3) Process Management

Page No. 28
Date

Process :- A process is a program in execution. It is the unit of work in a modern time sharing system.

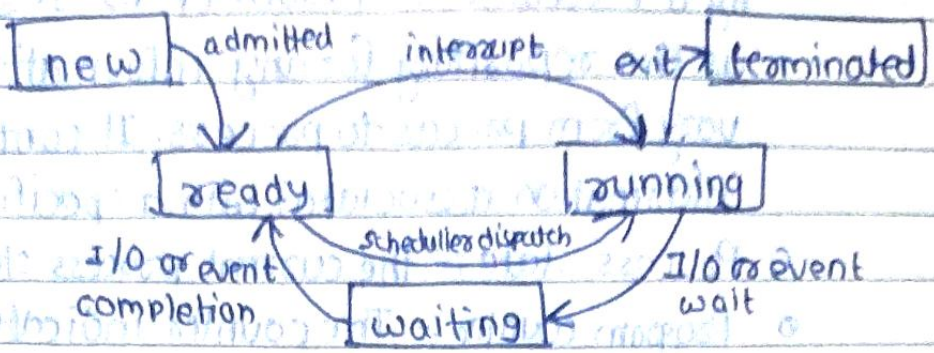
Concept :- Process may also be called as job or task. Program by itself is not a process. A program is a passive entity, such as a file containing a list of instructions stored on disk.
A process is an active entity having :
- Program counter specifying the next instruction to execute.
- set of associated resources.

A program becomes a process when an executable file is loaded into memory.

Diagram :

- Text section :- It contains program code, program counter and the contents of the processor's register. Program counter specifies the next instruction to execute.
- Data section :- It contains global variables
- Heap section :- Dynamically allocated memory during the process run-time
- Stack :- Contains temporary data such as function parameters, return addresses & local variables

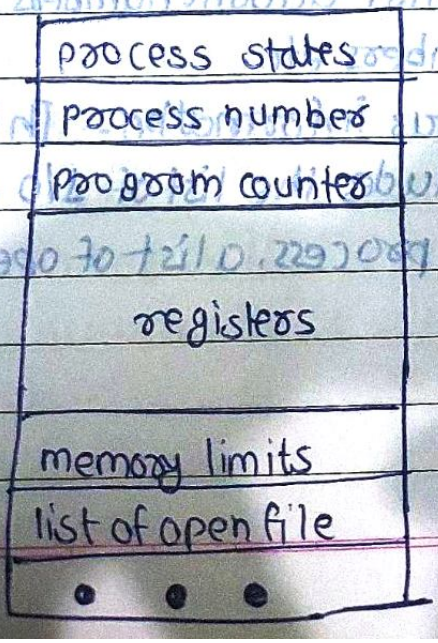
Process states:



Process changes the state during its execution. The state of a process is defined as a current activity of that process.
A process may be in one of the following states:

- New :- The process is being created.
- Running :- Instructions are being executed.
- Waiting :- The process is waiting for some event to occur.
- Ready :- The process is waiting to be assigned to a processor.
- Terminated :- The process has finished execution.

Process Control Block (PCB):



Each process is represented in the operating system by a Process Control Block (PCB). It simply serves as the repository for any information that may vary from process to process. It contains much information associated with a specific process

