSIONS

See: http://www.dartmouth.edu/~rc/help/faq/permissions.html

Every user on a Unix system has a unique username, and is a member of at least one group (the primary group for that user). This group information is held in the password file (/etc/passwd). A user can also be a member of one or more other groups. The auxiliary group information is held in the file /etc/group. Only the administrator can create new groups or add/delete group members (one of the shortcomings of the system).

Every directory and file on the system has an owner, and also an associated group. It also has a set of permission flags which specify separate read, write and execute permissions for the 'user' (owner), 'group', and 'other' (everyone else with an account on the computer) The 'ls' command shows the permissions and group associated with files when used with the -l option.

The following is an example from hpc-class where tmpdir is a directory with the first level having files ex1.f90, ex2.f90, a.out (an executable) and subdir (a directory). The directory subdir has one level with one member ex3.f90. Thus the path for each of these files is

/home/grl/tmpdir/ex1.f90
/home/grl/tmpdir/ex2.f90
/home/grl/tmpdir/a.out
/home/grl/tmpdir/subdir/ex3.f90

If one now issues **ls -l** or **ls -lh** from /home/grl/tmpdir, the following is displayed. Issuing **ls -lh** then it will give the data set size in bytes, i.e. "human readable", so this was used in the following.

```
total 716K
-rwxr-xr-x  1 grl dusers 708K Jul 24 14:02 a.out
-rw-r--r--  1 grl dusers   90  Jul 25 12:15 ex1.f90
-rw-r--r--  1 grl dusers   35  Jul 24 12:07 ex2.f90
drwxr-xr-x 2 grl dusers   20  Jul 24 12:08 subdir
```

The "total 716K" means the total number of system blocks used for the listed files. On the student cluster 1 block is 1KB, i.e 1024 Bytes.

In the above there is one line for each file and directory where the columns mean the following:

column 1 - if it is a normal file, d if it is a directory, and s if it is a socket file (a special file used for inter-processor communication)

columns 2,3,4 - read (r), write (w), execute (x) permission, and no permission (-) for the file Owner.

columns 5,6,7 - read (r), write (w), execute (x) permission, and no permission (-) for the Group.

columns 8,9,10 - read (r), write (w), execute (x) permission, and no permission (-) for Other.

columns 11,12,13 gives the "link" count. Notice above the directory "subdir" has a link count of 2 indicating it is not a regular file but a directory in tmpdir.

The next several columns lists the owner of he different files/directories.

The next several columns lists the group name. In this case the group name is "dusers".

The next several columns lists the size of the file.

The next several columns lists the date when the file was last modified.

The last several columns list the name of the file/directory.

Values for columns 1-10

- in any position means that flag is not set, i.e. cannot read, write or execute.
r file is readable by Owner, Group or Other.
w file is writeable. On a directory, write access means you can add or delete files.

x file is executable. On a directory means you can list the files in that directory.
s in a place where 'x' would normally go is called the set-UID or set-groupID flag.

The default file permissions (umask):

Each user has a default set of permissions which apply to all files newly created by that user, unless the software explicitly sets something else. This is often called the "umask", after the command used to change it. It is either inherited from the login process, or set in the .cshrc or .login file which configures an individual account, or it can be run manually. Typically the default configuration is equivalent to issuing **umask 22** which produces permissions of:

-rw-r--r-- for regular files, or
drwxr-xr-x for directories.

This gives read access to files and lookup access to directories to Owner, Group and Other. This is the default for all ITS clusters.

When working with group-access files and directories, it is common to use **umask 2** which produces permissions of:

-rw-rw-r-- for regular files, or
drwxrwxr-x for directories.

For private work, use **umask 77** which produces permissions:

-rw------- for regular files, or
drwx------ for directories.

The logic behind the number given to umask is not clear. The above is partially taken from http://www.computerhope.com/unix/uumask.htm

Use chmod to change permission flags

http://catcode.com/teachmod/
http://www.computerhope.com/unix/uchmod.htm

The chmod command allows one to change file/directory permissions (sometimes called file modes) for files and directories. The general form is

chmod options permissions filename

If no *options* are specified, **chmod** modifies the permissions of the file specified by *filename* to the permissions specified by *permissions*.

Example. Suppose you are the owner of file named **myfile** and you want to set permissions so that:

1. the user can read, write and execute myfile
2. the members of your group can read and execute it, and
3. Others may only read it

then issue

chmod u=rwx,g=rx,o=r myfile

where u, g and o stand for user, group and other. One can equivalently do this using the octal permission notation by issuing

chmod 754 myfile

where digits 7, 5 and 4 each individually represent the permissions of the user, group, and other in that order. Each digit is a combination of the numbers 4, 2, 1, and 0:

4 stands for read
2 stands for write
1 stands for execute
0 stands for “no permission”.

So 7 is the combination of 4+2+1 (read, write, and execute), 5 is 4+0+1 (read, no write, and execute), and 4 is 4+0+0 (read, no write, and no execute).

One may also use the +, -, and = operators with the chmod command;

1. The + operator causes the permissions selected to be added to the existing permissions of each file.
2. The - operator causes them to be removed.
3. The + operator causes them to be the only permissions that the file has.

The following letters select the new permission flags as follows:

r - read
w - write.
x - execute (access for directories).
X - execute only if the file is a directory or already has execute permission for some user.
s - set user or group ID on execution.
t – sticky.
u - the permissions granted to the user who owns the file.
g - the permissions granted to other users who are members of the file’s group.
o - permissions granted to users that are in neither of the two proceeding categories.
a - permissions granted to all, i.e. u, g and o.

Additional examples

1. chmod g+rwx myfile - gives group read, write and execution permission to "myfile", leaving all other permission flags alone.
2. chmod g+r myfile - gives group read permission to "myfile", leaving all other permission flags alone.
3. chmod g-rw myfile – removes group read and write access for “myfile”.
4. chmod -og= myfile - removes all permissions for myfile from other and group.
5. chmod og=r myfile - allows group and other to read myfile
6. chmod -R g+rwx mydir - gives group read, write and execution permission to everything in directory "mydir", leaving all other permission flags alone.

7. chmod u=rwx,go= privatefile - gives user read, write and execute access and revokes all group and other access to file 'privatefile'