Process Management

CPU executes a large number of program while its main concerned is the execution of user program, CPU is also needed for certain number of activities. These activities are called processes. Operating system is responsible for performing the following activities in connection with processes. This is called "process management" (process to manage.) Example—

(1) creation and deletion of processes,
(2) suspension/delay and resumption of processes etc.

Process

The notion (धारणा) of the process is centralised to the understanding of operating system. There are quite few definition presented in the literature but they are not perfect 'definition' as they appear.

Definition

The term 'process' was first used by designer of MULTICS in 1960. Since, then the term process used some what interchangeably with 'task' or 'job'.

(1) A program in execution (To perform to complete).
(2) A process is an instance of program.
(3) A process is the unit of work in the system.
(4) The entity assign to the processor.
(5) The Dispatchable unit.
(6) A process is a framework of set of sequential step for performing task.
(7) animated spirit of a procedure in a sequence.

Process States

A process state is indicative of the activity which is currently performed. Example-

w state: process is being created.
ady state: The process waiting for a chance to be allocated to C.P.U time. etc...

PROCESS STATES TRANSITION

(1) A process goes through various state for performing its task.

(2) The transition of a process from one state to another occur depending on the flow of execution of the process.

(3) It is not necessary for a process to undergo all the states.

(4) Various process states are-

a) New state ⇒ Process is being created.

b) Ready state ⇒ The process waiting for a chance to be allocated to CPU time.

c) Running state ⇒ When a process is being executed.

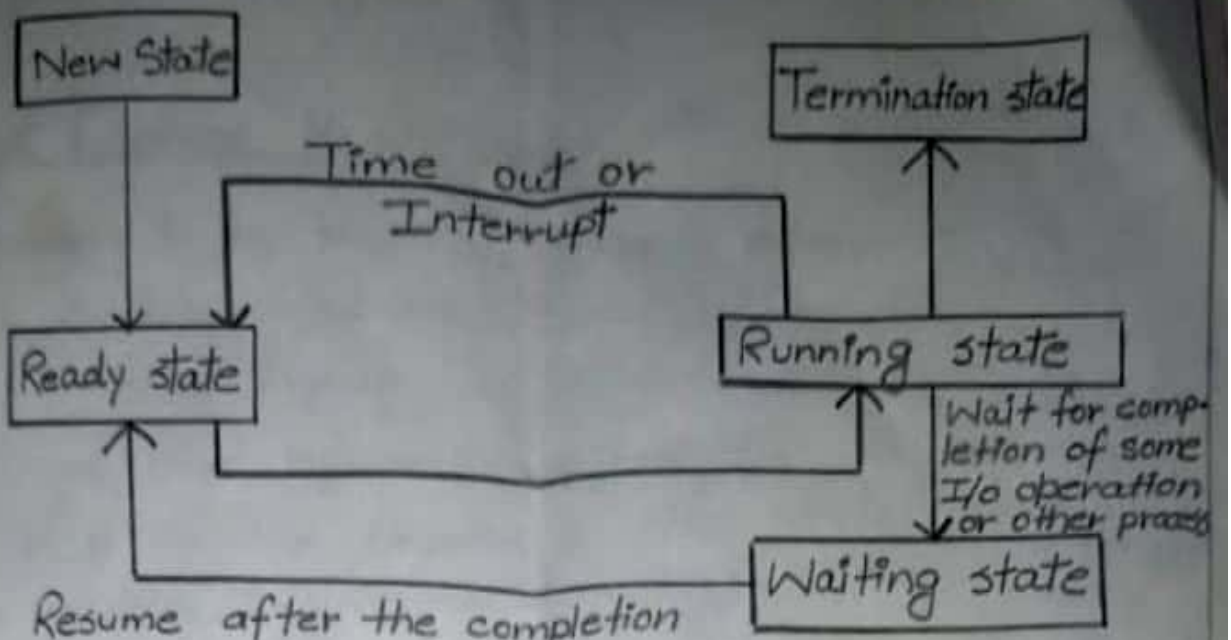
d) Waiting state ⇒ It is also called Block state. When a process waits for the termination of an I/o operation or another process.

e) Terminate state ⇒ The process has finished execution.