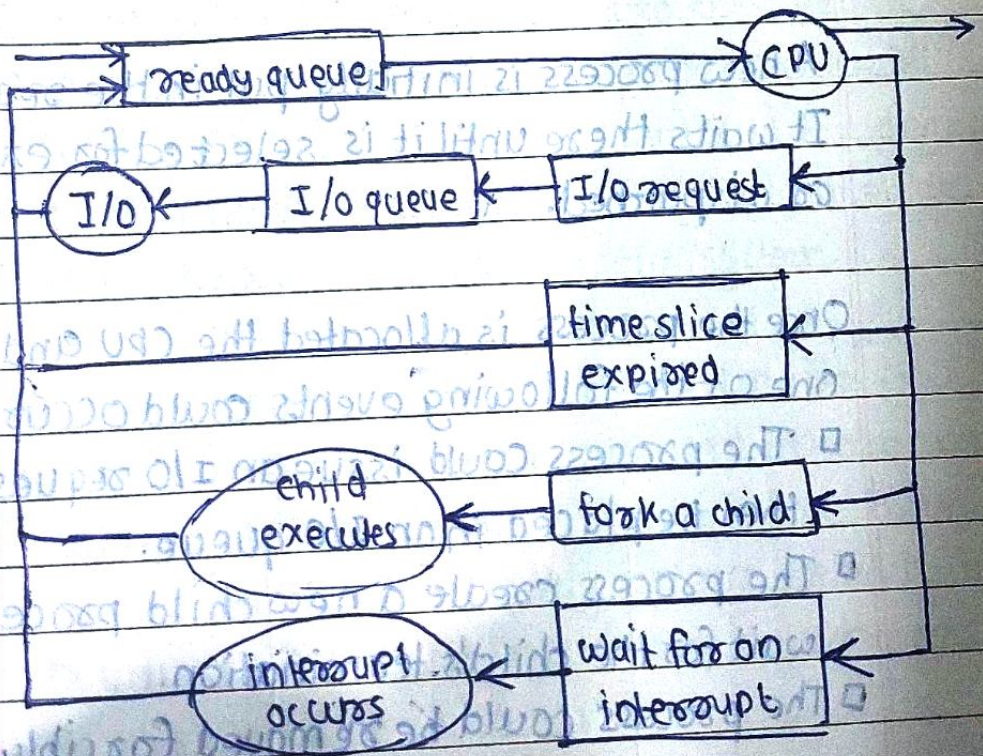
- Process state :- The current process state is shown.
- Program counter :- The counter indicates the address of the next instruction to be executed for this process.
- CPU registers :- All registers are stored here.
- CPU scheduling information :- The information includes a process priority, pointers to scheduling queues and any other scheduling parameters.
- Memory management information :- This information may include items as the value of the base and limit registers and the page tables or the segment table depends on the memory system used by OS.
- Accounting information :- The information include the amount of CPU and real time used, time limits, account numbers, job or process numbers etc.
- I/O status information :- This information includes the list of I/O devices allocated to the process, a list of open files, and so on.

Process schedulling :-

The objective of multiprogramming is to have some process running at all times, to maximize CPU utilization and the objective of time sharing is to switch the CPU among processes so frequently that users can interact with each program while it is running.

The meet these objectives, the process scheduller selects an available proces from a set of several available processes for program execution of the CPU.

Schedulling queues



When processes enter the system, they are put into a job queue, which consists of all processes in the system. The processes which are in main memory and are ready and waiting to execute are kept in the ready queue.

Two types of queues are present:

① The ready queue

② Set of device queues.

Notations used:

① Rectangle box represents a queue.

② Circle represents the resources that serves the queues.

③ Arrow indicates the flow of process in the system.

A new process is initially put in the ready queue. It waits there until it is selected for execution, or dispatched.

Once the process is allocated the CPU and is executing one of the following events could occur:

- The process could issue an I/O request and then be placed in an I/O queue.
- The process create a new child process and wait for the child's termination.
- The process could be removed forcibly from the CPU, as a result of an interrupt, and be put back in the ready queue.

Schedullers :- The operating system must select processes from these queues in some way. The selection process is carried out by the appropriate scheduller.

Types and functions of schedullers :
① Long-term scheduller or Job scheduller
② Short-term scheduler or CPU scheduler
③ Medium-term scheduller

① Long-term scheduler or Job scheduler :-
It selects processes from job pool and loads them into memory for execution. It executes much less frequently as compared to short-term scheduler. Minutes difference can be there between two processes. It controls the degree of multiprogramming. It may only need to invoke when process leaves the system.

② Short-term scheduler or CPU scheduler :-
It selects from the process that are ready to execute and allocates CPU to one of them. It executes more frequently as compared to long term scheduler. It must select new processes for the CPU frequently.

③ Medium-term schedulller :- It can be advantageous to remove a process from memory and thus reduce the degree of multiprogramming. Later the process can be reintroduced to into memory, and its execution can be continued from the previous position. This is called swapping.

Processes can be :-

① I/O bound process :- It spends more time doing I/O than doing computations.

