INFO I211 Information Infrastructure II

Spring 2014

Lab 2: Unix &c.

Unix as an operating system: an overview.

- Unix shells
- the Unix file system
- processes and PIDs
- process control in shells
- Unix commands
- Unix data I/O
- I/O redirection
- command piping
- command options

Unix Shells

A Unix shell is an interactive command interpreter which accepts the commands that you type, and runs the corresponding programs. There are several popular shells in Unix systems, including the "C" shell variants (csh, tcsh), korn shell variants (ksh, zsh) and bourne shell variants (sh, bash). The first in each of the variants is the original shell, and the second one is the recent improved version. Most shells have similar look and feel, but the newer ones (tcsh, zsh, bash) have advanced features such as command and file completion, command-line editing etc. You can change a shell by running the `chsh` (change shell) command.

This information is SoIC Burrow cluster specific: To change your shell, you will need to use the IU SoIC Linux Account Management Page, which also notes that "*Shell changes on all platforms [...] will not take effect until the next day after the change is made*".

Every shell has its own settings file in which you can specify special configuration options. See the manual page for your shell to get the details of the syntax and format of this file.

The Unix File System

The Unix *file system* provides the permanent storage for all the data used by the operating system. This includes all the data files (e.g. the data we'll serve from Apache) and all the program executable files (e.g. Apache itself, `httpd`). As in essentially all operating systems you are going to use, the Unix *file system* is organized as a tree-like structure of *directories* and *files*. More often than in other mainstream operating systems, any part of the file system can physically reside on *file servers,* usually interconnected by the *NFS* (Network File System) protocol. An important example for this class: your own *home directory* on the IU Burrow cluster (e.g. `/u/username/`) does not physically reside on any hard disk located in the LH004/LH035 workstations, but on the hard disks of an NFS server.

Unix Paths

Most Unix commands are actually programs that run when you type in the command. Some commands are built-in to the shell that you use. When you type in a command, the shell looks for this command in a list of directories that you specify. This list of directories is the "*path*." Depending on the shell you are using, the syntax for declaring the path will be different, but the usual set of directories to include in the path are (some of these paths are IU Burrows cluster specific):

```
.:/bin:/usr/bin:/usr/local/bin:/usr/ucb:/usr/etc:/sbin:/usr/sbin:/usr/ccs/bin:/usr/dt/bin:/usr/local/gnu/bin
```

Keep in mind that the longer the path is, the longer it takes the shell to search through all the commands in the path. You might want to look at these directories to see if there are commands in there that you do use. If not, you may want to remove the paths. Some shells create an index structure of all the commands when they start up, which reduces this searching time.

Also, having the "." in your path allows you to run your scripts and programs from the current direcory without having to type "./" in front of the script or program name.

Commonly used Unix Commands

Here is a list of commonly used unix commands in alphabetical order. Each command includes a description and typical invocations. Note that almost all the unix commands accept command line arguments and options that usually start with a dash (-). The options can be stacked (i.e., if a command accepts options -a and -b, it will also accept -ab).

`cat`
: Short for con*cat*enate, commonly used to dump the content of files (specified in the command line) in the standard output.

`chmod`
: chmod is the unix command to set and change the permission modes for files and directories. There are two main ways to invoke chmod:

  - `chmod [options] filename`
    In this mode, [options] is of the form x[+/-]y where x is zero or more of u,g,o (user, group, other) and y is one of (r,w,x). e.g., you can turn on read and execute permission for a file for the group and others using chmod go+rx filename.
  - `chmod nnn filename` This changes the permissions of the file to the 3-digit octal number nnn, the first digit corresponding to the user, second for the group, third for everyone else. e.g., `chmod 700 filename` changes the permission of the file to read, write, execute by you, and no access for anyone else.

`cp`
: This is the unix file copy command. Typical invocation is `cp file1 file2`. The -r option can be used to copy recursively.

`date`
: Prints out the current date and time. The -u option can be used to print Universal (UTC/GMT) time.

`diff`
: Displays the differences between two text files. The -c option can be used to show some "context". The -r option allows for recursive operation on directories.

`du`
: Displays the current disk usage, recursively for every subdirectory of the current directory. The -s option can be used to display only the total usage, without any breakdown.

`find`
: A very powerful file finder. The simplest invocation is: `find . -name "filename_with_wildcards" -print`. See the manpage for further options.

`grep`
: Search for strings in one or more files. Typical invocation is: `grep "string" filenames`. Note that `grep` actually searches for Regular Expressions, and if all you want to do is search for simple strings, `fgrep` is the way to go, since it's much faster than `grep`.

