Working With Files

Linux System Administration
COMP2018 Summer 2017

Path Names

- Any time you need to identify a file to work with, you need to know how to write it on the command line
- Linux provides 2 ways to specify a file name, and a few shortcuts for certain commonly accessed directories
- An absolute pathname is the full specification of the file's name starting at the root of the live filesystem and following the entire path to the file (specification starts with /) and is completely independent of your current working directory
- A relative pathname is the complete path to the file, starting from the current working directory (specification does not start with /)

Directory Name Shortcuts

- bash provides some shortcuts to commonly used directory names
- Some are stored in variables, such as the real user's home directory ($HOME), the current working directory (aka present working directory or $PWD)
- ~ (tilde) when typed as the first character of a pathname followed by a / (slash), or as a complete pathname, is the exact same thing as typing the absolute path to the real user's home directory
- ~accountname when typed as the first component of a pathname followed by a /, or as a complete pathname, is the exact same thing as typing the absolute path to accountname's home directory
- . (dot) as a pathname component means the current directory, .. (dot dot) as a pathname component means the directory which contains the current one (directory above the current one)

Current Working Directory

- When you login, your current working directory is set to your home directory, defined by your user account
- Your home directory is intended to be a repository for files unique to yourself and may or may not be private to you
- To change bash's notion of where you are virtually located in the live filesystem, use the cd command
- cd with nothing else on the command line sets your current working directory to your home directory
- cd with one argument on the command line sets your current working directory to whatever you specified on the command line, if you have permission to access that directory
- If you give an argument to cd, the argument must be an existing directory pathname

Directory Stack

- To temporarily change directory with the intention to return to where you started, use pushd and popd instead of cd

- pushd saves your current directory on a stack and changes to a new directory

- popd takes you to the last directory you pushed onto the stack and takes it off the stack

Directory Contents

- A directory is a file, albeit a special one
- Instead of containing solely user data, a directory contains a list of files which are considered to be "in" that directory
- Each file in the list also has a special number called an inode number associated with it that uniquely identifies that file within the filesystem where it is actually stored
- To view a list of files, we use the ls command

ls

- The ls command with nothing else on the command line displays the list of files in the current directory, excluding files whose names start with a dot (we call these hidden files)

- There are many options to the ls command to modify what and how it shows file lists

- Commonly helpful options include
  - -l (long listing displaying attributes for each file including type, permissions, link count, ownership, length, timestamp)
  - -d (display directory files themselves instead of the list of files they contain)
  - -s (display file sizes as well as names)
  - -t (display file listing sorted by last modification timestamp instead of name)
  - -F (display a character after the file names if they are a symbolic link, executable, or directory)
  - -a (display all files including hidden files (file names that start with a dot))
  - -A (display all files except . and ..)
  - -R (recursively display file hierarchy)
  - --color may display file names using colours on some systems
  - -h may display file lengths and sizes in human-friendly number specifications on some systems instead of as byte counts

- Options may be combined (e.g. ls -lht) and multiple file names (e.g. ls /etc/hostname /etc/hosts /etc/network) may be given as arguments

- Note that file length does not imply filesystem space allocated or consumed

- tree (lstree on some systems) can display a file hierarchy in a visually helpful way

du

- To find out how much disk space is being used by a file or directory, use the du command
- The general form is du [-s] [-h] file(s)
- The -s options tells du not to show sizes for every file found, but instead display a total for all the files
- The -h option tells du to show sizes in human-friendly numbers instead of byte counts

Creating Directories

- Directories are created using mkdir
- Multiple directories can be created by putting multiple pathnames on the command line as arguments
- -p tells mkdir to make all pathname components as need
- mkdir requires write and access permissions (wx) on the directory which will contain any directory being created

Removing Directories

- Directories are removed using rmdir
- Multiple directories can be removed by putting multiple pathnames on the command line as arguments
- rmdir requires write and access permissions (wx) on the directory which contains any directory being removed
- rmdir requires a directory to be empty before it can be removed (i.e. it cannot create orphan files)
- rm -r can be used to recursively remove a directory and the entire file hierarchy it contains

Copying Directories

- Since directories may contain file hierarchies, copying them requires copying the entire hierarchy and unless you are root, the copy will be owned by whatever user does the copying, no matter who originally owned which files in the hierarchy

- There are multiple commands to accomplish this depending on some nuances of copying unusual file types which may be present in the hierarchy

- cp -r source(s) destination

- Multiple sources may be specified if the destination is an existing directory

- Be careful with hierarchies which may contain files of types other than regular or directory and consider the effects of options --sparse, --preserve, --force, and --archive along with similar options (see http://www.gnu.org/software/coreutils/manual/html_node/cp-invocation.html#cp-invocation)

- Beware of copying between filesystems of different formats and the effects that may have on what you can preserve when copying files

- You must have read and access permissions (rx) as appropriate to copy directory hierarchies