BLOCK DIAGRAM OF PROCESS TRANSITION:-

OR

TRANSITION OF A PROCESS FROM ONE STATE TO ANOTHER



A new process is admitted into a data structure called ready queue or ready state.

A queue is a data structure that stores all data in first-in-first-out manner. The process is inserted first in the ready queue and then it is sent for execution.

Each process assign a time slice for execution. The C.P.U executes the process at the front end of the ready queue and send to the running state. A time out occurs if the process is not completed within the time slice.

When a process is interrupted, the running process is sent back to the ready queue. When a process waits for the completion of an I/o task or another process, it is sent to the waiting state, this process state is marked as waiting. Then after completion of I/o task, the process goes back to the ready state.

When execution of a process is, the process goes to terminated state.

PROCESS CONTROL BLOCK (PCB)

(1) A process in an operating system is represented by a data structure called process control block.

(2) It is also known as process descriptor.

(3) It get the information about the process.

(4) A PCB is the location in the main memory, where various information regarding a process is stored.

(5) Each process has a single PCB.

(6) When the process is terminated, the PCB is released from the memory.

(7) The information contained in PCB about the specific process includes -

i) Current state ⇒ current state of process i.e. whether it is ready state, running state, waiting state or whatever.

ii) process ID ⇒ unique identification of the process in order to track "which is which" information.

iii) Link to parent process ⇒ A new process (i.e. child can be created from an existing process.) The existing process is called parent process.

iv) Link to child ⇒ (If it exists)

v) Priority of a process ⇒ The priority of a process (a part of CPU scheduling information.)

vi) A register save area.

vii) The processor it is running on.

The PCB is certain store that allows the operating system to locate key (बहुत जरूरी) information about the process.

Thus a PCB is a data structure or descriptor that defines the process of the O.S.

OPERATION ON PROCESS OR

O.S SERVICES FOR PROCESS CONTROL

(1) Operating system services are set of routines required for operating and managing the application.

(2) In an Operating system, categories services are available such as process control, file management, device management, memory management etc.

(3) operating system services are invoked/started by system call, system call provide an interface between process and operating system.

(4) O.S system call for process management or operations available in various O.S are such as

a) Create system call/operation.
b) Fork/Join system call/operation.
c) Execute system call/operation.
d) Delete system call/operation.
e) Wait/suspend system call/operation.
f) Resume system call/operation.
g) Abort system call/operation.

a) Create system call ⇒ create a new process with a blank PCB. The Operating system generates process ID and assigned it to the new process.

ther attributes (field of PCB) are copied to the PCB of the new process and process is transferred to the ready queue. A process can be created by another process.

b) Fork/Join system call ⇒ Fork system call making easy creation of a process by bifurcation of an existing process.

In other words, fork system call creates a duplicate process. The new process is the child of the existing process from which the fork system call has been invoked/started. The values in the various field of the existing processID of the PCB are copied to the PCB of the new process.

c) Execute system call ⇒ Execute a process from the running state.

d) Delete system call ⇒ Delete system call terminates a process. A process can be deleted by itself or from other process. After the deletion of a process, the Operating System releases and reclaim the resources such as file or memory space allocated to the deleted process. It also releases the PCB of the deleted process.

e) Wait system call ⇒ It is also known as suspend system call. The wait system call suspend a running process and waits for the completion of an I/O operation or another process. It is also called locking of a process.

Resume system call ⇒ It is also called wake-up system call. It is used for restoring a block process. A process cannot involved the resume system call itself because it must be in the running state to invoke a system call. A suspended process needs for another process for restoring.

8) Abort system call ⇒ It is similar to delete system call except the abort is invoked for forcefully (अत्यधिक) termination of a process. While executing the abort system call, the operating system perform similar task as delete but it save the information regarding the process. Operating system displays the information such as reason for aborting the process and values of the register at the time of aborting in a file on the computer system.

CRITICAL SECTION

<1> A process is in critical section if it executes code that manipulates shared data and resources.

<2> In other word, critical section is that overlapping portion of each process where shared data and resources are accessed.