`gzip`
: gzip is one of the most common compression tools in UNIX, not dissimilar from *zip* Invocations:

  - To compress: `gzip filename`
  - To decompress: `gzip -d filename`

`kill`
: Kills (sends signals to) process. The simplest invocation is `kill -[SIGNAL] pid_list` where SIGNAL can be any valid unix signal (e.g., HUP, TERM, STOP, CONT) and pid_list is a list of process IDs (can be looked up by the ps command).

`ls`
: This is the command to display the list of files and directories in the current directory or a directory specified in the command line. Commonly used options include -l for detailed listing, -a to display hidden files (files starting with a .), -t to display files sorted by the modification time, etc.

`lynx`
: lynx is a text-based WWW browser for unix command-line sessions. In lynx, the tab key and the up and down arrows are used for moving around in a displayed document. The left and right arrows are used for following links and going back to the previously displayed document. It can be invoked without any arguments or by specifying an URL in the command line.

`mkdir`
: Creates a directory. Simplest invocation is `mkdir directoryname`

`mv`
: Moves a file or directory to another name. On some UNIX variants, directories can not be moved across different partitions.

`ps`
: ps is the unix command for displaying currently running processes. The behavior of ps is not consistent across different UNIX variants. See the manual pages for further details, and use the `which` command to check which `ps` you get by default.

`pwd`
: Prints the current working directory.

`rm`
: The unix file remove command. Can be used with the -r option to recursively remove everything (use with caution). The -i option can be used to turn on "interactive" mode - rm will ask for every file whether or not you want it to be removed.

`rmdir`
: Removes a directory, only if the directory is empty. To remove non-empty directories, use `rm -r`

`tar`
: Tar (Tape ARchive) is the unix format for archiving multiple files, both ascii and binary, in tapes for backing up purposes. Although originally designed for tape archiving, it is commonly used for archiving files into a single big file. The same command is used for archiving and extracting from the archives. Typical invocations are:

  - To archive: `tar cvf newarchivefilename.tar <list of files and directories>`
  - To 'un'archive (extract): `tar xvf archivefilename.tar`

`top`
: top displays the processes running on the system, sorted by CPU usage. The display typically updates itself every few seconds. Very useful to check if any of your programs are running out of control: when a process consistently uses 99% of CPU time, it might be time to `kill` it.

`whereis`
: Looks for executables (specifyied as arguments) in a predefined set of directories, and reports a list of what's available on the system.

`which`
: See where a particular executable is located. Useful to find out whether you are actually running the right program. (e.g., `which httpd` will tell you which httpd you will run if you just typed httpd at the prompt).

`who`
: Displays all the people who are logged on or have processes running on the system.

An Important/Useful/Noteworthy List

Here are some more UNIX commands and concepts. You can check the `man` pages for more information about the commands, typing `man` *`commandname`*.

- `man`
- `fg`, `bg`, <ctrl>-C, <ctrl>-Z, `&`
- `ssh`
- shells: `csh`, `tcsh`, `bash`, etc.
- ~~`chsh`~~ `->` use the IU SoIC Linux Account Management Page.

- the X Window system: X Servers and Clients (see e.g. The X Window System: A Brief Introduction)

- the cluster `burrow.soic` vs. the machines `degu.soic` etc.
- `ps aux` (user-oriented, and BSD style) vs. `ps -ef` ("-f" stands for "full", and POSIX style)
- `which` vs. `whereis`
- `kill -9` vs. `kill -TERM` etc.

Running Python Scripts For Your Distributed Application

For I211 / Spring 2014, we will rely on the SoIC CGI server to run distributed applications — CGI scripts written in Python. As from the KB article How do I run CGI scripts on the SoIC web server?:

> *The SoIC CGI server allows the running of arbitrary programs (eg. perl, **python**, scheme, bash, etc). In order to use the SoIC CGI server you must have an account on the SoIC Unified Linux systems. Furthermore, you must have an account in the Sharks domain since having a Burrow account is not sufficient. [...] You can **check the status of your account** per the KB article How do I get detailed information about my Linux accounts?.*
> *The SoIC server cgi.soic.indiana.edu is used for CGI program support. [...] your cgi programs will run under **your account** and not as the normal apache user. As a result, your scripts will have the full **permissions** (and limitations) of your account.*

Do You Really Want To Run Your Own Web Server For Your Distributed Application?

If you are curious, the SoIC systems would also support running your own server on the Burrow cluster:

- SoIC Help - Web Hosting Options
- SoIC Help - Can I run my own web, database, or other server on the unified linux systems?
- SoIC Help - How do I set up a web server on the Burrow?

We will *not* need to run separate web servers for I211 / Spring 2014 course purposes. We'll be relying on the SoIC CGI server as described above.
