Assignment 1

Operating Systems

1. What are the main functions of an operating system?

Ans-

Hardware Management

- One of the most important functions of an operating system is the management of all the computer's internal and external hardware.
- The operating system controls all connected devices, telling them how to operate and interact.
- The results of said operations and interactions are the basic performance of the computer.
- Some examples of the hardware that the operating system controls include the hard disk, optical drives, video cards and external port controllers, such as USB and Firewire.

Program Control

- Operating systems also run programs, enabling them to operate and function as designed.
- Programs have to be designed to run with a particular operating system in order to function, as the operating system needs to display any interfaces and facilitate communications among the programs and your computer's connected hardware. Without the operating system to run it, programs can't perform their tasks.

System Resource Management

- The operating system manages a computer's resources, allocating them as needed.
- Which tasks the CPU processes in what order, which functions or programs are assigned processing power or memory first, and many other important resource allocation tasks are controlled by the operating system

Network Communication

- The operating system also facilitates network communication by enabling connected network devices to communicate with the computer and with any programs that require them.
- Network cards, such as Ethernet ports and wireless LAN cards, have the tools to connect with networks but need the operating system to direct their communication correctly so information can be exchanged.
- The operating system also interprets the information that comes in and determines which installed program can best display it for you.

2. **What does the CPU do when it has no program to execute?**
Ans-

The CPU is in a halt state. That is, it stops fetching instructions and waits for an interrupt. In normal operation, CPUs continually fetch and execute new instructions, usually from sequential memory locations, with the occasional jump to a new location. Interrupts cause the CPU to save its state, jump to a different location and start doing something completely different. In a halt state, the CPU responds to interrupts but doesn't do anything else.

In a typical desktop or laptop system, the operating system will detect there are no processes needing to run, schedule an interrupt to occur a few milliseconds in the future and then run a special HALT instruction, at which point the CPU stops fetching new instructions. In older CPUs, there was no HALT instruction and they would just execute a busy loop, but this doesn't happen now as its very power inefficient

3. **Difference between Multi-programming, Multi-tasking and Multi-processing?.**
Ans-

Multiprogramming

Multiprogramming is also the ability of an operating system to execute more than one program on a single processor machine. More than one task/program/job/process can reside into the main memory at one point of time. A computer running excel and firefox browser simultaneously is an example of multiprogramming.

Multitasking

Multitasking is the ability of an operating system to execute more than one task simultaneously on a single processor machine. Though we say so but in reality no two tasks on a single processor machine can be executed at the same time. Actually CPU switches from one task to the next task so quickly that appears as if all the tasks are executing at the same time. More than one task/program/job/process can reside into the same CPU at one point of time

Multiprocessing

Multiprocessing is the ability of an operating system to execute more than one process simultaneously on a multi processor machine. In this, a computer uses more than one CPU at a time.

Q4. Intmain()

```
{
    inti, int j;
     scanf("%d", &i);
     for(j=0; j<i; j++)
     {
     sum= j+i;
     }
     printf("%d", sum);
     exit(0);
  }
```

In the above problem, differentiate each and every line of code as CPU execution or I/O execution.

Ans

In the following program

```
intmain()                // CPU execution
        {
        inti, intj;                  //CPU execution
scanf("%d", &i);         //  I/O execution
        for(j=0; j<i; j++)
          {                                  CPU execution
        sum= j+i;
        }
printf("%d", sum);      // I/O execution
        exit(0); }                  // CPU execution
```

5. Difference between a program and a process?

A **program** is a set of instructions that are to perform a designated task, where as the process is an operation which takes the given instructions and perform the manipulations as per the code, called 'execution of instructions'. A process is entirely dependent of a 'program'.

A **process** is a module that executes modules concurrently. They are separate loadable modules. Where as the program perform the tasks directly relating to an operation of a user like word processing, executing presentation software etc

Q6. Categorize the following state: NEW, READY, RUN, BLOCK, TERMINATE, SUSPEND READY, SUSPEND WAIT as main memory or secondary memory.
Ans.

NEW: secondary memory
READY: main memory
RUN: main memory
BLOCK: main memory
TERMINATE: secondary memory
SUSPEND READY: secondary memory
SUSPEND WAIT : secondary memory

Q7. Define the term context switch with a appropriate example.

Ans.

A context switch is a procedure that a computer's CPU (central processing unit) follows to change from one task (or process) to another while ensuring that the tasks do not conflict. Effective context switching is critical if a computer is to provide user-friendly multitasking.In a CPU, the term "context" refers to the data in the registers and program counter at a specific moment in time. A register holds the current CPU instruction. A program counter, also known as an instruction address register, is a small amount of fast memory that holds the address of the instruction to be executed immediately after the current one.A context switch can be performed entirely in hardware (physical media). Older CPUs, such as those in the x86 series, do it that way. However, most modern CPUs perform context switches by means of software(programming). A modern CPU can perform hundreds of context switches per second. Therefore, the user gets the impression that the computer is performing multiple tasks in a parallel fashion, when the CPU actually alternates or rotates between or among the tasks at a high rate of speed.

