NAME- Akhilesh agrhari
RN - 15
SEC - A

Assignment - 1

1- What are the main functions of an OS ?

ans- Resource management - Operating System Also Known as the Resource Manager Means Operating System will Manages all the Resources those are Attached to the System means all the Resource like Memory and Processor and all the Input output Devices those are Attached to the System are Known as the Resources of the ComputerSystem and the Operating system will Manage all the Resources of the System. The Operating System will identify at which Time the CPU will perform which Operation and in which Time the Memory is used by which Programs. And which Input Device will respond to which Request of the user means When the Input and Output Devices are used by the which Programs. So this will manage all the Resources those are attached to the Computer System.

Storage Management- Operating System also Controls the all the Storage Operations means how the data or files will be Stored into the computers and how the Files will be Accessed by the users etc. All the Operations those are Responsible for Storing and Accessing the Files is determined by the Operating System Operating System also Allows us Creation of Files, Creation of Directories and Reading and Writing the data of Files and Directories and also Copy the contents of the Files and the Directories from One Place to Another Place.

Process Management : The Operating System also Treats the Process Management means all the Processes those are given by the user or the Process those are System 's own Process are Handled by the Operating System . The Operating System will Create the Priorities foe the user and also Start or Stops the Execution of the Process and Also Makes the Child Process after dividing the Large Processes into the Small Processes.

Memory Management: Operating System also Manages the Memory of the Computer System means Provide the Memory to the Process and Also Deallocate the Memory from the Process. And also defines that if a Process gets completed then this will deallocate the Memory from the Processes.

Extended Machine : Operating System also behaves like an Extended Machine means Operating system also Provides us Sharing of Files between Multiple Users, also Provides Some Graphical

Environments and also Provides Various Languages for Communications and also Provides Many Complex Operations like using Many Hardware's and Software's.

ANS 2 - This depends on the OS and the CPU architecture. On x86 (Intel compatible) the operating system might execute HLT instructions, making the CPU wait until something interesting happens, such as a hardware interrupt. This supposedly consumes very little power. Operating systems report the time spent doing this as "idle" and may even assign it to a fictional "idle" process.

ANS 3-

```
int main() // CPU execution
{
Int I,j; // CPU execution
Scanf("%d",&i); // I/P execution
for(j=0;j<I;j++) // CPU execution
{
Sum=j+I; // CPU execution
}
printf("%d",Sum); // O/P execution
exit(0); // CPU execution
}
```

ANS 4 -

Multiprogramming

In a multiprogramming system there are one or more programs loaded in main memory which are ready to execute. Only one program at a time is able to get the CPU for executing its instructions (i.e., there is at most one process running on the system) while all the others are waiting their turn.

The main idea of multiprogramming is to maximize the use of CPU time. Indeed, suppose the currently running process is performing an I/O task (which, by definition, does not need the CPU to be accomplished). Then, the OS may interrupt that process and give the control to one of the other in-main-memory programs that are ready to execute (i.e. *process context switching*). In this way, no CPU time is wasted by the system waiting for the I/O task to be completed, and a running process keeps executing until either it voluntarily releases the CPU or when it blocks for an I/O operation. Therefore, the ultimate goal of multiprogramming is to keep the CPU busy as long as there are processes ready to execute.

Note that in order for such a system to function properly, the OS must be able to load multiple programs into separate areas of the main memory and provide the required protection to avoid the chance of one process being modified by another one. Other problems that need to be addressed when having multiple programs in memory is *fragmentation* as programs enter or leave the main memory. Another issue that needs to be handled as well is that large programs may not fit at once in memory which can be solved by using *pagination* and *virtual memory*. Please, refer to this article for more details on that.

Finally, note that if there are N ready processes and all of those are highly CPU-bound (i.e., they mostly execute CPU tasks and none or very few I/O operations), in the very worst case one program might wait all the other N-1 ones to complete before executing.

Multiprocessing

Multiprocessing sometimes refers to executing multiple processes (programs) at the same time. This might be misleading because we have already introduced the term "multiprogramming" to describe that before.

In fact, multiprocessing refers to the *hardware* (i.e., the CPU units) rather than the *software* (i.e., running processes). If the underlying hardware provides more than one processor then that is multiprocessing. Several variations on the basic scheme exist, e.g., multiple cores on one die or multiple dies in one package or multiple packages in one system.

Anyway, a system can be both multiprogrammed by having multiple programs running at the same time and multiprocessing by having more than one physical processor.

