UNIT - 1

1. different Linux flavors available

   Debia
   Kali Linux
   Cent OS
   Ubuntu
   Linux Mint
   Fedora
   Arch Linux

2. File Attributes.

   File type
   Access permission
   Hard link count
   Mode
   User id
   Group id
   Last access time
   Last modified time
   Last changed time
   Size

3.File types in LINUX.

   Regular
   FIFO
   DIR
   Block device
   Character device
   Socket file
   Link file

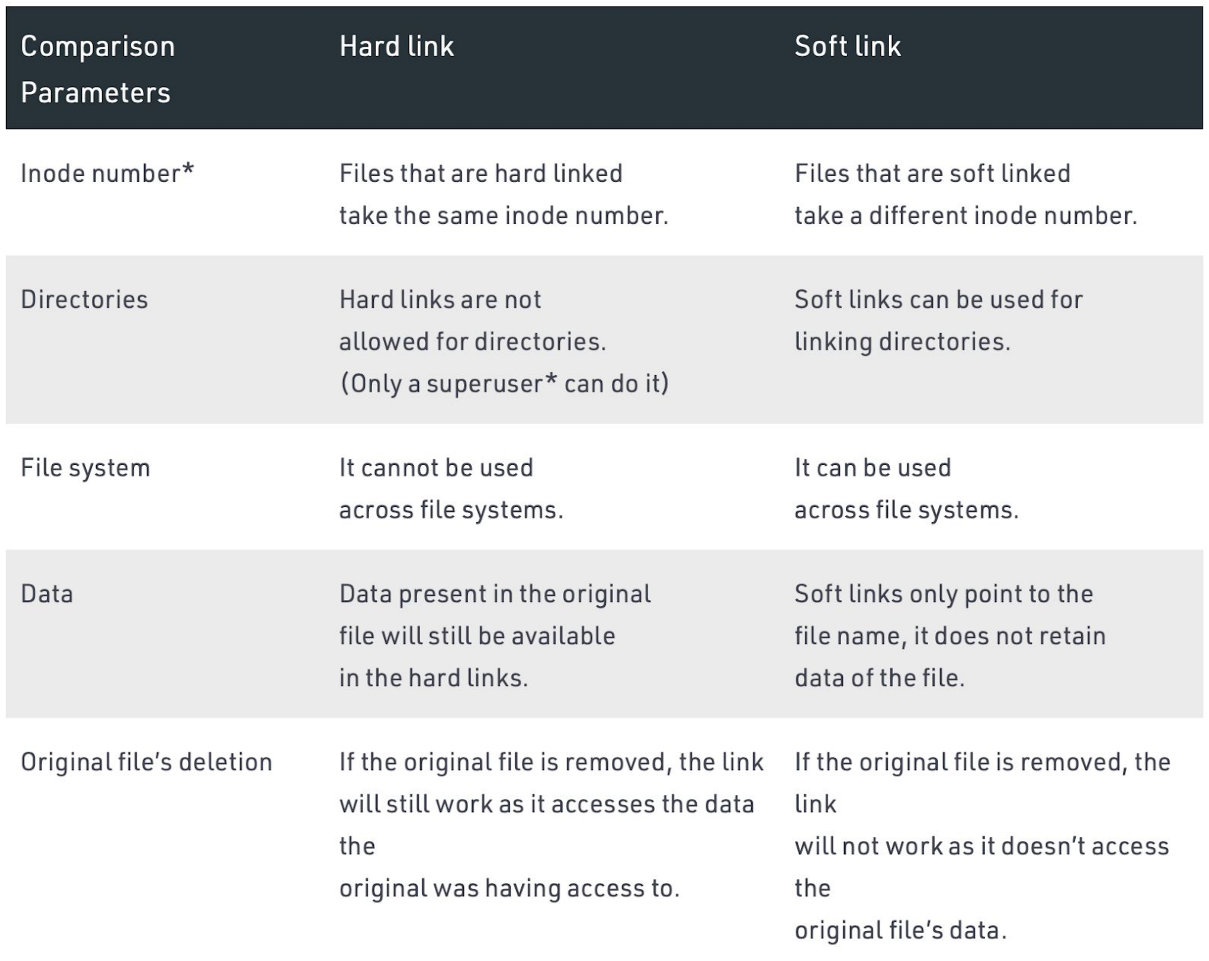
4.about the standard I/O and formatted I/O in C.
Taking input from
Formatted -input from console

5.Explain file descriptors in Linux.
Sdtin - 0
Stdout - 1
Stderr - 2
3 nudi kali
A file descriptor is a number that uniquely identifies an open file in a computer's operating system. It describes a data resource, and how that resource may be accessed.

