Processes, Threads and Memory Management in Windows Operating System

Processes

The process is that the execution of a program that performs the actions laid out in that program. It can be defined as an execution unit where a program runs. Windows Operating System helps you to make, schedule, and terminates the processes which are employed by the CPU. A process created by the main process is named a child process.

Process operations are often easily controlled with the assistance of PCB (Process Control Block). you'll consider it as the brain of the process, which contains all the crucial information associated with processing like process id, priority, state, CPU registers, etc.

What is Process Management?

Process management involves various tasks like creation, scheduling, termination of processes, and a deadlock. the process may be a program that's under execution, which is a crucial a part of modern-day operating systems. Windows Operating System must allocate resources that enable processes to share and exchange information. It also protects the resources of every process from other methods and allows synchronization among processes.

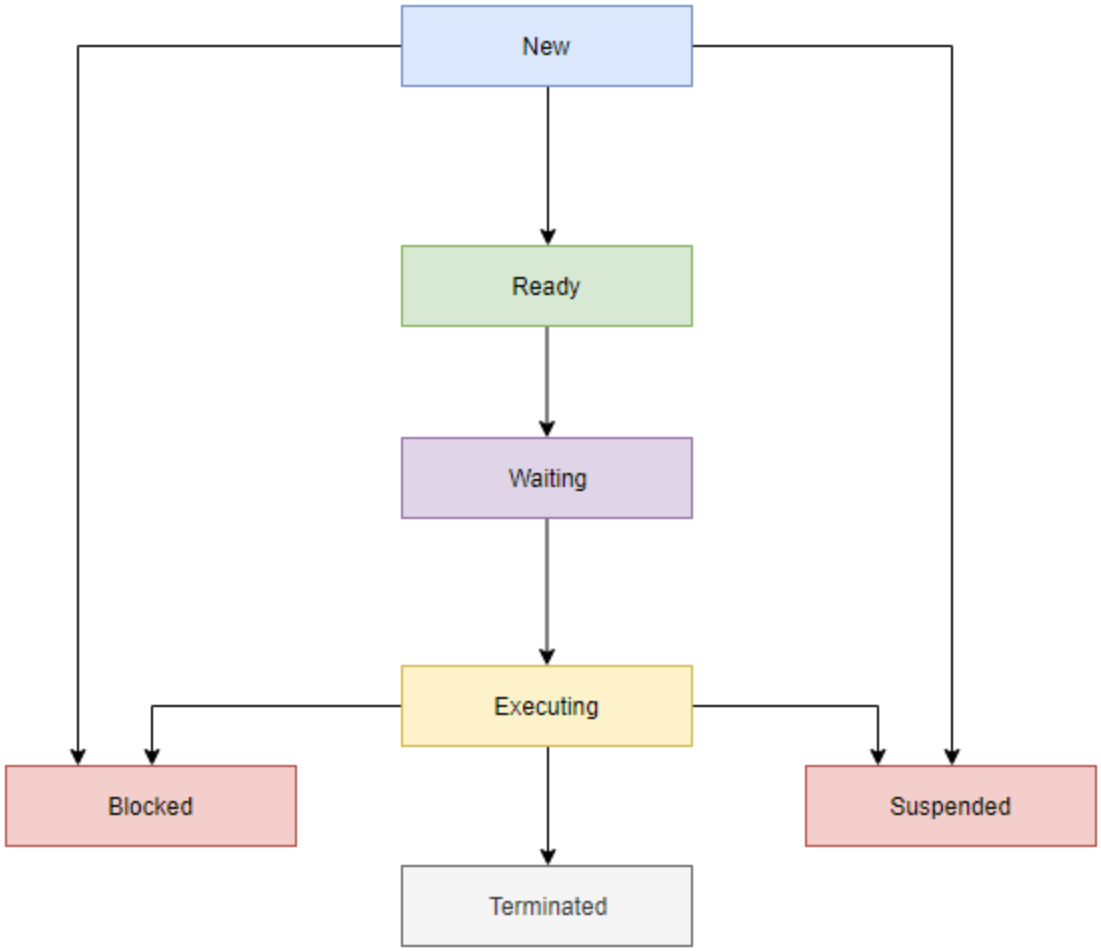
It is the work of Windows Operating System to manage all the running processes of the system. It handles operations by performing tasks like process scheduling and like resource allocation.

