**UNIT V- LINUX SYSTEM ADMINISTRATION**

**LINUX OPERATING SYSTEM**

1. **Introduction:**

Linux started out as a UNIX variant for the IBM PC (Intel 80386) architecture. Linus Torvalds, a Finnish student of computer science, wrote the initial version. Torvalds posted an early version of Linux on the Internet in 1991. Since then, a number of people, collaborating over the Internet, have contributed to the development of Linux, all under the control of Torvalds. Because Linux is free and the source code is available, it became an early alternative to other UNIX workstations, such as those offered by Sun Microsystems and IBM. Today, Linux is a fullfeatured UNIX system that runs on all of these platforms and more, including Intel Pentium and Itanium, and the Motorola/IBM PowerPC. Key to the success of Linux has been the availability of free software packages under the auspices of the Free Software Foundation (FSF). FSF's goal is stable, platform-independent software that is free, high quality, and embraced by the user community. FSF's GNU project provides tools for software developers, and the GNU Public License (GPL) is the FSF seal of approval. Torvalds used GNU tools in developing his kernel, which he then released under the GPL. Thus, the Linux distributions that you see today are the product of FSF's GNU project, Torvald's individual effort, and many collaborators all over the world. In addition to its use by many individual programmers, Linux has now made significant penetration into the corporate world. This is not only because of the free software, but also because of the quality of the Linux kernel. Many talented programmers have contributed to the current version, resulting in a technically impressive product.

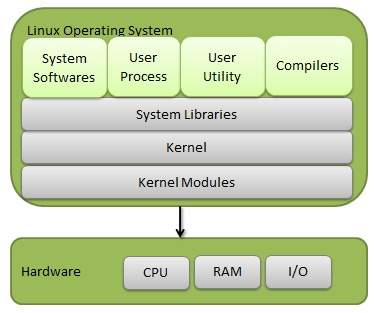
Basic features :

* Portable – Portability means softwares can works on different types of hardwares in same way.Linux kernel and application programs supports their installation on any kind of hardware platform.
* Open Source – Linux source code is freely available and it is community based development project. Multiple teams works in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
* Multi-User – Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
* Multiprogramming – Linux is a multiprogramming system means multiple applications can run at same time.
* Hierarchical File System – Linux provides a standard file structure in which system files/ user files are arranged.
* Shell – Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs etc.
* Security – Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

## B. Components of Linux System

Linux Operating System has primarily three components

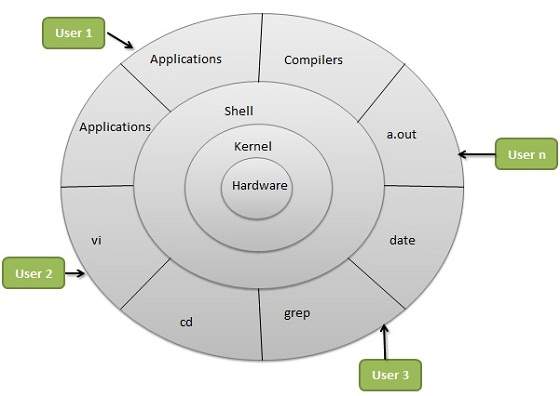
* Kernel − Kernel is the core part of Linux. It is responsible for all major activities of this operating system. It consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.
* System Library − System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries implement most of the functionalities of the operating system and do not requires kernel module's code access rights.
* System Utility − System Utility programs are responsible to do specialized, individual level tasks.



Components of Linux Operating system.

## 

## Linux Architecture

The following illustration shows the architecture of a Linux system −

The architecture of a Linux System consists of the following layers −

* Hardware layer − Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc).
* Kernel − It is the core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.
* Shell − An interface to kernel, hiding complexity of kernel's functions from users. The shell takes commands from the user and executes kernel's functions.
* Utilities − Utility programs that provide the user most of the functionalities of an
* operating systems.

## System calls

## Process Management and IPC

## (i) System calls related to process management – lab observation

## 1.To create a new process – fork () is used. 2. To run a new program = exec () is used. 3. To make the process to wait = wait () is used. 4. To terminate the process – exit () is used. 5. To find the unique process id – getpid () is used. 6. To find the parent process id – getppid () is used.

