LECTURE 16: SHELL COMMANDS CONTINUED

HOW TO CONNECT

- Mac/Linux:
  - No need to connect, just use your own command line(Terminal)
  - ssh –p 130 username@server
  - Enter password when prompted
- Windows:
  - Command Prompt
    - Not identical
  - We provide access to our UNIX terminal
    - Connect via an SSH client, such as PuTTY
    - https://www.putty.org/

UNIX FILESYSTEM ACCESS PERMISSIONS

- Every file has an owner and an owner group
- You own new objects that you create
- The owner can set the permissions of a file
- An owner group allows a file to be shared among members of the same project
  - The file owner specifies what the members of the group can do with the file

THE ROOT ACCOUNT

- UNIX's omnipotent administrative user(UID 0)
- Also known as superuser account, but actual username is root
- Traditional UNIX allows for superuser to any valid operation on any file or process
- Example root-only processes:
  - Setting hostname
  - Setting system clock
  - Shutting down system
  - Creating device files

THE CP COMMAND

- Used to copy files: make a copy of src_file
- Common form: cp <src_file> <dest_file>
  - If dest file exists, it will be overwritten. If it does not exist, it is created. src_file not changed
- Another form: cp <src_file> <dest_dir>

THE CP COMMAND

- A third form: copy each src file into the dest directory
- cp <src_file1> <src_file2> <src_file3> <dest_file>
- Examples:
  - cp /tmp/337/* . copies all files in /tmp/337 directory into the current directory. Pay attention to same-name files!
  - cp ~aaydin/* ~ copies all files in user aaydin's home directory into the current user's home directory
  - cp lab1.cpp lab2.cpp .. copies the two cpp files to parent directory

THE MV COMMAND

- Used to move or rename files and directories
- Same form as cp. Does not create a copy of source file, but moves the source itself to the new location
  mv <src_file> <dest_file>
  mv <src_file> <dest_dir>
  mv <src_file1> <src_file2> <src_file3> <dest_dir> move 3 source files into destination directory

THE MV COMMAND

- The source does not have to be a file, can be directories to
  mv <src_dir> <dest_dir>
  mv <src_dir1> <src_dir2> <src_dir3> <src_dir4> <dest_dir>

THE MAN COMMAND

- Allows to read online manual pages for commands and utilities
  man <command_name>
- Examples: man mkdir, man cd, man more

SHELL UTILITIES AND SHELL SCRIPTS

UNIX SHELL UTILITIES

- UNIX shell utilities
- Shell variables
- Quoting
- Running shell scripts
- Shell scripts for system administration

UNIX SHELL UTILITIES

- File manipulation utilities
  - To display a specified portion of a file for a quick look using head and tail
- Redirections
- Pipes
- Sequencing commands using semicolon

THE HEAD COMMAND

- head myfile
  - Shows the first 10 lines of myfile
- head -5 myfile
  - Shows the first 5 lines of myfile
- head myfile yourfile ourfile
  - Shows the first 10 lines of each of the files
- head –c5 myfile or head –c 5 myfile
  - Shows the first 5 lines of myfile

THE TAIL COMMAND

- tail myfile
  - Shows the last 10 lines of myfile
- tail -5 myfile
  - Shows the last 5 lines of myfile
- tail myfile yourfile ourfile
  - Shows the last 10 lines of each of the files
- tail –c5 myfile or tail –c 5 myfile
  - Shows the last 5 characters of myfile

REDIRECTION

- Among the most useful facilities that the shell provides are the shell redirection operators
- Every process has at least 3 communication channels to use: STDIN, STDOUT, STDERR
  - Each can connect to a terminal window, a file, a network connection or another process' channel
  - A process is any program in execution. The program can be the shell, an application, or a program that you write

REDIRECTION OPERATORS

- Most commands accept their input from STDIN, write their output to STDOUT, write error messages to STDERR
  - STDIN : the keyboard
  - STDOUT: the display
  - STDERR: the display
- < connects a command's STDIN to the content of a file instead
- > redirects STDOUT to replace an output file's existing content
- >> redirects STDOUT and append to a file

REDIRECTION EXAMPLES

- echo “This is a test message” > /tmp/mymessage
- cat < /tmp/mymessage
- >& redirects both STDOUT and STDERR to the same place
- 2> redirects stderr only
- /dev/null discards written data, but reports if the write operation is successful (https://en.wikipedia.org/wiki/Null_device)

PIPES

- The shell allows one to use standard output of one process as the standard input to another process.
- General form command A | command B
- ps –ef | grep sshd
- ls | sort

SEQUENCING COMMANDS

- One can enter series of commands separated by “;” in a single line
- date; pwd; ls

MORE UNIX SHELL UTILITIES

- find: locate files
- grep: search for a specified pattern in a file or lists of files
- sort: sort contents of a file
- cat: display contents

SHELL VARIABLES

- The shell handles the user interface and acts as a command interpreter
- It needs and keeps track of information such as your HOME directory, terminal type etc.
- This information stored in shell variables
  - Environment variables: defined by system admins (Some are in output of env)
  - Local variables: user defined

ENVIRONMENT AND SHELL VARIABLES

- One can define variables for a shell (e.g., x=3. No spaces, and the variable name should start with a letter)
- In csh or tcsh, use set x=3
- To use a shell variable, use $, e.g., echo $x
- To delete a variable, use unset x
- Important built-in shell variables:
  - $PWD stores a string recording the name of the current directory
  - $PATH stores the directories the shell uses to search for executables for each issued command

SHELL STARTUP FILES

- When you run bash, bash will first execute whatever bash commands that are in the system file /etc/profile
- It sets $PATH to a basic value, and some other shell variables
- You can include more directories to your path
- To do so, add a line to .bashrc or .profile files in your home directory
- /etc/profile, ~/.bashrc and ~/.profile are your shell startup files
  - System-wide: /etc/bash.bashrc
- A skeleton available at: /etc/skel/.bashrc

QUOTING

- Single and double quotes
- Strings in single or double quotes are treated similarly, except double quotes strings are subject to variable expansion

```
$ mylang="Pennsylvania Dutch”
$ echo "I speak $mylang.”
I speak Pennsylvania Dutch.
$ echo 'I speak $mylang.'
I speak $mylang.
```

BACK QUOTES

- Use back quotes to call a command inside a string

```
$echo "Today is `date`."
Today is Mon Feb 21 11:17:56 EST 2011.
```

FIN!