./

Learning Report

C AND LINUX OS PROGRAMMING



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be approved By** | **Remarks/Revision Details** |
| 1 |  | Name/PS No | Name/PS No | Module Owner Name | Comments |
| 2 | 28/02/21 | 99003769 |  |  |  |

**Document History**

[ACTIVITY 2: C and LINUX OS Programming](#_Toc53129064) **[.](#_Toc53129064)**

**[INTRODUCTION](#_Toc53129065)** [4](#_Toc53129065)**[.](#_Toc53129065)**

**componnent of LINUX system** [5](#_Toc53129066)**[.](#_Toc53129066)**

**[Functio](#_Toc53129067)**[n](#_Toc53129067) **[of OS](#_Toc53129067)** [7](#_Toc53129067)

**[LINUX OS ARCHITECTURE](#_Toc53129068)** [8](#_Toc53129068)-9

**[FEATURE OF LINUX OS](#_Toc53129069)****[10.](#_Toc53129069)**

**INTERRUPT** [10](#_Toc53129070)**[.](#_Toc53129070)**

**[SYSTEM CALL](#_Toc53129071)** [11](#_Toc53129071)**[.](#_Toc53129071)**

**[TYPES OF SYSTEM CALL](#_Toc53129072)****[12.](#_Toc53129072)**

**[PROCESS MANAGEMENT](#_Toc53129073)****[13-14](#_Toc53129073)**

**[PROCESS LIFE CYCLE...........................................................................................................................................15-16](#_Toc53129073)**

**[PROCESS TALE AND PROCESS CONTROL BLOCK................................................................................................. 15-16](#_Toc53129073)**

**[SIGNAL..................................................................................................................](#_Toc53129073)**....................................................17

**THREADS................................................................................................................................................................18**

**PROCESS VS THEREAD............................................................................................................................................19**

**PIPES......................................................................................................................................................................20**

**SHARED MEMORY..................................................................................................................................................**20

**REFERENCE--------------------------------------------------------------------------------------------------------------------------------------------21**

**INTRODUCTION TO C PROGRAMMING LANGUAGE :**

C is a general-purpose programming language, and is used for writing programs in many different domains, such as operating systems, numerical computing, graphical applications, etc. It is a small language, with just 32 keywords. It provides “high-level” structured programming constructs such as statement grouping, decision making, and looping, as well as “low level” capabilities such as the ability to manipulate bytes and addresses.

**INTRODUCTION TO LINUX OS:**

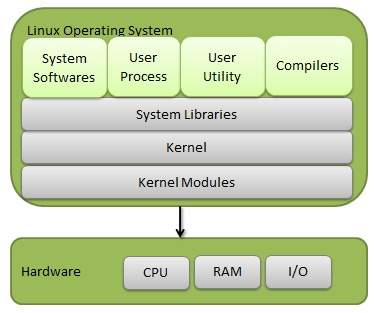
LINUX is an operating system or a kernel distributed under open-source license. Its functionality list is quite like UNIX. The kernel is a program at the heart of the Linux operating system that takes care of fundamental stuff, like letting hardware communicate with software.

**LINUX HISTORY:**

Kernel of Unix was developed by Linus Torvalds in 1991and its official name is ‘GNU/Linux.

## ***Components of Linux System***

1. **Kernel**
2. **System Library**
3. **System Utility**



* **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 task.

Types of Operating system:

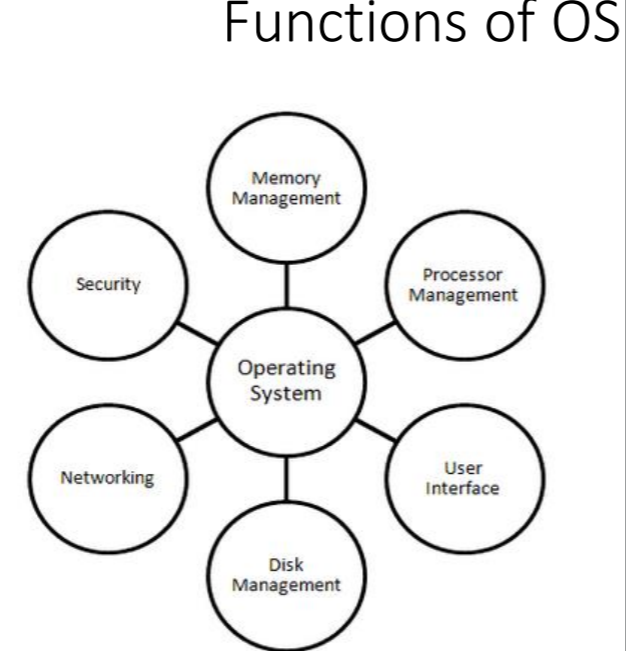
• Batch OS

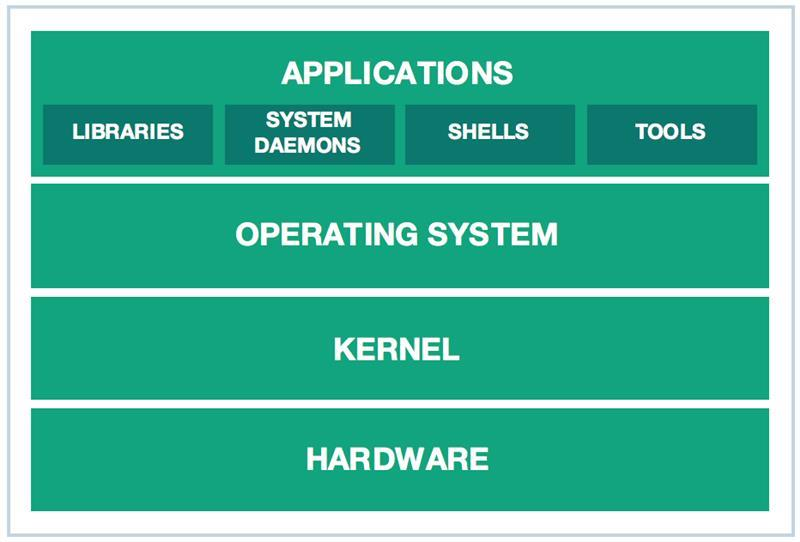
• Time-sharing OS (Linux)

• Distributed OS

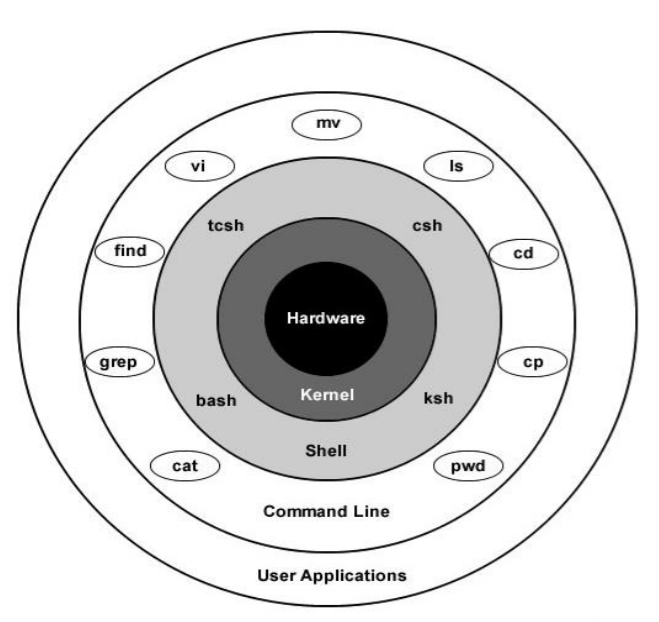
• Network OS

• Real-time OS





**Linux OS Architecture**



**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.

There are 3 types of Kernel:

1. Micro
2. Monolithic
3. Modular

• Compressed Kernel is stored at /boot/vmlinu\*

• Dynamic modules of kernel /lib/modules

**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.Types of shells – bash, ksh, csh, tcsh, zsh etc.

**Utilities** − Utility programs that provide the user most of the functionalities of an operating systems.

Features of Linux OS:

1. Open source

2. Portable

3. Multi-user

4. Multi programming

5. Hierarchical File system

6. Shell

7. Security

Interrupts:

An interrupt is an event that alters the normal execution of the program and can be generated by hardware devices or even by the CPU itself.