## (ii) IPC – Lab observation

## Pipe()

## Message queues : msgget(),msgsnd(),msgrcv().

## Shared memory: shmget(),shmmat(),shmctl().

## Shared Memory :

Shared Memory is an efficient means of passing data between programs.

Shared memory is a method of [inter-process communication](https://en.wikipedia.org/wiki/Inter-process_communication) (IPC), i.e. a way of exchanging data between programs running at the same time. One [process](https://en.wikipedia.org/wiki/Process_(computing)) will create an area in [RAM](https://en.wikipedia.org/wiki/Random-access_memory) which other processes can access.

## A  shared memory is a memory  that may be simultaneously accessed by multiple programs with an intent to provide communication among them or avoid redundant copies. Shared memory is an efficient means of passing data between programs. Depending on context, programs may run on a single processor or on multiple separate processors.

## [https://upload.wikimedia.org/wikipedia/commons/thumb/f/f2/Shared_memory.svg/300px-Shared_memory.svg.png](https://en.wikipedia.org/wiki/File:Shared_memory.svg)

An illustration of a shared memory system of three processors.

## One program will create a memory portion which other processes (if permitted) can access.A shared memory can be created using shmget() function. This function returns an identifier to the segment. (shmid : user defined)

## shmat() attaches the shared memory segment identified by shmid to the address space of the calling process. On success shmat() returns the address of the attached shared memory segment;

The command ipcs -l option gives the system limits for each ipc facility.

# ipcs -m -l

------ Shared Memory Limits --------

max number of segments = 4096

max seg size (kbytes) = 67108864

max total shared memory (kbytes) = 17179869184

min seg size (bytes) = 1

## Write a simple program to create a shared memory segment for IPC between parent and child.

## The child can place a value , which can be accessed by the parent process. Wait(0) is used as a synchronization primitive between the parent and the child process.

## System calls :

* **shmget()** is used to obtain create a shared memory segment.

## shmid = shmget((key\_t)integer, number of bytes,IPC\_CREAT |permissions);

* System call **shmat()** accepts a shared memory ID, shm\_id, and attaches the indicated shared memory to the program's address space. If this call is unsuccessful, the return value is -1. Normally, the second parameter is NULL. If the flag is SHM\_RDONLY, this shared memory is attached as a read-only memory; otherwise, it is readable and writable.

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

main()

{

key\_t key; /\* key to be passed to shmget() \*/

int shmflg; /\* shmflg to be passed to shmget() \*/

int shmid; /\* return value from shmget() \*/

int size; /\* size to be passed to shmget() \*/

int \*shm\_ptr; /\* ptr used to get the address of the shared memory

int value;

// creating a shared memory segment with key as 1, size as 10 bytes with permissions

size = 10;

shmflg = IPC\_CREAT|0666;

if ((shmid = shmget (key\_t)1, size, shmflg)) == -1)

{

printf("shmget: shmget failed"); exit(1);

}

// shm\_ptr is a pointer giving the address of the shared memory segment

// shmat function returns the address of the shared memory segment created.

shm\_ptr = shmat(shm\_id,0, 0); /\* attach \*/

pid = fork();

// child process creattion

if (pid==-1) printf(“fork err \n”);

if (pid==0)

{

// child region

\*shmptr=100;

// child is placing the integer 100 in the shared memory

}

else

{

wait(0);

// parent reading the value from the shared memory

value = \*shmptr;

printf(“ value read by the parent is %d\n”, value);

}

shmctl(shmid,IPC\_RMID,0);

// shared memory control function is used to delete the shared memory created.

}

Output :

Value read by the parent is 100

To display all the ipc structures maintained in the system:

$ ipcs

* Displays all the interprocess communication structures available
  1. **System administration:**
* **How to create users, groups, changing permissions to the file, directories.**
* **Scheduling of processes ( crontab, at commands)**
* **Go through 8.a,8.b,8.c ( lab observation)**

Scheduling of Processes

1. at command
2. crontab

// understand the difference between at and crontab

Crontab : In order to carry out a submitted job every day for years together, without needing any prompting from us it is used.

The jobs can be carried out on a regular basis using the crontab command as follows:

step 1 : create a cmdfile { use cat command or vi editor or gedit }

sai@ubuntu:~$ cat > cmdfile

\* \* \* \* \* echo "Aim higher ....." >> contents

step 2: execute the crontab command using cmdfile

sai@ubuntu:~$ crontab cmdfile

// The crontab executes cmdfile

// \* \* \* \* \* - Minute ,Hour, Day of Month, Month of year, day of week

// could see the text getting appended to the file contents every minute....

