Introduction of Process Management

Last Updated : 12 Jul, 2024

A process is a program in execution. For example, when we write a program in C or C++ and compile it, the compiler creates binary code. The original code and binary code are both programs. When we actually run the binary code, it becomes a process. A process is an 'active' entity instead of a program, which is considered a 'passive' entity. A single program can create many processes when run multiple times; for example, when we open a .exe or binary file multiple times, multiple instances begin (multiple processes are created).

In this article, we will discuss process management in detail, along with the different states of a process, its advantages, disadvantages, etc.

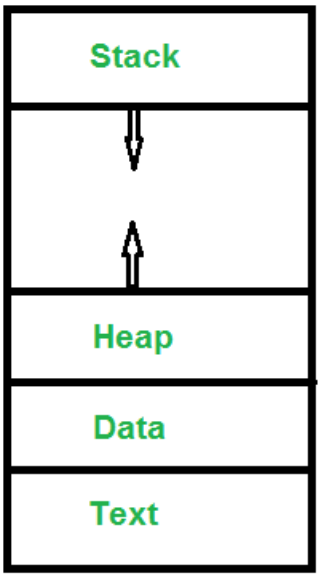
What is Process Management?

Process management is a key part of an operating system. It controls how processes are carried out, and controls how your computer runs by handling the active processes. This includes stopping processes, setting which processes should get more attention, and many more. You can manage processes on your own computer too.

The OS is responsible for managing the start, stop, and scheduling of processes, which are programs running on the system. The operating system uses a number of methods to prevent deadlocks, facilitate inter-process communication, and synchronize processes. Efficient resource allocation, conflict-free process execution, and optimal system performance are all guaranteed by competent process management. This essential component of an operating system enables the execution of numerous applications at once, enhancing system utilization and responsiveness.

How Does a Process Look Like in Memory?

A process in memory is divided into several distinct sections, each serving a different purpose. Here's how a process typically looks in memory:



- **Text Section**: A Process, sometimes known as the Text Section, also includes the current activity represented by the value of the Program Counter.
- **Stack**: The stack contains temporary data, such as function parameters, returns addresses, and local variables.
- **Data Section**: Contains the global variable.
- **Heap Section**: Dynamically memory allocated to process during its run time.

Characteristics of a Process

A process has the following attributes.

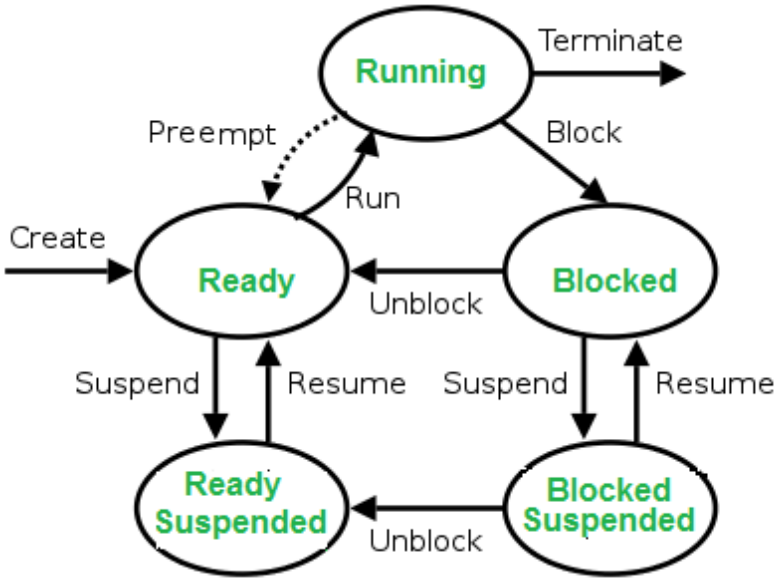
- **Process Id:** A unique identifier assigned by the operating system.
- **Process State:** Can be ready, running, etc.
- **CPU Registers:** Like the Program Counter (CPU registers must be saved and restored when a process is swapped in and out of the CPU)
- **Accounts Information:** Amount of CPU used for process execution, time limits, execution ID, etc
- **I/O Status Information:** For example, devices allocated to the process, open files, etc
- **CPU Scheduling Information:** For example, Priority (Different processes may have different priorities, for example, a shorter process assigned high priority in the shortest job first scheduling)

All of the above attributes of a process are also known as the **context of the process**. Every process has its own process control block(PCB), i.e. each process will have a unique PCB. All of the above attributes are part of the PCB.

