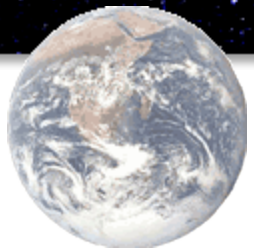
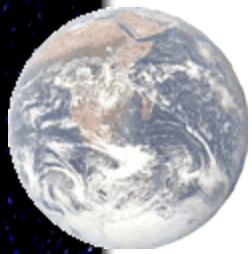
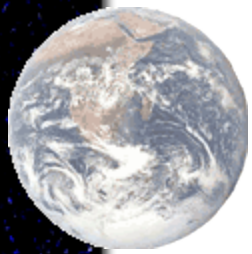


# Process Description and Control



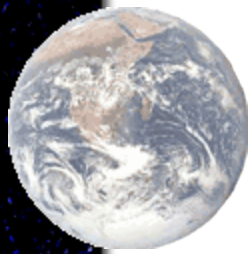
# Requirements of an Operating System

- Interleave the execution of multiple processes to maximize processor utilization while providing reasonable response time
- Allocate resources to processes
- Support interprocess communication and user creation of processes



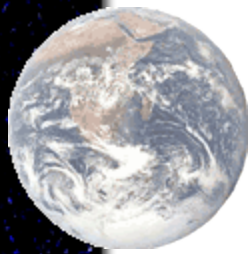
# Concepts

- Computer platform consists of a collection of hardware resources
- Computer applications are developed to perform some task
- Inefficient for applications to be written directly for a given hardware platform
- Operating system provides a convenient to use, feature rich, secure, and consistent interface for applications to use
- OS provides a uniform, abstract representation of resources that can be requested and accessed by application



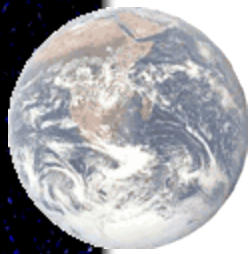
# Manage Execution of Applications

- Resources made available to multiple applications
- Processor is switched among multiptle application
- The processor and I/O devices can be used efficiently



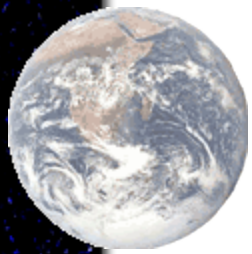
# Process

- 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 instructions



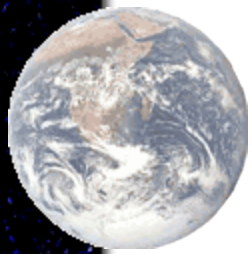
# Process Elements

- Identifier
- State
- Priority
- Program counter
- Memory pointers
- Context data
- I/O status information
- Accounting information

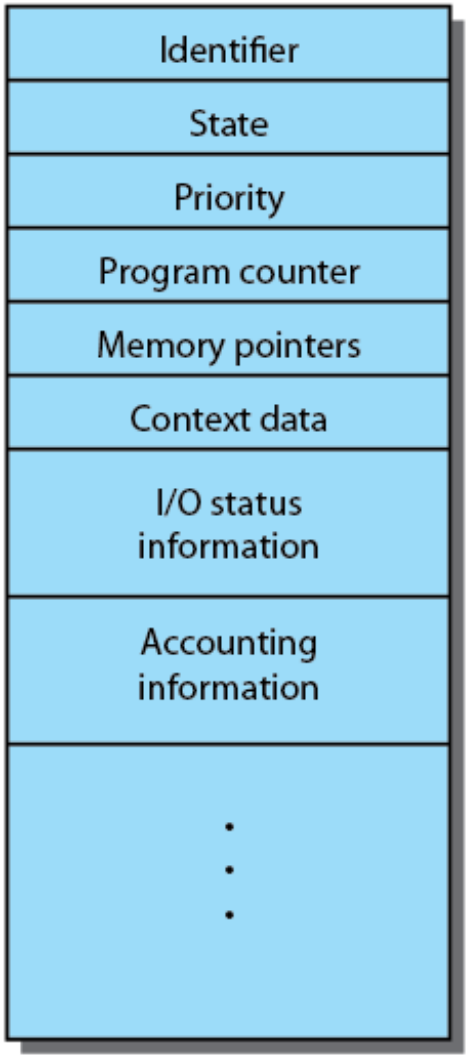


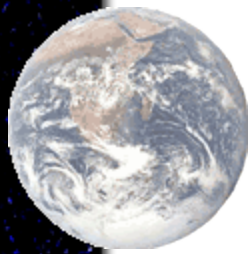
# Process Control Block

- Contains the process elements
- Created and manage by the operating system
- Allows support for multiple processes