② CPU bound process :- It spends more time doing computations than doing I/O.

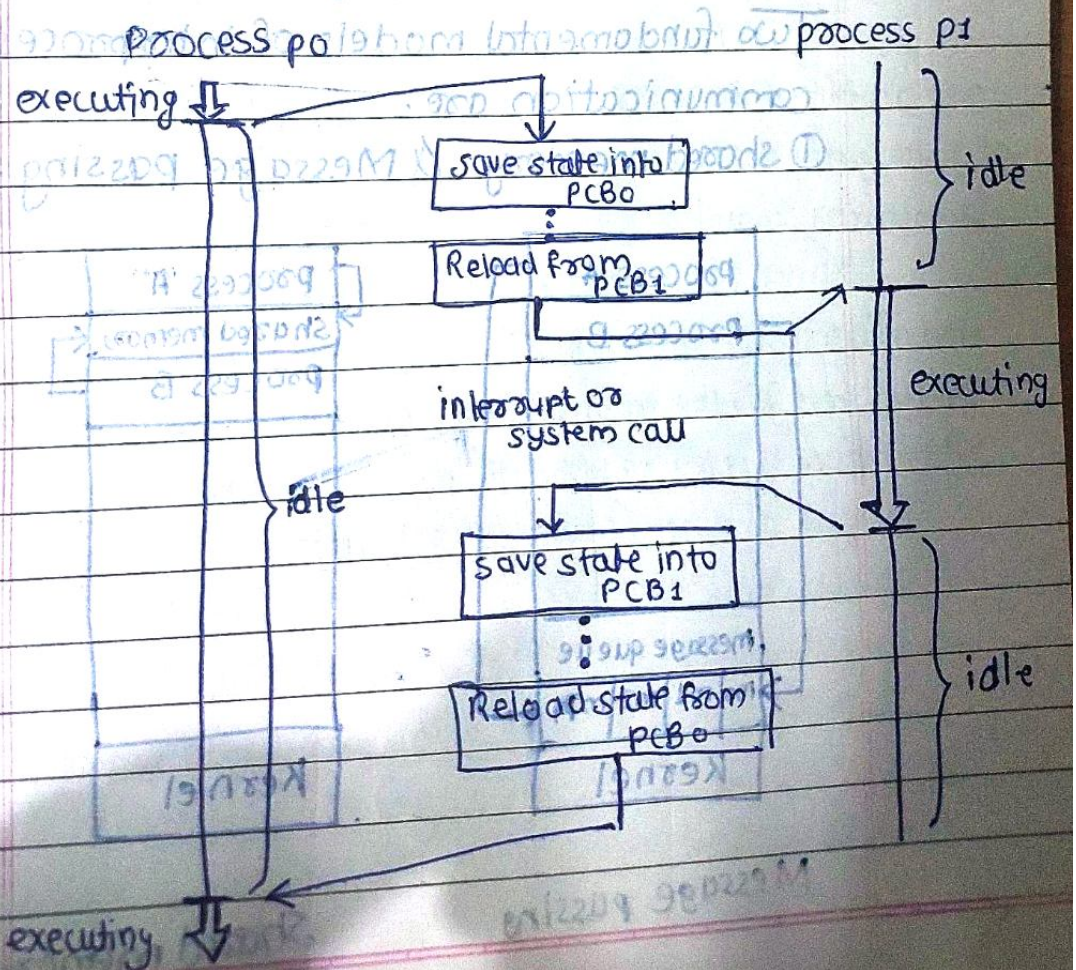
Long term schedulller must select a good mix of I/O ~~based~~ bound and CPU-bound processes. If all processes are I/O bound, the ready queue will almost empty, and the short term scheduler will have no work.

If all processes are CPU bound, the I/O waiting queue will almost empty, devices will be unused, and again the systems will be unbalanced.

The system with the best performance will have a combination of CPU bound and I/O bound processes.