States of Process

A process is in one of the following states:

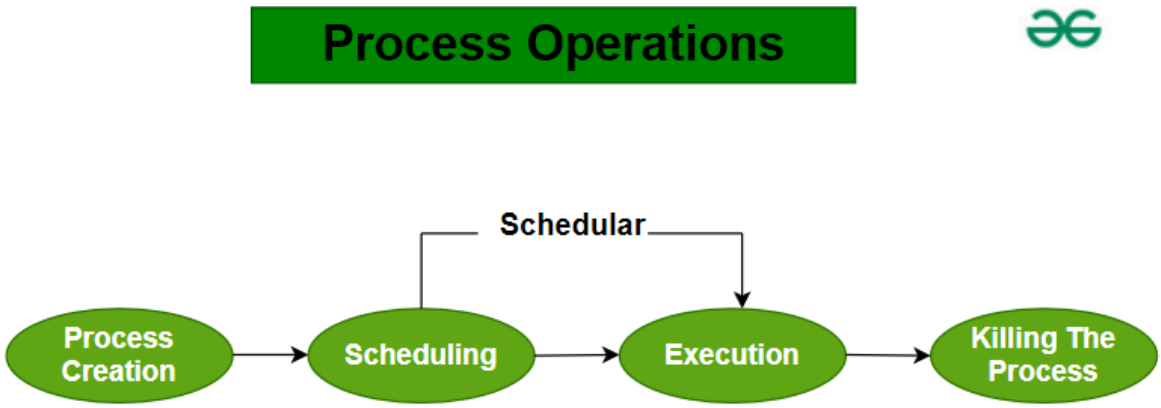
- **New:** Newly Created Process (or) being-created process.
- **Ready:** After the creation process moves to the Ready state, i.e. the process is ready for execution.
- **Running:** Currently running process in CPU (only one process at a time can be under execution in a single processor).
- **Wait (or Block):** When a process requests I/O access.
- **Complete (or Terminated):** The process completed its execution.
- **Suspended Ready:** When the ready queue becomes full, some processes are moved to a suspended ready state
- **Suspended Block:** When the waiting queue becomes full.



Process Operations

Process operations in an operating system refer to the various activities the OS performs to manage processes. These operations include process

creation, process scheduling, execution and killing the process. Here are the key process operations:



Process Operations

Process Creation

Process creation in an operating system (OS) is the act of generating a new process. This new process is an instance of a program that can execute independently.

Scheduling

Once a process is ready to run, it enters the “ready queue.” The scheduler’s job is to pick a process from this queue and start its execution.

Execution

Execution means the CPU starts working on the process. During this time, the process might:

- Move to a waiting queue if it needs to perform an I/O operation.
- Get blocked if a higher-priority process needs the CPU.

Killing the Process

After the process finishes its tasks, the operating system ends it and removes its Process Control Block (PCB).

Context Switching of Process

The process of saving the context of one process and loading the context of another process is known as Context Switching. In simple terms, it is like loading and unloading the process from the running state to the ready state.

When Does Context Switching Happen?

Context Switching Happen:

- When a high-priority process comes to a ready state (i.e. with higher priority than the running process).
- An Interrupt occurs.
- User and kernel-mode switch (It is not necessary though)
- Preemptive CPU scheduling is used.

Context Switch vs Mode Switch

A mode switch occurs when the CPU privilege level is changed, for example when a system call is made or a fault occurs. The kernel works in more a privileged mode than a standard user task. If a user process wants to access things that are only accessible to the kernel, a mode switch must occur. The currently executing process need not be changed during a mode switch. A mode switch typically occurs for a process context switch to occur. Only the kernel can cause a context switch.

CPU-Bound vs I/O-Bound Processes

A CPU-bound process requires more CPU time or spends more time in the running state. An I/O-bound process requires more I/O time and less CPU time. An I/O-bound process spends more time in the waiting state.

Process planning is an integral part of the process management operating system. It refers to the mechanism used by the operating system to determine which process to run next. The goal of process scheduling is to improve overall system performance by maximizing CPU utilization, minimizing execution time, and improving system response time.

Process Scheduling Algorithms

The operating system can use different scheduling algorithms to schedule processes. Here are some commonly used timing algorithms:

- **First-Come, First-Served (FCFS):** This is the simplest scheduling algorithm, where the process is executed on a first-come, first-served basis. FCFS is non-preemptive, which means that once a process starts executing, it continues until it is finished or waiting for I/O.
- **Shortest Job First (SJF):** SJF is a proactive scheduling algorithm that selects the process with the shortest burst time. The burst time is the time a process takes to complete its execution. SJF minimizes the average waiting time of processes.
- **Round Robin (RR):** Round Robin is a proactive scheduling algorithm that reserves a fixed amount of time in a round for each process. If a process does not complete its execution within the specified time, it is blocked and added to the end of the queue. RR ensures fair distribution of CPU time to all processes and avoids starvation.
- **Priority Scheduling:** This scheduling algorithm assigns priority to each process and the process with the highest priority is executed first. Priority can be set based on process type, importance, or resource requirements.
- **Multilevel Queue:** This scheduling algorithm divides the ready queue into several separate queues, each queue having a different priority. Processes are queued based on their priority, and each queue uses its own scheduling algorithm. This scheduling algorithm is useful in scenarios where different types of processes have different priorities.

