UNIT I

1) What is OS?

→

- OS is a middleware betⁿ user program & System hardware.
- An operating system (OS) is system software that manages computer hardware, software resources & provides common services for computer programs.
- For hardware functions such as i/p & o/p & memory allocation, the OS acts as an intermediate between programs and the computer hardware, although the application code is usually executed directly by the hardware and frequently makes system calls to an OS function / is interrupted by it.

2) Why it is required?

→

Need of Operating Software:

i) OS as a platform for Application programs -

- OS provides a platform, on top of which, other programs, called application programs can run.
- These programs help users to perform a specific task easily
- Acts as an interface betⁿ the computer & the user.

ii) Managing i/p - o/p unit -

- OS allows the computer to manage its own resources such as memory, monitor, keyboard, etc.
- Management of these resources is required for an effective utilization.

iii) Consistent user interface -

- OS provides the user an easy-to-work user interface, so

user doesn't have to learn a different UI every time.
- OS provides templates, UI components to make the working of a computer, easy for the user.

iv) Multitasking -
- OS manages memory & allow multiple programs to run in their own space & even communicate with each other through shared memory.

3) What are the functions of OS?

→ i) Process Management: Deals with management of the CPU
- The OS takes care of the allotment of CPU time to diff. processes.
- When a process finishes its CPU processing after executing for the alloted time period, then is called scheduling.
- Types of scheduling techniques used by OS -

a) Shortest Job first (SJF):
Process which need the shortest CPU time are scheduled first

b) Round Robin Scheduling:
Each process is assigned a fixed CPU execution time in cyclic way

c) Priority Based Scheduling (Non-Preemptive):
Processes are scheduled according to their priorities.

ii) Device Management:
OS communicates with hardware & the attached devices & maintains a balance betn them & the CPU

iii) Buffering

iv) Spooling (Simultaneous Peripheral Operation on line)

v) Memory management

vi) Virtual Memory

4) Memory Virtualization

→

- Memory Virtualization is seen as virtual memory/swap on servers & workstations
- It enhances performance by providing greater memory capacity without the expense of adding main memory.
- Memory virtualization decouples volatile RAM resources from individual systems in the data centre, & then aggregates these resources into a virtualized memory pool available to any computer in the cluster.

5) What are diff. types of OS? Explain any one

→

Types of OS:

1) Batch OS
2) Time-sharing OS
3) Distributed OS
4) Network OS
5) Real-Time OS
6) Multiprogramming OS
7) Multitasking OS
8) Multiprocessing OS

⇒ Multiprogramming OS:

- ~~Multiprogragramming~~ Multiprogramming means more than one process in main memory which are ready to execute
- Maximize CPU utilization
- Process generally require CPU time & I/O time
- CPU never idle unless there is no process ready to ~~use~~ execute

• Advantages :
1) High CPU utilization
2) Less working time, response time for the process
3) May be extended to multiple users
• Disadvantages :
1) Difficult scheduling
2) Memory management is required.

6) Explain fork system call with example

→ System call fork() is used to create processes
• It takes no arguments & returns a process ID
• 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, distinguishing the parent from the child can be done by testing the returned value of fork()
a) If fork() returns a -ve value, the creation of a child process was unsuccessful
b) fork() returns a zero to the newly created child process
c) fork() returns a +ve value, the process ID of the child process, to the parent.
↳ normally an integer
Process ID is of type pid_t defined in sys/types.h

Example:
```
#include<stdio.h>
# include<string.h>
# include <sys/types.h>

# define MAX_COUNT 200
# define BUF_SIZE 100
```

