

# Unix File System Commands

In addition to *pwd*, *cd*, and *ls*, here are some other key commands for working with files: *cp* (*copy*), *mv* (*move*), *rm* (*remove*), *rmdir* (*remove directory*), *mkdir* (*make directory*), and *ln* (*create link*)

- Renaming a file is done via *mv* — you’re “moving” the file to the same directory under a different name; it works on both files and directories/folders, and when used across devices, it copies then deletes
- *ln* creates two kinds of links — *symbolic* links (*ln -s*) link by a path, while *hard* links link by the *inode* number
- Many of the above commands support the *-r* (*recursive*) switch, for affecting subdirectories and files within them
- By default, *cp* creates “new” files (i.e., last modified today) — use *-p* to *preserve* the copied file’s metadata
- *rmdir* only works when a directory is completely empty, including invisible files...in Unix, a file is invisible if its filename starts with a period (.) — run *ls* with the *-a* switch to see them
- The recursive version of *rm*, *rm -r*, will delete non-empty directories if you’re in a hurry...just be sure that you want to do this
- You can create multiple levels of new directories via *mkdir -p*, e.g., *mkdir -p five/nested/directories/at/once*

# Disk-Related Commands

- File storage is typically available on multiple distinct devices; on Windows, these are identified by top-level *drive letters*, while on Unix and other platforms, these devices are *mounted* onto a unified directory tree — thus separating the device from the file structure
- *mount* and *umount* (no, that isn't a typo) are the command line equivalents for “inserting” and “ejecting” these storage devices; *mount* provides total control over where these devices can be accessed, though there are conventions (*/mnt* in Linux, */Volumes* in Mac OS X)
  
- *df*, though nominally meant for displaying *free disk space*, also shows the *mount points* of all accessible storage devices; run it with the *-h* or *-H* (“human-readable”) options for maximum readability
- *du* (“*disk usage*”) performs a similar function, but from the point of view of files/directories instead of devices
  - ◇ *du* takes files/directories as parameters, indicating the items for which disk use is to be calculated; default is the current working directory
  - ◇ The *-h* or *-H* options are also available with *du*
  - ◇ Default output lists usage for every subdirectory, going from bottom up; the *--max-depth* option (*-d* in some versions of *du*) controls how much is displayed

# Finding Files

- Recall that the *which* command displays the precise executable that gets invoked when a particular command is typed
  - ◇ Specifically, the search space used by *which* is your current *path*, stored in the *PATH* environment variable (use *echo \$PATH* in Unix to display this; *echo %PATH%* in Windows does the same, though *which* isn't available “out of the box” on that operating system)
- More general-purpose commands are also available for finding files of any kind
  
- The *locate* command is a fast find-by-name program; it uses a database for speed and control — configuration files determine which directories are included in the search, and the database requires periodic updates
- *find* is slower but more comprehensive; it performs a sequential search starting at a given location
  - ◇ You can search on a variety of properties, including name, type, size, user, and many more (as usual, *man find* spills all of the gory details)
  - ◇ By default, *find* displays the locations of files matching the given criteria; it can also be told to perform other actions, such as deleting the found files or executing another program with the found files as arguments

# Network File Commands

- Another file manipulation command category works with files on multiple networked computers
- *scp* (secure copy) is the *cp* workalike, but can transfer over a network too; the general format is *scp path-to-local-file user@machine:destination-path*
- *sftp* (secure ftp) is functionally similar, but provides an interactive shell for listing and transferring files
- *scp* and *sftp* are typically standard in Unix and can be gotten for free for Windows (PuTTY's *scp* is called *pscp*)
  
- *rcp* (remote copy) and *ftp* (file transfer protocol) are *scp* and *sftp*'s predecessors, but they don't encrypt files in transit and so are mainly used (*ftp* particularly) for public or anonymous servers
- *rsync* is also a file transfer utility, but it specializes in keeping directories "in sync" — great for backups or replication, and fast because *rsync* uses file comparison algorithms prior to transferring them (it's actually someone's [Andrew Tridgell's] PhD thesis!)
- File transfer commands using web protocols (i.e., *http*, *https*, etc.) are also available: these include *wget* and *curl* — they take the same URLs that you would enter into a browser, except of course they just download the file represented by that URL instead of displaying it