Advantages of Process Management

- **Running Multiple Programs:** Process management lets you run multiple applications at the same time, for example, listen to music while browsing the web.
- **Process Isolation:** It ensures that different programs don't interfere with each other, so a problem in one program won't crash another.
- **Fair Resource Use:** It makes sure resources like CPU time and memory are shared fairly among programs, so even lower-priority programs get a chance to run.
- **Smooth Switching:** It efficiently handles switching between programs, saving and loading their states quickly to keep the system responsive and minimize delays.

Disadvantages of Process Management

- **Overhead:** Process management uses system resources because the OS needs to keep track of various data structures and scheduling queues. This requires CPU time and memory, which can affect the system's performance.
- **Complexity:** Designing and maintaining an OS is complicated due to the need for complex scheduling algorithms and resource allocation methods.
- **Deadlocks:** To keep processes running smoothly together, the OS uses mechanisms like semaphores and mutex locks. However, these can lead to deadlocks, where processes get stuck waiting for each other indefinitely.
- **Increased Context Switching:** In multitasking systems, the OS frequently switches between processes. Storing and loading the state of each process (context switching) takes time and computing power, which can slow down the system.

Conclusion

In conclusion, process management is a important function of an operating system, ensuring that multiple programs can run smoothly and efficiently. It involves creating, scheduling, and terminating processes, as well as managing resources and handling communication between processes. Effective process management optimizes the use of system resources, maintains system stability, and enhances the overall performance and responsiveness of the computer. Understanding and implementing robust process management strategies are crucial for maintaining an efficient and reliable computing environment.

GATE-CS-Questions on Process Management

Q.1: Which of the following need not necessarily be saved on a context switch between processes? (GATE-CS-2000)

(A) General purpose registers

(B) Translation lookaside buffer

(C) Program counter

(D) All of the above

Answer: (B)

In a process context switch, the state of the first process must be saved somehow, so that when the scheduler gets back to the execution of the first process, it can restore this state and continue. The state of the process includes all the registers that the process may be using, especially the program counter, plus any other operating system-specific data that may be necessary. A translation look-aside buffer (TLB) is a CPU cache that memory management hardware uses to improve virtual address translation speed. A TLB has a fixed number of slots that contain page table entries, which map virtual addresses to physical addresses. On a context switch,

some TLB entries can become invalid, since the virtual-to-physical mapping is different. The simplest strategy to deal with this is to completely flush the TLB.

Q.2: The time taken to switch between user and kernel modes of execution is t1 while the time taken to switch between two processes is t2. Which of the following is TRUE? (GATE-CS-2011)

(A) t1 > t2

(B) t1 = t2

(C) t1 < t2

(D) nothing can be said about the relation between t1 and t2.

Answer: (C)

Process switching involves a mode switch. Context switching can occur only in kernel mode.

Frequently Asked Questions on Process Management – FAQs

Why process management is important?

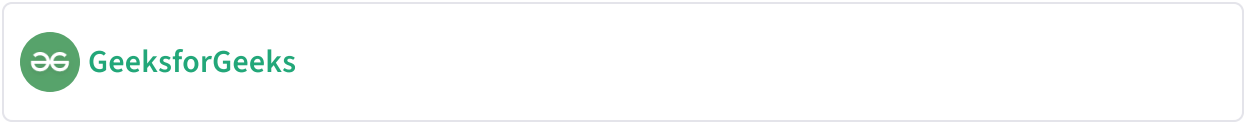
Process management is important in an operating system because it ensures that all the programs running on your computer work smoothly and efficiently.

What is the main difference between process manager and memory manager?

Processes in the system are manage by processor manager and also it is responsible for the sharing of the CPU. whereas, memory in the system is managed by memory manager and it is responsible also for allocation and deallocation of memory, virtual memory management, etc.

What is the difference between a process and a program?

A program is a set of instructions stored on disk (passive), while a process is an instance of the program in execution (active). A single program can be associated with multiple processes.



Previous Article

Real Time Operating System (RTOS)

Next Article

Process Table and Process Control Block (PCB)

Similar Reads

Difference Between Process, Parent Process, and Child Process

Running program is a process. From this process, another process can be created. There is a parent-child relationship between the two processes....

8 min read

Process Scheduler : Job and Process Status

When the jobs moves through the system and makes progress, it changes it's states from HOLD to FINISH. When the job is being processed by the...

2 min read

Difference between Process Image and Multi Thread Process image

1. Process Image : Process image is an executable file required during the execution of any process. It consists of several segments related to the...

2 min read

Process Table and Process Control Block (PCB)

While creating a process, the operating system performs several operations. To identify the processes, it assigns a process identification...

7 min read

Process states and Transitions in a UNIX Process

Process is an instance of a program in execution. A set of processes combined together make a complete program. There are two categories o...

4 min read

View More Articles

Article Tags : Operating Systems cpu-scheduling