Files and Directories @ NERSC
Unix Commands

Files and Directories

- What is a file?
  - a container for ordered data
  - persistent (stays around) and accessible by name
- Unix files
  - regular Unix files are pretty simple
    - essentially a sequence of bytes
    - can access these bytes in order
  - Unix files are identified by a name in a directory
    - this name is actually used to resolve the hard disk name/number, the cylinder number, the track number, the sector, the block number
      - you see none of this
    - it allows the file to be accessed

Files and Directories

- Unix files come in other flavors as well, such as
  - Directories
    - a file containing pointers to other files
    - equivalent of a “folder” on a Mac or Windows
  - Links
    - a pointer to another file
    - used like the file it points to
  - Devices
    - access a device (like a soundcard, or mouse, or ...) like it is a file

Directories

- Current Working Directory
  - the directory you are looking at right now
  - the shell remembers this for you

- To determine the Current Working Directory, use the command pwd (Print Working Directory)

  Use: pwd

  Result: print the current working directory

Directories

- Moving about the filesystem
  - Use the "cd" (Change Directory) command to move between directories and change the current directory

    Use: cd /project/projectdirs/astro250

    Result: Makes /project/projectdirs/astro250 the current working directory – this is the NERSC GFS
- Listing the contents of a directory
  - Use the "ls" (LiSt directory) command to list the contents of a directory

```
[nugent:/global/homes/n/nugent] ls -F
tmp/      a.out*      smit.script    nyx@
```

Directories

- The upside-down tree
  - the Unix filesystem is organized like an upside-down tree
    - at the top of the filesystem is the root
      - write this as a lone slash: **/**
    - For example, you can change to the root directory:

    [nugent:/global/homes/n/nugent] cd /

    Do you ever need to go here?…not likely.

    Other important NERSC directories:

    $CSCRATCH – cori scratch seen on all systems for now

    $SCRATCH – local scratch, only current machine (edison, or cori)

    $TMPDIR – temporary directory – often used by codes

Directories

- You can locate a file or directory by this way:
  - look at the first character of the pathname
    - / start from the root
    - . start from the current directory
    - .. start from the parent directory
    - ~ start from a home directory
    - Or you start from the current directory going down to the subdirectories in the pathname, until you complete the whole pathname.

mkdir/rmdir – for making/removing directories

touch – for making an empty file

Directories

- Some standard directories and files in a typical Unix system (like carver.nersc.gov)
  - / the root
  - /bin BINaries (executables)
  - /dev DEVices (peripherals)
  - /devices where the DEVICES really live
  - /etc startup and control files
  - /lib LIBraries (really in /usr)
  - /opt OPTional software packages
  - /proc access to PROCesses
  - /sbin Standalone BINaries
  - /tmp place for TeMPorary files
  - /global/homes/[a-z]
    where home directories are mounted

Directories

- /usr — USeR stuff
- /usr/bin — BINaries again
- /usr/include — include files for compilers
- /usr/lib — LIBraries of functions etc.
- /usr/local — local stuff
- /usr/local/bin — local BINaries
- /usr/local/lib — local LIBraries
- /usr/openwin — X11 stuff
- /usr/sbin — sysadmin stuff
- /usr/tmp — place for more TeMPorary files
- /var — VARiable stuff
- /var/mail — the mail spool

Pathnames

- A typical Unix file system spans many disks
  - As a user you don't know or need to know which physical disk things are on
    - in fact, you don't even know which machine they are attached to: disks can be "remote" (eg: your home directory is stored on a disk attached to a server in the machine room)
    - Look at the *df* command to see different disks and space used
  - Inside each directory may be more directories

- The Absolute Path
  - to identify where a file is, string the directories together
    - separating names with slashes:
    - e.g. /global/homes/n/nugent
    - this is the absolute path for my home directory
    - lists everything from the root down to the directory you want to specify
    - Key for running codes in the batch system is knowing where you are…
    - ~ (tilde for your home directory… not so useful)

Some Useful Unix File Commands

- man command / info command Most important one
  - RTFM….

- ls (with options, -F, -lt, -a, …)
- du (with options –sh, -h) – disk used
  - don't do this in home dir ;-)
- df (with option –h) – disk free
- prjquota (followed by name of project) – project quota
- myquota - my own quota

- mkdir –p – makes a directory and all the subdirectories

Files

- Almost any character is valid in a file name
  - all the punctuation and digits
  - the one exception is the / (slash) character
  - the following are not encouraged
    - ? * [ ] “ ” ’ ( ) & : ; !
  - the following are not encouraged as the first character
    - - ~
  - control characters are also allowed, but are not encouraged
- UPPER and lower case letters are different
  - A.txt and a.txt are different files

Hard and Symbolic Links

- When a file is created, there is one link to it.
- Additional links can be added to a file using the command ln. These are called hard links.
- Each hard link acts like a pointer to the file and are indistinguishable from the original.

```
[nugent:] echo “This is junk.” > readme.txt
[nugent:] ls –lia readme.txt
[nugent:] ln readme.txt unix_is_easy
[nugent:] ls –lia
```

- There is only one copy of the file contents on the hard disk, but now two distinct names!