Process Control Blocks

The PCB may be a full sort of Process Control Block. it's a data structure that's maintained by Windows Operating System for each process. The PCB should be identified by an integer Process ID (PID). It helps you to store all data required to stay track of all the running processes.

It is also in charge of storing the contents of processor registers. These are saved when the process moves from the running state then returns back thereto. the knowledge is quickly updated within the PCB by the windows operating system as soon because the process makes the state transition.

Process States



Threads

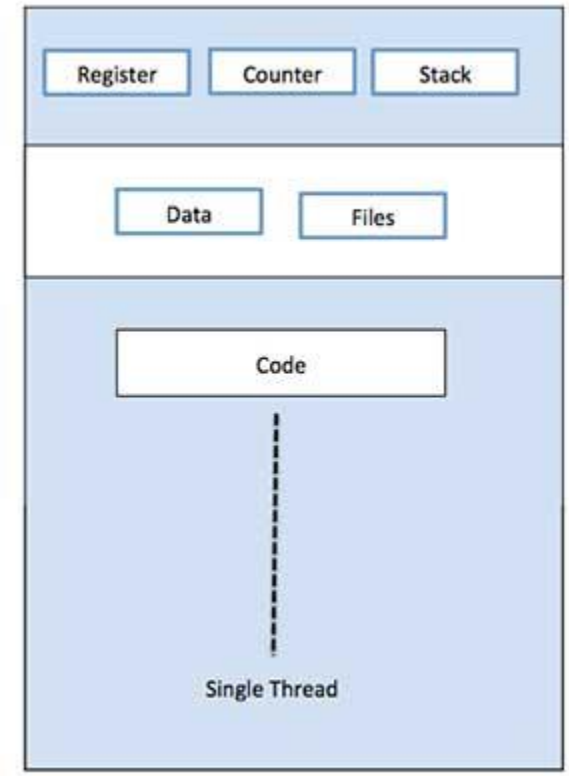
A thread may be a flow of execution through the process code, with its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack that contains the execution history.

A thread shares with its peer threads few information like code segment, data segment, and open files. When one thread alters a code segment memory item, all other threads see that.

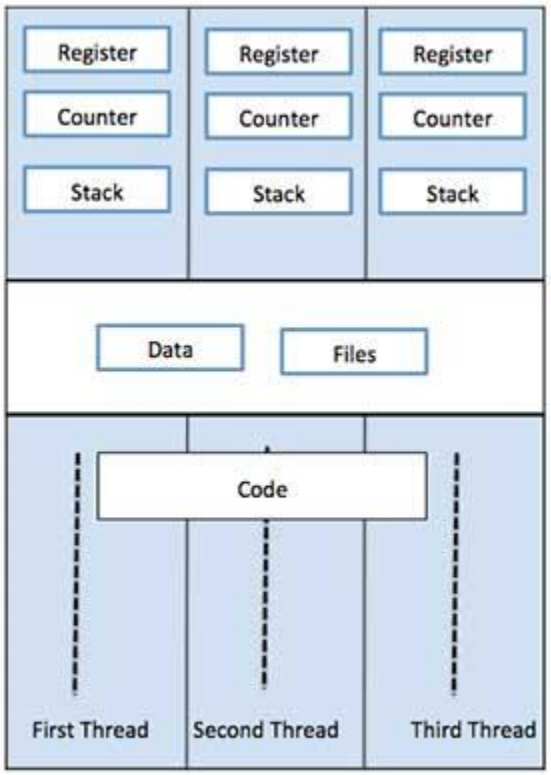
A thread is additionally called a lightweight process. Threads provide how to enhance application performance through parallelism. Threads represent a software approach to improving the performance of the Windows Operating System by reducing the overhead thread is like a classical process.

