Process Model

Modern operating systems introduce concept of process to differentiate it from a program. A list of instructions in a programming language is a program, which is static. Once a program is placed in computer memory, and allocated resources it becomes a process. A process is program in action. Information regarding a process (Process-ID, Process status, resources allocated, PC, Special registers values, stack, data section, etc.) is maintained in a Process Control Block (PCB). A process is dynamic entity and goes through different states during its life cycle from creation to completions as listed below.

New/Creation:	The process is being created.
Ready:	The process is ready to get a processor when available.
Running:	Instructions are being executed, i.e., process is using CPU.
Blocked:	The process is waiting for some event to occur. Completion of an IO operation.
Terminated:	The process has completed. Resources of the process are released.

A process could be CPU bound or IO bound. If a process spends more time in using CPU during its lifetime, it is called **CPU bound** process. If a process spends more time in performing IO operations during its lifetime, it is called **IO bound** process.
A process model (Figure-1) is created to illustrate different states a process goes through during its lifecycle.

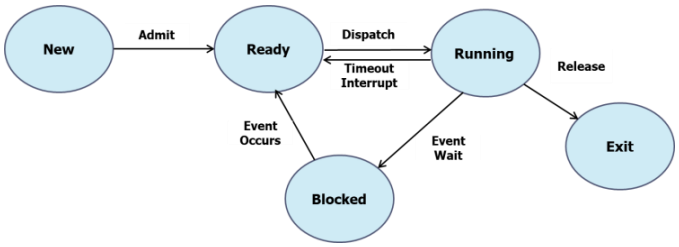

Figure-1: General Purpose Process Model

Operating system maintains lists/queues for resources allocation, and a process waits in a queue before a resource is allocated. We can use this 5-state process model to illustrate resources utilization by different types (Batch, Multi-Programming, Time-Sharing) of operating systems.

Process Model of Batch OS
In Batch OS, there is only one user process at any given time. It means there will be only one entry in different queues of the system. Process model of Batch OS is depicted in Figure-2.

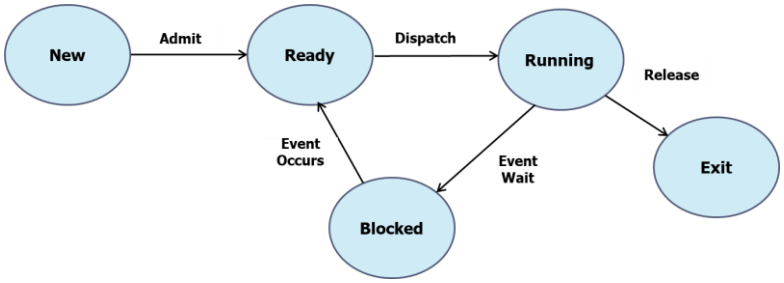

Figure-2: Process Model of Batch OS

Since there is only one process, initially Ready list will have that entry. Running and waiting lists are empty. Once CPU is allocated to that process, running list will have that entry. When the process initiates an IO operation, process status is changed from Running to Waiting. While process is in waiting state, running state is empty, i.e. CPU is idle. While process is in Running state using CPU, no IO operation is performed, and IO devices are idle. We can easily visualize, how one process is moved through different states, and how at any given time either CPU, or IO devices are underutilized.

Process Model of Pure Multiprogramming OS

To overcome the underutilization of resources, multiprogramming OS was designed. In this case more than programs are memory resident, and memory and CPU are shared among these processes. Initially pure multiprogramming systems were designed, where a process once allocated CPU (in running state), it keeps using CPU, until it voluntarily leaves CPU, either initiating an IO operation (moving to waiting state) or exit the running state due to completion. This CPU scheduling is non-preemptive. If we have mixed of CPU bound and IO bound jobs, then there is possibility that CPU bound processes can monopolize CPU usage and during that time IO devices are also underutilized. Process model of pure multiprogramming system is shown in Figure-3. There is not much difference in static models of batch OS and pure multiprogramming OS, in case of pure multiprogramming at any given time there will be more than entries in ready and blocked states.

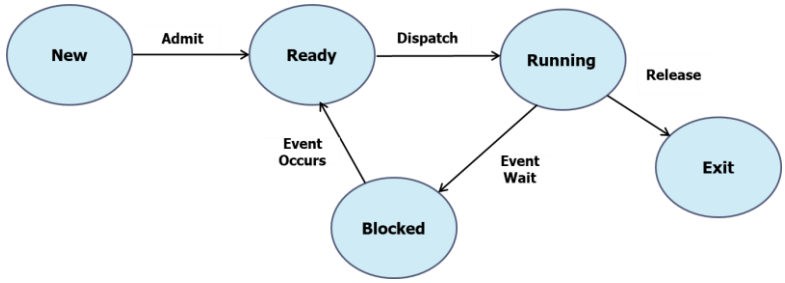

Figure-3: Process Model of Pure Multiprogramming OS

We can visualize resources utilization by placing different processes in different states, since more than one processes are active. Initially a few processes are in ready list, one from the ready list will be allocated CPU by the scheduler. That process is now in running state, if that process is IO bound process, in a short time, it will initiate an IO operation, and change its state from running to waiting. Due to CPU and IO devices speed disparity that process will be in wait state for a while. Meanwhile due to availability of CPU, another process from ready list will be allocated CPU and its state being changed to running. If that process is CPU bound, it will remain in running state. At this stage we can visualize through the process model that CPU and IO devices are busy. However, a CPU bound process can monopolize CPU usage. Meanwhile the process in wait state will change its state as soon as IO operation initiated by the process is completed. The process will be in ready list. At this stage we can visualize, IO devices being idle.

