# Processes

**Process State Models** In studying operating systems, the term **process** can be defined in different ways, depending on its actual composition or utilization. Below are some of its basic definitions (Stallings, 2018).

* A process can be a program in execution.
* A process can be an instance of a program running on a computer.
* A process can be considered as an entity that can be assigned to and executed on a processor.
* A process can be treated as a unit of activity that executes a sequence of instructions, a current state, and an associated set of system resources.
* A process can also be defined as an environment that consists of a number of elements for executing a user-level program.

There are two (2) essential elements of a process, the *program code* – which can be shared with other processes that are executing the same program, and the *set of data* associated with the program code. Note that a computer's kernel can execute multiple processes at a time, thus supporting thousands of processes on a single system (Gregg, 2021). At any given point in time, while a program is executing, the process itself can be uniquely characterized by the following elements:

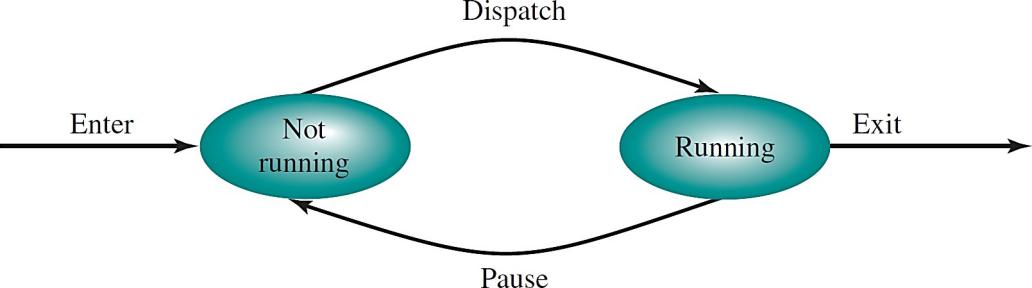
* **Identifier** – It is a unique identifier for each process.
* **Process State** – It indicates the current activity of a process. A process state can be any of the following (Silberschatz, Galvin & Gagne, 2018)
  + **New** – The process is being created.
  + **Ready** – The process is waiting to be assigned to a processor.
  + **Running** – The instructions are being executed.
  + **Waiting** – The process is waiting for a particular event to occur.
  + **Terminated** – The process has finished the execution.
* **Priority** – It refers to the priority level relative to other processes.
* **Program counter** – It indicates the address of the next instruction to be executed for a particular process.
* **Memory pointer** – It contains pointers to program codes and data associated with the process, including any memory blocks shared with other processes.
* **Context data** – These are the data present in a processor's registers during process execution.
* **I/O status information** – This includes different information associated with the input/output status of a process, such as the outstanding I/O requests and I/O devices assigned to the process.
* **Accounting information** – This includes different tracking information associated with the process, such as the amount of processor time, the clock time used, and time limits.

The elements of a process are stored in a data structure called **process control block**. The core concept of the process control block is that it contains sufficient information so it is possible to interrupt a running process and later resume the operation as if no interruption occurred. Thus, making the process control block the key tool that enables the operating system (OS) to support multiple processes and perform multiprocessing (Stallings, 2018).

# Two-State Process Model (Stallings, 2018)

An operating system controls the course of execution that involves the process of determining an interleaving pattern for execution and allocating resources to processes. In a two-state process model, a process can either be in a running state or in a not running state.

* **Not running state** – When the OS creates a new process, it creates a process control block for the new process and adds the new process into the not running state of the system. Then, the process waits for an opportunity to execute.
* **Running state** – The dispatch portion of the OS selects a particular process to run. Then, it enters the running state. Intermittently, the currently running process will be interrupted and the dispatch portion has to select other processes to run. Subsequently, the interrupted process returns to the queue in the not running state, while a completed process exits the system.



***Figure 1***. The state transition in a two-state process model.

*Source*: Operating Systems: Internal and Design Principles (9th ed.), 2018 p. 136

