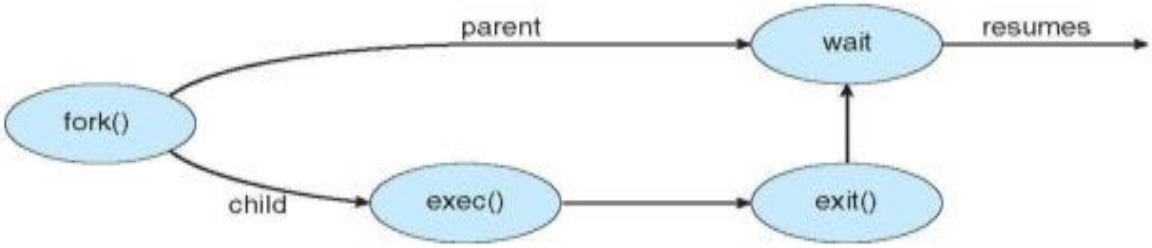
Solved Questions Module-2

1. With a figure, explain how process is created using fork () system call?



Parent process create children processes, which, in turn create other processes, forming a tree of processes. Generally, process identified and managed via a process identifier (pid) fork() system call creates new process

2. What is the use of Process Control Block (PCB) in operating system?

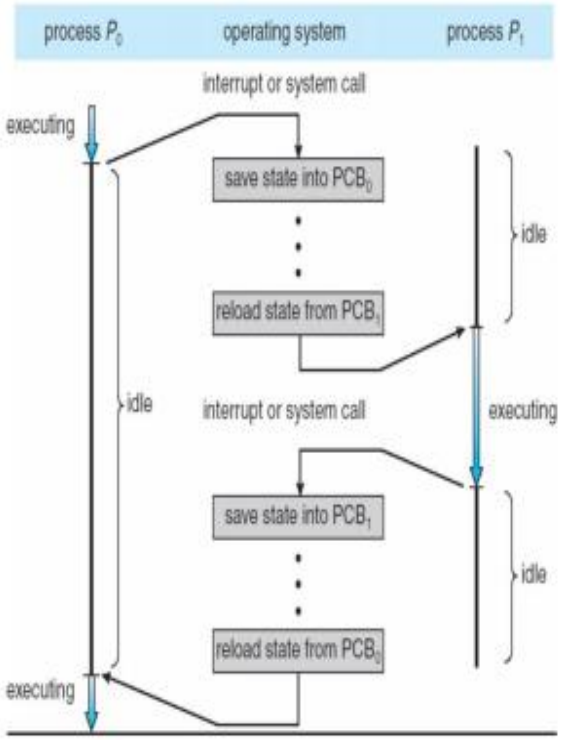
3. Explain process control block.

Each process is represented in the operating system by a Process Control Block (PCB)-also called a task control block. In brief, the PCB simply serves as the repository for any information that may vary from process to process. - 1 mark

PCB diagram + field details – 2 Marks

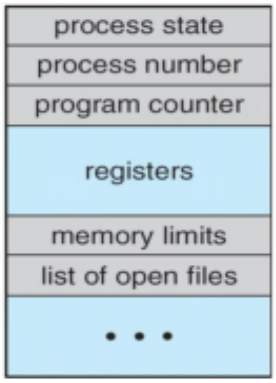
Process Control Block

- Each process is represented on the OS by a Process Control Block(PCB) also called a *task control block*..
- It includes
  - Process state – The state may be new, ready running, waiting, halted etc
  - Program counter – The counter indicates the address of the next instruction to be executed for this process.
  - CPU registers – The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information. Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward



  - CPU scheduling information- includes a process priority, pointers to scheduling queues, and any other scheduling parameters.

- Memory-management information -include such information as the value of the base and limit registers, the page tables, or the segment tables, depending on the memory system used by the operating system.
- Accounting information –includes the amount of CPU and real time used, time limits, account numbers, job or process numbers, and so on.
- I/O status information – I/O devices allocated to process, list of open files



Process Control Block

4. How many times 'Good Luck' and 'Do well' will be printed after executing the following code. Justify your answer.

```
void main()
{
fork();
printf("Good Luck\n");
fork();
fork();
printf("Do well\n");
}
```

7 child processes and Good Luck printed 2 times and Do well printed 8 times.

2^n is used to find count of printed statements.

2^1 = 2 times printed Good Luck.

2^3 = 8 times printed Do well.

$2^n - 1$ =7 child processes.

5. How many times will 'Forked' get printed by the below code and justify your answer.

```
int main() {
fork();
fork();
printf("Forked\n");
return 0;}
```

3 child processes will be created and hence together with parent, it will print Forked 4 times.

6. List and explain the various synchronous and asynchronous methods of message passing in IPC.

Blocking send. Non blocking send, Blocking receive, Non blocking receive

Message passing may be either blocking or nonblockingalso known as synchronous and asynchronous.

- Blocking send. The sending process is blocked until the message is received by the receiving process or by the mailbox.
- Nonblocking send. The sending process sends the message and resumes operation.
- Blocking receive. The receiver blocks until a message is available.
- Nonblocking receive. The receiver retrieves either a valid message or a null.

7. Explain the different buffering mechanisms used in message passing systems?

Whether communication is direct or indirect, messages exchanged by communicating processes reside in a temporary queue. Basically, such queues can be implemented in three ways:

- Zero capacity -The queue has a maximum length of zero; thus, the link cannot have any messages waiting in it. In this case, the sender must block until the recipient receives the message.
- Bounded capacity. The queue has finite length n; thus, at most n messages can reside in it. If the queue is not full when a new message is sent, the message is placed in the queue, and the sender can continue execution without waiting. The link's capacity is finite. If the link is full, the sender must block until space is available in the queue.
- Unbounded capacity. The queue's length is potentially infinite; thus, any number of messages can wait in it. The sender never blocks.

