

Operating System: - It is a system software that controls the execution of an application program and acts as an interface between computer hardware and users.

❖ Objectives of Operating Systems:-

(1) Convenience:- An OS makes computer system more convenient to use.

(2) Efficiency:- An OS allows the computer system resources to be used in an efficient manner.

(3) Ability to evolve:- An OS should be designed in such a way that as to permit introduction of new system functions without interfering with existing OS.

❖ Services(Functions) of OS:-

(1) Program Development:- The OS provides a variety of services and facilities like editors and debuggers, that assist the programmer in creating programmes. These services are not core services but utility services.

(2) Program Execution:- A number of steps need to be performed to execute a program such as

(a) Program and data must be loaded in main memory.

(b) I/O devices and files must be initialized.

(c) Other resources must be prepared.

OS provides scheduling activities of above mentioned tasks.

(3) Access to I/O device:- Each I/O device requires its own set of instructions or control signals for operation. The OS provides uniform interface that hides these technical details from users so users can access them with simple read and write operations.

(4) Controlled Access to files:- For file access, OS must reflect a detailed understanding of nature of I/O devices (i.e. disk or tape) and the structure of data contained in the file so it is possible for users to access them in a controlled fashion. In case of a system with multiple users, OS may provide protection mechanisms to control access to the files.

(5) System access:- For a public or shared system, OS controls the access of system as a whole or to specific system resources. The access function must provide protection of resources and data from unauthorized users.

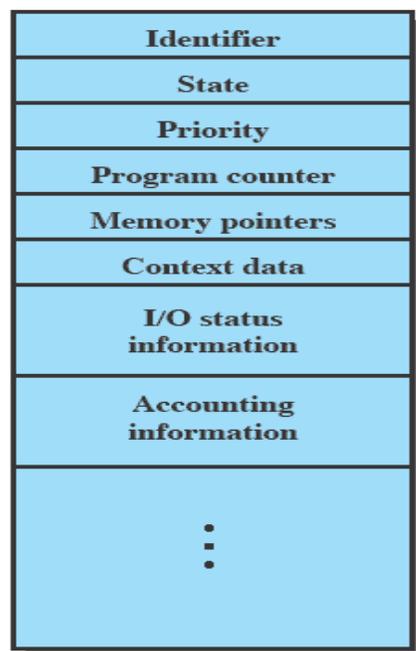
(6) Error detection and Response:- A variety of errors can occur while computer is running. These includes hardware errors (i.e memory error, device failure) and software errors(i.e division by zero, attempt to access forbidden memory location).In each case OS must provide a response that clears the error condition with the least impact on running program.

(7) Accounting:- A good OS collect usage data for various resources and monitor the performance parameters such a response time. For any system this information is useful for future enhancement of the system.

❖ Process

- *A program in execution*
- An instance of a program running on a computer
- The entity that can be assigned to and executed on a processor
- A unit of activity characterized by the execution of a sequence of instructions, a current state, and an associated set of system resources

❖ PCB (Process Control Block)



- Identifier:- Unique identifier associated with this process, to distinguish it from other processes.
- State:- Stores the state of process (i.e running, not running).
- Priority:- Store the priority of process.
- Program Counter:- It stores the address of next instruction to be executed.
- Memory Pointers:- Includes the pointer to the program code and data associated with this process plus any shared memory block.

Figure 3.1 Simplified Process Control Block

- Context Data:- These are the data that are present in registers of processor while the process is executing.
- I/O Status Information:- Includes I/O requests,I/O devices assigned to this process, a list of files in use by process.
- Accounting :- It includes statistical information like processor and clock time to be used,time limits etc.

- **Process States**

Process Trace: - Listing of sequence of instructions that execute for a process.