Interrupts can be grouped into two categories based on the source of the interrupt:

1. ****synchronous****, generated by executing an instruction.
2. ****asynchronous****, generated by an external event.

* ****Maskable****
* ****Non-maskablse****

Interrupts must be serviced with utmost priority

• Types

• Hardware Interrupts

• Software Interrupts

**System calls:**

System call is a programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. A system call is a way for programs to interact with OS. A computer program makes a system call when it makes a request to the operating system’s kernel. System call provide the services of the operating system to the user programs via Application Program Interface(API).

SERVICE PROVIDED BY SYSTERM CALL:**:**

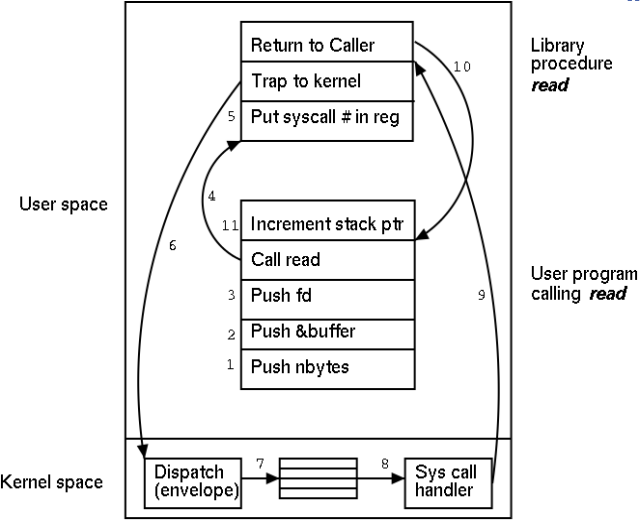
1. Process creation and management
2. Main memory management
3. File Access, Directory and File system management
4. Device handling(I/O)
5. Protection
6. Networking, ETC.

**TYPES OF SYSTEM CALL:.**

1. Process Control.
2. File management.
3. Device management.
4. Information maintenance.
5. Communication.

**Some Example of Window and Unix system call:**

|  |  |
| --- | --- |
| WINDOW | UNIX |
| 1. CreateProcess() 2. ExitProcess() 3. ReadFile() 4. WriteFile() 5. CloseHandle() | 1. fork() 2. exit() 3. wait() 4. open() 5. read() |



**PROCESS MANAGEMENT:**

**Process is a program under Execution.**

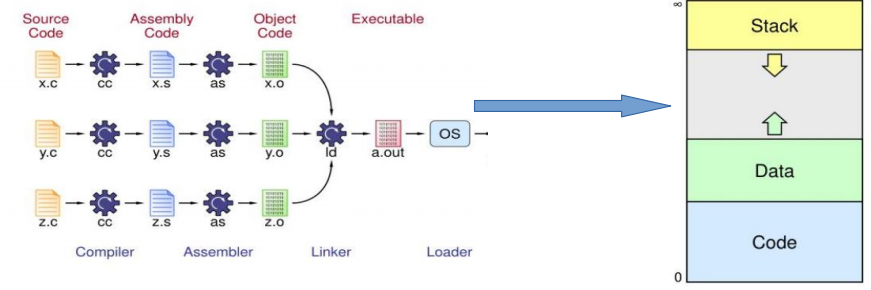
Every process has its own independent stack.

Kernel maintain process list table in the form of doubly linked list.