# Process Control Block

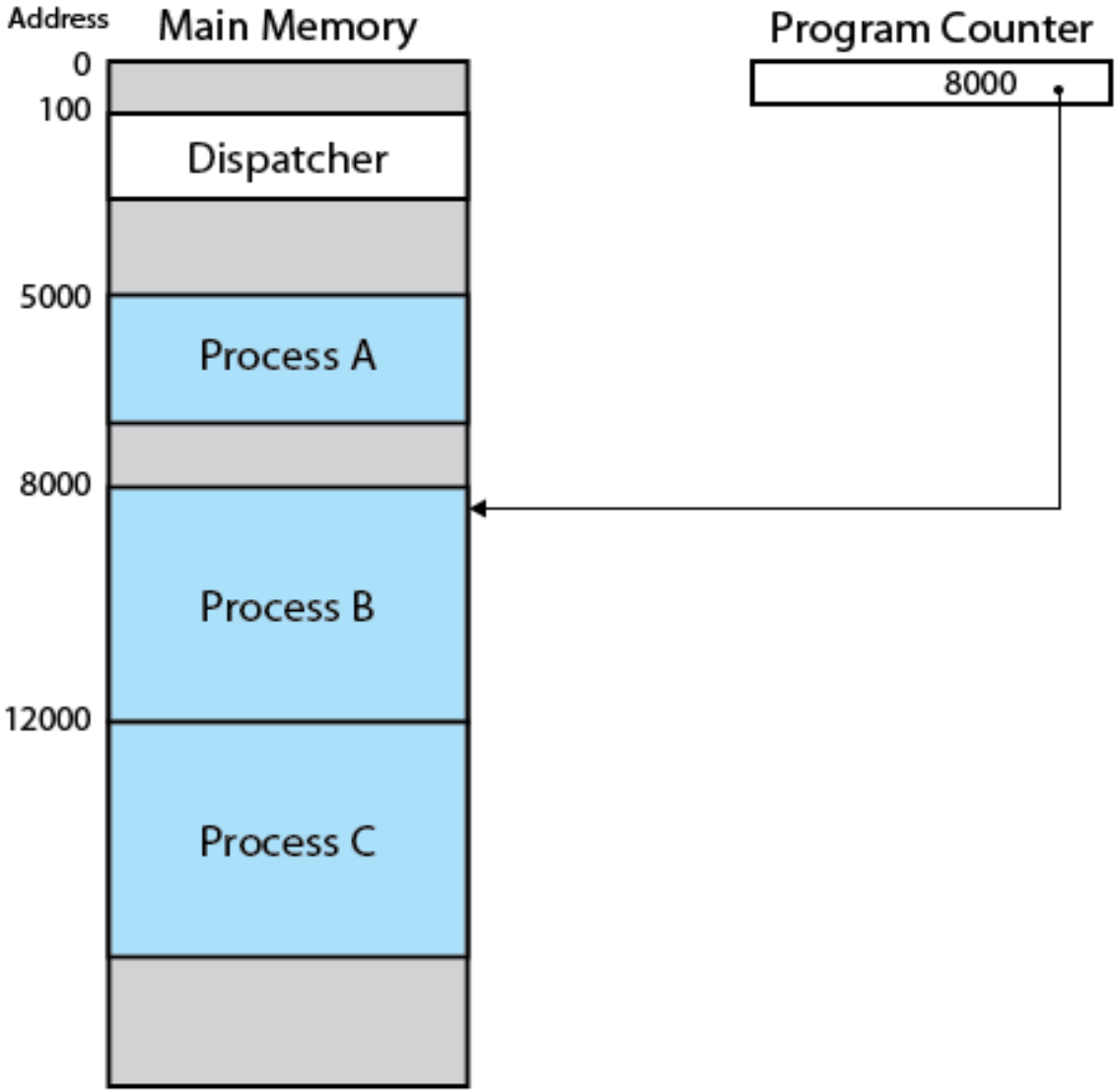


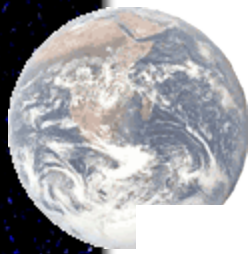


# Trace of Process

- Sequence of instruction that execute for a process
- Dispatcher switches the processor from one process to another