Each thread belongs to precisely one process and no thread can exist outside a process. Each thread represents a separate flow of control. Threads are successfully utilized in implementing

network servers and web servers. They also provide an appropriate foundation for the parallel execution of applications on shared-memory multiprocessors. the subsequent figure shows the working of a single-threaded and a multithreaded process.



Single Process P with single thread



Single Process P with three threads

Advantages of Thread

Threads minimize the context switching time.

- Use of threads provides concurrency within a process.
- Efficient communication.
- It is more economical to make and context switch threads.
- Threads allow utilization of multiprocessor architectures to a greater scale and efficiency.

Types of Thread

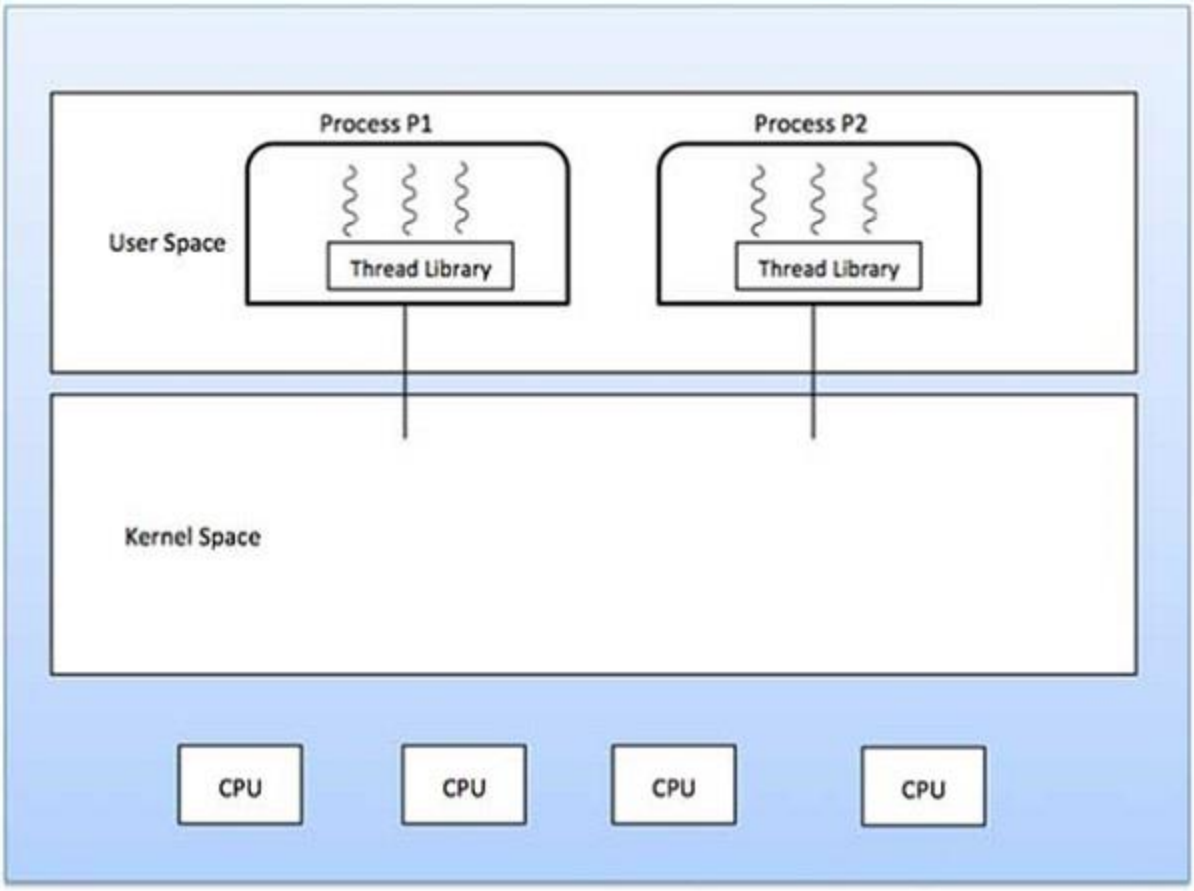
Threads are implemented within the following two ways −

- **User Level Threads** − User-managed threads.
- **Kernel Level Threads** – Windows OS manage threads working on the kernel, an OS core.