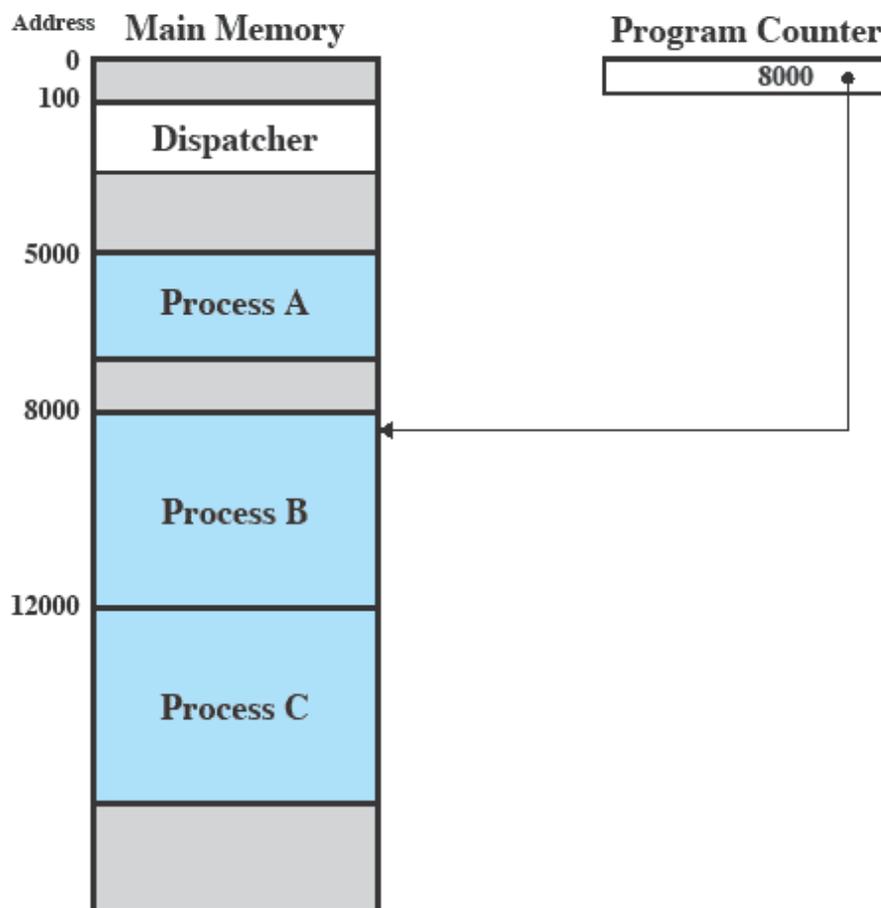


Figure 3.2 Snapshot of Example Execution (Figure 3.4) at Instruction Cycle 13

5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004		12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011

(a) Trace of Process A (b) Trace of Process B (c) Trace of Process C

5000 = Starting address of program of Process A
8000 = Starting address of program of Process B
12000 = Starting address of program of Process C

Figure 3.3 Traces of Processes of Figure 3.2

1	5000			27	12004		
2	5001			28	12005		
3	5002					-----	Timeout
4	5003			29	100		
5	5004			30	101		
6	5005			31	102		
		-----	Timeout	32	103		
7	100			33	104		
8	101			34	105		
9	102			35	5006		
10	103			36	5007		
11	104			37	5008		
12	105			38	5009		
13	8000			39	5010		
14	8001			40	5011		
15	8002					-----	Timeout
16	8003			41	100		
		-----	I/O Request	42	101		
17	100			43	102		
18	101			44	103		
19	102			45	104		
20	103			46	105		
21	104			47	12006		
22	105			48	12007		
23	12000			49	12008		
24	12001			50	12009		
25	12002			51	12010		
26	12003			52	12011		
						-----	Timeout

100 = Starting address of dispatcher program

Shaded areas indicate execution of dispatcher process;