	Long-term	Short-term	Medium term
1	It selects processes from job pool and loads them into memory for execution	It selects from the processes that are ready to execute and allocates the CPU to one of them	It removes the process from main memory and again reloads when required.
2	Frequency of execution is low	Frequency of execution is high	Frequency of execution is medium
3	Deals with main memory for loading processes	Deals with CPU	Deals with main memory for removing & reloading processes
4	Works slowly	Works fast	Moderate

Context Switch

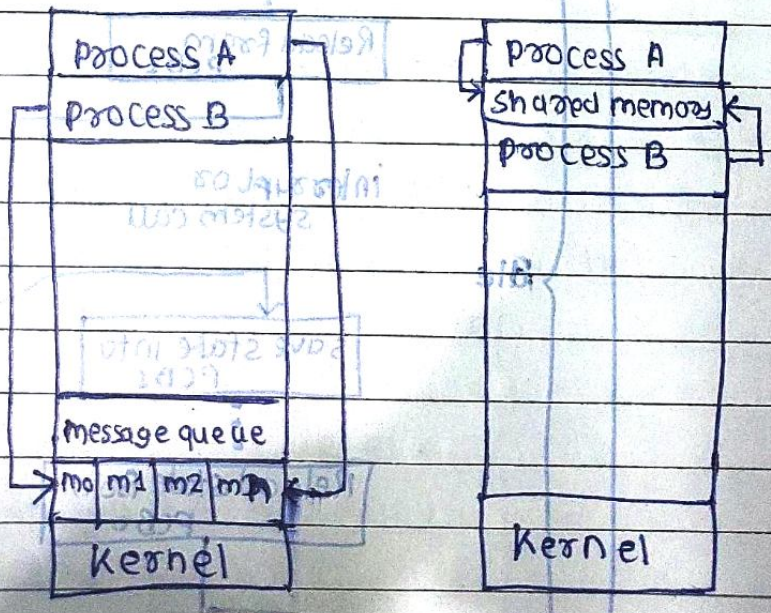


Switching the CPU to another process requires performing a state save of current process and a state restore of different process. This task is known as context switching. When context switching occurs, the kernel saves the context of the old process ~~int~~ in it's PCB and loads the saved context of the new process. It is a pure time overhead, because the system does no useful work while switching. Switching speed varies from machine to machine depending on the memory speed, the number of registers that must be copied and the existance of special instructions.

Interprocess communication

Two fundamental models of interprocess communication are:

① Shared memory ② Message passing.



Message passing Shared memory

Shared memory systems :-

Interprocess communication using shared memory requires establishing a shared memory region between the cooperating processes.

Shared memory regions resides in the address space of the process creating the shared memory segment. Other processes that want to communicate using this shared memory segment must attach to their address space.

Normally, OS tries to prevent one process from accessing another process's memory. Shared memory requires that two or more processes agree to remove this restriction. They can exchange information by reading and writing data in shared areas.

Two types of buffers can be used

① Unbounded buffer :- No practical limit on the size of the buffer. The consumer may have to wait for new items, but producer can always produce new items.

② Bounded buffer :- Fixed buffer size, In this case, the consumer must wait if the buffer is empty, and the producer must wait if the buffer is full.

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)
* 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 establish between them.

Methods for logical implementation of link for send receive operations are:

① Direct or Indirect communication
② Synchronous or Asynchronous communication
③ Automatic or explicit buffering

Direct or Indirect communication:

Direct communication:- Each processes that wants to communicate must explicitly name the recipient or sender of the communication

In this scheme, the send() and receive() primitives are defined as:

① send (P, message) - send msg to process P
② receive (Q, message) - Receive msg from Process Q

A communication link in this scheme has the following properties:

① A link is established automatically between every pair of processes that want to communicate
② The process need to know only each others identity.
③ A link is associated with exactly two processes.
④ Between each pair of processes, there exists only one link.

In symmetry, addressing both sender process and the receiver process must name the other to communicate

In assymmetry, addressing only sender names the recipient; the recipient is not required to name the sender

In this scheme, the send() and receive() primitives are defined as follows:

① send(P, message) - send a message to process P.
② receive(id, message) - Receive a message from any process whose name is unknown.

Indirect communication:- The messages are sent to and are received from mailbox or ports. A mailbox can be viewed abstractly as an object into which messages can be placed by processes and from which messages can be removed. Each mailbox has a unique identification.

Date

The send() and receive() primitives are defined as follows:

① send(A, message):- Send a message to mailbox A

② receive(A, message):- Receive a message from mailbox A.

