Lecture 3: Running Programs in the Shell

CSE 374: Intermediate Programming Concepts and Tools
Administrivia

- Just joined?
- Linux access fixes coming later today
- Discord up and running!
Shell State

- Each new instance of the bash shell maintains a “state”
  - Current location in the file system

- PATH variable
  - Whenever a command is executed there is a list of pre-defined locations bash looks
  - PATH holds the list of pre-designed directories
  - echo $PATH

- Bash rc file
  - A shell script that is automatically run whenever Bash starts up
  - Used to initialize your shell state
  - You can find example files shard online
  - Exists locally on your machine, not on the remote linux server

- Bash history file
  - .bash_history is a hidden file that automatically logs your command history

https://astrobiomike.github.io/unix/modifying_your_path
Shell Variables

▪ Shell variables = string substitution
  - Declare variables in the shell to easily refer to a given string

▪ Declare variables in the terminal with a name and a string value
  - `<var name>="<var string>"`
  - EX: `myvar="myvalue"
    - Note: no white space allowed on either side of the "="

▪ Refer to your variable using the "$" symbol before the var name
  - `$$<var name>
  - EX: echo $myvar
    - myvalue

▪ Alias
  - Rename a bash command, create your own shortcut
  - alias <string>="substitution string"
    - EX: alias cheer="echo hip hip hurray!"
  - Only exists within the current state of your shell
  - Can store alias in bashrc file to preserve alias across all shells
Math in Bash

- Everything is interpreted as a string
  - No concept of integer values

- To do math use double parenthesis to interpret as arithmetic expression
  - RES=$( (1+2) )
    - $RES will be set to “3”
  - MYVAR=$( (RES+2) )
    - $MYVAR will be “5”
Quotes in the Shell

- Double quotes can be used to wrap a string with white space into a single argument
  - **EX:** `myvar="Some string"`

- Single quotes tell the shell to treat the string as a literal
  - No variable expansion or command substitution
  - **EX:** `echo '$myvar'`
    - `$myvar`
    - `echo "$myvar"`
    - `Some string`
Demo: Variables and Aliases
Processes in the Shell

- Programs running in the shell are called “processes”
  - We refer to the code/instructions as the “program”
  - We can run a given program many times, creating many processes
  - Terminal can only run one process in the foreground at a time
    - Use the “&” special character to launch a process in the background
    - EX: emacs &

- Bash Shell has many built in programs
  - Commands like cd and ls

- Processes have Input and Outputs
  - Inputs come in two main forms: arguments and stdin
    - Arguments are strings separated by spaces given after the command
    - EX: cp my/src dest/folder
      - Arguments: “my/src” and “dest/folder”
      - Arguments with spaces need to be wrapped in quotes EX: echo “hello world”
    - Stdin or Standard Input is a stream that the user enters into the terminal
  - Outputs can be stdout, stderr or a directed to an output file
  - All redirections & string expansions or substitutions are done by the shell before the command
Terminal Text Editing

- While working with the text interface for the OS often it is helpful to use the text interface editors
  - You do not have to use any of these in this course
  - You can use Visual Studio Code if you prefer
  - Editors are a matter of preference

PICO

EMACS

VIM
Streams

- **stdin** – Standard in (File Descriptor = 0)
  - Keyboard input typed into the terminal

- **stdout** – Standard out (File Descriptor = 1)
  - Results of a process after it has completed printed to the screen of the terminal

- **stderr** – Standard error (File Descriptor = 2)
  - Results of a process if it exits in error printed to the screen of the terminal

- I/O streams can be redirected
  - `< yourInput`
  - `>` yourOutput
    - Redirect stdout to a file
  - `>>` appendYourOutput

- **Pipes** – special character “|”
  - Redirect the output of one process to the input of another
  - `cmd1 | cmd2`

---

https://wiki-dev.bash-hackers.org/syntax/redirection
grep

- Search for a given string within a given file
  - `grep [options] pattern [files]`
  - EX: `grep "computer" /usr/share/dict/words`

- Helpful Options
  - `-c`: prints count of lines with given pattern
  - `-h`: display matched lines (without filenames)
  - `-i`: ignore case when matching
  - `-l`: display list of filenames with matches

https://www.geeksforgeeks.org/grep-command-in-unixlinux/
Demo: Grep
Questions

Lecture Participation Poll #3

Log onto [pollev.com/cse374](pollev.com/cse374)

Or

Text CSE374 to 22333
Lists in Bash

- Lists in bash are strings with multiple words separated by white space
  - Bash does not have arrays
Writing Scripts

- In a new file...
  - Make first line “#!/bin/bash” this specifies which interpreter to execute
  - Add your commands, save file
  - Make the file executable with chmod u+x