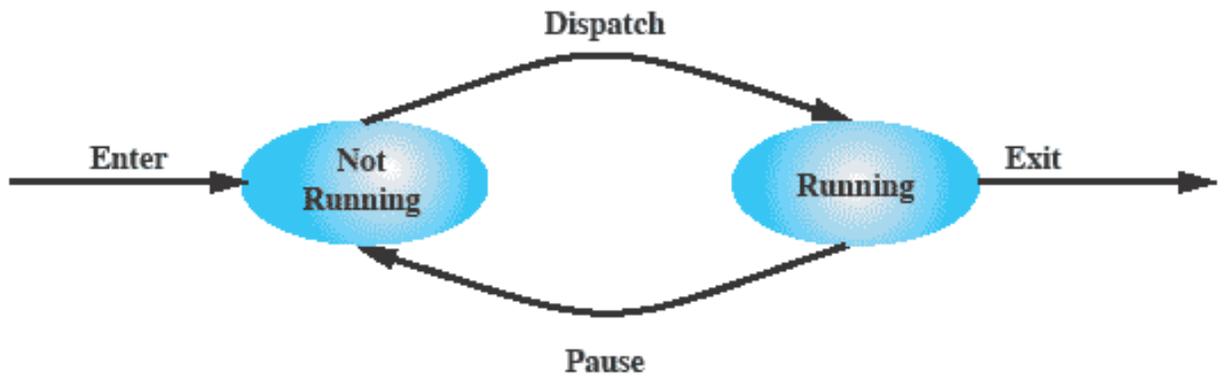
first and third columns count instruction cycles;

second and fourth columns show address of instruction being executed

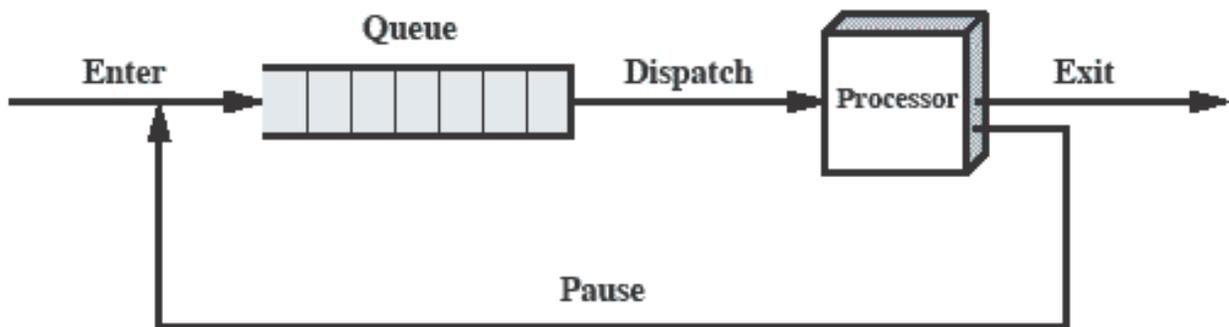
Figure 3.4 Combined Trace of Processes of Figure 3.2

Dispatcher:- It is a program that switches processor from one process to other.

❖ Two State Process Model



(a) State transition diagram



(b) Queuing diagram

→States:-

(1) Not Running :- Process is in main memory and available for the execution.

(2)Running:- Process is in main memory and currently executed by processor.

→Transitions:-

(1) Not Running→Running:- When it is the time to select the process to run, the OS select one of the processes from not running state into running state.

(2)Running→Not Running:- When process reaches to the maximum allowable time , the process is transferred into Not Running state.

→Limitations

(1) Main memory constraint is not considered.

(2) I/O event is not considered by this model.