6.What is meant by low level file access?
In lower level file handling we will study the basic opening, closing, reading, and writing a file through a file descriptor which acts as a pointer or cursor to a file.It is called lower level file handling because it will execute or do operations on kernel level

7. What is the difference between system call and library function call
A system call is implemented in kernel space while a library call is implemented in the user space.
The main difference between System Call and Library Call is that System call is a request to the kernel to access a resource while library call is a request to use a function defined in a programming library.
Library calls internally use system calls.

8. Differentiate hard link and symbolic links to a file.



Comparison Parameters	Hard link	Soft link
Inode number*	Files that are hard linked take the same inode number.	Files that are soft linked take a different inode number.
Directories	Hard links are not allowed for directories. (Only a superuser* can do it)	Soft links can be used for linking directories.
File system	It cannot be used across file systems.	It can be used across file systems.
Data	Data present in the original file will still be available in the hard links.	Soft links only point to the file name, it does not retain data of the file.
Original file's deletion	If the original file is removed, the link will still work as it accesses the data the original was having access to.	If the original file is removed, the link will not work as it doesn't access the original file's data.

9.What are the contents of an i-node

- User ID of file
- Group ID of file
- Device ID
- File size
- Date of creation
- Permission
- Owner of the file
- File protection flag
- Link counter to determine number of hard links

10)Write the prototypes and usage of stat system calls.
The stat() system call actually returns file attributes. The file attributes of an inode are basically returned by Stat() function. An inode contains the metadata of the file. An inode contains: the type of the file, the size of the file, when the file was accessed (modified, deleted) that is time stamps, and the path of the file, the user ID and the group ID, links of the file, and physical address of file content.
The stat () system call actually returns file attributes.
int stat(const char *path, struct stat *buf)

Unit - 2

1.Process and process attributes

AProcessisaprogram underexecutioninaUNIXorPOSIXsystem.

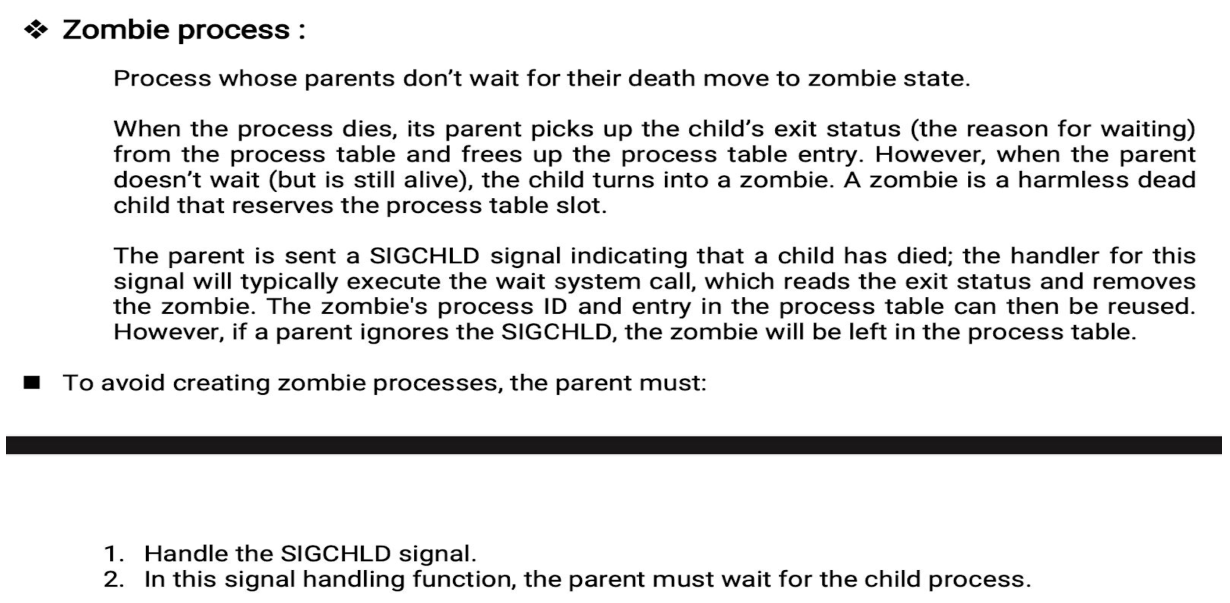
Process ID The identifier of the process.
Process Parent ID The identifier for the parent process
Process State The state of the process (Sleeping, Disk, Running, Zombie, Trace, Dead,)

A real user identification number

A real group identification number

Session ID The session ID.