// can wait for some 2 minutes and open the file named contents : twice the message would have appeared in the file. Afer 1 hour we could see the file for 60 messages .. continues..................

[optional step : To view the commands submitted use -l option with crontab command ]

sai@ubuntu:~$ crontab -l

\* \* \* \* \* echo "Aim higher ....." >> contents ]

To remove the submitted job to the crontab:

sai@ubuntu:~$crontab -r

Note :

using a crontab we can submit only one command file.

To schedule few more jobs , edit the cmdfile and resubmit it using $crontab cmdfile.

***Shell scripts***

1. Executing some basic commands with shell script.

#! /bin/sh ls who pwd cal 2015

1. Interactive shell script. (using shell variables)

#!/bin/sh

echo what is your name

read name

echo Hello $name, happy programming

1. [ using if-then-fi statement]

#!/bin/sh

echo “ Enter the source and target names”

read source,target

if cp $source $target

then

echo “ File copied successfully”

else

echo “failed to copy”

fi

1. Write a shell script to check whether the user has a write permission to file.

If so, append some contents to it.

#! /bin/sh

echo “enter filename”

read filename

if [ -w $filename ]

then

echo “type the content to append. Press ctrl D to stop”

cat >> $filename

else

echo “No permission”

fi

1. Check for the existence of a directory, if so create a file in that.

#!/bin/sh

echo “ enter the directory name”

read dirname

if [ -d $dirname ]

then

cat > xyz

else

echo “dir doesnot exist”

fi

DNS : The domain name system (DNS) maps internet [domain names](http://searchwindevelopment.techtarget.com/definition/domain-name) to the internet protocol ([IP](http://searchunifiedcommunications.techtarget.com/definition/Internet-Protocol)) network addresses they represent and enables websites to use names, rather than difficult-to-remember [IP addresses](http://searchwindevelopment.techtarget.com/definition/IP-address).

Importance of DNS :

<http://dyn.com/blog/dns-why-its-important-how-it-works/>

----------------------------------------------------------------------------------------------------

# Network configuration and information

## 1. Configuration of network interfaces

User friendly Linux distributions come with various graphical tools, allowing for easy setup of the computer in a local network, for connecting it to an Internet Service Provider or for wireless access. These tools can be started up from the command line or from a menu:

* Ubuntu configuration is done selecting System->Administration->Networking.

## 2. Network configuration files

The graphical helper tools edit a specific set of network configuration files, using a couple of basic commands. The exact names of the configuration files and their location in the file system is largely dependent on your Linux distribution and version.

However, a couple of network configuration files are common on all UNIX systems:

### 1. /etc/hosts

The /etc/hosts file always contains the localhost IP address, 127.0.0.1, which is used for interprocess communication. Never remove this line! Sometimes contains addresses of additional hosts, which can be contacted without using an external naming service such as DNS (the Domain Name Server).

A sample hosts file for a small home network:

|  |
| --- |
| # Do not remove the following line, or various programs  # that require network functionality will fail.  127.0.0.1 localhost.localdomain localhost  192.168.52.10 tux.mylan.com tux  192.168.52.11 winxp.mylan.com winxp |

Read more in man hosts.

### 2. /etc/resolv.conf

The /etc/resolv.conf file configures access to a DNS server. This file contains your domain name and the name server(s) to contact:

|  |
| --- |
| search mylan.com  nameserver 193.134.20.4 |

### 

Commands to know :

### 3. /etc/nsswitch.conf

The /etc/nsswitch.conf file defines the order in which to contact different name services. For Internet use, it is important that dns shows up in the "hosts" line:

|  |
| --- |
| [bob@tux ~] grep hosts /etc/nsswitch.conf  hosts: files dns |

This instructs your computer to look up hostnames and IP addresses first in the /etc/hosts file, and to contact the DNS server if a given host does not occur in the local hosts file. Other possible name services to contact are LDAP, NIS and NIS+.

More in man nsswitch.conf.

## 4. Network interface names

On a Linux machine, the device name lo or the local loop is linked with the internal 127.0.0.1 address. The computer will have a hard time making your applications work if this device is not present; it is always there, even on computers which are not networked.

