Lecture 3: The UNIX Style

CS2042 - UNIX Tools

October 3, 2008

Lecture Outline

Windows' Structure

- Highest-level directory is Desktop
- Uses C:, D:, etc. to represent different disks/volumes
- User programs in Program Files, Windows programs in Windows, kernel programs and libraries in System/System32
- Most settings found in the Registry

Style Note:

Windows allows complex filenames with spaces. However, it is not case-sensitive. This means that **rm myfile.txt** would remove **MyFile.txt**. This is not the case in UNIX. Use lowercase letters and underscores instead of spaces - your personal MP3 collection is exempt from this rule.

Unix's Structure

Where are my programs?

- The binary folders:
- **/bin:** System programs
- **/usr/bin:** Most user programs
- **/usr/local/bin:** A few other user programs

These three directories are always included in a UNIX system's PATH - the place it looks for executables.

Other Important Directories

- **/dev:** Hardware devices can be accessed here - usually you don't mess with this stuff.
- **/lib:** Stores libraries, along with **/usr/lib, /usr/local/lib**, etc.
- **/mnt:** Frequently used to "mount" disk drives
- **/usr:** Mostly user-installed programs and their related files

Settings

System-wide settings for programs can be found in **/etc**

- Stored in plaintext files, easily editable

User-specific Settings:

User-specific settings are usually stored in each user's home directory. These files (or directories) start with a period, which marks them as "hidden" in UNIX.

- Example: User settings for the GIMP image editor are stored in ∼/**.gimp/**

- View all files (including hidden files) with **ls -a**, or **ls -al** for long format

Lecture Outline

Plaintext Files

Text Files

Plaintext files are written in a human-readable format. They are frequently used for simple documentation, application settings, source code, logs, and anything that someone might want to read via a terminal.

- Think of something you might create in Notepad
- Editable using many existing editors - for this class, use **nano**
- To view a plaintext file without changing it, use **less** or **more**

Binary Files

Binaries

Binary files are written in machine code. They aren't human-readable, and if you try to open them with a plaintext viewer/editor, you'll just see gibberish with some weird symbol characters mixed in.

- Common examples are binaries (executables), libraries, media files, and non-handwritten files (.zips, .docs, .pdfs, etc)
- Technically editable using "hex editors" - for our purposes, not editable
- To create a binary file, you need to use some sort of binary-outputting program.

Lecture Outline

What is a Command?

- **ls -al ../otherdir**
- **more readme.txt**
- **chmod o+rw somefile**

With few exceptions, most "commands" are just programs made to run from the command line. All three example programs are located in **/usr/bin**.

Parameters

Command Parameters

Parameters are the extra bits of information that you feed a command-line program to tell it specifically what you want it to do. For example, filename operands are parameters. We have already covered a number of common parameters for several of our commands, such as **ls -[al] [filename]**.

- Most option parameters we'll deal with will be preceded by a dash -

Lecture Outline

The PATH

Earlier we mentioned the PATH, those 3 binary folders that Unix searches for programs. How does this affect the running of commands?

- Command in the PATH:
  - Just run the command - Linux will find the program you're looking for automatically.
- Command not in the PATH:
  - Type the full path to the program, ex: **/home/user/program1**
  - If the program is in the current directory, we have to specify that: **./program1**

Lecture Outline

What's the Difference?

Links

Pretty much the same as links in Windows - just a redirect to another location. Limited to files on the same file system.

Symbolic Links

Causes the linked item to appear as if it were actually in both locations - useful for resolving relative paths. Also useful for linking directories, or linking across disk drive volumes.

Creating Links

Link Creation

ln [options] <target_file> [link_name]

- Creates a link to <target_file> at [link_name], defaulting to the current directory

Symlink Creation

ln -s <target_file> [link_name]

- Creates a symlink to the target file or directory