Q8. Consider a system with 'n' CPU processors and 'm' processes, then answer the following queries regarding minimum and maximum number of processes:

	Minimum	Maximum
Ready	**?**	**?**
Running	**?**	**?**
Block	**?**	**?**

Ans.

	Minimum	**Maximum**
Ready	0	M
Running	0	N
Block	0	M

Q9. Define the responsibilities of: Short term scheduler, Middle term scheduler and Long term scheduler.

Ans.

Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run. Schedulers are of three types −

- Long-Term Scheduler
- Short-Term Scheduler
- Medium-Term Scheduler

Long Term Scheduler :

It is also called a **job scheduler**. A long-term scheduler determines which programs are admitted to the system for processing. It selects processes from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling

Short Term Scheduler :

It is also called as **CPU scheduler**. Its main objective is to increase system performance in accordance with the chosen set of criteria. It is the change of *ready state to running state* of the

process. *CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.*

Short-term schedulers, also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are *faster* than long-term schedulers.

Medium Term Scheduler:

Medium-term scheduling is a part of **swapping**. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium-term scheduler is in-charge of handling the swapped out-processes.

A running process may become suspended if it makes an I/O request. A suspended processes cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage. This process is called **swapping**, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

Q10. State some principles of giving a better user interactivity.
Ans.

ANSWER 10:

a) **LEARNABILITY:**
Another very important core principle is the ability to easily learn and use an interface after using it for the first time.

b) **CONSISTENCY:**
As well as **matching people's expectations** through terminology, layout and interactions the way in which they are used should be consistent throughout the process and between related applications. By maintaining consistency users learn more quickly, this can be achieved by re-applying in one part of the application their prior experiences from another.

c) **MATCH USER EXPERIENCE AND EXPECTATION:**
By matching the sequence of steps, layout of information and terminology used with the expectations and prior experiences of the user, the friction and discomfort of learning a new system will be reduced.

Q11. Why can’t a virus just take hold of the system and keep running? Think and try to
Ans.

Virus programs out there in the Internet are not that smart to get into a computer hardware. A CPU is a hardware component and that cannot be attacked directly by a software program (Virus). So the Virus program first needs to get into the Motherboard firmware (BIOS) to take control of the whole system. But currently there are no malwares known to the world which could do such complicated things. It doesn’t mean that they do not exist. A CPU is at the lowest end of all computer components. It is coded in a different language then your operating system (CPU is coded in Assembly and OS mainly in C++, the virus has to take over loads of parts before it can reach the CPU. Also a CPU isn't a single component. It has two more units inside it called the Arithmetic Logic Unit and the Control Unit. It is basically super complex to code a virus to do so.

Q12. What is the responsibility of dispatcher?
Ans.

The dispatcher is the module that gives control of the CPU to the process selected by the short-time scheduler(selects from among the processes that are ready to execute).
The function involves :
Swithching context
Switching to user mode
Jumping to the proper location in the user program to restart that program.

Q13. What are the applications of real time operating system?
Ans.

Real time operating system are known for -
1.Deterministic- they execute functions in fixed amount of time
2.Correctness- time at which result produced
3.Predictability- all constrints related to timing meet

Application based on classification of real time operating system -

*SOFT RTOS - performance degraded but not destroyed by failure to meet response time constraints.
Used in Multimedia, interactive video games

*FIRM RTOS - missing more than few deadline, may lead to catastrophe.
In robot weed killer

*Hard RTOS- failure to meet single deadlines may lead to catastrophic failure
In aircraft control system, aviation, nuclear power systems, cheamical plants ,life support system.

Q14.What do understand by the term system call?
Ans.

In computing, a **system call** is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. This may include hardware-related services (for example, accessing a hard disk drive), creation and execution of new processes, and communication with integral kernel services such as process scheduling. System calls provide an essential interface between a process and the operating system.

In most systems, system calls can only be made from userspace processes, while in some systems, OS/360 and successors for example, privileged system code also issues system calls

Q15. What is the use of Fork and Exec system call?
Ans.

Fork

When you come to metaphorical "fork in the road" you generally have two options to take, and your decision effects your future. Computer programs reach this fork in the road when they hit the fork() system call.

At this point, the operating system will create a new process that is exactly the same as the parent process. This means all the state that was talked about previously is copied, including open files, register state and all memory allocations, which includes the program code.

The return value from the system call is the only way the process can determine if it was the existing process or a new one. The return value to the parent process will be the Process ID (PID) of the child, whilst the child will get a return value of 0.

At this point, we say the process has forked and we have the parent-child relationship as described above.

Exec

Forking provides a way for an existing process to start a new one, but what about the case where the new process is not part of the same program as parent process? This is

the case in the shell; when a user starts a command it needs to run in a new process, but it is unrelated to the shell.

This is where the exec system call comes into play. exec will *replace* the contents of the currently running process with the information from a program binary.

Thus the process the shell follows when launching a new program is to firstly fork, creating a new process, and then exec (i.e. load into memory and execute) the program binary it is supposed to run.