Operating Systems Concepts

1. The major differences between a process and a thread are:

Process	Thread
a) A process is a program under execution i.e an active program.	a) A thread is a lightweight process that can be managed independently by a scheduler.
b) Processes require more time for context switching as they are more time than between threads.	b) Threads require less time for context switching as they are lighter than
c) Processes are totally independent and does not share memory.	c) A thread may share some memory with its peer threads

Thread is called a light weight process. because it has its own stack but can access shared data. Because ~~threads~~ threads share the same address as the process and other threads within the process; the operational ~~cost~~ cost of communication between the threads is low, which is an advantage.

2. Zombie process is the one which has finished the execution but still has entry in the process table to report to its parent process is known as a zombie process. A child process always first becomes a zombie before being removed from

the process table. The parent process reads the exit status of the child process which reaps off the child process entry from the process table.

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int main()
{
    parent_id  child_id = fork();
    if (child_id >0).
      sleep (50); //Parent process sleeps for 50 seconds.
    else
      exit(0); // child process finish execution after this call.
    return 0;
}
```

Orphan process is a process where parent process no more exists i.e either finished or terminated without waiting for its child process to terminate is called an orphan process.

```
#include <stdio.h>
#include <~~stdlib.h~~ sys/types.h>
#include <unistd.h>
int main ()
{
    int pid = fork(); // child process
    if (pid >0)
    {
      printf ("Parent process"); }
      // pid is 0 in child process and negative
      // if fork() fails.
    else if (pid==0)
    {
      sleep(30);
```