<3> Critical section problem is to design a protocol to achieve cooperation among processes.

Each process should seek permission to enter its critical section.

<4> The code section that implements this request is called entry section.

<5> An exit section may follow critical section.

Remainder section → Remaining codes comes under remainder section.

General Structure of a Typical Process

```
Repeat {
        Entry section;
        Critical section;
        Exit section;
        Remainder section;
        until false.
}
```

(7) There are three necessary and sufficient requirement for critical section solution. These are as follows:

(a) Mutual Exclusion
(b) Bounded Waiting
(c) Progress

(a) Mutual Exclusion ⇒ It states that if one process is executing in its critical section, no other process can execute in that critical section.

(b) Bounded Waiting ⇒ It states that if a process makes a request to enter its critical section and before that request is granted there is limit number of time, other processes are allowed to enter that critical section.

(c) Progress ⇒ It states that a process cannot stop other processes for entering their critical section if it is not executing in its critical section.

V.V.I

RULES OF MUTUAL EXCLUSION

OR

MUTUAL EXCLUSION PRINCIPLE

(1) Resources like memory and certain input output cannot be used by more than one process at a time.

(2) The principle of mutual exclusion states that a process has unique access to shared resources in critical section.

(3) For a given resource among competing processes, one process can be in its critical section at an instant of time.

For Example- If a process is using a printer then no other process can use it until first process is over-come with its work.

SEMAPHORES

Semaphore is a variable with an integer value on which three basic operations are defined.

(1) Initialization - to non-negative value.

(2) Wait operation → Decrement semaphore value. A process executing wait is blocked if its value become negative.

Pseudocode for wait operation

```
wait(s)
while(s<=0)
{
//no-operation
    s--;
}
```

signal operation → Increment semaphore value.
A process blocked by wait operation is unblocked if value become positive.

Pseudocode for signal operation

```
signal(s)
{
  s++;
}
```

V.V.I

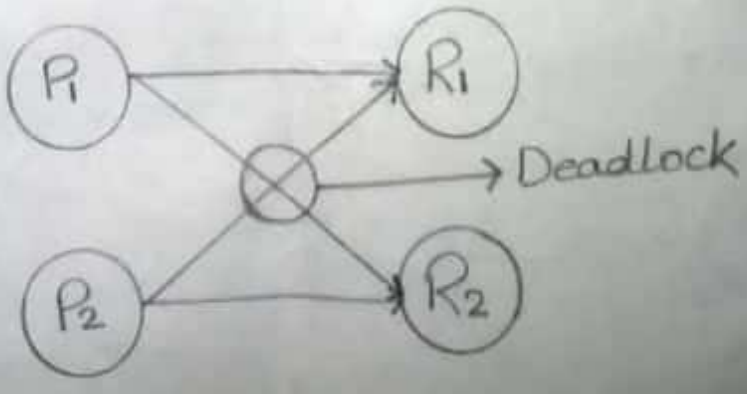
DEADLOCK

(1) In a multiprogramming O.S, resources such as I/O devices, CPU cycle and memory are used by multiple number of processes. There can be multiple number of instances of a resource.

(2) In a computer system deadlock occurs when two or more process are attempt to access the resource that the other process has locked. Hence, cannot be shared as a result. Each process wait for a resource that the other has locked and neither transaction can finished.

(3) A deadlock is a situation where two or more competing action are waiting for the other to finish and this neither ever does.

For Example-



resources, a CD-writer drive (i.e. R_1) and a printer (i.e. R_2) attached to a computer. To process P_1 and P_2 exist in the ready queue. Both the process P_1 and P_2 wants to read a document, write it back to the CD and get modified document printed. Both the processes P_1 and P_2 require both the resources, the CD-writer drive (i.e. R_1) and printer (i.e. R_2). The process P_1 acquire the CD-writer drive (i.e. R_1) and lock it in non-sharable mode. The process P_2 acquire the printer (i.e. R_2) and lock it in non-sharable mode. The process P_1 waits for obtaining the printer and process P_2 waits for obtaining the CD-writer drive. This led to a situation which is called a deadlock.

