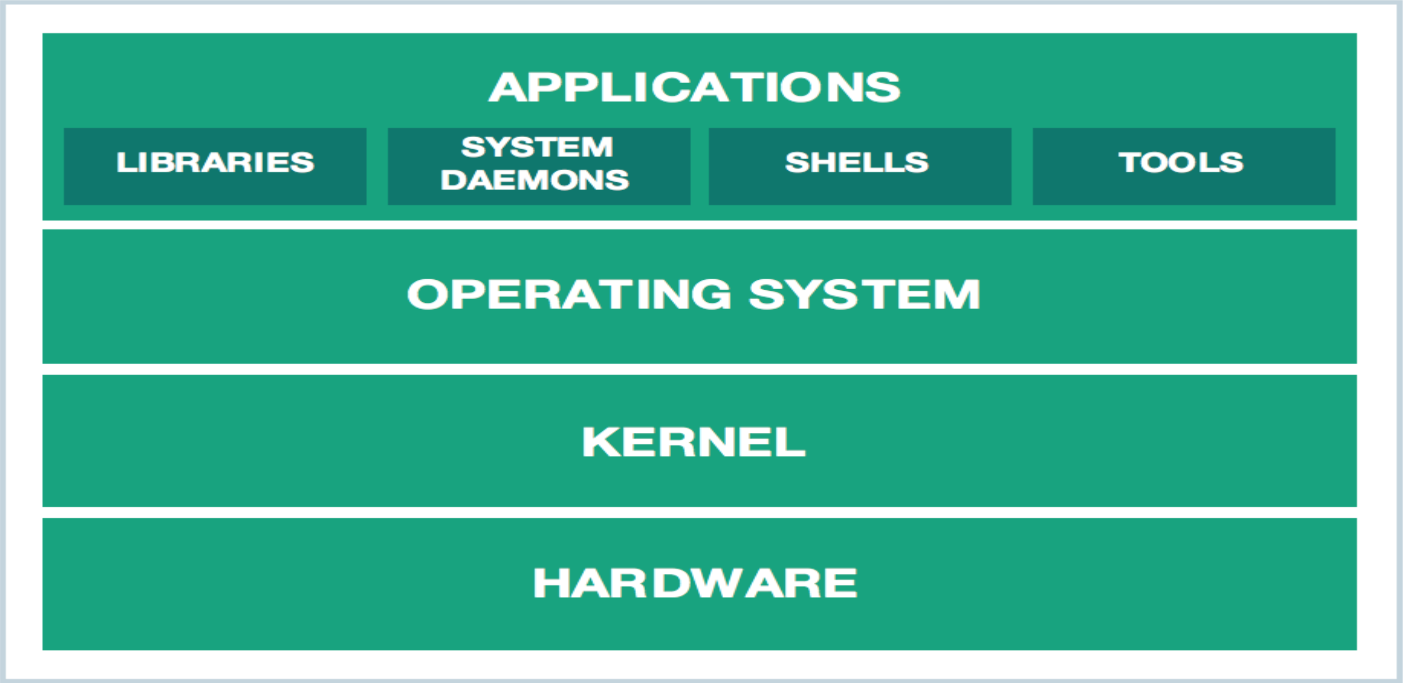
UNIX

Unix Architecture:



Command	Description	Flags
man	Displays information about any Linux command, including the commands used to start GroupWise programs.	
Users	List users in your machine	
whoami	Displays who you are logged in as.	
uname	Prints system info	a/r/m/v : all info/kernel-release/kernel-machine/version
Logout	To logout	
Cd	Change directory	~/- / ~username
Date	To display current date	
Cal	To display calendar	
Clear	It is used to clear the screen	
Ls	List files	-l / -a / -s / h / -r / -t
		*.doc
Cat *filename*	Displays content of file	
Wc *filename*	Word count file name	
Mkdir *dir_name*	Make directory	-p
Rmdir *dir_name*	Remove directory	
Cp *file1 file2*	Copy files	-r /-f
Mv file1 location	Move files	-r / -f

rm *filename*	Remove file	
Pwd	Current working directory	
Echo	Prints content on screen	
File	Finds a file in current working directory	

Going to home directory

Cd ~

Access modes in unix:

Read ->	**r**
Write->	**w**
Execute->	**x**

Permissions in unix:

First 3 chars(2-4): permission for file owner
Second group of 3 chars(5-7): permission for group of users
Third group of 3 chars(8-10): permission for others

Types of permission:

U -> User
G -> Group
O -> Other

Changing the file permissions:

1	**+** Adds the designated permission(s) to a file or directory.
2	**-** Removes the designated permission(s) from a file or directory.

```
Chmod u+rwx filename
Chmod u-rwx filename
Chmod u+r filename

Chmod g+rx filename
Chmod o-x filename
```

```
chmod o+wx,u-x,g+rx filename
```

Using chmod with Absolute Permissions

The second way to modify permissions with the chmod command is to use a number to specify each set of permissions for the file.

Each permission is assigned a value, as the following table shows, and the total of each set of permissions provides a number for that set.

Number	Octal Permission Representation	Ref
0	No permission	---
1	Execute permission	--x
2	Write permission	-w-
3	Execute and write permission: 1 (execute) + 2 (write) = 3	-wx
4	Read permission	r--
5	Read and execute permission: 4 (read) + 1 (execute) = 5	r-x
6	Read and write permission: 4 (read) + 2 (write) = 6	rw-
7	All permissions: 4 (read) + 2 (write) + 1 (execute) = 7	rwx

```
chmod 755 testfile

chmod 743 testfile

chmod 043 testfile
```

To check commands history

```
history

history -d line_number
```

Environment Variables

An important Unix concept is the **environment**, which is defined by environment variables. Some are set by the system, others by you, yet others by the shell, or any program that loads another program.

```
TEST="Unix Programming"
$echo $TEST
```

It produces the following result.

```
Unix Programming
```

Note that the environment variables are set without using the **$** sign but while accessing them we use the $ sign as prefix. These variables retain their values until we come out of the shell.

When you log in to the system, the shell undergoes a phase called **initialization** to set up the environment. This is usually a two-step process that involves the shell reading the following files −

- /etc/profile
- profile

The .profile File

The file **/etc/profile** is maintained by the system administrator of your Unix machine and contains shell initialization information required by all users on a system.

The file **.profile** is under your control. You can add as much shell customization information as you want to this file.

Setting the PATH

When you type any command on the command prompt, the shell has to locate the command before it can be executed.

The PATH variable specifies the locations in which the shell should look for commands. Usually the Path variable is set as follows −

```
$PATH=/bin:/usr/bin
```

Environment Variables set by System

HOME

Indicates the home directory of the current user: the default argument for the cd **built-in** command.

PATH

Indicates the search path for commands. It is a colon-separated list of directories in which the shell looks for commands.

PWD

Indicates the current working directory as set by the cd command.

RANDOM

Generates a random integer between 0 and 32,767 each time it is referenced.

UID

Expands to the numeric user ID of the current user, initialized at the shell startup.

Piping: Connecting one or more commands together

You can connect two commands together so that the output from one program becomes the input of the next program. Two or more commands connected in this way form a pipe.

To make a pipe, put a **vertical bar (|)** on the command line between two commands.