# Example Execution

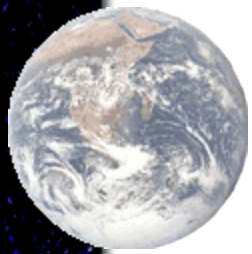




# Trace of Processes

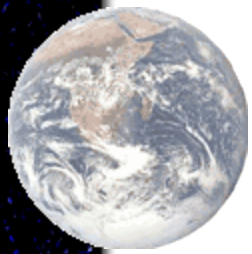
| (a) Trace of Process A | (b) Trace of Process B | (c) Trace of Process C |
|---|---|---|
| 5000 | 8000 | 12000 |
| 5001 | 8001 | 12001 |
| 5002 | 8002 | 12002 |
| 5003 | 8003 | 12003 |
| 5004 | | 12004 |
| 5005 | | 12005 |
| 5006 | | 12006 |
| 5007 | | 12007 |
| 5008 | | 12008 |
| 5009 | | 12009 |
| 5010 | | 12010 |
| 5011 | | 12011 |

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



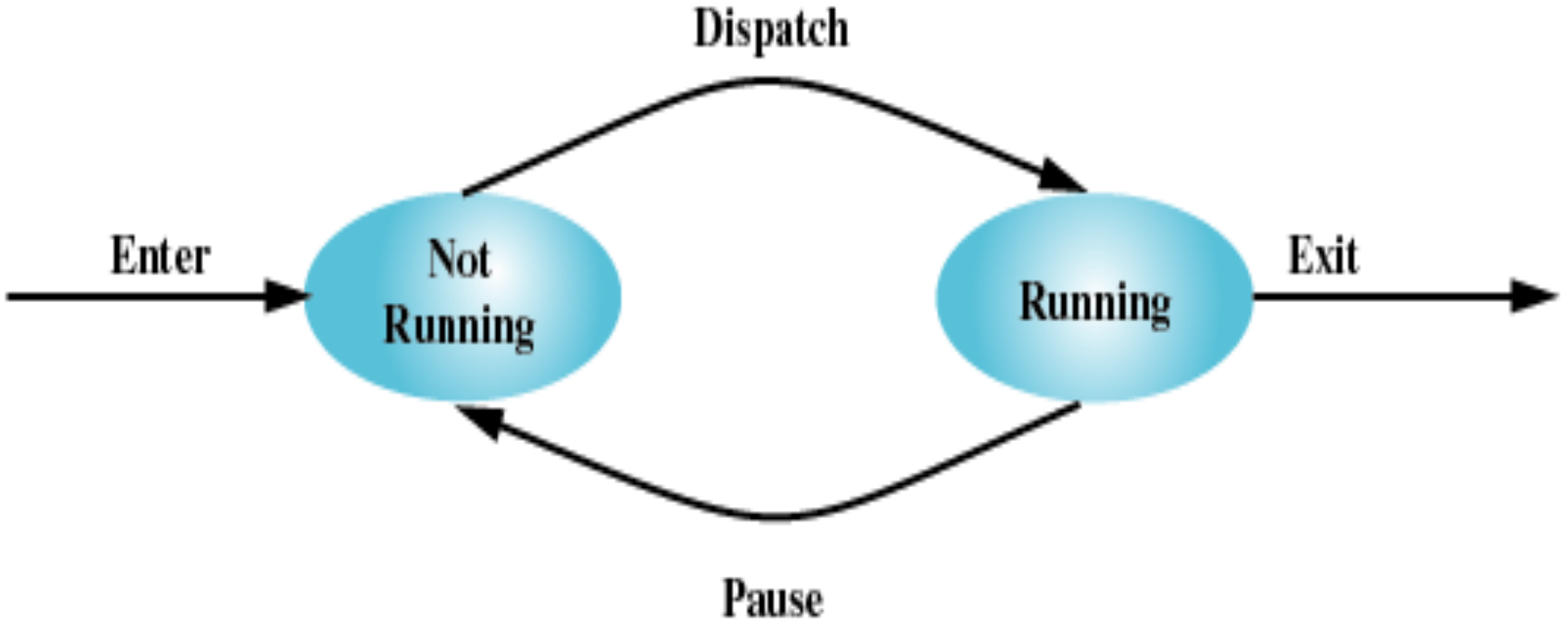
| | |
|---|---|
| 1 | 5000 |
| 2 | 5001 |
| 3 | 5002 |
| 4 | 5003 |
| 5 | 5004 |
| 6 | 5005 |
| ---------- | Time out |
| 7 | 100 |
| 8 | 101 |
| 9 | 102 |
| 10 | 103 |
| 11 | 104 |
| 12 | 105 |
| 13 | 8000 |
| 14 | 8001 |
| 15 | 8002 |
| 16 | 8003 |
| ---------- | I/O request |
| 17 | 100 |
| 18 | 101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

