Advanced Unix Programming

Module 01

Raju Alluri
askraju @ spurthi.com

Advanced Unix Programming: Module 1

- Why Unix
- The Unix Login Session
- Working with the Unix Filesystem
- The `vi`-editor
- File Handling and File Permissions
- Process Utilities
- Disk Utilities
- Network Utilities

Why Unix

- Unix is robust, proven, seasoned...
  - Invented in 60's
  - Modular, Secure by design, Efficient
- Unix is fun to play with
  - Misunderstandings about Unix being tough
- Unix is open
  - The OS itself gives various hooks and entries into the operating system

Why Unix, #2

- Unix is everywhere
  - Several variants available for use of differing needs
    - Solaris, Linux, HP-UX etc.
  - Scales from single CPU systems to the order of 100+
- Classic & Strong fundamentals, yet evolving to the needs of the industry
- Has GUI, but best way to learn is using non-GUI interactions.

Typical Unix Login Session

- User logs in with Username, password
  - Username: `student`
  - Password: `********`
- User is logged into a command interpreter called “shell”
  - Bourne Shell (`sh`), C-Shell (`csh`), Korn Shell (`ksh`), Bourne Again Shell (`bash`)

Typical Unix Login Session, #2

- User logs into a directory called “home directory”
  - e.g. `/home/student`
- User executes “commands”
  - Most commands work same in all shells
    - Input/Output redirection, shell level commands differ
- User logs out

Unix Files Basics

- Primitive uses of interactions with file system are
  - Directories (Folders)
  - Files
- Directory Operations
  - Directory Creation
    - `mkdir`
  - Directory Deletion
    - `Rmdir`

Unix Files Basics, #2

- File Operations
  - Create a file
    - `touch, cp, mv`
    - Create a file by using an editor like vi
  - Delete a file
    - `rm, unlink`

Using the `vi` editor

Open a file (say helloworld.txt) using

```
vi helloworld.txt
```

- By default, you will be in what is known as “command” mode
- Press `a` to go to “edit” mode and start typing the text
- Once done, press `Esc` to return to “command” mode
- Press `:w<enter>` to save the file, `:q<enter>` to quit
- Check the contents of the file

```
cat helloworld.txt
```

Command mode operations of vi

- l – go to right
- h – go to left
- j – go down
- k – go up
- a – append
- A – append at end of line
- o – add a new line after the current line
- O – add a new line before the current line
- x – delete a character
- dd – delete the line
- w – go to next word
- dw – delete word
- yy – copy current line into buffer
- p – print buffer contents after current line
- P – print buffer contents before current line

Command mode operations of vi, #2

- You have to press <enter> after following
  - :w – save the file
  - :q – quit the vi editor
  - :q! - quit the vi editor without saving
  - :!<command> - execute the <command> within the shell
  - :r <filename> - read file to location after current line

For Your Practice

- Create a file called helloworld.txt that contains the following text

```
Hello World!
```

- Edit the file from the above exercise and make the text repeat for 10 lines. Try not to retype the text in all the 10 lines
- Edit the file from the above exercise and reduce it to 5 lines, by using line-wise deletion
- Edit the file from the above exercise and make the first two lines contain "`Hello`" only.

Disks & File Systems

- An entire disk is partitioned into file systems
  - Each filesystem has
    - A base directory
    - One or more subdirectories and files
    - Each subdirectory might contain further subdirectories
  - Each filesystem has
    - Space arranged into what is known as i-nodes
    - (more details about inodes and filesystems later)
- Each filesystem needs to be “mounted” for it to be usable

Disks & File Systems, #2

- How to mount a filesystem

```
mkdir /mnt/cdrom
mount /dev/cdrom /mnt/cdrom
```

- How to unmount a filesystem

```
umount /mnt/cdrom
```

- How to see the space left on a file system

```
df <name_of_filesystem>
df -k <name_of_filesystem>
```

-

Disks & File Systems, #3

- Mounting a filesystem can also be done with the help of /etc/mnttab entries
  - /etc/fstab entry

```
/dev/cdrom    /mnt/cdrom     iso9660
 noauto,owner,ro  0   0
```