2.Zombie process and avoiding



❖ **Zombie process :**

Process whose parents don't wait for their death move to zombie state.

When the process dies, its parent picks up the child's exit status (the reason for waiting) from the process table and frees up the process table entry. However, when the parent doesn't wait (but is still alive), the child turns into a zombie. A zombie is a harmless dead child that reserves the process table slot.

The parent is sent a SIGCHLD signal indicating that a child has died; the handler for this signal will typically execute the wait system call, which reads the exit status and removes the zombie. The zombie's process ID and entry in the process table can then be reused. However, if a parent ignores the SIGCHLD, the zombie will be left in the process table.

- To avoid creating zombie processes, the parent must:

1. Handle the SIGCHLD signal.
2. In this signal handling function, the parent must wait for the child process.

3.Orphan process

A process whose parent has exited.

• Orphaned processes are inherited by init

• Its slot in the process table is immediately released when an orphan terminates

4. Sigaction ()

The sigaction() system call is used to change the action taken by a process on receipt of a specific signal.

```
int sigaction(int signum,
              const struct sigaction *act,
              struct  sigaction *oldact );
```

5.how is process terminated when abort is called

Causes an abnormal program termination and returns control to the host environment. The abort() function flushes all buffers and closes all open files.

6. signal that cannot be caught by a system call

The signalsSIGKILL and SIGSTOP cannot be caught or ignored.

Unit - 3

1. `FIFO`

A **FIFO file** is a special kind of **file** on the local storage which allows two or more processes to communicate with each other by reading/writing to/from this **file**
mkfifo(“demo.txt”);

UNIT-4

1) Explain about semaphores.

Semaphore in Linux plays an important role in a multiprocessing system. ... It is a variable or abstract data type used to control access to a common resource by multiple processes in a concurrent system such as a multiprogramming operating system.

-It can be used to implement critical-section problems; allocation of resources

2) Explain about shared memory.

-an area of memory accessible by multiple processes.

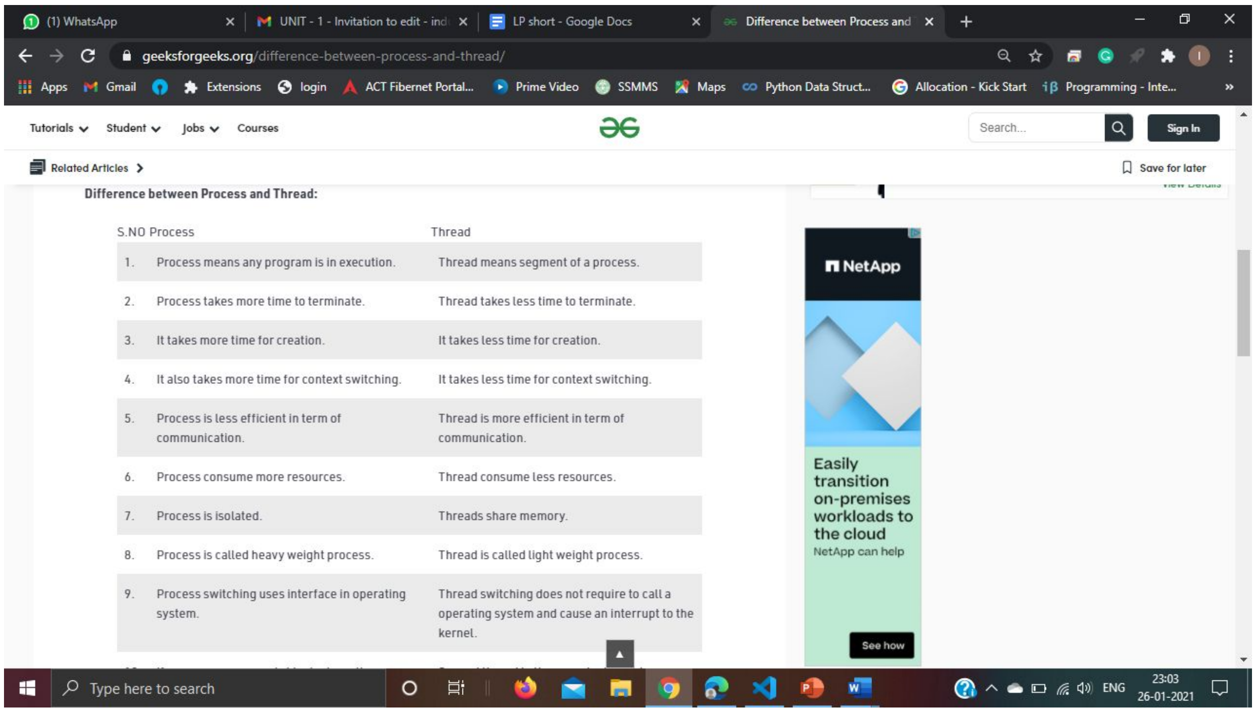
3) Explain with example semaphores.