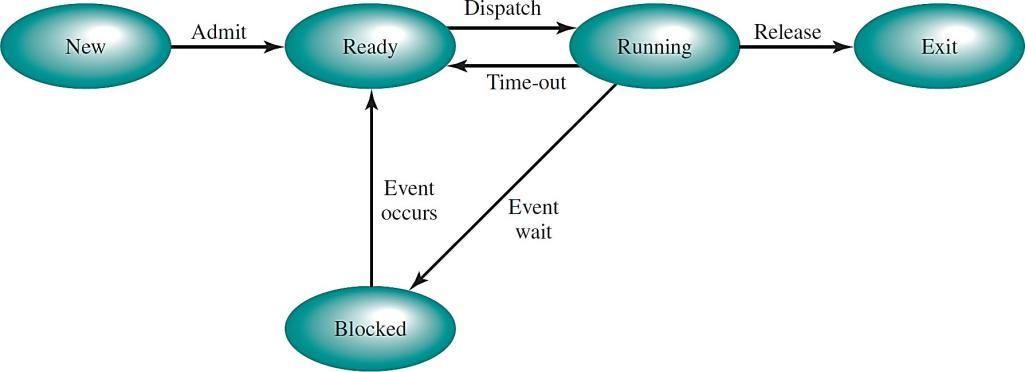
This process model is effective only if all processes are always ready to execute. The queue in the not running state would be in a first-in-first-out (FIFO) arrangement, wherein each process in the queue is given a certain amount of time, alternately, to execute and then return to the queue, unless completed.

However, some processes in the not running state are ready to execute but are blocked, waiting for an I/O operation to complete. The dispatcher could not just select any process since there is only one (1) queue. Rather, the dispatcher would have to scan the queue looking for the process that is not blocked and that has been in the queue the longest. This makes the implementation of the two-state process model inadequate.

# Five-State Process Model (Stallings, 2018)

This model naturally handles processes through the implementation of five unique states which are the following:

* **New state**: A process that has just been created and has not yet been admitted to the pool of executable processes by the OS.
* **Ready state**: A process that is prepared to execute when given the opportunity.
* **Running state**: The process that is currently being executed.
* **Blocked state/Waiting state**: A process that cannot execute until some event occurs, such as the completion of an I/O operation.
* **Exit state**: A process that has been released from the pool of executable processes by the OS, either because it halted or it was terminated.



***Figure 2***. The state transition in a five-state process model.

*Source*: Operating Systems: Internal and Design Principles (9th ed.), 2018 p. 140

Note that the not running state in a two-state process model was technically divided into two (2): The Ready state and the Block state. This is to improve the overall efficiency of the operating system. In addition, the New and the Exit state are useful constructs for process management. The following are the possible state transitions for a five-state process model:

|  |  |  |
| --- | --- | --- |
| Null → New | Running → Exit | Blocked → Ready |
| New → Ready | Running → Ready | Ready → Exit |
| Ready → Running | Running → Blocked | Blocked → Exit |

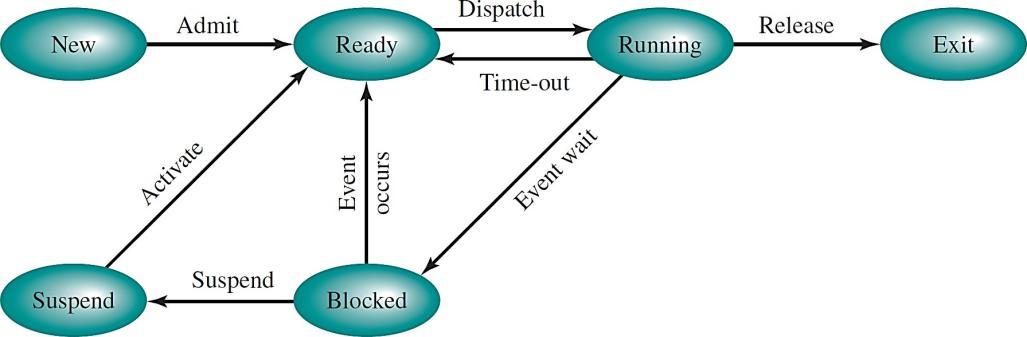
# Suspended Processes (Stallings, 2018)

The three (3) principal states which are Ready, Running, and Blocked, provide a systematic way of modeling the behavior of a process and guide the implementation of an operating system. Some OS are constructed using only these three (3) states. However, there are good justifications for adding other states to a model.

In a suspended process model, processes that are not immediately available for execution can occur. These processes may or may not be waiting for a specific event. If a process is waiting for a particular event, this means that the blocked state is independent of the suspended state, and the occurrence of the blocking event does not enable the process to be executed immediately.

A process can be placed in a suspended state by any of the following:

1. The process itself
2. A parent process
3. The operating system



***Figure 3***. A five-state process model with a suspended state.

*Source*: Operating Systems: Internal and Design Principles (9th ed.), 2018 p. 145

**Process Description (Stallings, 2018)** An operating system (OS) controls all the events within a computer system. It schedules and dispatches processes for execution by the processor, allocates resources to different processes, and responds to requests by user processes for basic services.

The OS can generally be considered as an entity that manages the utilization of system resources by processes. During the course of a process execution, each process requires access to certain system resources such as the processor, input and output devices, and main memory. In a multiprogramming environment, there are numerous processes that can be created and exist in a virtual memory.