A communication link has the following properties:

① A link is established between a pair of processes only if both members of the pair have a shared mailbox.

② A link may be associated with more than two process.

③ Between each pair of communicate processes, a number of different links may exists, with each link corresponding to one mailbox.

Synchronisation:- Communication between processes takes place through calls to send() and receive() primitives.

Message passing may be either:

① Blocking / Synchronous

② Non-blocking / Asynchronous

Blocking send:- The sender process is blocked until the message is received by the receiving process or the mail box.

Non-blocking send:- The sending process sends the message and resumes operation.

Date

Blocking receive :- The receiver blocks until a message is available.

Non-blocking receive :- The receiver retrieves either a valid message or a null.

✳ Buffering :- In direct or indirect communication, message exchanged by communicating processes reside in a temporary queue.
Such queues can be implemented in 3 ways.

① Zero capacity :- The queue has a maximum length of zero ; thus, it cannot have any messages waiting in it.

② Bounded capacity :- The queue has finite length (n), thus, at most n messages can reside in it.

③ Unbounded capacity :- The queue's length is infinite; thus, any number of messages can wait in it.

Threads :- A process is a program in execution. A process can be divided into number of threads. A thread is the basic unit of CPU utilization ; it contains a thread ID, a program counter, a register set and a stack.
One thread can share code section, data section and other operating system resources, such as open files and signals with other threads of the same process.

Benefits of Multithreaded programming:

① Responsiveness :- Multithreading an interactive application allows a program to continue running even if part of it is blocked or is performing a lengthy operation, thereby increasing responsiveness

② Resource sharing :- Processes can share resources through techniques such as shared memory or message passing. However threads share the memory and the resources of the process to which they belong by default.

③ Economy :- Allocating memory and resources for process creation is costly. But threads share the resources of the process to which they belong, it is more economical to create and context switch threads.

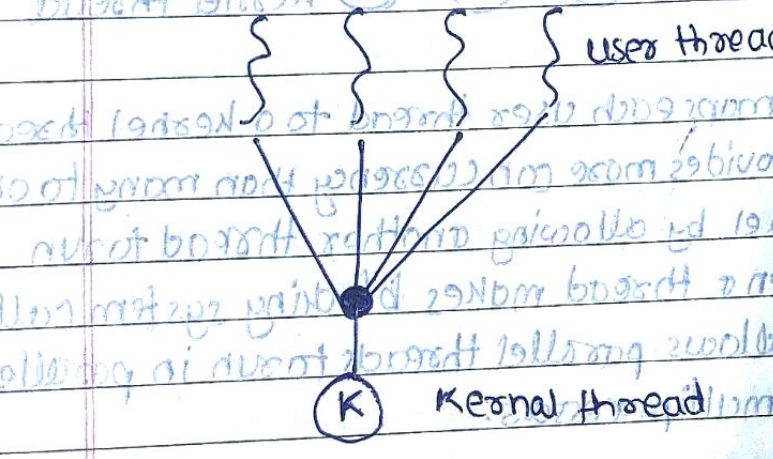
④ Scalability :- The benefits of multithreading can be even greater in a multiprocessor architecture, where threads may be running in parallel on different processing cores

User and kernel threads:

User threads are supported above the kernel and are managed without kernel support.
Kernel threads are supported and managed directly from the OS.
A relationship must exist between user level and kernel level threads. There are 3 ways to establishing such relationship:

- Many to one model
- One to one model
- Many to many model

- Many to one model:



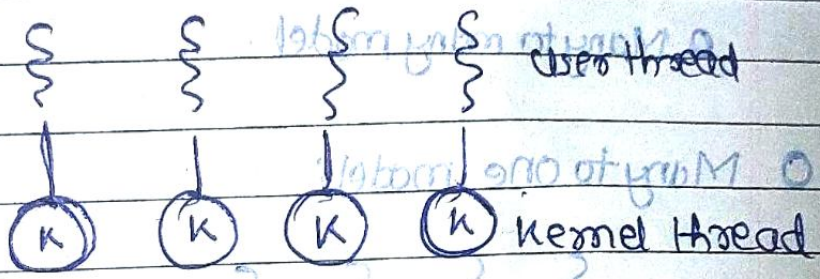
It maps many user threads to one kernel thread. Thread management is done by thread library in user space, so it is efficient. However the entire process will block if a thread makes a blocking system call.