User Level Threads

In this case, the thread management kernel isn't conscious of the existence of threads. The thread library contains code for creating and destroying threads, for passing messages and data between threads, for scheduling thread execution, and for saving and restoring thread contexts. the application starts with one thread.

.



Kernel Level Threads

In this case, thread management is completed by the Kernel. there's no thread management code within the application area. Kernel threads are supported directly by the windows OS. Any

application is often programmed to be multithreaded. All of the threads within an application are supported within one process.

The Kernel maintains context information for the process as an entire and for people threads within the process. Scheduling by the Kernel is completed on a thread basis. The Kernel performs thread creation, scheduling, and management within the Kernel space. Kernel threads are generally slower to make and manage than user threads

Memory Management

Memory management is that the functionality of windows OS that handles or manages primary memory and moves processes back and forth between main memory and disk during execution. Memory management keeps track of every and every memory location, no matter either it's allocated to some process or it's free. It checks what proportion of memory is to be allocated to processes. It decides which process will get memory at what time. It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

Process Address Space

The process address space is that the set of logical addresses that a process references in its code. for instance, when 32-bit addressing is in use, addresses can range from 0 to 0x7fffffff; that's, 2^31 possible numbers, for a complete theoretical size of two gigabytes.

The OS takes care of mapping the logical addresses to physical addresses at the time of memory allocation to the program. There are three sorts of addresses utilized in a program before and after the memory is allocated −

S.N.	Memory Addresses & Description
1	**Symbolic addresses** The addresses utilized in an ASCII text file. The variable names, constants, and instruction labels are the essential elements of the symbolic address space.
2	**Relative addresses** At the time of compilation, a compiler converts symbolic addresses into relative addresses.
3	**Physical addresses** The loader generates these addresses at the time when a program is loaded into the main memory.

Virtual and physical addresses are equivalent in compile-time and load-time address-binding schemes. Virtual and physical addresses differ within the execution-time address-binding schemes.

The set of all logical addresses generated by a program is mentioned as a logical address space. The set of all physical addresses like these logical addresses is mentioned as a physical address space.

The runtime mapping from virtual to physical address is completed by the memory management unit (MMU) which may be a hardware device. MMU uses the subsequent mechanism to convert virtual addresses to physical addresses.

- The value within the base register is added to each address generated by a user process, which is treated as an offset at the time it's sent to memory. for instance, if the base register value is 10000, then an effort by the user to use address location 100 are going to be dynamically reallocated to location 10100.

- The user program deals with virtual addresses; it never sees the important physical addresses.

Memory Allocation

Main memory usually has two partitions −

- **Low Memory** − Windows OS resides in this memory.
- **High Memory** − User processes are held in high memory.

The windows OS uses the subsequent memory allocation mechanism

S.N.	Memory Allocation & Description
1	**Single-partition allocation** In this sort of allocation, a relocation-register scheme is employed to guard user processes from one another, and from changing operating-system code and data. The relocation register contains the worth of the smallest physical address whereas the limit register contains a variety of logical addresses. Each logical address must be but the limit register.
2	**Multiple-partition allocation** In this sort of allocation, the most memory is split into a variety of fixed-sized partitions where each partition should contain just one process. When a partition is free, a process is

	chosen from the input queue and is loaded into the free partition. When the process terminates, the partition becomes available for an additional process.

Fragmentation

As processes are loaded and far away from memory, the free memory space is broken into little pieces. It happens after a while that processes can't be allocated to memory blocks considering their small size and memory blocks remain unused. This problem is understood as Fragmentation.

