CIS 18A
Introduction to Linux / Unix

Directories

De Anza College
Instructor: Clare Nguyen

Files in Linux

- The Linux philosophy is "everything is a file."
  The term "file" not only means a file of data (the typical meaning), it can also refer to an input device (such as a scanner), an output device (such as the monitor), a hardware component (such as the hard drive), or a process (such as the shell that you work with).
- Linux divides files into 7 different types:
  1. *regular file*: a file of data. Data can be text (human readable) or binary (machine readable).
  2. *directory*: a file that can "contain" other files (equivalent to folders in Windows).
  3. *character special file*: IO device that processes one character at a time, such as a printer.
  4. *block special file*: IO device that processes a block of characters at a time, such as a hard disk.
  5. *symbolic link*: a file that holds the address of another file.
  6. *FIFO*: a file used for inter-process communication.
  7. *socket*: a file used for network communication.
- This class covers regular files, directories, and links. This section covers directories.

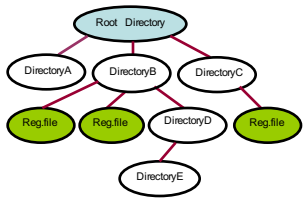
Directories

- Directories help you organize files by providing ways to group similar files together.
- In Linux, files are grouped into directories (which are files also), and directories are grouped under other directories, in a "tree" form called a *directory tree* or *directory hierarchy*.
- Each file in the hierarchy is called a *node*.
- The top node is called *root*.
- Except for root, each node can have:
  - 1 node above it, which is its *parent node*.
  - 0 or more nodes below it, which are its *child nodes*.
- A parent directory can have directory child nodes, called *subdirectories*.

Directory Tree Diagram

In this sample directory tree:

- There are 9 files.
- Each file is a node.
- One root node or root directory.
- DirectoryC is the child node of root, and is the parent node of a regular file.
- DirectoryD is the subdirectory of DirectoryB.



Root Directory

- There is only one root node for every directory hierarchy
- The actual name you type in for root is: /
- Some common subdirectories under root are
  - bin: (for **bin**ary) contains general Linux utilities, in binary format
  - sbin: (for **s**ystem **bin**ary) contains utilities for system administration
  - etc: contains configuration files for the system
  - usr: (for **usr**) contains applications for the users
  - lib: (for **lib**rary) shared libraries
  - var: (for **var**iable) contains data files that change when the system is running
  - tmp: (for **tmp**orary) contains temporary files
  - dev: (for **dev**ice) contains files for hardware devices
  - mnt: (for **mnt**) contains mount points for storage devices
  - home: contains home directories of users

Home Directory

- Every user is assigned a unique directory to store his / her own files. This directory is called the *home directory*.
- Your home directory has the same name as your log in name.
- You cannot change your home directory name or its location in the system.
- When you first log in the system, you are automatically at the home directory.
- Often while you do work on the system you will not be at the home directory. Instead you will move to a different directory, depending on the task you are doing. The directory where you are working is called the *current directory* or the *working directory*.

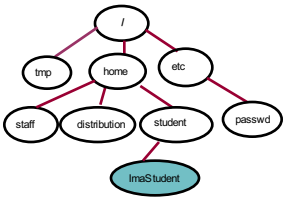
Directory Path (1 of 3)

- Since the Linux directory tree is a huge hierarchy of files, it is important to know the path to a file in order to access it.
- The path of a file is the location of the file in the directory tree.
- A path is formed by:
  - listing node names that are connected to each other in the tree structure.
  - separate node names with / (*not* the same as root).
- 2 types of paths: *absolute path* and *relative path*
- *Absolute path:*
  - Shows the location of a file, starting from root /
  - The absolute path of a file will not change unless the file is moved to another place in the hierarchy.
- Relative path:
  - Shows the location of a file, starting from your current directory.
  - A relative path never starts with /
  - Since the relative path is relative to your current directory, the relative path of a file will change if you change your current directory.

Directory Path (2 of 3)

Assume your home directory is ImaStudent:

- The *absolute path* of your home directory is /home/student/ImaStudent
- If you are currently at the home directory, you can use a *relative path* to get to your home directory by traversing down to student, and then to your directory. The *relative path* from home to your directory is: student/ImaStudent



- Likewise, if you see a file path /etc/passwd, then you know that the file passwd is under the directory etc, which is under root.