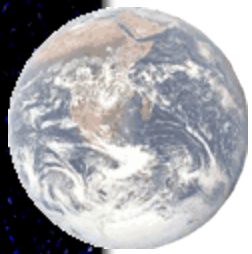
100 = Starting address of dispatcher program



# Two-State Process Model

- Process may be in one of two states
- Put in Not running queue when created

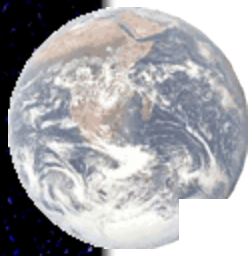




# Process Creation

**Table 3.1 Reasons for Process Creation**

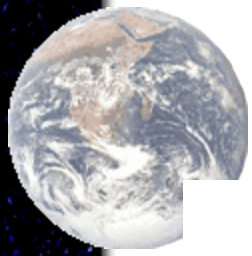
| | |
|---|---|
| New batch job | The operating system is provided with a batch job control stream, usually on tape or disk. When the operating system is prepared to take on new work, it will read the next sequence of job control commands. |
| Interactive logon | A user at a terminal logs on to the system. |
| Created by OS to provide a service | The operating system can create a process to perform a function on behalf of a user program, without the user having to wait (e.g., a process to control printing). |
| Spawned by existing process | For purposes of modularity or to exploit parallelism, a user program can dictate the creation of a number of processes. |



# Process Termination

**Table 3.2 Reasons for Process Termination**

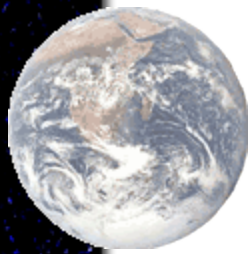
| | |
|---|---|
| Normal completion | The process executes an OS service call to indicate that it has completed running. |
| Time limit exceeded | The process has run longer than the specified total time limit. There are a number of possibilities for the type of time that is measured. These include total elapsed time ("wall clock time"), amount of time spent executing, and, in the case of an interactive process, the amount of time since the user last provided any input. |
| Memory unavailable | The process requires more memory than the system can provide. |
| Bounds violation | The process tries to access a memory location that it is not allowed to access. |
| Protection error | The process attempts to use a resource such as a file that it is not allowed to use, or it tries to use it in an improper fashion, such as writing to a read-only file. |
| Arithmetic error | The process tries a prohibited computation, such as division by zero, or tries to store numbers larger than the hardware can accommodate. |



# Process Termination

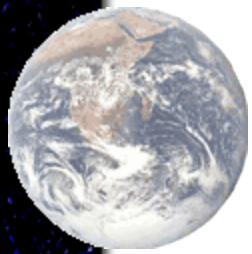
**Table 3.2 Reasons for Process Termination**

| | |
|---|---|
| Time overrun | The process has waited longer than a specified maximum for a certain event to occur. |
| I/O failure | An error occurs during input or output, such as inability to find a file, failure to read or write after a specified maximum number of tries (when, for example, a defective area is encountered on a tape), or invalid operation (such as reading from the line printer). |
| Invalid instruction | The process attempts to execute a nonexistent instruction (often a result of branching into a data area and attempting to execute the data). |
| Privileged instruction | The process attempts to use an instruction reserved for the operating system. |
| Data misuse | A piece of data is of the wrong type or is not initialized. |
| Operator or OS intervention | For some reason, the operator or the operating system has terminated the process (for example, if a deadlock exists). |
| Parent termination | When a parent terminates, the operating system may automatically terminate all of the offspring of that parent. |
| Parent request | A parent process typically has the authority to terminate any of its offspring. |



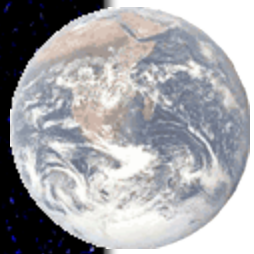
# Processes

- Not-running
  - ready to execute
- Blocked
  - waiting for I/O
- Dispatcher cannot just select the process that has been in the queue the longest because it may be blocked