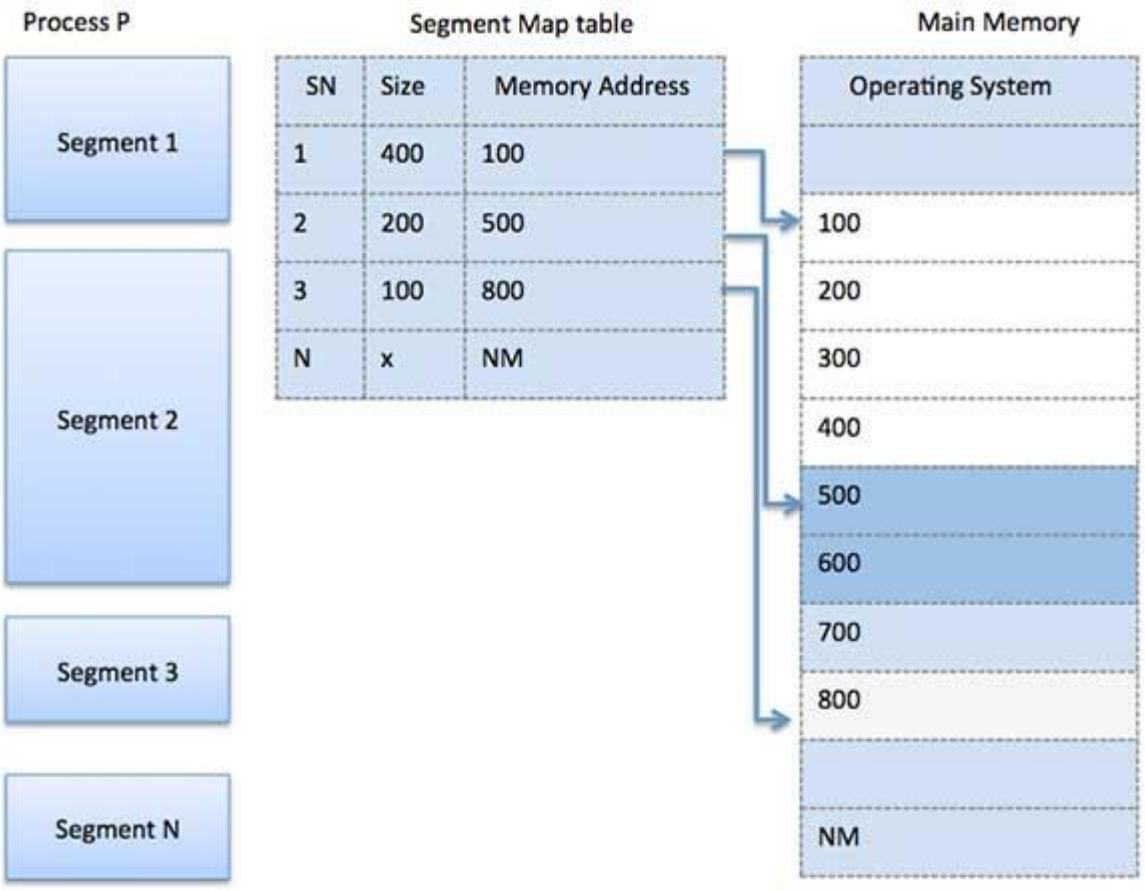
Paging

A computer can address more memory than the quantity physically installed on the system. This extra memory is really called virtual storage and it's a part of a hard that's found out to emulate the computer's RAM. The paging technique plays a crucial role in implementing virtual storage.

Paging may be a memory management technique during which process address space is broken into blocks of an equivalent size called pages (size is that the power of two, between 512 bytes and 8192 bytes). the scale of the process is measured within the number of pages.

Segmentation

Segmentation may be a memory management technique during which each job is split into several segments of various sizes, one for every module that contains pieces that perform related functions. Each segment is really a special logical address space of the program.


Process P
Segment Map table
Main Memory
Segment 1
Segment 2
Segment 3
Segment N
SN
Size
Memory Address
1
400
100
2
200
500
3
100
800
N
x
NM
Operating System
100
200
300
400
500
600
700
800
NM

References

Tutorialspoint.com. 2021. *Operating System - Multi-Threading - Tutorialspoint*. [online] Available at: <https://www.tutorialspoint.com [Accessed 4 May 2021].

Guru (2020) *Processes* , Available at: *https://www.guru99.com* (Accessed: 05-05-2021