
# Digital Assignment – 2

**Name : G. Nithish**

**Reg no: 19BCS0012**

**Course: Operating System**

**1. What is the purpose of system calls?**

**System Calls are a way for user programs (running in user mode) to request some service from Operating System. In other words, system calls allow the user programs to ask OS to do some stuff on behalf of the user program. For example, read and write of file which requires the I/O from/to the storage device.**

**For example, read and write of file which requires the I/O from/to the storage device. Such system call operations are exposed to the end user in the form of simple library calls/APIs - read(), write(), open() in like etc . Also Operating System is there as an interface between user**

**programs and the hardware. It interacts with hardware on behalf of operations requested by the user programs.**

**2.what is the purpose of system programs?**

**System programs can be thought of as bundles of useful system calls. They provide basic functionality to users so that users do not need to write their own programs to solve common problems.**

**3.what are the five major activities of an operating system with regard To process management?**

**the operating system on a computer manages the applications running on it. Each running program on a computer has at least one process associated with**

**it. A process therefore represents some or all of a program while it is running. Most of the time when you use a computer, more than one application will be running. The operating system has to manage the resources on the computer to make this possible. This process management operation requires a range of activities, all administered by the operating system.**

**THERE ARE FIVE TYPES:**

**1. Creating and deleting**

**2. Suspending and resuming**

**3. Synchronizing**

**4. Communicating**

**5. Deadlock handling**

**Creating and Deleting**

**Some of the processes on your computer may run for short periods of time, with others running continuously over longer periods. For example, some background processes will begin when the computer first boots up, such as those associated with input and output. Other processes will start when you launch applications. The processes created when a software application is**

**launched will typically then stop when you exit or quit the application. The operating system manages the creation and deletion of all running processes.**

**Suspending and Resuming**

**Although the processes on a computer may appear to be running continuously, they will often enter paused states for short periods of time. If a process is not currently executing -- for example, if it is waiting for an input or output operation to complete -- it may be suspended. The operating system manages the suspension and resumption of such processes when the required resources become available.**

**Synchronizing**

**A computer has a finite range of processing resources that must be shared between all running processes. The operating system creates the impression that several processes are being executed at the same time, but the available resources are in fact being switched between them so quickly that they appear to be running simultaneously. The operating system carries out process synchronization to keep any running programs functional and available for user interaction.**

**Communicating**

**In order to keep the running processes synchronized and receiving the necessary resources, the operating system must be able to communicate with the processes. For example, the operating system must be able to determine when a process is suspended or ready for resource allocation. If processes need access to the same system resource, this communication activity becomes even more vital.**

**Deadlock Handling**

**When a number of running processes are all in a paused state, each one waiting for resources currently being used by another running process, deadlock can occur. This could cause all programs to halt indefinitely if the operating system did not intervene. The operating system can take steps both to avoid and end deadlock should it occur. Operating systems use various strategies to handle deadlock.**

**4. Which of these are the major activities of an operating system with regard to memory management?**

**The three major activities of an operating system in connection with regard to memory management are:**

- **Keep track of which parts of memory are currently being used and by whom.**
- **Decide which processes are to be loaded into memory when memory space becomes available.**
- **Allocate and de allocate memory space as needed.**

**5. Which of these are the major activities of an operating system with regard to secondary-storage management?**

**The three major activities of an operating system in regard to secondary storage management are: Managing the free space available on the secondary-storage device. Allocation of storage space when new files have to be written. Scheduling the requests for memory access.**

**6. Explain following unix system calls**

| | |
|---|---|
| **Process control** | **Fork()**<br>**Exit()**<br>**WAIT()** |
| **FILE MANIPULATION** | **OPEN()**<br>**READ()**<br>**WRITE()** |

## Process control:

### Fork ()

One use of the fork function is to create a new process (the child) that then causes another program to be executed by calling one of the exec functions. When a process calls one of the exec functions, that process is completely replaced by the new program which starts executing at its main function.

Exit ()

exit () void exit(int status) -- terminates the process which calls this function and returns the exit status value. Both UNIX and C (forked) programs can read the status value. By convention, a status of 0 means normal termination any other value indicates an error or unusual occurrence.

### Wait ()

Wait () int wait (int *status _location) -- will force a parent process to wait for a child process to stop or terminate. Wait () return the pid of the child or -1 for an

Error. The exit status of the child is returned to status _location.

File manipulation:

## Open ()

The Unix Shell. If you have no experience with the Unix command shell, it will be best to work through this primer. The last section summarizes the basic file manipulation commands. In OS X, you can open file using the command "open".

## Read ()

File Management - In this chapter, we will discuss in detail about file ... To list the files and directories stored in the current directory, use the following command – $ls .... The Unix program will read the default input from STDIN.

## Write ()

Use of cat The cat command is one of the most basic commands. It is used to create, append, display, and concatenate files. We can create a file with cat using the '>' to redirect standard input ( stdin ) to a file. Using the '>' operator truncates the contents of the output file specified.