Process Model of Multiprogramming with Priorities OS

To overcome the CPU monopolization by CPU bound processes, which have poor utilization of IO devices, multiprogramming operating system with priorities was designed. Here processes are assigned priorities at process creation time. To simplify our description of the system, we consider two types of priorities (low and high). The system can create

separate ready lists for low and high priority processes. In this case the CPU scheduling is preemptive (once CPU is allocated to a process, due to certain conditions, OS can preempt and take back CPU and allocate it to another process. The process model of multi-programming OS with priorities depicted in Figure-4.

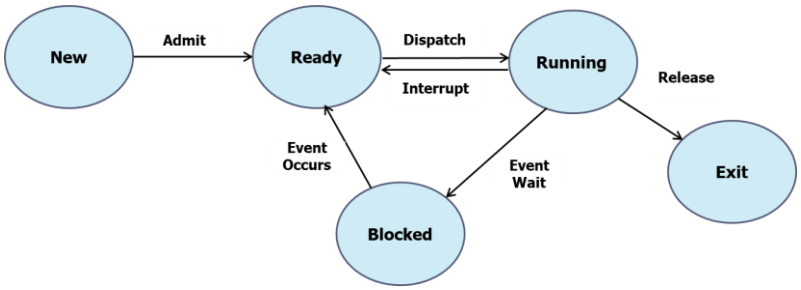

Figure-4: Process Model of Multiprogramming with priorities OS

First, we visualize working of multi-programming OS with priorities through process model. Initially a few processes (priority: low or high assigned) are in ready list, one from the ready list will be allocated by the scheduler. Assume the process in running state has low priority. Meanwhile a process with a high priority joins ready list. At this stage, scheduler will preempt and stop the execution of process in running state, make a context switch by placing that process in ready list and allocating CPU to high priority process by changing its state to running. As described in previous sections, we have CPU bound processes and IO bound processes. If you are asked to assign a high priority to IO bound or CPU bound processes, which processes you will assign high priority. Verify your choice by visualizing the process model. You can also draw the process model where system has separate ready lists for low and high priority processes along with separate waiting lists for low and high-speed devices.

Process Model of Time-Sharing OS

Time-Sharing OS is multiprogramming OS with interactivity. More than one program is memory resident, using computing resources. Using a preemptive scheduling, CPU is allocated to a process for a fixed time slot. At the expiry of time slot, CPU is taken from that process and its status is changed to Ready state. Another process which is at head of the Ready list is allocated CPU and its status is changed to Running. A process may initiate an IO operation or is terminated before expiry of time slot, in this case, CPU is allocated to another process from the Ready list. The process model of Time-Sharing OS is depicted in Figure-5.

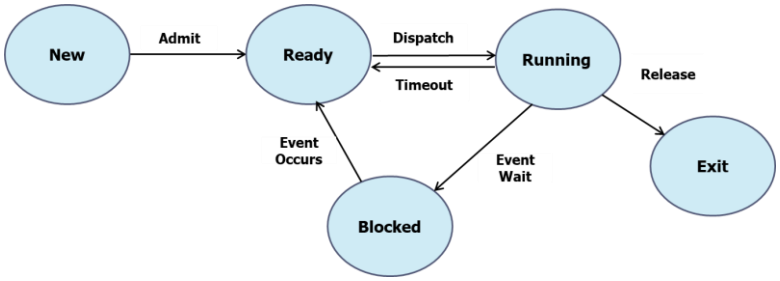

Figure-5: Process Model of Time-Sharing OS

You can easily see the process models of Time-Sharing OS and Multiprogramming OS with Priorities are identical. Only difference is in preemptive policy being implemented. In case of Multiprogramming OS with Priorities, preemption is made at the arrival of a high

priority process in ready list. Whereas in Time-Sharing OS, preemption is made at the expiry of time slot.

Process Model of General-Purpose OS

In main frame computers era, computer installations were managed by a general-purpose OS providing services to different types of applications. In such environments, OS is managing some jobs in batch mode, some jobs are executed in multi-programed mode, and others interactive jobs are managed in timesharing mode. In that case a process goes through different states and the process model is depicted in Figure-6.

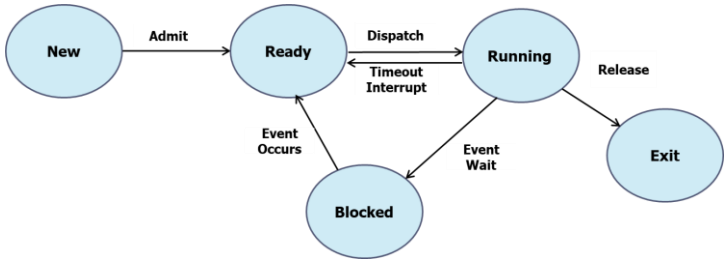

Figure-6: Process Model of General-Purpose OS

In this model, batch, multi-programmed and timesharing operating systems processes can go through different states.

Network OS

You may recall from history of operating systems section that Network OS manages, distributed system (loosely coupled independent computers, with its own local OS). Since resources are shared in an explicit way, and each unit's local OS is responsible for management of its resources. For each unit, process model will depend on OS type.

Distributed OS

You may recall from history of operating systems section that Distributed OS manages, distributed system like one operating system managing all distributed resources without users knowing, where programs are stored, and where executed. For one Ready list will be having links to multiple Running state, and one Running state will be linked to multiple Waiting states.

Process Models for Network OS and Distributed OS

Try to draw process models for homogeneous and heterogeneous Network OS and visualize flow of a process through different states. Similarly draw process model for Distributed OS, by assuming number of units with varying CPU power, Memory size, and IO devices are being managed by OS and allocated to user processes.