LINUX COMMAND LINE INTERFACE

Lab 9

EECS 448

Meenakshi Mishra and Dr Swapan Chakrabarti

Command Line Interface

- GUI is easier to operate for new users
- Lots of tasks that are easier done with command line interface
- GUI often impose limitations on speed and efficiency
- Keyboard is often faster to use than mouse
- Makes it possible to write scripts for repetitive tasks
- Professional software engineers and system administrators often prefer command line interface over GUIs

Limitations of GUI

- By presenting all options, add complexity to each specific use since all uses are supported
- Clicking mouse repetitively makes user feel busy, but not often the best way to do a task
- GUI based interfaces cannot be used by script based languages
- Remote access of CLI has lower BW
- Often GUI are wrappers around CLI
- Many systems do not provide GUIs for all tasks
- We will learn how to use the CLI

GUI/CLI Design Patterns

- Two common ways to design software to robustly support both CLI and GUI
- One way emphasizes library or module layers that provide powerful high level interfaces
  - GUI and CLI are then independently implemented thin wrappers for library routine calls
- Another uses CLI implementation as the primary interface for all applications
  - GUI is then independent application which constructs and emits CLI command strings

Shell

- Program that takes commands from keyboard and gives them to operating system
- Used to be the only user interface available for Unix-like systems
- Now, we have GUIs in addition to CLIs like shell
- On most systems, bash is the shell program
- Terminal or terminal emulator is program that opens the window and lets you interact with the shell
- Open the terminal on your screen
- Type random characters and press enter
- Press the up-arrow key (Notice that command history is saved)
- Mouse can still be helpful for scrolling, copying, pasting etc

Navigation

- Files arranged in hierarchical directory structure
  - First directory is root directory (No concept of drive letter, just one tree system present)
  - / : Root directory (Mostly only contains subdirectories)
  - /boot : Linux kernel and boot files stored. Kernel file is called vmlinuz
  - /etc : contains configuration files for the system
  - /bin, /usr/bin : contains most of programs for the system. /bin directory has essential programs that the system requires to operate; /usr/bin has applications for system's users
  - /sbin, /usr/sbin : programs for system administration
  - /usr : variety of stuff to support user applications
  - /usr/local : used for installation of software and files for use on local machine
  - /var : contains files that change as the system is running
  - /home : users keep personal work
  - /lib : Shared libraries are kept here
  - /root : Superuser's home directory
  - /tmp : temporary files are written here
  - /dev : contains devices that are available to the system.

Navigation

- pwd
  - Present working directory
  - When you first login, usually your pwd is /home/your_user_name
- cd
  - Change directory
  - cd *pathname*
  - Pathname can be absolute or relative to pwd
    - Pathname beginning with ‘/’ indicates absolute path (‘/’ indicating the root directory)
    - Pathname beginning with ‘.’ indicates the path with respect to the current directory
    - Pathname beginning with ‘..’ indicates the path with respect to the parent directory
    - Filenames are case sensitive
    - cd ~username changes to home directory of specified user
    - cd followed by nothing changes the directory to home directory of current user
- ls
  - List the items in pwd
  - Files starting with ‘.’ are hidden; to list those type ls –a
  - ls *pathname*
  - ls –l
  - ls –la
  - ls –d pathname
- Use tab to complete pathname

Try This

- Get your present working directories
- List the contents of your present working directory
- Navigate to some other directory
  - There is no need to type the entire path; use tab
- Come back to your home directory

Manipulating Files

- Wildcards
  - * : Matches any characters
  - ? : Matches a single character
  - [characters] : Matches any character that is member of set characters
    - [:alnum:]
    - [:alpha:]
    - [:digit:]
    - [:upper:]
    - [:lower:]
  - [!characters] : Matches any character that is not a member of set characters
  - Can use wildcards with any commands that accepts filename arguments
- cp
  - cp *file1 file2*
  - cp –i *file1 file2*
  - cp *file1 dir1*
  - cp –R *dir1 dir2*

Manipulating Files

- mv
  - mv *file1 file2*
  - mv –i *file1 file2*
  - mv *file1 file2 file3 dir1*
  - mv *dir1 dir2*
- rm
  - rm *file1 file2*
  - rm –i *file1 file2*
  - rm –i *dir1 dir2*
  - Removed stuff cannot be restored. Careful when using with wildcards. Make habit of listing all files first before removing.
- mkdir
  - mkdir *dir*
- du
  - du pathname

Try This

- Make a directory named Lab_9
- Go inside that directory
- Copy some existing file in some other directory to this directory
- Try removing some file from this directory
- Do not move a file, and then remove it unless you really want to delete it

Files

- File
  - file path
  - Tells type of object specified
  - Some are text and some are binary
- Reading files
  - cat path
  - more path
  - less path
- File content processing
  - grep “expression” path
    - Searches files in path line by line to find occurrences of “expression”
  - diff file1 file2
    - Find differences between two files