8. Define the parameters for multilevel feedback queue scheduling? How it is better compared to multilevel queue scheduling?

6. Multilevel Feedback Queue Scheduling

- In multi level queue scheduling, processes are permanently assigned to a queue on entry to the system. Processes do not move from one queue to the other, since processes do not change their foreground or background nature.
- For avoiding starvation Multilevel feedback queue scheduling, allows a process to move between queues.
- The idea is to separate processes with different CPU-burst characteristics.
- If a process uses too much CPU time, it will be moved to a lower-priority queue. This scheme leaves I/O-bound and interactive processes in the higher-priority queues.

- Similarly, a process that waits too long in a lower- priority queue may be moved to a higher-priority queue. This form of aging prevents starvation.

Figure: Multilevel feedback queues

- The processes are classifies on the basis of CPU burst time.
- Highest priority is given to the I/O bound processes and interactive processes.
- The incoming processes are entered in Q0. In Q0, each process has same priority.
- The first process in Q0 is taken and allocates CPU for execution. After completing 8 milliseconds, the process is preempted and it is added to the end of Q1.
- After completing one turn in Q0, the processes are taken from Q1. After completing 16 time quantum, next go to FCFS.
- Last performs FCFS.

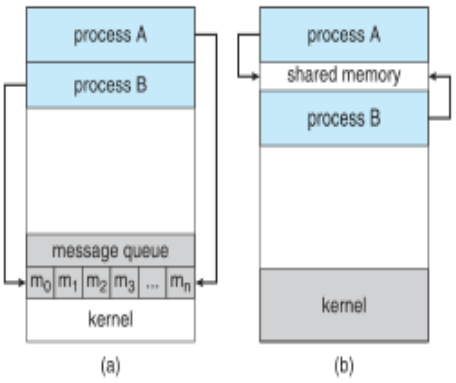
9. Write the difference between process and thread.

- The primary difference is that threads within the same process run in a shared memory space, while processes run in separate memory spaces.

- Threads are not independent of one another like processes are, and as a result threads share with other threads their code section, data section, and OS resources (like open files and signals).
- But, like a process, a thread has its own program counter (PC), register set, and stack space.

10. Differentiate between the two common models of interprocess communication.

- Inter process communication (IPC) refers to the coordination of activities among cooperating processes.
- Reasons for cooperating processes:
    - Information sharing
    - Computation speedup
    - Modularity
    - Convenience
- Two models of IPC
    - **Shared memory-**
    - **Message passing**



Shared-Memory Systems

- Interprocess communication using shared memory requires communicating processes to establish a region of shared memory.
- A shared-memory region resides in the address space of the process creating the shared memory segment. Other processes that wish to communicate using this shared memory segment must attach it to their address space.
- They can then exchange information by reading and writing data in the shared areas.
- The communication is under the control of the users processes not the operating system.
- Example for cooperating processes.:-Producer-consumer problem.
    - A producer process produces information that is consumed by a consumer process.
    - One solution to the producer-consumer problem uses shared memory.
    - A buffer which reside in a region of memory that is shared by the producer and consumer processes is used
    - The producer and consumer must be synchronized, so that the consumer does not try to consume an item that has not yet been produced
    - Two types of buffers can be used.
        - **unbounded-buffer** places no practical limit on the size of the buffer
        - **bounded-buffer** assumes that there is a fixed buffer size

Message-Passing Systems

- Message passing provides a mechanism to allow processes to communicate and to synchronize their actions without sharing the same address space.
- A message-passing facility provides at least two operations: send (message) and receive (message).
- If processes P and Q want to communicate, they must send messages to and receive messages from each other; a communication link must exist between them.
- methods for logically implementing a link and the send() /receive() operations:
    - Direct or indirect communication.
    - Synchronous or asynchronous communication.
    - Automatic or explicit buffering

11. Differentiate Pre-emptive and Non-pre-emptive scheduling giving the application of cach of them.

- Scheduling fell into one of the two general categories:
  - **Non Preemptive Scheduling:** When the currently executing process

gives up the CPU voluntarily.

- **Preemptive Scheduling:** When the operating system decides to favour another process, pre-empting the currently executing process.

Preemptive Scheduling

- CPU-scheduling decisions may take place under the following four circumstances:
  1. When a process switches from the running state to the waiting state
  2. When a process switches from the running state to the ready state.
  3. When a process switches from the waiting state to the ready state
  4. When a process terminates
- Scheduling under 1 and 4 is nonpreemptive.
- All other scheduling is preemptive
- Consider access to shared data
- Consider preemption while in kernel mode

- Consider interrupts occurring during crucial OS activities

12. Why is context switching considered to be an overhead to the system?

- **What is Context Switch?**
  - Switching the CPU to another process requires saving the state of the old process and loading the saved state for the new process.
  - This task is known as a Context Switch.

  - The context of a process is represented in the Process Control Block(PCB) of a process;
  - it includes the value of the CPU registers, the process state and memory-management information.
  - When a context switch occurs, the Kernel saves the context of the old process in its PCB and loads the saved context of the new process scheduled to run.
  - Context switch time is pure overhead, because the system does no useful work while switching.
  - Its speed varies from machine to machine, depending on the memory speed, the number of registers that must be copied, and the existence of special instructions(such as a single instruction to load or store all registers).
  - Typical speeds range from 1 to 1000 microseconds.
  - Context Switching has become such a performance bottleneck that programmers are using new structures(threads) to avoid it whenever and wherever possible.