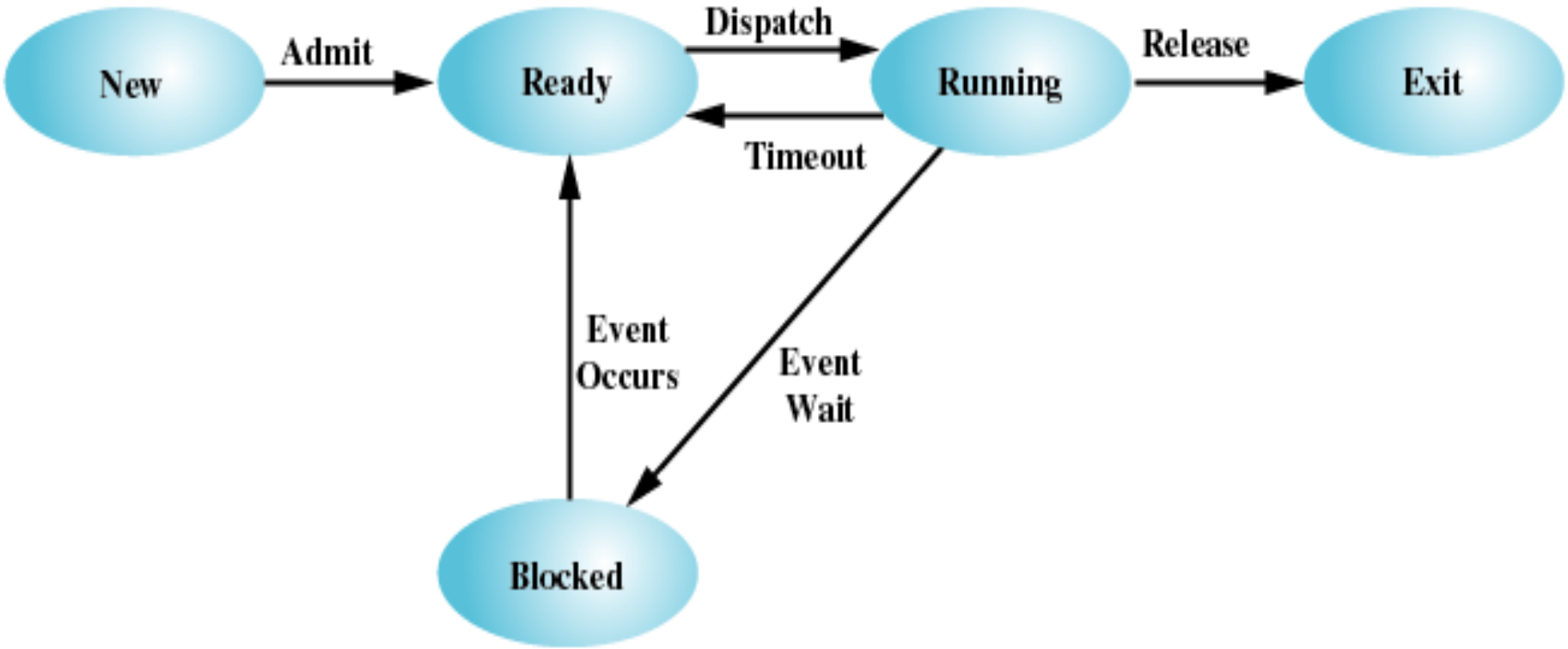


# A Five-State Model

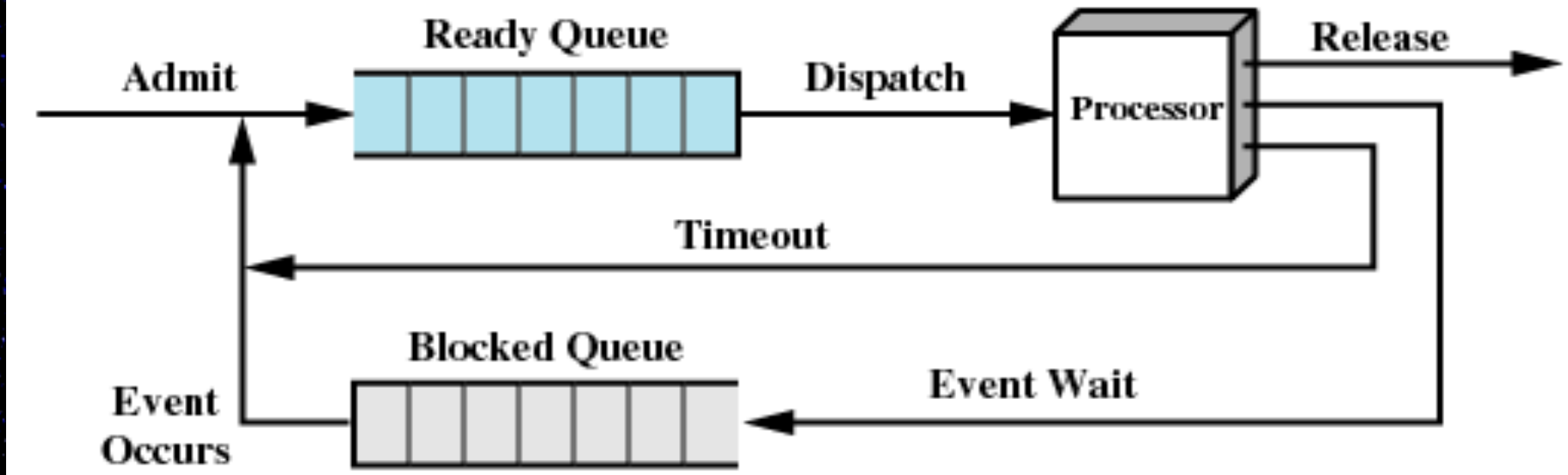
- New
- Ready
- Running
- Blocked
- Exit