```
    printf("child process");
}
return 0;
}
```

Q3. What is the difference between fork and vfork? When should we use fork and wait system call combination instead of using vfork?

Ans:

fork()	vfork()
a) In fork() system call, child and parent process have separate memory space.	a) While in vfork() system call, child and parent process share same address space.
b) There is a wastage of address space.	b) There is a no wastage of addressing space

Q4. What is context? How context switching takes place? "Context switching is overhead to the system" - Justify.

Ans: A process has got various states for sequential execution of it. These process states is also called as context. In context switching, it involves storing the context or state of a process so that it can be reloaded when required and execution can be resumed from the same point as earlier.

Context switching time is pure overhead because the system does no useful work while switching, that is no user code is executing. The round robin algorithm incurs a greater number of task switches than non preemptive

algorithms. Each task switches means a greater amount of CPU time is spent doing task switches instead of useful work. If the quantum is too large then R/R degenerates to FCFS with poor response times.

5) Which multithreading model is best according to you & why?

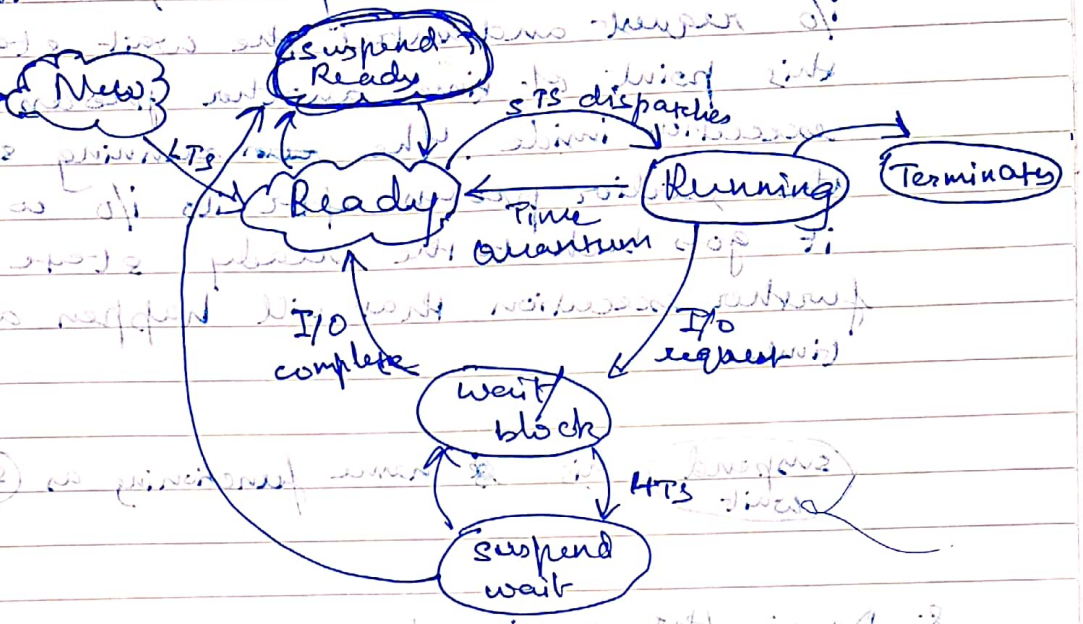
Ans- FCFS stands out for me in this. It is the simplest scheduling algorithm that schedules according to arrival time of processes. Hence, here the process requests first to CPU, and is allocated the CPU first. Therefore, no process is on hold and all are completed as well as it implies it is non-preemptive.

6) Differentiate Kernel level & user level thread.

User Level Thread	Kernel Level Thread
a) User managed threads.	a) Operating system managed threads acting on kernel, an operating system core.
b) User level threads are fast to create and manage.	b) Kernel threads are generally slower to create and manage than user threads.
c) Thread switching does not require kernel mode privileges.	c) Transfer of control from one thread to another within the same process requires a mode switch in the kernel.

Q7. Draw and explain process state diagram including both active and suspended states.

Ans:



In New state a process enters the RAM for execution LTS (Long Term Scheduler) brings the processes into ready queue from New state. When ready state has stared accumulating processes which might lead to complete full storage where Ready state can't take any more processes. But if a priority process results to come from New state at this point of time then Ready state won't have any space to give the priority process for execution. Since priority process to go to the Running state. After which the processes from suspend ready state will go to Ready state and then to Running state for further execution.

In the Running state the processes needs to be dispatched in CPU or scheduled in CPU for execution of the processes.

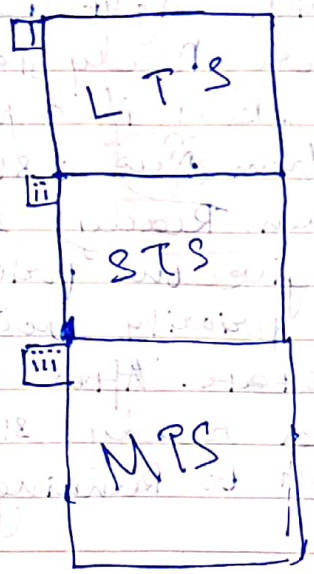
In the Terminated state a process is completely executed deallocated.

In (wait/block), a process has some i/o operation that needs to be performed while it is under execution in the running state, then it urges a i/o request and enters the wait state. At this point of time another process is under execution inside the running state so the earlier process after its i/o completion it gets back to the ready state for further execution that will happen at later time.

(suspend wait) his name functioning as (suspend Ready)

8. Draw the queuing diagram and state at which transition which scheduler works.

Ans-



From the New State to Ready state each process travels using the long Term Scheduler.

Processes from ready state dispatched to running state using the short term scheduler.

Medium Term Scheduler brings process from wait/block state to suspend wait if RAM is getting full due to many processes in wait/block state.

Q. Where is PCB? Why we need to have pointer in PCB? What is the purpose of storing CPU register and special purpose register values in PCB.?

Ans:- While creating a process the operating system performs several operations. To identify the processes, it assigns a PID to each process. As the operating system supports multi-programming, it needs to keep track of all the processes. For this task, the PCB is used to track the process execution status.

The pointer is also called as a stack pointer which is needed to saved when the process is switched from one state to another state to retain the current position of the process. It helps to access the PCB quickly.

10. What is Mid-Term scheduler? When it is required? Why the name is so?

Ans: Mid-term scheduler is responsible for suspending and resuming the process. It mainly does swapping. This scheduler is required when a running process can become suspended due to an i/o request. This scheduler works in between a process suspension and resuming. ~~This scheduler works in between a process itself.~~ ~~suspension~~ Suspension might happen due to some i/o request after which it again resumes. Thus, it is a mid-term scheduler.