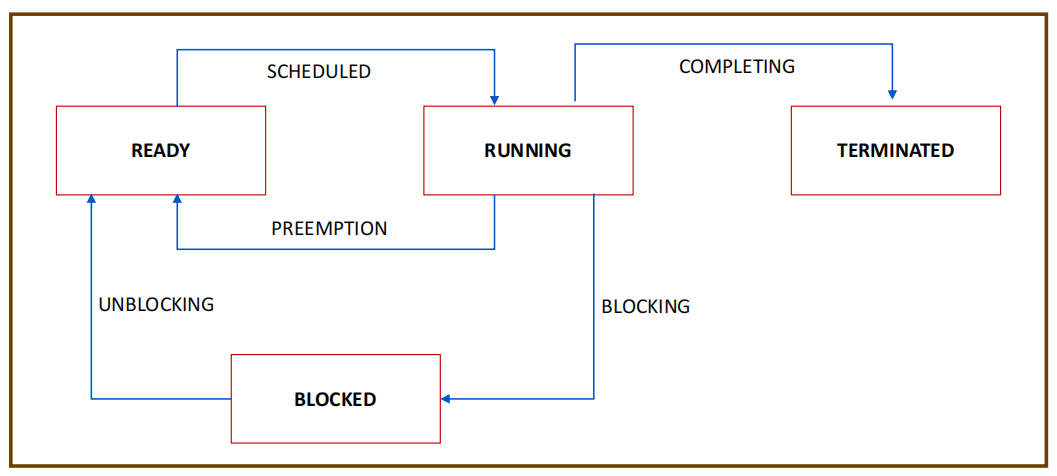
Each process has a unique id.

There are basically 2 types of processes:

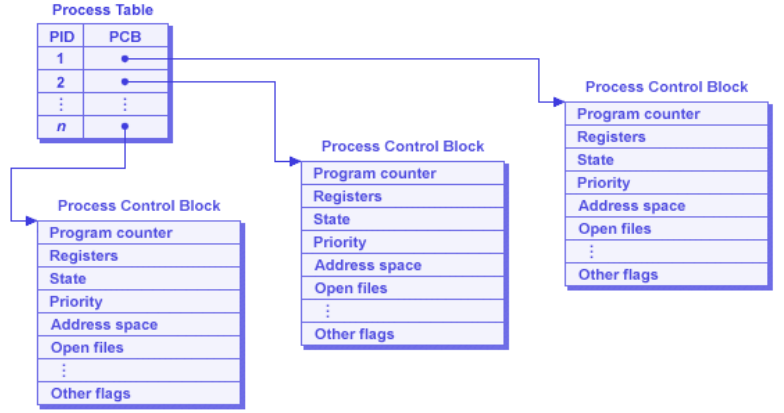
1. **Foreground processes: This process is also known as interactive process.**
2. **Background processes: This process is known as non interactive process.**



**Process life cycle:**



**Process Tale and Process Control block:**



• To start a process in Background use **&** symbol in command

• cat file.txt &

• jobs

• Lis the running processes

• ps , ps –f

• Stop a process

• kill -9, pkill

• Parent and Child

• Each Process(PID) has a parent (PPID)

• Zombie and Orphan Process

• Orphan process is one whose parent is killed/terminated before itself.

• Processes which completed the execution but still have entry in process table.

• Daemon Processes

• Processes that run in background

Context Switching:

Itis a switching of the central processing unit from one process or thread to another.

**SIGNAL:**

Signals operate at process level.

Used for communication of abnormal termination, illegal memory access & events that go wrong.

Signals communicate between applications at user level.

Signals are considered as software interrupts, but it has no interrupt vector tale.

SOME COMMAND OF SIGNAL:

• kill –l, will list all the signals

• kill –SIGxxxx <pid>

• kill -<signo> <pid>

• kill <pid>

• kill -9 <pid>  terminate is SURE KILL

• kill (pid, signal number)  system call

• pkill, killall, pgrep  process kill

**Non Maskable Signa**l:

SIGKILL, SIGSTOP

**THREADS:**

thread is basically a path of execution within a process. A process can contain multiple threads.