The first ethernet device, eth0 in the case of a standard network interface card, points to your local LAN IP address. Normal client machines only have one network interface card. Routers, connecting networks together, have one network device for each network they serve.

If you use a modem to connect to the Internet, your network device will probably be named ppp0.

There are many more names, for instance for Virtual Private Network interfaces (VPNs), and multiple interfaces can be active simultaneously, so that the output of the ifconfig or ip commands might become quite extensive when no options are used. Even multiple interfaces of the same type can be active. In that case, they are numbered sequentially: the first will get the number 0, the second will get a suffix of 1, the third will get 2, and so on. This is the case on many application servers, on machines which have a failover configuration, on routers, firewalls and many more.

## 5. Checking the host configuration with netstat

Apart from the ip command for displaying the network configuration, there's the common netstat command which has a lot of options and is generally useful on any UNIX system.

Routing information can be displayed with the -nr option to the netstat command:

|  |
| --- |
| bob:~> netstat -nr  Kernel IP routing table  Destination Gateway Genmask Flags MSS Window irtt Iface  192.168.42.0 0.0.0.0 255.255.255.0 U 40 0 0 eth0  127.0.0.0 0.0.0.0 255.0.0.0 U 40 0 0 lo  0.0.0.0 192.168.42.1 0.0.0.0 UG 40 0 0 eth0 |

This is a typical client machine in an IP network. It only has one network device, eth0. The lo interface is the local loop.

|  |  |
| --- | --- |
|  |  |

## Other hosts

An impressive amount of tools is focused on network management and remote administration of Linux machines.

### 1. The host command

To display information on hosts or domains, use the host command:

|  |
| --- |
| [emmy@pc10 emmy]$ host www.eunet.be  www.eunet.be. has address 193.74.208.177  [emmy@pc10 emmy]$ host -t any eunet.be  eunet.be. SOA dns.eunet.be. hostmaster.Belgium.EU.net.  2002021300 28800 7200 604800 86400  eunet.be. mail is handled by 50 pophost.eunet.be.  eunet.be. name server ns.EU.net.  eunet.be. name server dns.eunet.be. |

Similar information can be displayed using the dig command, which gives additional information about how records are stored in the name server.

### 2. The ping command

To check if a host is alive, use ping. If your system is configured to send more than one packet, interrupt ping with the Ctrl+C key combination:

|  |
| --- |
| [emmy@pc10 emmy]$ ping a.host.be  PING a.host.be (1.2.8.3) from 80.20.84.26: 56(84) bytes of data.  64 bytes from a.host.be(1.2.8.3):icmp\_seq=0 ttl=244 time=99.977msec  --- a.host.be ping statistics ---  1 packets transmitted, 1 packets received, 0% packet loss  round-trip min/avg/max/mdev = 99.977/99.977/99.977/0.000 ms |

### 3. The traceroute command

To check the route that packets follow to a network host, use the traceroute command:

|  |
| --- |
| [emmy@pc10 emmy]$ /usr/sbin/traceroute www.eunet.be  traceroute to www.eunet.be(193.74.208.177),30 hops max,38b packets  1 blob (10.0.0.1)  0.297ms 0.257ms 0.174ms  2 adsl-65.myprovider.be (217.136.111.1)  12.120ms 13.058ms 13.009ms  3 194.78.255.177 (194.78.255.177)  13.845ms 14.308ms 12.756ms  4 gigabitethernet2-2.intl2.gam.brussels.skynet.be (195.238.2.226)  13.123ms 13.164ms 12.527ms  5 pecbru2.car.belbone.be (194.78.255.118)  16.336ms 13.889ms 13.028ms  6 ser-2-1-110-ias-be-vil-ar01.kpnbelgium.be (194.119.224.9)  14.602ms 15.546ms 15.959ms  7 unknown-195-207-939.eunet.be (195.207.93.49)  16.514ms 17.661ms 18.889ms  8 S0-1-0.Leuven.Belgium.EU.net (195.207.129.1)  22.714ms 19.193ms 18.432ms  9 dukat.Belgium.EU.net (193.74.208.178) 22.758ms \* 25.263ms |

------------------------------------------------------------------

**Virtualization:**

**Basics, hypervisor, guest operating system,**

**Chapter 16 – 16.1, 16.2, 16.3 and 16.7**

**-----------------------------------------------------------------------------------------------------------**