An OS must have information about the current status of each process and resource in order to effectively manage all operations. Thus, the operating system constructs and maintains tables of information about each process that it is managing. The following are the four (4) different types of tables maintained by an operating system:

1. **Memory tables** – These are used to keep track of both the main and the secondary memory. A part of the main memory is reserved for OS utilization, while the remaining parts are available for the processes' utilization. Note that processes are maintained on the secondary memory using some sort of virtual memory or a simple swapping mechanism. The memory tables must encompass the following information:
   * The allocation of the main memory processes
   * The allocation of the secondary memory processes
   * Any protection attribute for the main or the virtual memory
   * Any information needed to manage the virtual memory
2. **I/O tables** – These are used by the OS to manage the input and output devices and channels of a computer system at any given point in time.
3. **File tables** – These tables hold information regarding the existence of files and its corresponding attributes through a file management system.
4. **Process tables** – These tables are maintained by the OS to manage processes.

**Process Location**: At a minimum, a process must include a program to be executed. Associated to this program is a set of data locations for local and

global variables and any defined constants. Thus, a process will always encompass sufficient memory to hold a program and its corresponding data.

Process control blocks are also referred to as **process images**. The location of a process image will depend on the memory management scheme being used. In the simplest case, a process image is maintained as a continuous block of memory maintained in the secondary memory. To execute a process, the entire process image must be loaded into the main memory, or at least to the virtual memory. Thus, the OS needs to know the location of each process on the disk.

**Process Attributes**: A multiprogramming system requires a great deal of information about each process that resides in a process control block. There are different types of information that might be of use to an OS without considering any details as to how that information is organized. The information in a process control block can be grouped into three (3) categories:

1. **Process identification** – This involves unique numeric identifiers that are assigned to specific processes, which can simply be an index into a primary process table, or identifiers for cross-referencing process tables.
2. **Processor state information** – This consists of processor registers' information. During process execution, information are in the registers. When a process is interrupted, all information in the registers must be saved so it can be restored when the process resumes execution.
3. **Process control information** – This contains the additional information needed by the OS to control and coordinate various active processes, such as the data structure, resource ownership and utilization information, and process privileges.

The process control block is the most important data structure in operating systems, since it contains all the information about a process that are needed by the OS. The process control blocks are read and/or modified by every module in an OS, including those involved in resource allocation, scheduling, interrupt processing, analysis, and performance monitoring.

**Process Control (Stallings, 2018) Modes of Execution**: Specific processes require certain privileges in order to protect the hardware and the OS from impending malfunction. Most processors support at least two (2) modes of execution which are the following:

* **User mode** – This is referred to as the *less-privileged mode*, since user programs execute in this mode.

In some operating systems, a process can be terminated by the process that created it, or when the parent process itself is terminated.

* **Kernel mode** – This is referred to as the *more-privileged mode*, where the

software has complete control of the processor and its instructions, registers, and memory.

Certain instructions and memory regions can only be executed and accessed in the more-privileged mode. This is necessary to protect the OS and the key operating system tables, such as process control blocks, from interferences by user programs.

**Process Creation**: When a new process is to be added to those currently being managed, the operating system builds the data structure that will manage the process and allocates address space in the main memory for the process. Contemporary operating systems commonly create processes in a way that is transparent to the user or application program. With the evolution of operating systems, programmers realized that it can be useful and efficient, especially in structuring applications, to allow one (1) process to cause the creation of another. Process creation can proceed as follows:

1. Assign a unique process identifier to the new process
2. Allocate space for the process
3. Initialize the process control block
4. Set the appropriate linkages
5. Create or expand other data structures

**Process Switching**: When the operating system controls the currently running processes, a process switch may occur at any time. Thus, the OS needs to provide a regaining control feature if a process is blocked, suspended, or paused. Process switching usually involves interrupts or traps. **Interrupts** are external events that are independent of the currently running process, while **traps** are related to errors or exception conditions generated within the currently running process. On the other hand, a **supervisor call** due to an explicit request can still occur to interrupt a process execution.

**Process Termination**: Any computer system must provide means for a process to indicate its completion. A batch job may include a *halt instruction* or an explicit OS *service call for termination*. Generally, the halt instruction generates an interrupt to alert the OS that a process has been completed. For interactive applications, specific action of user can trigger process completion.

**References:**

Gregg, B. (2021). *System performance: Enterprise and Cloud (2nd ed.)*. Pearson Education, Inc.

Silberschatz, A., Galvin, P. & Gagne, G. (2018). *Operating systems concepts (10th ed.)*. John Wiley & Sons, Inc. Stallings, W. (2018). *Operating systems: Internal and design principles (9th ed.)*. Pearson Education Limited