  - Mount command

```
mount /mnt/cdrom
```

-

Disk & File Utilities

- Linking one file to another

```
ln -s orig_file [link_name]
```

  - Soft link: works across filesystems

```
ln orig_file [link_name]
```

  - Hard link: Works only within filesystems

- Finding all the files that match with a name pattern

```
Find dirname [-name pattern] [-print] [-exec
  ...]
```

  - Find all the files starting with dirname that match with pattern and print them. Execute the command that follows -exec.
  - Very powerful command

Disk & File Utilities, #2

- du: Disk usage

```
du filename|dirname
du -s dirname
```

Users and Groups

- Each person that needs to login to a Unix system needs a username and password
  - e.g.: `student, ********`
- Groups are logical groupings of zero or more users
- Each file/directory belongs to a user and a group
- The user and group will have specific permissions on the file/directory
  - Read (r)
  - Write (w)
  - Execute (x)

File Permissions

- Each file has permissions for user (u), group (g) and all others (o)

```
-rw-rw-r--  1 student  student 0 Aug 31 21:58 helloworld.txt
```

- The first character denotes the type of the file
  - (to be discussed in detail later)
- The first {`rwx`} set indicates the permissions for user
- The second {`rwx`} set indicates the permissions for group
- The third {`rwx`} set indicates the permissions for others
- You can use decimal equivalents of above numbers

File Permissions, #2

- Using decimal notation
  - `rwx = 7`
  - `rw- = 6`
  - `r-x = 5`
  - `-w- = 2`
- You can use chmod command to change permissions for file

```
chmod 777 helloworld.txt
chmod 644 helloworld.txt
```

- The listing also gives the user and group information of the file

File Permissions, #3

- Another way of changing file permissions
  - Use u for user, g for group, o for others, a for all
  - Use + for giving permissions, - for not giving permissions
  - Use r for read, w for write, x for execute

```
chmod a+r helloworld.txt
chmod a-w helloworld.txt
chmod g-x helloworld.txt
```

-

Processes

- Each program in unix runs as a process
  - Processes can create other processes
  - The process that creates another one is called parent process
  - The process that is created is called child process
  - e.g. The shell you login to is a process by itself
- Processes can interact with the files, processes and other resources
- Each process opens 3 files by default
  - standard input, standard output and standard error

Processes

- Process Characteristics
  - Each process will be identified by a unique ID called ProcessID or PID.
  - Each process belongs to a user and is denoted by UID
  - Each process is executed from a file called command
  - Each process consumes resources
    - CPU Time
    - Memory

Commands related to processes

- ps: the process listing

```
ps -u <username>
ps -l
ps -f
ps -aef
```

- ulimit: usage limits for all processes that are spawned by this session

```
ulimit -a
```

Networking Basics

- A network connects two or more systems
- Each system on the network is addressed by a hostname or an IP address
  - Hostname looks like `myserv` (hostname) or `myserv.rdom` (hostname in another domain)
  - IP address looks like 192.168.0.2 (IPv4 Address)
    - IPv6 addresses will be discussed later
- Networking utilities are used to communicate over the network

Basic Networking Utilities

- ftp: File Transfer Protocol

  `ftp [ <hostname> ]`

  - You will be asked for username (a valid username on the target hostname) and password
  - Once you login, you can execute several commands (`cd, get, put, bin, asc, help` etc.)

- telnet: TELNET protocol based communication

  `telnet [-l <username>] [<hostname>]`

  - Telnet to hostname using username

Basic Networking Utilities, #2

- rlogin: Remote login

  ```
  rlogin [-l username] hostname
  ```

  - Allows you to login to a remote host
- finger: User information lookup

  ```
  finger [ username | username@hostname ]
  ```

  - Lookup user's details such as when was the last login, what shell is used, if there is mail etc.

Basic Networking Utilities, #3

- arp: ARP (Address Resolution Protocol) cache management

```
arp -a
arp hostname
```

Other Utilities

- who: who is logged in currently

  `who`

  – Who is logged in currently

  `who -b`

  – What is the last boot time

- w: who logged in and what are they doing

  `w`

  – List all the users currently logged in and what they are doing