# Five-State Process Model

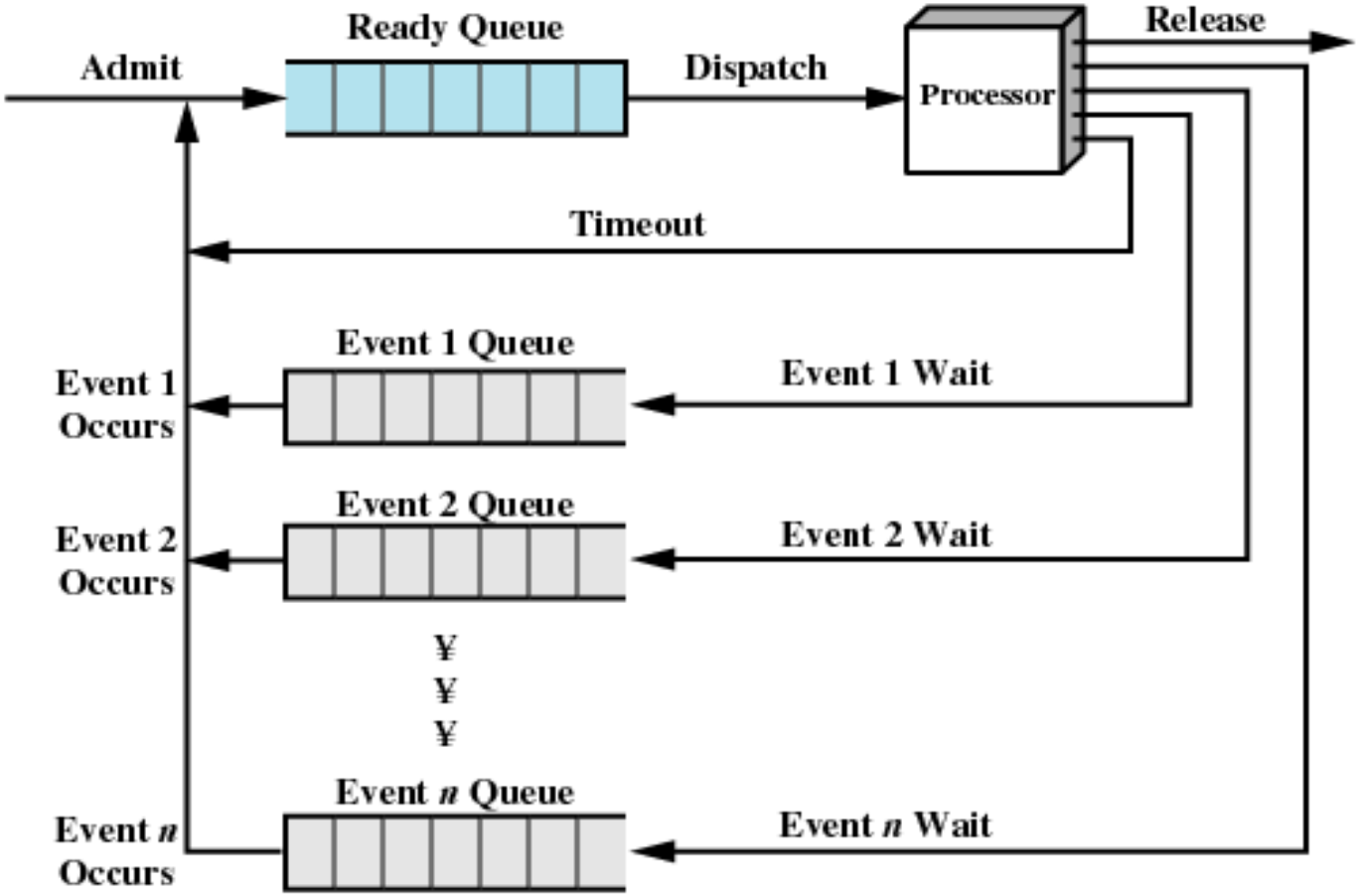


# Using Two Queues

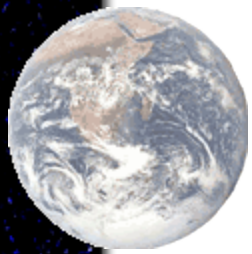



(a) Single blocked queue

# Multiple Blocked Queues

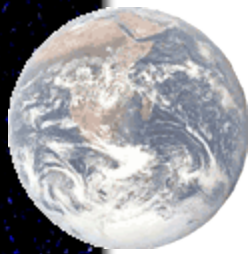



(b) Multiple blocked queues



# Suspended Processes

- Processor is faster than I/O so all processes could be waiting for I/O
- Swap these processes to disk to free up more memory
- Blocked state becomes suspend state when swapped to disk
- Two new states
  - Blocked/Suspend
  - Ready/Suspend

