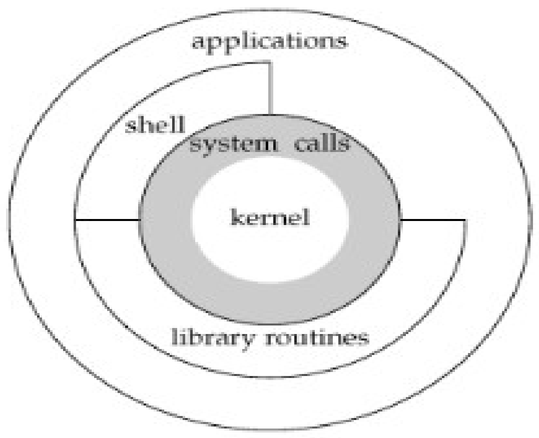
Linux Operating System and Programming

TOPIC#2 : The UNIX Architecture and Command Usage

Architecture of the UNIX operating system

Architecture of the UNIX operating system



Courtesy: Advanced Programming in the UNIX Environment
by W. Richard Stevens, Stephen A. Rago II Edition

The interface to the kernel is a layer of software called the system calls (the shaded portion in Figure 1.1). Libraries of common functions are built on top of the system call interface, but applications are free to use both. The shell is
a special application that provides an interface for running other applications.

n a broad sense, an operating system is the kernel and all the other software that makes a computer useful and gives the computer its personality. This other software includes system utilities, applications, shells, libraries of common functions, and so on.

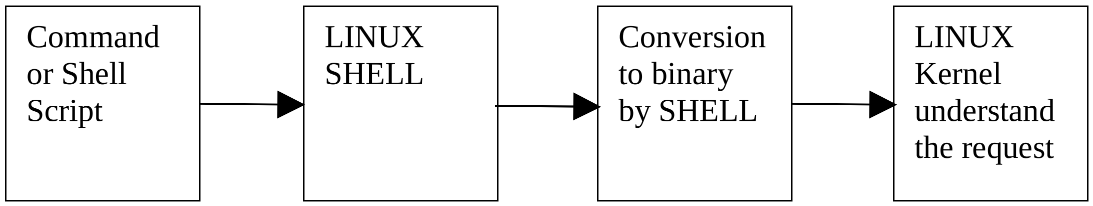
Kernel and Shell

The kernel is the core of the operating system- a collection of routines mostly written in C. It is loaded in to memory when the system is booted and communicates directly with the hardware.

Kernel is heart of Linux O/S. It manages resource of Linux O/S. Resources means facilities available in Linux. For example: dFacility to store data, print data on printer, memory, file management etc. Kernel decides who will use this resource, for how long and when. It runs your programs (or set up to execute binary files).

The following tasks are performed by the kernel
1) I/O management
2) Process management
3) Device management
4) File management
5) Memory management

The Shell accepts the users' instruction or commands in English and translates it into computers native binary language.


Command or Shell Script
LINUX SHELL
Conversion to binary by SHELL
LINUX Kernel understand the request

Shell is actually the interface between the user and the kernel.

<u>Different Types of Shell:-</u>
1) BASH (**B**ourne-**A**gain **SH**ell) developed by **Brain Fox** and **Chet Ramey** is most common shell in Linux.
2) CSH (**C SH**ell), developed by **Bill Joy** has similar **syntax (?)** and usage to **C programming language**.
3) KSH (**K**orn **SH**ell) developed by **David Korn**.

To check or know which shell is running use the command **echo $SHELL**

File and Process

A **file** is just an array of bytes and can contain virtually anything. It is also related to another file by being part of a single hierarchical structure.

Process is the name given to a file when it is executed as a program.

System Call

The functions used to communicate with the kernel are known as **system calls**.

Main features of UNIX

1. **Multiuser system**

UNIX is multiprogramming system i.e. it permits multiple programs to run and compete for the attention of the CPU.

- Multiple users running separate jobs
- Single user running multiple jobs

2. **Multitasking system**

UNIX is a multitasking system. A single user can run multiple tasks concurrently.

3. **Building block approach**

The commands can be connected using pipes and filters. This is the building block approach in UNIX.

4. **UNIX toolkit**

There are general-purpose tools, text manipulation utilities (called filters), compilers and interpreters, networked applications and system administrator tools available as UNIX toolkits.

5. **Pattern matching**

The * is a special character used by the system to indicate that it can match a number of filenames. There are several other *metacharacters*. This comprises the pattern matching feature of UNIX.

6. **Programming Facility**

Controls structures, loops and variables are the features used to design **shell scripts** – programs that can invoke the UNIX commands.

7. **Documentation**

Man command and UNIX documentation is the most important reference for commands and their configuration. Also many resources are available on the Internet and there are several newsgroups where problems related to shell programming, network configuration, etc are discussed.

There is a group of standards **Portable Operating System Interface for Computer Environments** (POSIX) developed by **Institution of Electrical and Electronics Engineers** (IEEE). Two most cited standards from the POSIX family are known as POSIX.1 (It specifies C application program interface – the system calls) and POSIX.2 (It deals with the shell and utilities).

Internal and External Commands

The commands that are in-built are known as **internal commands**. Like **echo** command.

The commands like **ls** that has an independent existence as a program or file in the /bin or /usr/bin directory are known as **external commands**.