Directory Path (3 of 3)

- All commands that accept a filename as an argument will accept a filename with a path:
  - When a filename with no path is the argument, the file must be in the current directory.
  - When a filename with a path is the argument, the file needs to be wherever the path indicates.
- Examples:

ls fileA	will list fileA in the current directory
ls /labs/fileA	will list fileA that is in the labs directory which is under root.
cp fileA fileB	will make a copy of fileA in the current directory and store it as fileB in the current directory.
cp fileA /files/fileB	will make a copy of fileA in the current directory and store it as fileB in the files directory, which is under root.

pwd and cd

- pwd: (for **p**rint **w**orking **d**irectory) shows the absolute path of where you are in the directory tree.
- cd: (for **c**hange **d**irectory) moves you to another directory.
- Common format: cd path

  where path can be:
  - Nothing: moves you to your home directory.
  - Absolute path: make sure it starts with / (root).
  - Relative path: make sure the first node in the path is in your current directory:
    - To go to a subdirectory: type the name of the subdirectory
    - To go to a parent directory: type ..
    - To go to the current directory: type .
  - Path with special symbols:

    ~ the absolute path of home directory

    ~userID the absolute path of the home directory of user userID

ls and Directories

- Recall:

ls	list filenames in the current directory
ls filename	show filename if file exists and is a regular file

- If filename is the name of a directory, then ls will list filenames *under* that directory.
- To see the file type of a file:
  - Use the long listing: ls –l

    The first character in the mode column will tell you the file type: d for directory, l for link, - for regular file.
  - Use: ls –F

    The filenames will be listed with an additional symbol at the end: / for directory, @ for link, * for executable regular file, nothing for text file (which is also a regular file).

mkdir and rmdir

- mkdir: (for **mk**e **dir**ectory) creates a new directory in the current directory, or under a different directory if a path is given.
- Common format: mkdir directory_name
- When first created, the directory is an *empty directory* (no files in it, except 2 hidden files . and ..).
- rmdir: (for **rm**ove **dir**ectory) deletes an *empty* directory.
  - If the directory is not empty, you must delete all files in it first.
- Common format: rmdir directory_name
- Alternatively, to remove a non-empty directory, use:

  rm –r directory_name where -r is for **r**ecursive
  - Removes the directory and recursively go down all its subdirectories and remove all the files under them.
  - Caution: this can remove a large number of files, make sure you don't run this command by accident.

cp and Directories

- Recall that cp will copy the source file to the destination file. Now we discuss all the combinations of cp with regular files and directories.
  File below means regular file, Dir below means directory.
- Copying a source regular file

cp existingFile nonExistingFile	new nonExistingFile is created
cp existingFile1 existingFile2	existingFile2 is overwritten
cp existingFile nonExistingDir	new regular file created with the name of nonExistingDir
cp existingFile existingDir	new file called existingFile created under existingDir

- Copying a source directory (need to use –r option, for recursively copy, which means all files and subdirectories will also get copied).

cp –r existingDir nonExistingDir	new nonExistingDir is created
cp –r existingDir1 existingDir2	existingDir1 is copied and put under existingDir2
cp –r existingDir nonExistingFile	new directory called nonExistingFile is created
cp –r existingDir existingFile	not possible

mv and Directories

- Recall that mv will move the source file to the destination file, and the source file will no longer exist. Now we discuss all the combinations of mv with regular files and directories.
  File below means regular file, Dir below means directory.
- Move a source regular file

mv existingFile nonExistingFile	new nonExistingFile is created
mv existingFile1 existingFile2	existingFile2 is overwritten
mv existingFile nonExistingDir	new regular file created with the name of nonExistingDir
mv existingFile existingDir	new file called existingFile created under existingDirectory

- Moving a source directory (don't need option, all files and subdirectories will move)

mv existingDir nonExistingDir	new nonExistingDir is created
mv existingDir1 existingDir2	existingDir1 moves under existingDir2
mv existingDir nonExistingFile	new directory called nonExistingFile is created
mv existingDir existingFile	not possible

which and whereis

- These commands give the path (location) of a particular utility. They only accept a utility name as an argument.
- which utility_name
  - shows the actual utility that runs when utility_name is typed on the command line
  - Example:
```
[cnguyen@voyager ~]$ which vi
alias vi='vim'
/usr/bin/vim
```
- whereis utility_name
  - shows the location of all files pertaining to the utility
  - Example:
```
[cnguyen@voyager ~]$ whereis vi
vi: /bin/vi /usr/share/man/man1/vi.1.gz
/usr/share/man/man1p/vi.1p.gz
```
- If you know the location of a utility and you want to run it, type the absolute path of the utility on the command line.
  - Example: To run vi instead of vim:
```
[cnguyen@voyager ~]$ /bin/vi filename
```

Next stop: Links and find