❖ Five states process model

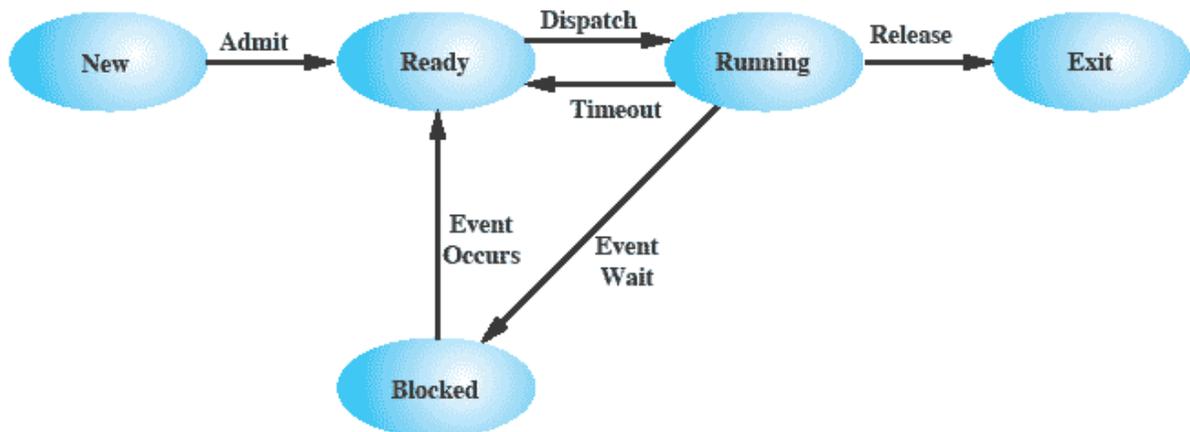
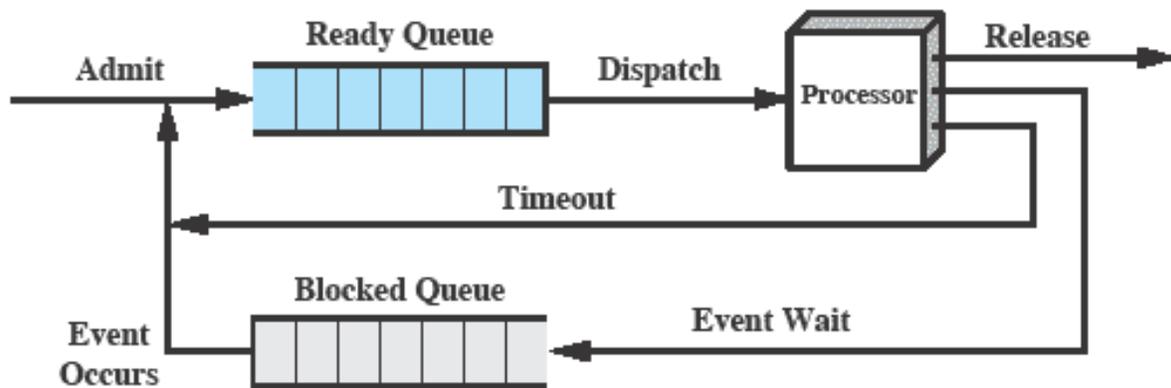
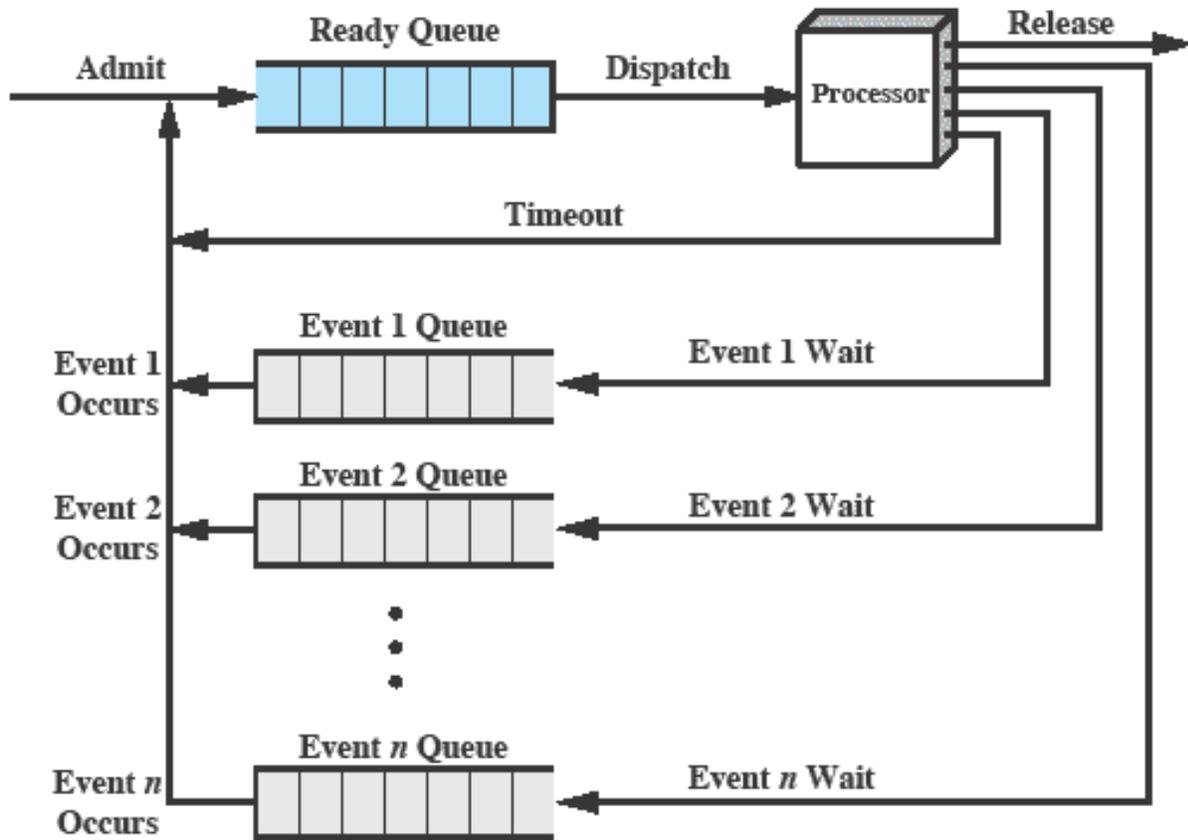