Date

Advantages:
① Use of library
② Efficient system in terms of performance
③ One kernel threads controls multiple user threads

Disadvantages:
① Cannot run multiple threads parallel.
② One block call blocks all the user threads.

○ One-to-one model:



It maps each user thread to a kernel thread.
It provides more concurrency than many to one model by allowing another thread to run when a thread makes blocking system call.
It allows parallel threads to run in parallel on multiprocessors.

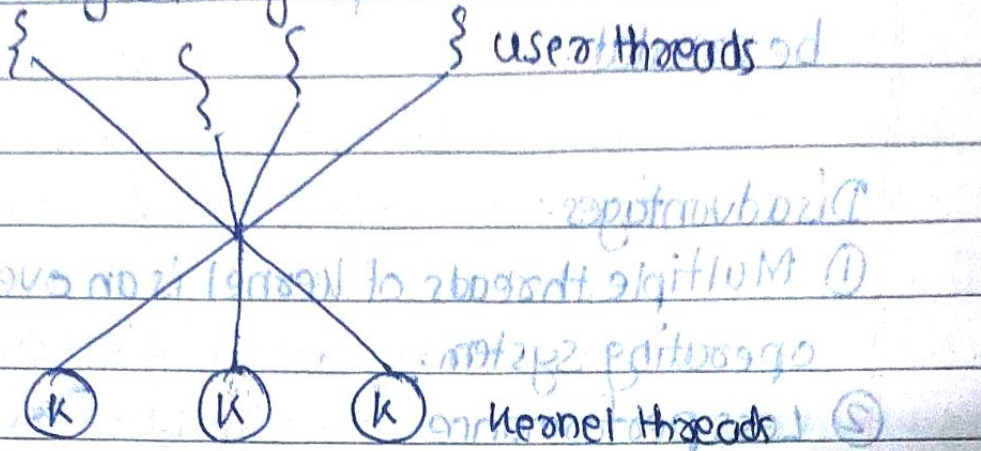
Advantages:-
① More concurrency
② Multiple threads can run in parallel
③ Less complication in the processing.

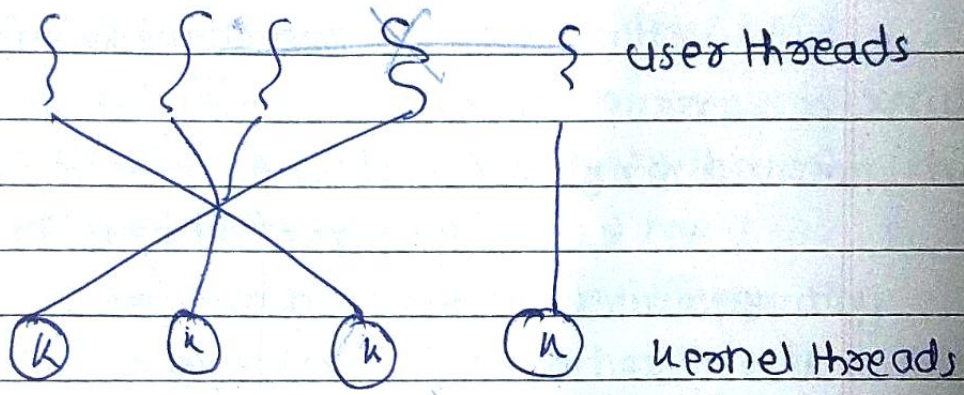
Disadvantages:-
① Everytime with user's thread, kernel thread is created.

② Kernel thread is an overhead.

③ It reduces the performance of the system.

④ Many to many model:





Two level model

It multiplexes many user level threads to a smaller or equal number of kernel threads. The number of kernel threads may be specific to either a particular application or to a particular machine.

Advantages:

① Many threads can be created as per user's requirement.

② Multiple kernel or equal to user threads can be created.

Disadvantages:

① Multiple threads of kernel is an overhead for operating system.

② Less performance.