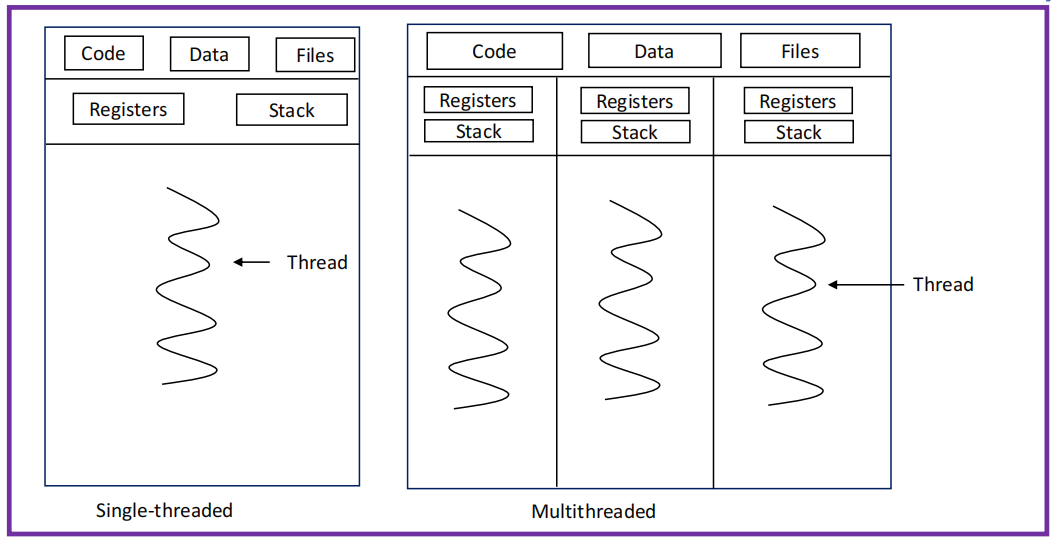
Scheduled threads interchangeably use CPU based on time sharing

Every process is run initially as a single thread, then multiple threads spawn.

Threads are faster than fork.

Common resources during execution run independently

**Basics of Threads:**



**Process vs Thread:**  
The threads within the same process run in the shared memory space, where as processes run in separate memory spaces.  
Threads are not independent of one another like processes are, and as the result threads share there code section , data section, and OS resources (like open files and signals)with other thread. But, like process, a thread has its own program counter (PC), register set, and stack space.

**TYPES OF THREAD:**

1. User Thread.
2. Kernel Thread.

INTER PROCESS COMMUNICATION:

IPC shows the information on the inter-process communication facilities for which the calling process has read access. By default shows information about all three resources of shared memory segments, message queues, and semaphores.

Process that writes/updates data is **PRODUCER .**

and process that reads is **CONSUMER.**

**PIPES:**

Pipe is a connection between two related processes.

It send the output of one command to another command for further processing.

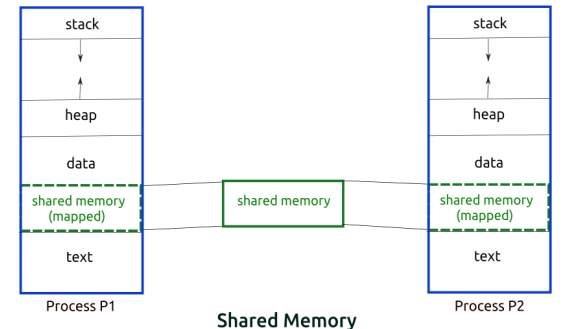
Pipe is one way communication only.

If the process tries to read before anything is written to the pipe, the process is suspended until something is written in it.

for two way communication using pipe two pipe should e used.

**SHARED MEMORY:**

Memory Segment is created by kernel and mapped to data segment of the address space of the requesting process it can be used like a global variable in address space.



**REFERENCE:**

1. **<https://www.geeksforgeeks.org/>**
2. **<https://en.wikipedia.org/>**
3. **<https://stackoverflow.com/>**