Figure 3.6 Five-State Process Model





(b) Multiple blocked queues

→States

- (1) **New**:- Process is in secondary memory and not yet admitted to pool of executable processes.
- (2) **Ready**:- Process is in main memory and available for the execution.
- (3) **Running**:- Process is in main memory and currently executed by processor.
- (4) **Blocked**:- Process is in main memory and waiting for an I/O event to occur.
- (5) **Exit**: Process is in secondary memory and release from pool of executable processes.

→Transitions

- (1) **New→Ready**:-When OS is prepared to take an additional process, the OS will move a process from New state to Ready state.

(2) **Ready→Running**:- When it is the time to select the process to run, the OS select one of the processes from ready state into running state.

(3) **Running→Ready**:- When process reaches to the maximum allowable time , the process is transferred into Not Running state.

(4) **Running→Blocked**:- When any running process requires I/O , the OS will put it in blocked state.

(5) **Blocked→Ready**:- When blocked process got an I/O , the OS will move it into Ready state.

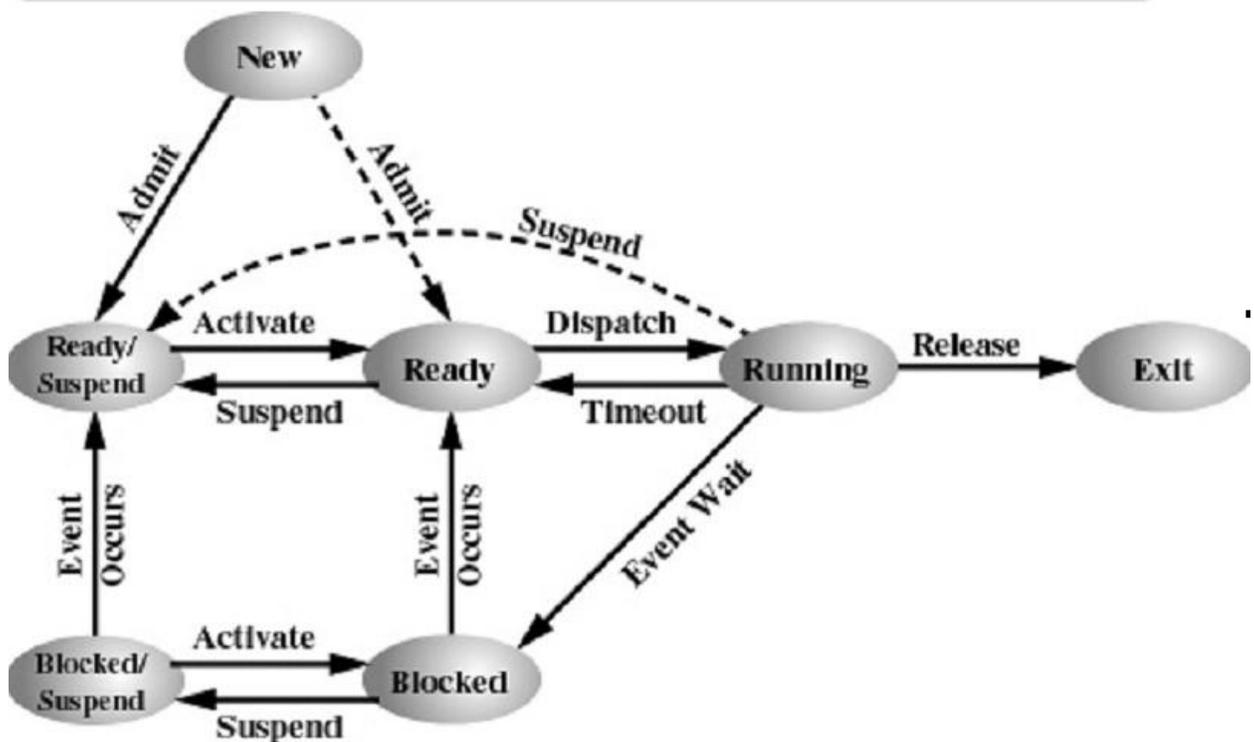
(6) **Running→Exit**:- When running process is terminated due to any reason, the OS will move it into Exit state.

