**What is Unix?**

The Unix operating system is a set of programs that act as a link between the computer and the user.

Kernel

Operating System

Shell -> Users communicates with the kernel through a program known as the **shell.**

The shell is a command line interpreter; it translates commands entered by the user and converts them into a language that is understood by the kernel.

* Unix was originally developed in 1969 by a group of AT&T employees Ken Thompson, Dennis Ritchie, Douglas McIlroy, and Joe Ossanna at Bell Labs.
* There are various Unix variants available in the market. Solaris Unix, AIX, HP Unix and BSD are a few examples. Linux is also a flavor of Unix which is freely available.
* Several people can use a Unix computer at the same time; hence Unix is called a multiuser system.
* A user can also run multiple programs at the same time; hence Unix is a multitasking environment.

The main concept that unites all the versions of Unix is the following four basics –

1. Kernel
2. Shell
3. Commands and Utilities
4. Files and Directories

# File Management:-

Ordinary Files

Directories

Special Files

# File Permission

* **Owner permissions** − The owner's permissions determine what actions the owner of the file can perform on the file.
* **Group permissions** − The group's permissions determine what actions a user, who is a member of the group that a file belongs to, can perform on the file.
* **Other (world) permissions** − The permissions for others indicate what action all other users can perform on the file.

## File Access Modes:/ Directory Access Modes/

### Read

### Write

### Execute

## Changing Permissions

|  |  |
| --- | --- |
| 1 | (+)Adds the designated permission(s) to a file or directory. |
| 2 | (-)Removes the designated permission(s) from a file or directory. |
| 3 | (=)Sets the designated permission(s). |

# Environment

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 process is as follows −

* The shell checks to see whether the file **/etc/profile** exists.
* If it exists, the shell reads it. Otherwise, this file is skipped. No error message is displayed.
* The shell checks to see whether the file **.profile** exists in your home directory. Your home directory is the directory that you start out in after you log in.
* If it exists, the shell reads it; otherwise, the shell skips it. No error message is displayed.

As soon as both of these files have been read, the shell displays a prompt −

$

## 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. The minimum set of information that you need to configure includes −

* The type of terminal you are using.
* A list of directories in which to locate the commands.
* A list of variables affecting the look and feel of your terminal.

You can check your **.profile** available in your home directory. Open it using the vi editor and check all the variables set for your environment.

## Setting the Terminal Type

## Setting the PATH

## PS1 and PS2 Variables

The characters that the shell displays as your command prompt are stored in the variable PS1. You can change this variable to be anything you want. As soon as you change it, it'll be used by the shell from that point on.

or example, if you issued the command −

$PS1='=>'

=>

=>

=>

Your prompt will become =>. To set the value of **PS1** so that it shows the working directory, issue the command −

# Pipes and Filters

# Processes Management

* Foreground Processes
* Background Processes
* Foreground Processes => By default, every process that you start runs in the foreground.
* Background Processes => The advantage of running a process in the background is that you can run other commands; you do not have to wait until it completes to start another!

The simplest way to start a background process is to add an ampersand (**&**) at the end of the command.

## Parent and Child Processes

Each unix process has two ID numbers assigned to it: The Process ID (pid) and the Parent process ID (ppid). Each user process in the system has a parent process.

## Zombie and Orphan Processes

Normally, when a child process is killed, the parent process is updated via a **SIGCHLD** signal. Then the parent can do some other task or restart a new child as needed. However, sometimes the parent process is killed before its child is killed. In this case, the "parent of all processes," the **init** process, becomes the new PPID (parent process ID). In some cases, these processes are called orphan processes.

When a process is killed, a **ps** listing may still show the process with a **Z** state. This is a zombie or defunct process. The process is dead and not being used. These processes are different from the orphan processes. They have completed execution but still find an entry in the process table.

## The top Command

The **top** command is a very useful tool for quickly showing processes sorted by various criteria.

# Network Communication Utilities

## The ping Utility

## The ftp Utility

* Connect and login to a remote host.
* Navigate directories.
* List directory contents.
* Put and get files.
* Transfer files as **ascii**, **ebcdic** or **binary**.

Command => ftp hostname or ip-address