LECTURE 17: UNIX SHELL UTILITIES

UNIX SHELL UTILITIES

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

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

PIPES VS REDIRECTION

- Pipes: From command(or program) to command
  - ls | sort
- Redirection: From command to file(Or from file to command)
  - ls > ls_output.txt
  - ls < folders.txt

SEQUENCING COMMANDS

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

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.
```

RUNNING SHELL SCRIPTS

- A shell script is a script written for the shell, i.e., command line interpreter, of an OS
- A shell is often considered a simple domain-specific programming language
- Typical operations performed by shell scripts include file manipulation, program execution and printing text

HOW TO RUN A SHELL SCRIPT

- First, get the script
  - scp, wget, nano ☺
- To run it
  - First give the exec permission: chmod u+x helloworld.sh
  - Type “./helloworld.sh”
- Or bash helloworld.sh

SAMPLE SCRIPTS

```
#!/bin/bash
# Let's name this file helloworld
echo "Ni Hao, World!"
```

SHELL POSITIONAL VARIABLES

- A shell script can read up to 10 command line parameters (also called arguments) into special variables called shell positional variables
  - $0: Contains the name of the script, as typed on command line
  - $1, $2, ..., $9 Contains the first through 9th command line arguments
  - $# Contains the number of command line arguments
  - $@ Contains all command line parameters
  - $? Contains the exit status of the last command

SHELL SCRIPTS FOR SYSTEM ADMIN

- Pile shell commands into a shell script
- Scripting for system administration
  - logrotate
  - cron daily backups

SCRIPTING FOR SYSTEM ADMINISTRATION

- Example: Managing log files automatically
- Logs are produced when running programs
- /usr/sbin/logrotate is an executable that rotates the log of different programs
- If rotated daily, after one day, log.2 becomes log.3, log.1 becomes log.2, current log becomes log.1
- The/etc/cron.daily/logrotate is the shell script that executes the logrotate command everyday

SCRIPTING FOR SYSTEM ADMINISTRATION

- /etc/sbin/logrotate uses /etc/logrotate.conf for configuration
- /usr/sbin/logrotate /etc/logrotate.conf
- Conf file can include more configuration files in a directory
  - /etc/logrotate.d/
- Other cronjobs under: /etc/cron.daily/

FIN!