Multitasking

Multitasking has the same meaning of multiprogramming but in a more general sense, as it refers to having multiple (programs, processes, tasks, threads) running at the same time. This term is used in modern operating systems when multiple tasks share a common processing resource (e.g., CPU and Memory). At any time the CPU is executing one task only while other tasks waiting their turn. The illusion of parallelism is achieved when the CPU is reassigned to another task (i.e. *process* or *thread context switching*).

There are subtle differences between multitasking and multiprogramming. A *task* in a multitasking operating system is not a whole application program but it can also refer to a "thread of execution" when one process is divided into sub-tasks. Each smaller task does not hijack the CPU until it finishes like in the older multiprogramming but rather a fair share amount of the CPU time called quantum.

Just to make it easy to remember, both multiprogramming and multitasking operating systems are **(CPU) time sharing** systems. However, while in multiprogramming (older OSs) one program as a whole keeps running until it blocks, in multitasking (modern OSs) time sharing is best manifested because each running process takes only a fair quantum of the CPU time.

ANS 5 -

PROGRAM - Program is an executable file containing the set of instructions written to perform a specific job on your computer. For example, **notepad.exe** is an executable file containing the set of instructions which help us to edit and print the text files.

Programs are not stored on the primary memory in your computer. They are stored on a disk or a secondary memory on your computer. They are read into the primary memory and executed by the kernel. A program is sometimes referred as **passive entity** as it resides on a secondary memory.

PROCESS - Process is an executing instance of a program. For example, when you double click on a notepad icon on your computer, a process is started that will run the notepad program.

A process is sometimes referred as **active entity** as it resides on the primary memory and leaves the memory if the system is rebooted. Several processes may related to same program. For example, you can run multiple instances of a notepad program. Each instance is referred as a process.

ANS 6 -

New State- Secondary memory.
Ready State- Main memory.
Run State- Main memory.
Block State- Main memory.
Terminate State- Main memory.
Suspend Ready State- Main memory.
Suspend Block State- Secondary memory

ANS 7 - A *context switch* (also sometimes referred to as a *process switch* or a *task switch*) is the switching of the CPU (central processing unit) from one *process* or *thread* to another.

Eg- p1 and p2 are two process are ready to execute. Suppose p1 is assigned to the cpu first. After some time p1 is removed from the cpu and cpu is given to the process p2. This removing and assigning of processes to the cpu is context switch

ANS 8 -

	Minimum	Maximum
Ready	0	M
Running	0	N
Block	0	M

ANS 9- **SHORT TERM SCHEDULER** - The short-term scheduler(also known as the CPU **scheduler**) decides which of the ready, in-memory processes is to be executed (allocated a CPU) after a clock interrupt, an I/O interrupt, an operating system call or another form of signal.

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.

MIDDLE TERM SCHEDULER - **Medium term scheduling** is part of the 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.

ANSWER 10:

LEARNABILITY:

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

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.

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.

ANSWER 11:

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.

ANSWER 12:

It is the module that gives control of the CPU to the process selected by the short-term scheduler. It receives control in kernel mode as the result of an interrupt or system call. The functions of a dispatcher involve the following:

Context switches, in which the dispatcher saves the state (also known as context) of the process or thread that was previously running; the dispatcher then loads the initial or previously saved state of the new process.

Switching to user mode.

Jumping to the proper location in the user program to restart that program indicated by its new state.

The dispatcher should be as fast as possible, since it is invoked during every process switch. During the context switches, the processor is virtually idle for a fraction of time, thus unnecessary context switches should be avoided.

ANSWER 13:

Applications of Real Time operating system:

Almost all the modern telecommunication systems make use of RTOS.

Radar systems, network switching control systems, satellite monitoring systems, satellite launch-control and maneuvering mechanisms, global positioning systems all have their roots in RTOS.

Now a days RTOs are increasingly finding use in strategic and military operations. These are used in guided missile launching units, track-and-trace spy satellites, etc.

ANSWER 14:

It provides an interface to the service available by an OS. These calls are generally available as routines written in C and C++.

ANSWER 15:

fork():

The purpose of fork() is to create a new process, which becomes the child process of the caller. After a new child process is created, both processes will execute the next instruction following the fork()system call. Therefore, we have to distinguish the parent from the child.

exec():

In computing, exec is a functionality of an operating system that runs an executable file in the context of an already existing process, replacing the previous executable. This act is also referred to as an overlay.