Operating System

Week 2 - Long Descriptive Questions

1.What is a system call ? Explain any 3 system calls with appropriate example program.

Ans. A system call is a way for a user program to interface with the operating system. The program requests several services, and the OS responds by invoking a series of system calls to satisfy the request. A system call can be written in assembly language or a high-level language like **C** or **Pascal**. System calls are predefined functions that the operating system may directly invoke if a high-level language is used.

A system call is a method for a computer program to request a service from the kernel of the operating system on which it is running. A system call is a method of interacting with the operating system via programs.

Type of System Calls:

Process Control: Process control is the system call that is used to direct the processes. Some process control examples include creating, load, abort, end, execute, process, terminate the process, etc.

File Management : File management is a system call that is used to handle the files. Some file management examples include creating files, delete files, open, close, read, write, etc.

Device Management : Device management is a system call that is used to deal with devices. Some examples of device management include read, device, write, get device attributes, release device, etc.

Information Maintenance : Information maintenance is a system call that is used to maintain information. There are some examples of information maintenance, including getting system data, set time or date, get time or date, set system data, etc.

Communication: Communication is a system call that is used for communication. There are some examples of communication, including create, delete communication connections, send, receive messages, etc.

2.Write a simple program for Process creation with sample output.

Ans.

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main() {
```

```
    fork();
    printf("Called fork() system call\n");
    return 0;
}
```

Output :

Called fork() system call
Called fork() system call

3.Differentiate between wait() and exit() system calls.

Ans.

wait()

In some systems, a process may wait for another process to complete its execution. This happens when a parent process creates a child process and the execution of the parent process is suspended until the child process executes. The suspending of the parent process occurs with a wait() system call. When the child process completes execution, the control is returned back to the parent process. In some systems, a process may have to wait for another process to complete its execution before proceeding. When a parent process makes a child process, the parent process execution is suspended until the child process is finished. The **wait()** system call is used to suspend the parent process. Once the child process has completed its execution, control is returned to the parent process.

exit()

The exit() system call is used by a program to terminate its execution. In a multithreaded environment, this means that the thread execution is complete. The operating system reclaims resources that were used by the process after the exit() system call. The **exit()** is a system call that is used to end program execution. This call indicates that the thread execution is complete, which is especially useful in multi-threaded environments. The operating system reclaims resources spent by the process following the use of the **exit()** system function.

4.What are the advantages of a multiprocessor system?

Explain about time sharing system.

Ans.

Multiprocessing operating system or the parallel system support the use of more than one processor in close communication.
The advantages of the multiprocessing system are:

- Increased Throughput − By increasing the number of processors, more work can be completed in a unit time.
- Cost Saving − Parallel system shares the memory, buses, peripherals etc. Multiprocessor system thus saves money as compared to multiple single systems. Also, if a number of programs are to operate on the same data, it is cheaper to store that data on one single disk and shared by all processors instead of using many copies of the same data.
- Increased Reliability − In this system, as the workload is distributed among several processors which results in increased reliability. If one processor fails then its failure may slightly slow down the speed of the system but system will work smoothly.
- Less electricity usage – In a single processor system, there is more load as many processes have to be executed at a time. But in multiprocessor system execution of multiple processes in done in a few times. That means multiprocessor CPUs consume low electricity than a single processor.

Time shared operating system : Time sharing is a logical extension of multiprogramming. The CPU performs many tasks by switches are so frequent that the user can interact with each program while it is running. A time shared operating system allows multiple users to share computers simultaneously. Each action or order at a time the shared system becomes smaller, so only a little CPU time is required for each user. As the system rapidly switches from one user to another, each user is given the impression that the entire computer system is dedicated to its use, although it is being shared among multiple users. A time shared operating system uses CPU scheduling and multi-programming to provide each user with a small portion of a shared computer at once. Each user has at least one separate program in memory.