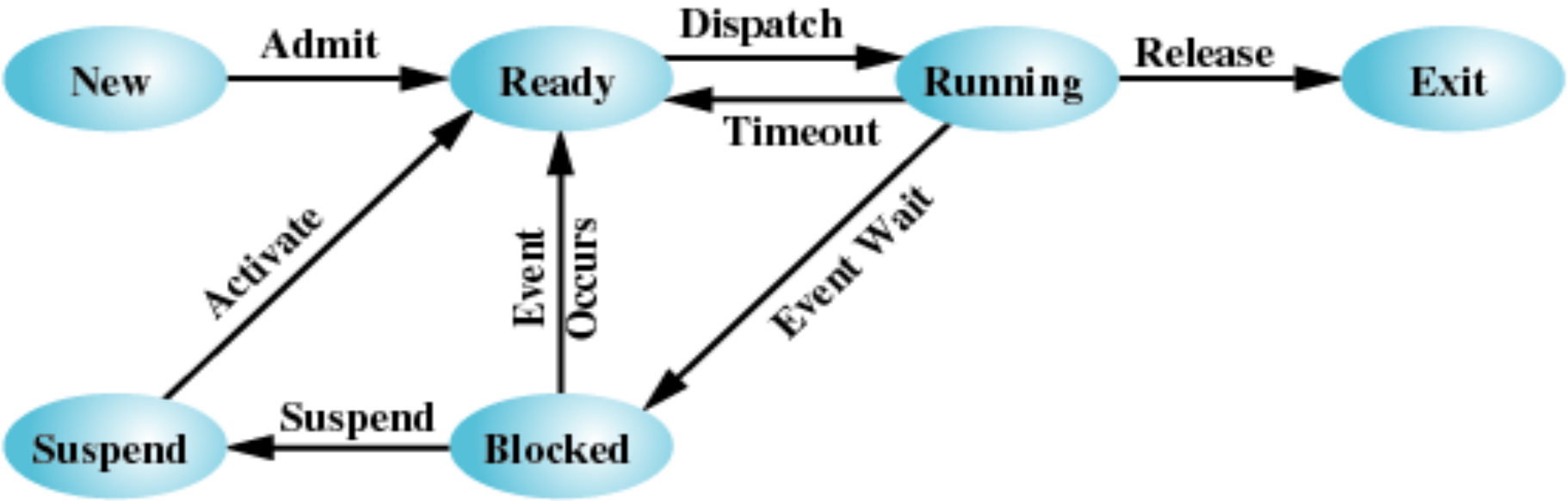


# Reasons for Process Suspension

**Table 3.3 Reasons for Process Suspension**

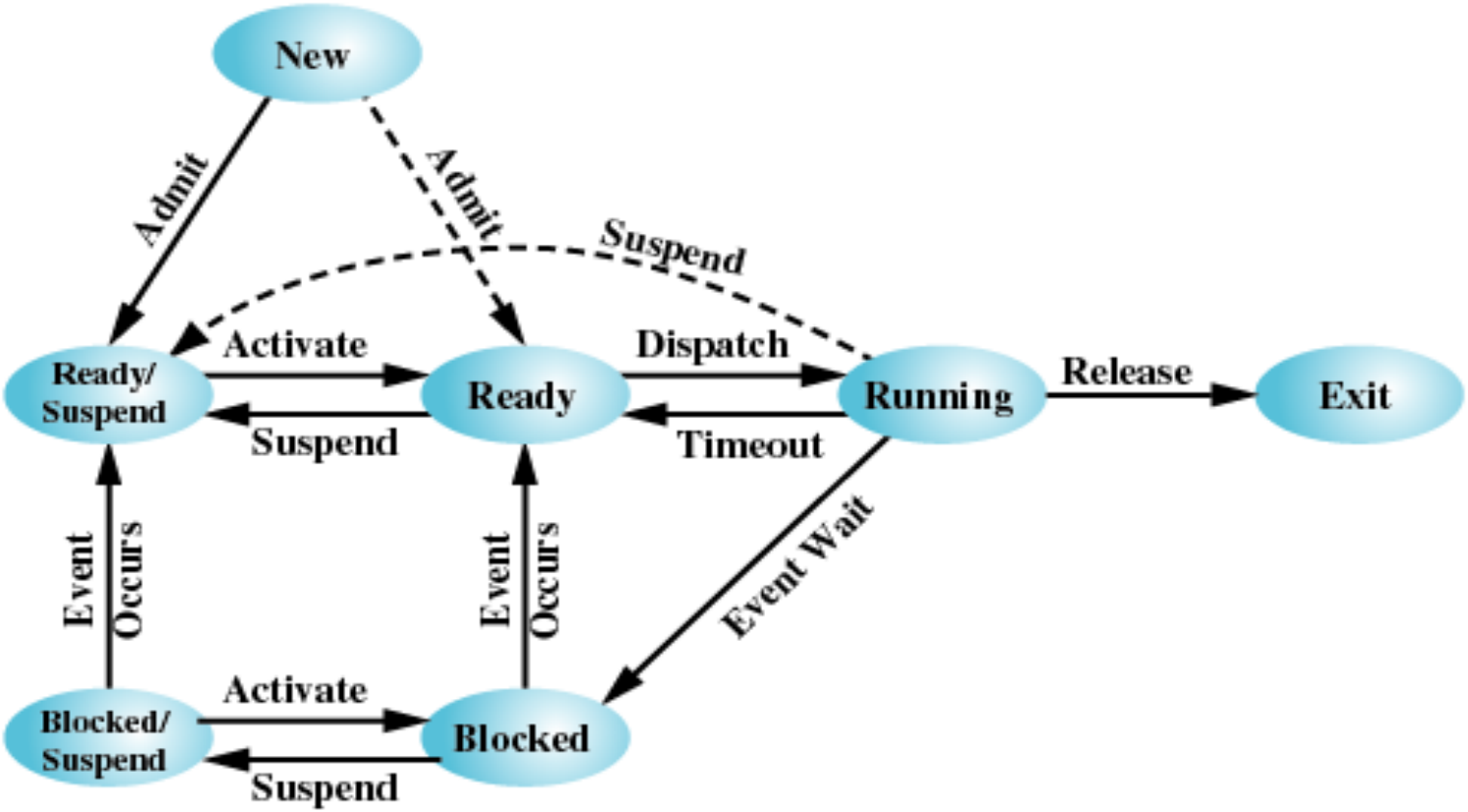
| | |
|---|---|
| Swapping | The operating system needs to release sufficient main memory to bring in a process that is ready to execute. |
| Other OS reason | The operating system may suspend a background or utility process or a process that is suspected of causing a problem. |
| Interactive user request | A user may wish to suspend execution of a program for purposes of debugging or in connection with the use of a resource. |
| Timing | A process may be executed periodically (e.g., an accounting or system monitoring process) and may be suspended while waiting for the next time interval. |
| Parent process request | A parent process may wish to suspend execution of a descendent to examine or modify the suspended process, or to coordinate the activity of various descendents. |

# One Suspend State




(a) With One Suspend State

# Two Suspend States




(b) With Two Suspend States