Commands often accept several types of **arguments**. **Option** is a special type of argument beginning with – sign. Many commands take filename(s) as argument so that command can take input from the file. The command with its arguments and options is known as **command line**. This line can be considered complete only after the user has hit enter key. The complete line is then fed to the shell as its input for interpretation and execution. Also commands like **pwd** and **who** will not take any arguments.
UNIX allows you to specify more than one command in the command line. Each command has to be separated from the other by a ; (semicolon).

PATH and export

PATH is an *environmental variable* in Linux and other Unix-like operating systems that tells the *shell* which directories to search for *executable files* (i.e., ready-to-run programs) in response to commands issued by a user.

To see PATH variable use:
echo $PATH
To set PATH variable use **PATH=$PATH:new_folder_contain_binaries**
e.g. export PATH=$PATH:/usr/share/java

A path is a unique location to a file or a folder in a file system of an OS. A path to a file is a combination of / and alpha-numeric characters.

Please, note that if you export any value to a variable, that is available within that current session only. You may like to write this command in rc profile files so that that settings are available on evey login of that user.

```
cd  ~
vi .bash_profile

PATH=$PATH:$HOME/bin:$HOME/<myprogrampathgoeshere>
export PATH
```

If you change .bash_profile and want have its effect immediately in active session, source the profile once manually as below:
source .bash_profile

or

. .bash_profile

Helplines in linux os:

- man command stands for manual which provide manual of most linux commands.

  e.g. man cat

  It displays manual of cat command and to quit from man press q.

- --help

Most linux command shows help using command option –help. e.g. cat --help

- Using Up and down arrow keys, you may traverse through recent commands
- command 'history' gives you history of commands executed on certain computer.
- Tab key provides choices available and fulfils text when matched.

Interested in Linux networking and administration or development? Few basics follows below:

- **Superuser**

The **superuser** is a special user account used for system administration. **sudo** is a program for Unix-like computer operating systems that allows users to run programs with the security privileges of another user (normally the superuser, or root).

Command 'visudo' as root let you edit sudo configurations.

Most commands like useradd, userdel, passwd -u etc are reserved by superuser.

- **Remote Login - SSH (Secure Shell)**

Secure Shell (**SSH**) is a cryptographic network protocol for secure data communication, remote command-line login, remote command execution, and other secure network services between two networked computers that connects, via a secure channel over an insecure network, a server and a client (running SSH server and SSH client programs, respectively).

i.e ssh <username>@<hostip>

On windows pcs, you may use putty or xshell for schools to do remote login to a linux server in the network.

You can login to a remote Linux server without entering password in 3 simple steps using ssky-keygen and ssh-copy-id as explained in this article.

ssh-keygen creates the public and private keys. **ssh-copy-id** copies the local-host's public key to the remote-host's authorized_keys file. ssh-copy-id also assigns proper permission to the remote-host's home, ~/.ssh, and ~/.ssh/authorized_keys.

- **X-Manager**

If you wish to have X-Manager work and be able to export display to remote screen likewise. As root you may need to do following:

```
vi /etc/ssh/ssh_config

  ForwardAgent yes

  ForwardX11 yes
```

```
/sbin/service sshd status
/sbin/service sshd restart
```

For exporting display, in xshell configuaration/preferences there is option to choose how you want to export display.

To see current value of environment variable env and DISPLAY do as below:

```
env
echo $DISPLAY
```

To set a display parameter manually,

```
export DISPLAY=localhost:0.0
```

To test

```
xclock
```

- **Scheduling execution of commands to day and time wise using CRON**

The software utility **cron** is a time-based job scheduler in Unix-like computer operating systems. People who set up and maintain software environments use cron to schedule jobs (commands or shell scripts) to run periodically at fixed times, dates, or intervals. It typically automates system maintenance or administration—though its general-purpose nature makes it useful for things like connecting to the Internet and downloading email at regular intervals.

Cron is driven by a ***crontab*** **(cron table)** file, a configuration file that specifies shell commands to run periodically on a given schedule. The crontab files are stored where the lists of jobs and other instructions to the cron daemon are kept. Users can have their own individual crontab files and often there is a system wide crontab file (usually in /etc or a subdirectory of /etc) that only system administrators can edit. Each line of a crontab file represents a job, and is composed of a CRON expression, followed by a shell command to execute.

[Courtesy: http://www.thegeekstuff.com/2009/06/15-practical-crontab-examples/]

Linux Crontab Format

```
MIN HOUR DOM MON DOW CMD
```

Table: Crontab Fields and Allowed Ranges (Linux Crontab Syntax)

Field	Description	Allowed Value
MIN	Minute field	0 to 59
HOUR	Hour field	0 to 23
DOM	Day of Month	1-31

MON	Month field	1-12
DOW	Day Of Week	0-6
CMD	Command	Any command to be executed.

As root:

vi /etc/crontab

Add entry like below:

The basic usage of cron is to execute a job in a specific time as shown below. This will execute the Full backup shell script (full-backup) on **10th June 08:30 AM**.

Please note that the time field uses 24 hours format. So, for 8 AM use 8, and for 8 PM use 20.

```
30 08 10 06 * /home/ramesh/full-backup
```

See that it is necessary that the file path and permissions are properly set for the user to be able to execute script.

For any particular user:

List current user's crontab:
crontab -l

Edit current user's crontab:
crontab -e

- **Rc files**

RC file is a Script file containing startup instructions for an application program (or an entire operating system), usually a text file containing commands of the sort that might have been invoked manually once the system was running but are to be executed automatically each time the system starts up.

Contributors (CE Department, DDU):
- Prof. Niyati J. Buch
- Prof. Jigar M. Pandya