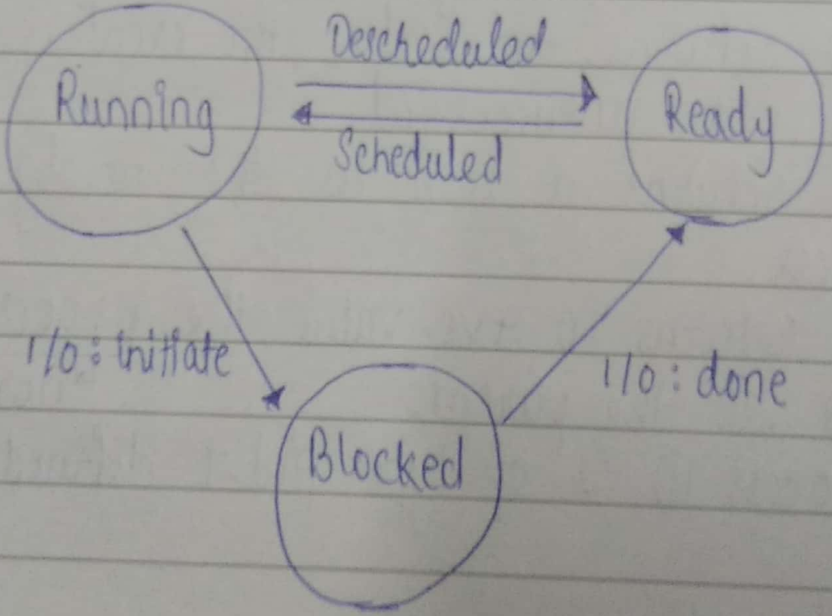
Scanned with CamScanner

```
void main(void)
{
   pid_t pid;
   int i;
   char buf[BUF_SIZE];

   fork();
   pid = getpid();
   for (i=-1; i<=MAX_COUNT; i++)
   {
    Sprintf(buf, "This line is from pid %d, value = %d\n", pid, i
      write (1, buf, strlen(buf));
   }
}
```

7) Draw & explain process steps diagram

⟶



- Running : Currently executing on CPU
- Ready : Waiting to be scheduled, ready to run but not yet run / the process is waiting to be assigned to the processor.

- Blocked: suspended, not ready to run
  → Process is ~~ready~~ waiting for some event to occur (such as I/O completion)
- New: being created, yet to run
- Dead: terminated: The process has finished execution

8) How OS keeps track of processes?

→ As the OS supports multi-programming, it needs to keep track of all the processes. For this task, the process control block (PCB) is used to track the process's execution status.

- A PCB contains information about the process, i.e. registers, quantum, priority, etc.
- The process table is an array of PCB's, that means logically contains a PCB for all of the current processes in the system.

1) Pointer: It is a stack pointer which is req. to be saved when the process is switched from one state to another to retain the current position of the process.
2) Process State: Stores the respective state of the process
3) Process No.: Every process is assigned with a unique id known as process ID / PID which stores the process ~~id~~ identifier.
4) Program Counter: Stores the counter which contains the address of the next instruction that is to be executed for the process

Pointer
Process State
Process No.
Program Counter
Registers
Memory Limits
Open File Lists
Misc. Accounting & Status Data

PCB

5) Register : CPU registers which includes : accumulator, base, registers & general purpose registers
6) Memory limits : ~~The~~ Contains the info about memory manage system used by OS. This may include the page tables, segment tables etc.
7) Open list : This info. includes the list of files opened for a process.
8) Miscellaneous accounting & status data :
This field includes info. about the amount of CPU used, time constraints, jobs/process no., etc.

9) Why / explain shell is required?
→
- A shell is a program which allows a user interactive text based control of the computer.
- It receives keyboard i/p, echoes it to the screen, & interprets the user's i/p to run commands, then it reads the o/p from the commands & sends it to the screen
- The way the shell talks to the kernel is by system calls. These system calls allows the user to do things like open files & create processes.
- Shell can manipulate the child in strange way.

10) What is context switch?
→
- Context Switching involves storing the context / state of a process so that it can be reloaded when required & execution can be resumed from the same point as earlier.
- This is a feature of a multitasking operating system &

allows a single CPU to be shared by multiple processes

- A context switch is the process of storing the state of a process / thread, so that it can be restored & resume execution at a later point.
- This allows multiple processes to share a single CPU, and is an essential feature of a multitasking OS.