Necessary Conditions for Deadlock

Following are the conditions under which deadlock can occur:-

<1> Mutual Exclusion
<2> Hold-and-wait
<3> No-Preemption
<4> Circular wait

<1> Mutual Exclusion ⇒ One or more resources should be require in exclusive mode. A process 'P' has locked a resource. This means no other process can acquire the resource 'R'.

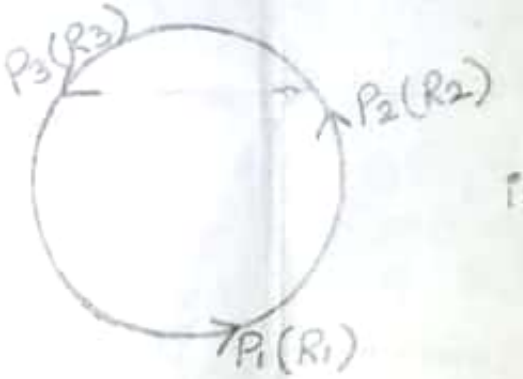
In other word, mutual exclusion means that a resource can be utilized by one process at a time.

Hold-and-wait ⇒ The process involved in a deadlock acquires atleast one process in exclusive mode and wait for atleast another resource that has been acquired by another process in exclusive mode.

<3> No-Preemption ⇒ processes do not releases the resource allocated to them without utilizing the resource. For Example ⇒ process 'P_1' hold the resource 'R_1' exclusively waits for the resource 'R_2' which is locked by process 'P_2' in exclusive mode.

The concept of no-preemption denotes that operating system cannot revoke the resource from the process forcefully.

<4> Circular wait ⇒



i = 1, 2, 3, 4 ..... n

A close chain or circular relationship exist among the processes involved in a deadlock.

For Example- There are three processes P_1, P_2 and P_3. The process P_1 is waiting for a resource that has been exclusively acquired by the process P_2, the process P_2 is waiting for a resource which is hold exclusively by the process P_3. The process P_3 is waiting for a resource which is acquired exclusively by the process P_1.

Deadlock Prevention

Deadlock prevention techniques allocate the resources in such a manner that atleast one of the four necessary condition of deadlock cannot occur. It denies atleast one of the condition acquired for occurrence of a deadlock. Deadlock prevention technique do not utilize a resource properly.

We have already discussed that there are four conditions to be satisfied for the occurrence of a deadlock. These conditions are:-

<1> Mutual Exclusion
<2> Hold-and-wait
<3> No-Preemption
<4> Circular wait

<1> Mutual Exclusion ⇒ The mutual exclusion principle is usually difficult to denied/displaced with, so one or more of the remaining conditions are to be prevented.

The mutual exclusion condition indicates locking of resources in exclusive mode, also known as non-sharable mode. An operating system can avoid blocking the resources in sharable mode if visible.

For Example- Two processes needs to read data from a file. The file is locking in sharable mode by two processes simultaneously. Locking of resource in sharable mode prevent waiting for a resource.

Some resources cannot be shared by multiple processes. For Example- Two process cannot simultaneously share a scanner.

• Hold-and-wait ⇒ The second condition for the occurence of deadlock is hold-and-wait. Operating system can avoid this by adopting a strategy that a process must request for all resources of the same time. This strategy can be implemented in two ways:-

i) A process can request all the resources it need when execution begin.

ii) Secondly, a process must release all the resources it presently hold before requiring any other required resource.

For Example- A process 'P' require resources R_1, R_2, R_3 and R_4 in exclusive mode in the sequence. R_3 and R_1 simultaneously at the beginning of the execution. R_1 and R_2 resources, then R_3 simultaneously and lastly R_1, R_4 simultaneously.

When applying the first strategy, the process 'P' acquires all the four resources simultaneously before it start execution.

When applying the second strategy, the process 'P' need not acquire all the four resources at the beginning. The process 'P' acquire the resources R_1 and R_2 at the time of start of execution. The process 'P' request the operating system for acquiring the resources R_1, R_2, R_3. The process 'P' works and releases all i.e. R_1, R_2, R_3 and send request for R_1 and R_4.