11) Why CPU scheduler is called short term scheduler and Job scheduler is called long term scheduler? State this consider after clarifying the tasks of these two types of scheduler.

Ans: A CPU scheduler controls or maintains a context switch and CPU is switched among multiple threads; short-term scheduler also known as CPU scheduler.

The job scheduler maintains a queue of programs/jobs which are selected for system to process. Programs are selected based on the scheduling mechanism and processed. This is long-term scheduler that controls degree of multi-programming.

12: What is cooperating process and why we need them?

Ans: Cooperating process are those that can affect or are affected by other processes running on the system. They are needed as it can share data with each other.

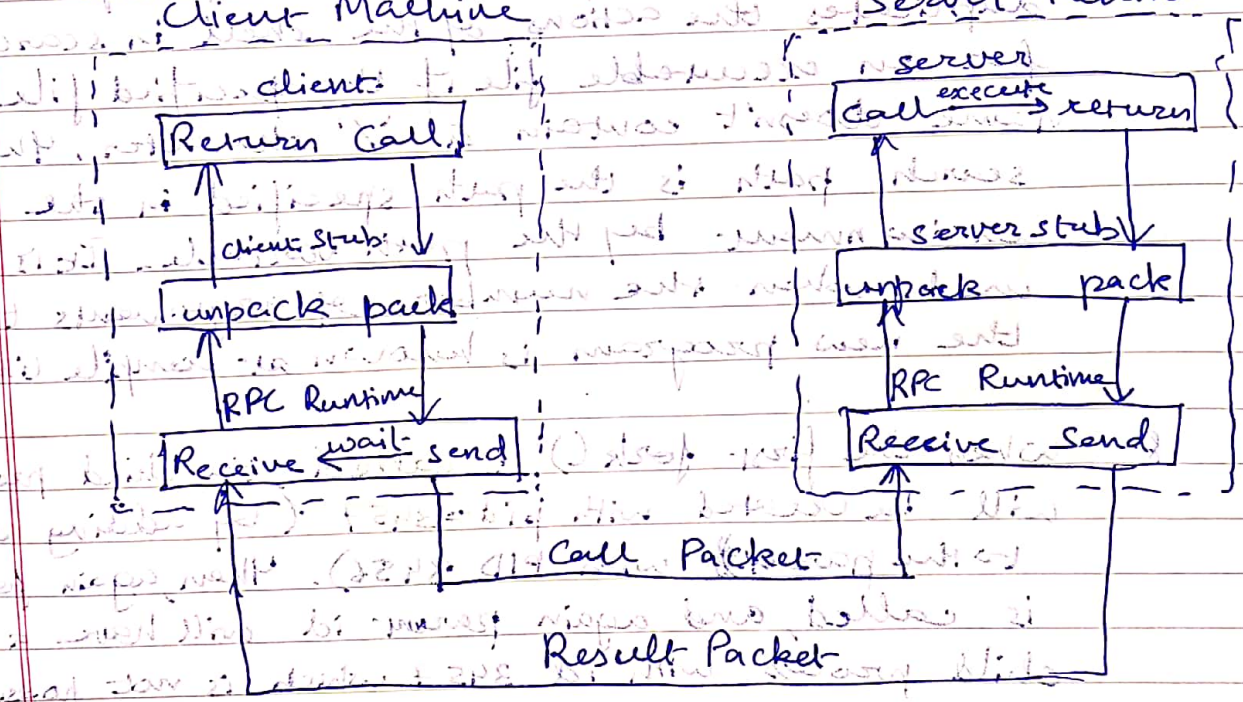
We need cooperating process for information sharing, computation speedup, modularity and convenience.

13: State the different types of message passing architectures and which one is required when, when should we use shared memory to communicate between processes?

14: The steps are:

- The client invokes a client stub procedure, passing parameters in usual way.
- The client stub marshalls the parameters into a message.

- The client stub passes it to the transport layer, which sends it to the remote server machine.
- On the server, the transport layer passes the message to a server stub, which demarshalls the parameters.
- It returns to the server stub after process completion and marshalls the return value into a message & send to transport layer.
- It sends the message back to client transport layer, which hands to the client stub.
- It demarshalls the return parameters and execution returns to the caller.



15. A socket is a software object that acts as an end point establishing a bidirectional network communication link between a server-side and a client-side program. It is used to tie the program's client and server sides. When a client establishes connection with the server, they read or write on sockets tied to specific communication

channel. These are of two types: active and passive. These are used to establish connection between client and server.

16. If one process is terminated, its related processes are also terminated abnormally. This phenomena is known as cascaded termination of process. It happens when a process creates a new process, the identity of the newly created process is passed to its parent. When a parent process is terminating, then all of its children process is also terminated. Thus, cascaded termination takes place.

17. The ~~exec~~ execlp system call ~~of duplication~~ duplicates the actions of the shell in searching for an executable file if the specified file name doesn't contain a '/' character. The search path is the path specified in the environment by the path variable. It is used when the number of arguments to the new program is known at compile time.

18. When the first fork() is called, the child process will be created with pid = 3457 (by adding one to the parent with PID 3456). Then, again fork is called, and again parent id will have a child process with id 3457 which is not possible and the 1st child will have a child with id 3458.

∴ One 1st printf :- 3456
3457
3458

Then, on third fork(), 3 child process will be created and two child will not be accepted due to same id.

∴ Four times Hello will be printed.

19. An address space is a range of valid addresses in memory that are available for a program or process. What is, it is the memory that a program or process can access. The memory can be either physical or virtual and is used for executing instructions and storing data. On a computer, each process and device is allocated an address space, which holds a certain portion of the processor's address space.

20. After p = fork();

if parent process will execute first, it will wait until the termination of the child.

Then, q = 100. → It will be printed by child child process.

Then, a child will be created & parent will be blocked.

∴ q = 200

Then, parent will proceed and will print.

q = 200 — Terminated

Now, the parent will again create a child.

∴ q = 200

q = 200.

13- In the message-passing model of interprocess communication, communication takes place by means of messages exchanged between the cooperating process.

The different types of message passing architecture are:-

- Direct or indirect communication
- Synchronous or asynchronous communications.
- Automatic or explicit buffering.

Shared memory provides a way by letting two or more processes share a memory segment. We should use shared memory when we want the data to be copied only twice - from input file into shared memory and from shared memory to the output file.