→Limitation

When all ready processes got blocked and there is no space in main memory, the system will not able to accommodate any new process for execution as a result processor remains in ideal state.

- ❖ **Swapping** : The OS needs to release sufficient main memory to bring new process in main memory for execution.

❖ Seven states process model



→States

(1) New:- Process is in secondary memory and not yet admitted to pool of executable processes.

(2)Ready:- Process is in main memory and available for the execution.

(3) Running:- Process is in main memory and currently executed by processor.

(4)Blocked:- Process is in main memory and waiting for an I/O event to occur.

(5)Exit: Process is in secondary memory and release from pool of executable processes.

(6) Bloked/Suspend: Process is in secondary memory and waiting for an I/O event.

(7) Ready/Suspend: Process is in secondary memory and available for execution.

→Transitions

(1)New→Ready / New-Ready/Suspend :- When new process is created , it can be either added to Ready state or Ready/Suspend state. In either case, the OS needs to create PCB and address space to the process. It is preferable to OS to perform this house keeping activities at early time, so large pool of processes can maintain. With this strategy , there would be often insufficient room in main memory for new process so process is put in Ready/Suspend state.

2) Ready→Running:- When it is the time to select the process to run, the OS select one of the processes from ready state into running state.

(3)Running→Ready:- When process reaches to the maximum allowable time , the process is transferred into Not Running state.

(4)Running→Blocked:- When any running process requires I/O , the OS will put it in blocked state.

(5)Blocked→Ready:- When blocked process got an I/O , the OS will move it into Ready state.

(6)Running→Exit:- When running process is terminated due to any reason, the OS will move it into Exit state.

7) Blocked→Blocked/Suspend:- If there are no ready processes and main memory is full, OS swapped out at least one blocked process to make a room in main memory.

(8) Blocked/Suspend→Ready/Suspend:- OS moves blocked/suspend process to ready/suspend when block/suspend process got an I/O.

(9)Blocked/Suspend→Blocked:- This transition shows poor design of model. However sometimes, there is a process in blocked/suspend queue has a higher priority than any other processes in ready/suspend

state and OS believes that I/O event will occur soon for the process, in this case OS moves Blocked/Suspend process into Blocked.

(10) Running → Ready/Suspend:- Generally running process is put in ready state by OS but however when any higher priority process in blocked/suspend state got unblock, the OS preempt any running process to accommodate execution to that higher priority process.

(11) Ready → Ready/Suspend:- Generally blocked process is suspended by OS to make a room in main memory, however it may be necessary to suspend a ready process if it is the ultimate solution to free up large block of main memory.

(12) Ready/Suspend → Ready:- When there are no ready processes and OS will need to bring one in for continuation, any ready/suspend process is placed in ready state.

❖ Process Vs. Thread

Process	Thread
(1) Program in execution.	(1) Execution path within a process.
(2) Process has its own address space.	(2) Thread share an address space of process.
(3) One process requires Inter Process Communication (IPC) to communicate with other process.	(3) Direct communication is possible among threads.
(4) One process cannot control other process.	(4) One thread can control all other threads of same process.
(5) Process creation is difficult.	(5) Thread creation is simple.
(6) Overhead associated with process is more in comparison of thread.	(6) Overhead associated with thread is less in comparison of process.