CIS 18A
Introduction to Linux / Unix

Links

De Anza College
Instructor: Clare Nguyen

Links

- Links are files that contain the address of another file.
  For those of you who are programmers, links are pointers or references to other files.
  In Windows, links are similar to shortcuts.
- Links help users to quickly access data that are in a different directory than the current directory. Instead of typing a long path to get to a file, you create a link to the file and put the link in your own directory.
- Then you access the file by using the name of the link. When the shell sees the link name as an argument, it accesses the file for you.
- 2 types of links: *hard links* and *symbolic* (or *soft*) *links.*
- Hard links contain the physical address (memory location) of another file.
- Symbolic links contain the path to another file.

Hard Links

- Any time you create a new file, you create the *first hard link* to the file. The filename you use is the hard link to the file. When you access the file by typing the filename, the hard link takes you to the physical address of the file.
- To create subsequent hard links to a file:
  ln existingFile linkName
  - ln is for **l**i**n**k
  - existingFile is the file you want to link to, and can contain a path.
  - linkName is the name of the new link, and can contain a path.
- To remove a hard link: rm linkName
  where linkName can have a path.

Use of Hard Links

Advantages of using hard links:

- If several people have hard links to the same file, then when one person changes the file, everyone sees the change because everyone is "looking" at the same file (same memory location).
- When you have a hard link to a file, you have the physical address (direct access to the memory location) of the file. So if the owner of the file deletes the file from his/her directory, you still have access to this file from your directory. A file is "gone" only when all hard links are removed.

Disadvantages of using hard links:

- As a regular user, you cannot create a hard link to a directory.
- Because hard links contain physical address of memory, you cannot create hard links over different file systems. If someone's file is on a different hard disk or a different sector of the disk than where your directory is, you cannot create a hard link to his/her file.
- To overcome these limitations of hard links, you need to use symbolic links

Number of Hard Links

- Every regular file has at least 1 hard link, created from when the file is created and given a name.
- Every directory has at least 2 hard links, the second link is to link up to the parent directory, in order to maintain the tree hierarchy.
- Every time someone uses ln to create a subsequent hard link, the number of hard links of the file increases by 1.
- Every time someone uses rm to remove a hard link, the number of hard links decreases by 1.
- The number of hard links of a file called filename is shown in the second column of ls –l filename

Check for Hard Links

- Every file in the system has a unique ID called the *inode number.*
- To see the inode number for a file:
  ls –i filename
  where i is for **i**node
  and filename can contain a path.
- If a file has 3 hard links, each of the hard links will have the same inode number because they are all links to the same memory location (same file).
- Therefore, to determine if 2 filenames are actually hard linked to the same file, check their inode numbers.
- Files that are hard linked together have the same inode number.

Symbolic Links (1 of 2)

- Symbolic links contain the path from the link to the actual file.
- The file type link refers to symbolic links, not to hard links.
- Since hard links are direct access to memory locations, hard links are considered regular files.
- To create a symbolic link to a file:
  ln –s existingPath linkName
  - existingPath is the path *from the link to the destination file.* This path will be stored in the link, so the link can help you access the file. If you create a symbolic link, it is best to use an absolute path to the destination file.
  - linkName is the name of the link, and can have a path.
- Creating a symbolic link to a file will have no effect on the number of hard links the file has.
- To remove a symbolic link: rm linkName
  where linkName can have a path.

Symbolic Links (2 of 2)

- Advantages of symbolic links:
  - Can link over file systems
  - Can link to directories
- Disadvantage of symbolic links:
  - Since the link contains the *path* to the actual file, if the file is deleted or is moved to another location, the link will be broken. You will no longer be able to access the file through the link.
- For both symbolic links and hard links: you can create a link to a file only if the owner of the file allows you to access the file (file access is covered in the Permissions section).

Check for Symbolic Links

To check that a file is a link, there are 2 ways:

- ls –l filename
  the first character in the mode column is l and the filename will show where the link is pointing to.
- ls –F filename
  the last character in the filename is @

find (1 of 2)

- find: searches a given part of the system directory tree for any file that matches some given criteria.
- Basic format: find start_dir criteria_list
  - start_dir is the directory from which find will start the search.
  - start_dir can be an absolute path or a relative path.
  - If it is a relative path, start_dir is relative to the directory where find is run.
  - If no start_dir is given, find starts the search from the current directory.
  - criteria_list tells search what to look for.
  - The criteria_list can be 1 or more criteria.
- find does a recursive search from the start directory, which means it will go down all subdirectories of each directory that it encounters.
- When ls is used to search for a file, the search only occurs at the directory that is given as the argument to ls.
  When find is use to search for a file, the search starts at the given directory and proceeds down all subdirectories, so it is a deeper search.

find (2 of 2)

- find is a powerful command that can do work (take action) on the files it found that match the criteria.
- The action on the files can include removing the files, modifying the files, copying the files to another location, etc.
- For this class, we will only use find to print the location of all the matched files. Printing the location of the matched files is the default action of find.
- When printing the location of the matched files, find prints the path of the matched files with respect to the start_dir.

Criteria for find (1 of 3)

- Files matching filename:
  - -name filename — all files matching filename
  - -name 'name_with_wildcards' — all filenames matching name_with_wildcards (single quotes are required)
- Files of a certain file type:
  - -type d — all files that are directories
  - -type f — all files that are regular files
  - -type l — all files that are links
- Files with a certain permission:
  - -perm octal_mode — all files with mode matching octal_mode (see Permissions section for octal_mode)
- Files that are empty:
  - -empty — applies to regular files and directories

Criteria for find (2 of 3)

- Files with a certain number of hard links:
  - -links +num — all files with number of hard links greater than num
  - -links num — all files with number of hard links equal num
  - -links –num — all files with number of hard links less than num

  num is a number
- Hard links to a file:
  - -inum inode_num — all files with a certain inode_num

    Recall: inode_num is found by using ls –i
- Symbolic links pointing to a certain file:
  - -lname path — all links that contain a specific path
  - -lname 'path_with_wildcards' — all links that contain paths that match the path_with_wildcards (single quotes are required)

  Recall: when a symbolic link points to a file, it contains the path to that file. The –lname option looks at the path in the symbolic links for a match. If there is a match, it means the link points to that file.

Criteria for find (2 of 3)

- find accepts one or more criteria.
- To use more than one criteria:
  - Criteria that are ANDed together:
    - the file has to match all criteria listed.
    - List the criteria separated by space.
    - Example: find ~ -type d -empty
  - Criteria that are ORed together:
    - the file has to match at least 1 criterion in the list.
    - List all criteria separated by -o (for **o**r)
    - Example: find ~ -type d -o -type f

Next stop: File Permission