(3) No-Preemption ⇒ The third condition is no preemption. An operating system can adopt a policy to avoid this condition. Operating system stop the resources if it is responsible for deadlock and starting for further requesting the resource.

(4) Circular wait ⇒ The fourth condition of occurrence of deadlock is circular wait. In this approach, system resources are divided into different classes j (where $j = 1, 2, \ldots\ldots n$).

Deadlock are prevented by requiring all processes to request and acquired the resources in strictly increasing order of specified system resource class. Once a process acquire a resource belonging to the class j (i.e. C_j), it can only request resource of C_{j+1} or higher thereafter.

VNI

DEADLOCK AVOIDANCE

(1) In deadlock avoidance, operating system requires additional information about which process requires which resource in advance to avoid a deadlock.

(2) The fundamental concept is that operating system only entertain those request for resource by a process that will not lead to a deadlock.

(3) Banker algorithm is a theoretical approach to avoid deadlock in resource scheduling.

(4) Banker algorithm adopts a strategy. The name banker depicts (comes from) the similarity of concept of the field of banking.

It is usually explained using an analog
i) The customer are process.
ii) Money is the resource.
iii) Bank is the Operating system.
iv) Each customer has credit limit.

There are two part of Banker algorithm
I. Safety algorithm
II. Resource Request algorithm

(I.) Safety Algorithm
Safety algorithm determines whether or not a system is in a safe state.

Step 1: Let Work and Finish, be arrays of length m and n respectively.
Initialize: Work = Available and
Finish[i] = False; for i=1, 2, ... n

Step 2: Find an i such that both
Finish[i] = False
Needi ≤ Work

Step 3: If no such i exists goto step 4; otherwise
Work = Work - Allocationi
Finish[i] = True
goto step 2

Step 4: if Finish[i] = True ∀, or, for all i
then the system is in safe state.

(II.) Resource Request Algorithm
The resource request algorithm determines whether a resource request can be safely granted.
Step 1: Let Requesti be the request array for process Pi.

Step 2: If Requesti > Needi that means it get errors because resource claim is maximum.

Step 3: If Requesti > Available then block process Pi for resource request that available resource.

Step 4: Here the system pretend to allocate the requested resources to the process Pi by modifying the state as follows:

Available = Available - Requesti

Needi = Needi - Requesti

Allocationi = Allocationi + Requesti

If the resulting resource - allocation state is safe, the transaction is completed and process Pi is allocated its resources. However, if the new state is unsafe, then Pi must wait for Requesti and the old resource - allocation state is restored.

DEADLOCK DETECTION AND RECOVERY

(1) Another approach of deadlock management is detection and recovery.

(2) In this approach, the system is not prevented from occurring a deadlock.

(3) Operating system periodically searches and analyses whether a deadlock occurs

(4) This approach of deadlock management has two phases - Detection and Recovery of a deadlock.

(5) The algorithm to detect deadlock in an operating system can use data structure similar to that

of data structure(array) used in the context of deadlock avoidance.

(6) In this algorithm, data structure 'Allocation', 'Claim', 'Available' and 'Flag' are used.

(7) Flag is the array or vector of length equals to (=) the number of processes in the system.

(8) Flag is used to mark and unmark a process.

Flag[x] = True, means process Px is marked.

Flag[y] = False, means process Py is unmarked.

(9) An operating system verifies after a certain interval whether a deadlock has occurred.

(10) This algorithm find a sequence of execution of processes that doesnot lead in unsafe state.

Step 1: Initialize : Flag[i] = False ; for i = 1, 2, ... n.
; denotes the number of processes.

Step 2: Find a process Pi such that
Flag[i] = False and Claim$_i$ ≤ Available$_i$; otherwise
goto step 6 if no such process found.

Step 3: update Available vector because of termination of process. Operating system revoke to control all the resources held by Pi process.

Available$_i$ = Available$_i$ + Allocation

Step 4: Mark the process Pi, Flag[i] = True

Step 5: goto step 2

Step 6: Flag[i] = True, process Pi is marked and no deadlock is occurred.

Operating system can adopt a strategy for deadlock recovery. It states that all the processes involved in deadlock can be terminated forcefully.