Hard and Symbolic Links

- A symbolic link is an indirect pointer to another file or directory.
- It is a directory entry containing the pathname of the pointed to file.

```
[nugent:] ln -s  /usr/local/bin  ubin
```

These are very useful for the project and scratch directories.

Hard and Symbolic Links

- Two hard links have the same authority to a file
  - Removing any of them will NOT remove the contents of the file
  - Removing all of the hard links will remove the contents of the file from the hard disk.
- A symbolic link is just an entry to the real name
  - Removing the symbolic link does not affect the file
  - Removing the original name will remove the contents of the file – and your symbolic link is foobarred.
- Only super users can create hard links for directories
- Hard links must point to files in the same Unix filesystem

Useful Unix commands

- set – in bash shows all of your environment variables
- module – at NERSC is the way you load software packages.
  - module list : This lists all the modulefiles that are currently loaded into your environment.
  - module avail : This option lists all the modulefiles that are available to be loaded. Notice that many of them have version numbers associated with them and that where there is more than one version, one is labeled as the default. You can also restrict this command to a single package; for example,
  - module avail visit
  - module load package: This loads a modulefile(s)
  - module unload package: This unloads a modulefile(s)

Useful Unix Commands

- module display – Shows what it will do to your environment

I usually load all my module files from my .bashrc.ext

Standard I/O

- Standard Output (stdout) 1
  - default place to which programs write
- Standard Input (stdin) 0
  - default place from which programs read
- Standard Error (stderr) 2
  - default place where errors are reported
- To demonstrate -- cat
  - Echoes everything you typed in with an <enter>
  - Quits when you press Ctrl-d at a new line -- (EOF)

Redirecting Standard Output

- cat file1 file2 > file3
  - concatenates file1 and file2 into file3
  - file3 is created if not there
- cat file1 file2 >! file3
  - file3 is clobbered if there
- cat file1 file2 >> file3
  - file3 is created if not there
  - file3 is appended to if it is there
- cat > file3
  - file3 is created from whatever user provides from standard input

Redirecting Standard Error

- Generally direct standard output and standard error to the same place:

  [nugent:] > cat myfile >& yourfile

  - If myfile exists, it is copied into yourfile
  - If myfile does not exist, an error message

    cat: myfile: No such file or directory

    is copied in yourfile

/dev/null

- /dev/null
  - A virtual file that is always empty.
  - Copy things to here and they disappear.
    - cp myfile /dev/null
    - mv myfile /dev/null
  - Copy from here and get an empty file.
    - cp /dev/null myfile
  - Redirect error messages to this file

Useful Unix Commands

find – this is the best way to find files. Only tricky part is to figure out how to do things with them afterwards.

[nugent:] find . –name foo

Redirection can be quite useful...

```
echo “This is junk.” > junk.dat
echo “This is more junk.” >> junk.dat
echo “This is less junk.” > junk.dat
cd ~/
find . –name foo
find . –name foo 2> /dev/null
find . –name foo >> ~/junk.dat 2>&1
```

Filters

- grep patternstr:
  - Read stdin and write lines containing patternstr to stdout
- wc:
  - Count the number of chars/words/lines on stdin
  - Write the resulting statistics to stdout
- sort:
  - Sort all the input lines in alphabetical order and write to the standard output.

Pipes

- The pipe:
  - Connects stdout of one program with stdin of another
  - General form:

    command1 | command2

    - stdout of command1 used as stdin for command2
  - Example:

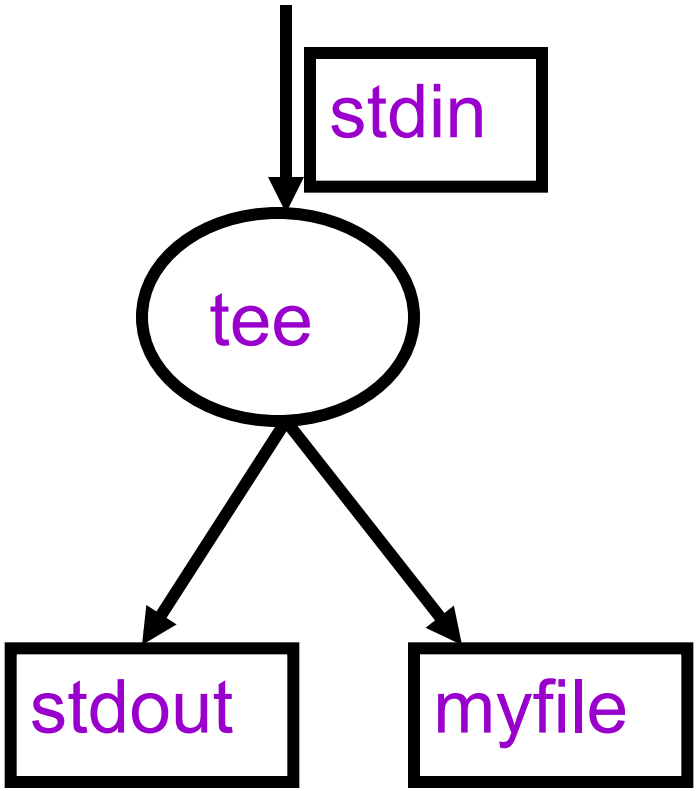
    [nugent:] > cat readme.txt | grep unix | wc -l
- An alternative way (not efficient) is to:

  [nugent:] > grep unix < readme.txt > tmp

  [nugent:] > wc -l < tmp
- Can also pipe stderr: command1 |& command2

The tee Command

- tee - replicate the standard output
  - cat readme.txt | tee myfile



File Permissions

- We should also know how to manipulate file persmission and groups for directories and files, use ls –l to see everything:
  - chmod go+rwX file.dat (give group and other
  - chgrp group file
  - chown owner file
  - use –R to change things recursively

File persmissions

Examples from my ~/tmp directory

- -rwxrwxr-x 1 nugent nugent 167 Nov 10 21:59 cleanit
- -rw------- 1 nugent nugent 19949 Nov 11 00:10 cleanup.o10619395
- -rwxrwxr-x 1 nugent nugent 850642 Nov 10 11:08 list.dat
- -rwxrwxr-x 1 nugent nugent 20222 Nov 10 22:24 list_r.dat
- -rw-r--r-- 1 nugent nugent 8389440 Jan 14 11:55 nov3096wiyn32dbs_wcs.fits
- -rw-r--r-- 1 nugent nugent 8389440 Jan 14 11:55 nov3096wiyn33dbs_wcs.fits
- -rw-r--r-- 1 nugent nugent 8389440 Jan 14 11:55 nov3096wiyn34dbs_wcs.fits
- -rw-rw---- 1 nugent nugent 178 Nov 10 22:25 runit
- -rw-r--r-- 1 nugent nugent 3904 Jan 15 14:31 sn9569.r.v5
- drwxrwxr-x 2 nugent nugent 4096 Jan 14 16:27 tmp
- drwxrwxr-x 2 nugent nugent 4096 Jan 15 18:16 tst

The bash environment

- Lets start slowly, and not screw this up.

First do this: touch .no_default_modules

- Logging in and the .bashrc.ext file
  - open up two terminals when you play with your .bashrc.ext, one to edit, and one to test logging in and out
  - cp .bashrc.ext .bashrc.ext.orig

- All of us should have the following set in our .bashrc.ext, regardless of the machine we are on
  - umask 002 or umask 022

umask

The octal umasks are calculated via the bitwise AND of the unary complement of the argument using bitwise NOT. The octal notations are as follows:

- **Octal value** : Permission
- **0** : read, write and execute
- **1** : read and write
- **2** : read and execute
- **3** : read only
- **4** : write and execute
- **5** : write only
- **6** : execute only
- **7** : no permissions

bash environment continued

```
if [ $NERSC_HOST == ”edison" ]
then
#  I am going to do this for edison
set -o emacs
HOSTNAME=`uname -n`
EDITOR=/usr/bin/emacs
export HOSTNAME EDITOR MANPATH
PS1='[$USER:$PWD]’
alias dir='ls -F -s -h’
alias la='ls -a -F -s’
alias lsd='ls -F|grep /’
fi
```

fun modules

I like to pull in Ted Kisner's hpcp modules. They have a ton of useful astro stuff.

I do the following in my .bashrc.ext:

```
hpcp () {
  machine=$1
  source /project/projectdirs/cmb/modules/hpcports_NERSC.sh
  hpcports gnu
  . ${HOME}/.modules
}
```

fun modules

Then in my .modules file:

```
if [ $NERSC_HOST == "edison" ]; then
 module load astromatic-hpcp wcstools-hpcp astrometry_net-hpcp
 module load cfitsio-hpcp
 module load python-hpcp scipy-hpcp matplotlib-hpcp astropy-hpcp psycopg2-hpcp ipython-hpcp
 module load grace
 module load postgresql-hpcp
fi

if [ $NERSC_HOST == "cori" ]; then
 module load astromatic-hpcp wcstools-hpcp astrometry_net-hpcp
 module load cfitsio-hpcp
 module load python-hpcp scipy-hpcp matplotlib-hpcp astropy-hpcp psycopg2-hpcp ipython-hpcp
 module load grace
 module load postgresql-hpcp
fi
```