Moving Directories

- Directories can be moved trivially within filesystem containers using the mv command (mv source(s) destination) because it is essentially just a change of containing directory for the directory being moved
- Multiple sources may be specified if the destination is an existing directory
- Moving a directory moves the file hierarchy from that directory on down, intact, to the destination
- Moving directories between filesystem containers invokes copy and delete
- Consider the effects on timestamps and ownerships when moving directory hierarchies
- Required permissions for moves and ownership changes caused by moves can be less than obvious - whoever creates files owns them

Creating Regular Files

- Regular files can created by any command capable of writing data to a file as long as you have write and access permission (wx) on the containing directory
- Shell redirection can save the output of any command to a file, creating it if necessary
- touch pathname(s) can quickly create empty regular files
- Editors are also commonly used to create regular files

Removing Regular Files

- rm pathname(s) can be used to remove regular files providing you have write and access permissions (wx) on the containing directory
- A removed file in a typical Linux filesystem CANNOT be undeleted
- File names are not stored inside files, they are stored in directories, removing an entire directory tree is done by using the -r option with rm to remove the directory at the top of the tree you want removed
- Once you remove a file from a directory, the space for it is immediately reclaimed by the filesystem and there is generally no way to find it back (some tools exist to try to do this, but it is neither straightforward or reliable)
- The -i option can be helpful as it makes each removal interactive and prompts you to confirm each deletion

Copying Regular Files

- Files are copied in various ways, depending on your goals and what you are copying
- Typical users who are just copying regular files will use `cp source-pathname(s) destination-pathname`
- If there are multiple source files, the destination must be a directory
- If the source files have multiple pathname components (i.e. `X/Y`) and the destination is a directory, those components will be created in the destination directory
- The copies are owned by the user/group who does the copying, with the same permissions as the source files
- Copying non-regular files may require command options to get the desired results
- Interesting options are `-i` (interactive), `-n` (no overwrite), `-v` (verbose), and `-r` (recursive)
- Consider, what do you want to happen if you copy a directory, or a symbolic link?

Moving and Renaming Regular Files

- Moving is like copying, but removes the source pathname(s) from the directories they were listed in before the move
- Moving to a non-existent pathname is renaming
- Moving between filesystems causes a copy-then-delete to be performed
- Moving within a filesystem just adds the existing source file to a destination directory, then removes the old name from the source directory
- Moving is done with the `mv source-pathname(s) destination-pathname` command
- The `-i`, `-n`, and `-v` options are available just like with the `cp` command
- File permissions and ownership may not end up the way you might guess, moving is only relocating, whereas copying makes new files

Linking Files

- We can create symbolic links between filenames and use those links to access the linked files
- The general form is `ln -s target linkname`
- You can then use linkname to refer to target
- Removing a link does not affect the target file
- Removing or renaming a target file does not affect the link, which might leave it pointing at something that no longer exists (a dangling link)

Remote Files

- There are several programs which are helpful when you want to retrieve some file(s) from a remote machine
- If the remote machine is one you have an account on, you can try ftp or sftp to transfer the file(s) of interest
- If the remote machine is a web server, you can use wget or curl to retrieve the file(s)

ftp And sftp

- ftp requires the remote host to be running an ftp daemon, sftp requires the remote host to be running an ssh daemon
- Both require either a valid login on the remote host, or to be configured to permit anonymous access
- Both allow you to connect and execute commands at a prompt, with a help command showing you what commands are available at the prompt
- Examples of commands available include cd, ls, get, mget, put, mput
- If you enter [s]ftp [user[:password]@]host on the command line, it tries to connect to that host before giving you a prompt

wget And curl

- Both wget and curl give you a way to retrieve a local copy of some file(s) from web servers, as well as upload data to those web servers using http and https protocols
- They have different default behaviours
- wget URL retrieves a file and stores it using the same name it had in the URL
- curl URL outputs the file contents onto your screen by default
- Both have many options to allow you to control the download, perform upload, etc.

Finding Files

- When a filesystem gets a lot of files in it, the tree structure can become complex
- Naming files in a future-friendly way is not a common skill
- Sometimes files get moved somewhere unintended or renamed unintentionally (click-drag syndrome)
- The ls command allows visually inspecting directory lists, but isn't usually very helpful to locate lost files
- The locate command can be used to find files, but depends on a database which may not be accurate
- The whereis command helps you find files in specific system directories, not helpful for finding lost user files

find

- The find command is the general-purpose file finding tool
- It allows searching for files based on many different characteristics or combinations of them, or simple searching by name
- It allows you to perform complex commands on the files found, or you can just list them out as found
- The general form is

```
find [starting-pathname] [options] [pattern(s)] [actions-to-take]
```

File Content vs. Names

- Linux does not generally care what you name your files
- GUI tools often use filename extensions as hints about file content for the purpose of finding apps to open them
- The same filename extensions you would use in other operating systems are useful in Linux when using a GUI, but are not typically relevant for command line programs
- On the command line, the quickest way to identify the actual content of a regular file is to use the `file file(s)` command