Let us assume, multiple processes are using the same region of code and if all want to access parallelly then the outcome is overlapped. Say, for example, multiple users are using one printer only (common/critical section), say 3 users, given 3 jobs at same time, if all the jobs start parallelly, then one user output is overlapped with another. So, we need to protect that using semaphores i.e., locking the critical section when one process is running and unlocking when it is done. This would be repeated for each user/process so that one job is not overlapped with another job.

4)Explain the concept of shared memory with example.

Shared memory is a technology that enables computer programs to simultaneously share **memory** resources for higher performance and fewer redundant data copies.

5)


geeksforgeeks.org/difference-between-process-and-thread/
Easily transition on-premises workloads to the cloud
NetApp can help
See how

Difference between Process and Thread:

S.NO	Process	Thread
1.	Process means any program is in execution.	Thread means segment of a process.
2.	Process takes more time to terminate.	Thread takes less time to terminate.
3.	It takes more time for creation.	It takes less time for creation.
4.	It also takes more time for context switching.	It takes less time for context switching.
5.	Process is less efficient in term of communication.	Thread is more efficient in term of communication.
6.	Process consume more resources.	Thread consume less resources.
7.	Process is isolated.	Threads share memory.
8.	Process is called heavy weight process.	Thread is called light weight process.
9.	Process switching uses interface in operating system.	Thread switching does not require to call a operating system and cause an interrupt to the kernel.

6)Thread attributes

pthread_create

Pthread_exit

7)10. Explain Mutex in Linux.

A **Mutex** is a lock that we set before using a shared resource and release after using it. When the lock is set, no other thread can access the locked region of code.

UNIT-5

1)What are I/O Models in I/O Multiplexing?

- blocking I/O
- nonblocking I/O
- I/O multiplexing (`select` and `poll`)
- signal driven I/O (`SIGIO`)
- asynchronous I/O (the POSIX `aio_`functions)

2) Explain Select Function.

The **select function** is used to determine the status of one or more sockets

3. Explain poll function.

;

`poll` provides functionality that is similar to `select`, but `poll` provides additional information when dealing with STREAMS devices.

```
#include <poll.h>

int poll (struct pollfd *fdarray, unsigned long nfds, int timeout);

/* Returns: count of ready descriptors, 0 on timeout, -1 on error */
```

Arguments:

The first argument (*fdarray*) is a pointer to the first element of an array of structures. Each element is a `pollfd` structure that specifies the conditions to be tested for a given descriptor, `fd`.

4)Explain getsockopt and setsockopt functions.

The **setsockopt function** sets the current value for a socket option associated with a socket of any type, in any state.

The **getsockopt**() function manipulates options associated with a socket. The **getsockopt**() function shall retrieve the value for the option specified by the option_name argument for the socket specified by the socket argument.

5)Explain socket option and TCP socket options.

The **socket** mechanism provides two **socket**-**option** interfaces for us to control the behavior of **sockets**. One interface is used to set an **option**, and another interface allows us to query the state of an **option**

`6)usage of select system call?`

select is a system **call** and application programming interface (API) in Unix-like and POSIX-compliant operating systems for examining the status of file descriptors of open input/output channels.