Files

- sed
  - Stream editor
  - Used for performing text transformation on an input stream
  - sed –e ‘expression’ path
  - One common use is substitution: s/e1/e2/g
- sort
  - sort path
- head
- tail
- find
  - find . –print :(Print paths of all files under current directory)
  - find / -name foo
  - find . –type f : (find . –type d)

File Permissions

- chmod
  - -rwx rwx rwx (owner, group, other users
  - chmod 777 filename
    - rwx rwx rwx = 111 111 111 = 777
    - rw- = 110 = 6
    - r-w = 101 = 5
    - r-- = 100 = 4
    - rw- --- --- = 110 000 000 =600
- chown
  - Change owner of a file
  - chown new_owner_name filename
- chgrp
  - Change group ownership of a file

Try This

- Check the permission of some file in your directory
- Change the permission of this file, so that all the other users except you can only read the files and you can read, write and execute
- Find all the files in your current directory

I/O Redirection

- Standard I/O ports
  - STDIN –all commands read STDIN (keyboard by default)
  - STDOUT – all commands write to STDOUT (screen by default)
  - STDERR – can be used separately for error output
- Redirect output port
  - > filename
    - Redirects the output of command to be written to file called filename
    - ls > filename.txt
    - ls >> filename.txt (appends file)
- Redirect input port
  - < filename
    - sort <filename.txt
    - sort <filename.txt >sortedfilename.txt
    - Order of redirection does not matter
    - Redirection should be after other options

Pipelines

- Redirect output of one command to input of other command
  - cmd1|cmd2 | cmd3 | cmd4
  - du ~ | sort –nr >hogs
  - ls –l | less
  - tar tzvf filename.tar.gz | less

Try This

- Create a list of files in your current directory in the file file_list.txt
- Take input from file_list.txt, sort it and write output in sorted_file_list.txt
- Create a list of files, sort it and then display it in a single line (use pipeline)

Expansions

- echo simple shell built in command to print its text argument on standard output
  - echo this is a test
  - echo *
  - echo D*
  - echo ~
- Pathname expansion
  - Wild card characters immediately expand the filename
- Arithmetic expansion
  - echo $((2+2))
  - echo $(((5**2)*3))
- Brace expansion
  - echo Number_{1..6}
  - mkdir {2003,2005}_{1..3}
- Parameter expansion
  - echo $USER
  - echo $SUER
    - Expansion still takes place, but results in empty string
- Command Substitution
  - echo $(ls)

Controlling Expansions

- echo try this
- echo The price is $2.00
- Double quotes
  - All special characters loose their meaning inside double quotes
    - Exceptions- $, \ (backslash), `(back-quote)
  - Word-splitting, pathname expansions, tilde expansion, brace expansion are suppressed
  - Parameter expansion, arithmetic expansion and command substitution still valid
  - ls two words.txt
  - ls “two words.txt”
  - echo “$USER $((5+3))”
  - echo $(cal)
  - echo “$cal”
  - Can use escape character in double quote to suppress expansion (“The price is \$2.00”)
- Single quotes
  - Suppresses all expansions

CLI Context

- All programs in linux are called processes
- Process runs within what is called an environment
- The parameters of environments are defined using environment variables
- Environment is defined by the environment variable
  - env lists the environment variables
  - echo $HOME
- PATH variable defines and ordered set of directories which are searched for command names
  - Members separated by colons
  - First match found is used
  - $Home/bin often put first for per-user commands

CLI Context

- ~/.bashrc file is used to provide customized definitions of environment variables, add to existing ones or create new ones
- which emacs : outputs path of file named emacs used as the command
- Store a script named emacs in HOME/bin
  - emacs command will execute this script
- history : records a history of commands
- !<reg-exp> lets you repeat first match to <reg-ex> while searching back in history
- alias cmd='string'
- Command Line Editing
  - Emacs commands work for navigation
  - Movement (C-a, C-e), cut (C-k), paste (C-y) etc

Job Control

- Each CLI command runs as separate child process
- Parent Bash process waits until child completes, then gives prompt
- Ampersand symbol tells the Parent not to wait and let child run in background
  - emacs &
- If you forget to put ampersand, stop the current job using C-z, then give command bg
- fg : puts a job to foreground
- ps lists all active processes
- kill processID : kill a job

Shell Scripts

- BASH script is a file containing set of statements
- Can look it up : man bash
- Write your own scripts
- Use a text editor of your choice
- Save the script below as helloworld
- Type ./helloworld
- #!/bin/bash

  # My first script

  echo “Hello World!”

Try this

- Write a shell script to create an html file
- The filename should be title.html
- The contents of the file should be

```
<html>
<head>
        <title>
        Sample html file
        </title>
</head>
<body>
        <h1>Sample</h1>
</body>
</html>
```