Chapter 2 Process Control Block (PCB)

Degree of multi programming .

Context Switching in OS .

Context switching triggers.

PCB.

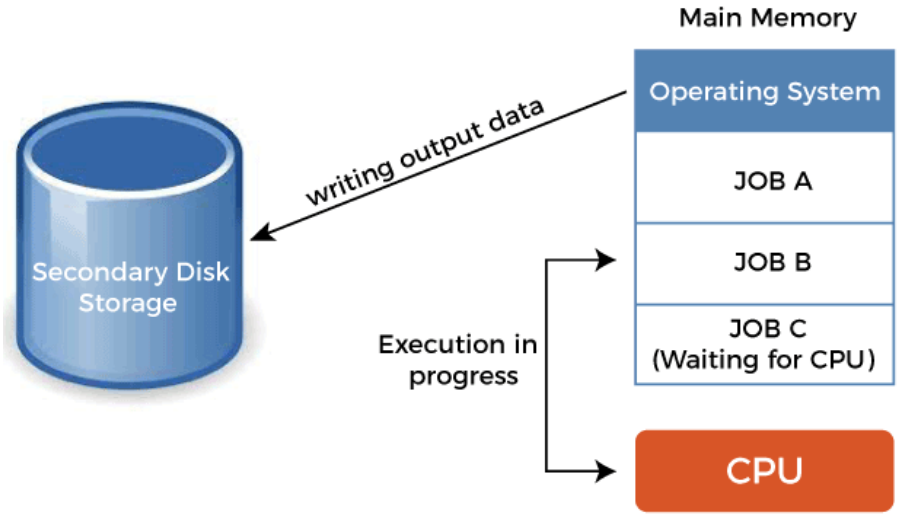
Memory management Stack & heap

- **What is Program in OS?**
- **What is Process?**
- **Features of Program.**
- **Features of Process.**
- **What is the Difference between Program and Process?**
- **States of Process**.
- **Attributes of a process.**
- **States of Process: A process is in one of the following states.**

Degree of multi programming (ch:1)

- Batch Operating System.
- Multiprogramming Operating System.
- Multiprocessing Operating System.

- used to keep the CPU busy



Context Switching

The Context switching is a technique or method used by the operating system to switch a process from one state to another to execute its function using CPUs in the system. When switching perform in the system, it stores the old running process's status in the form of registers and assigns the CPU to a new process to execute its tasks. While a new process is running in the system, the previous process must wait in a ready queue. The execution of the old process starts at that point where another process stopped it. It defines the characteristics of a multitasking operating system in which multiple processes shared the same CPU to perform multiple tasks without the need for additional processors in the system.

The need for Context switching

A context switching helps to **share a single CPU across all processes to complete its execution and store the system's tasks status**. When the process reloads in the system, the execution of the process starts at the same point where there is conflicting.

Following are the reasons that describe the need for context switching in the Operating system.

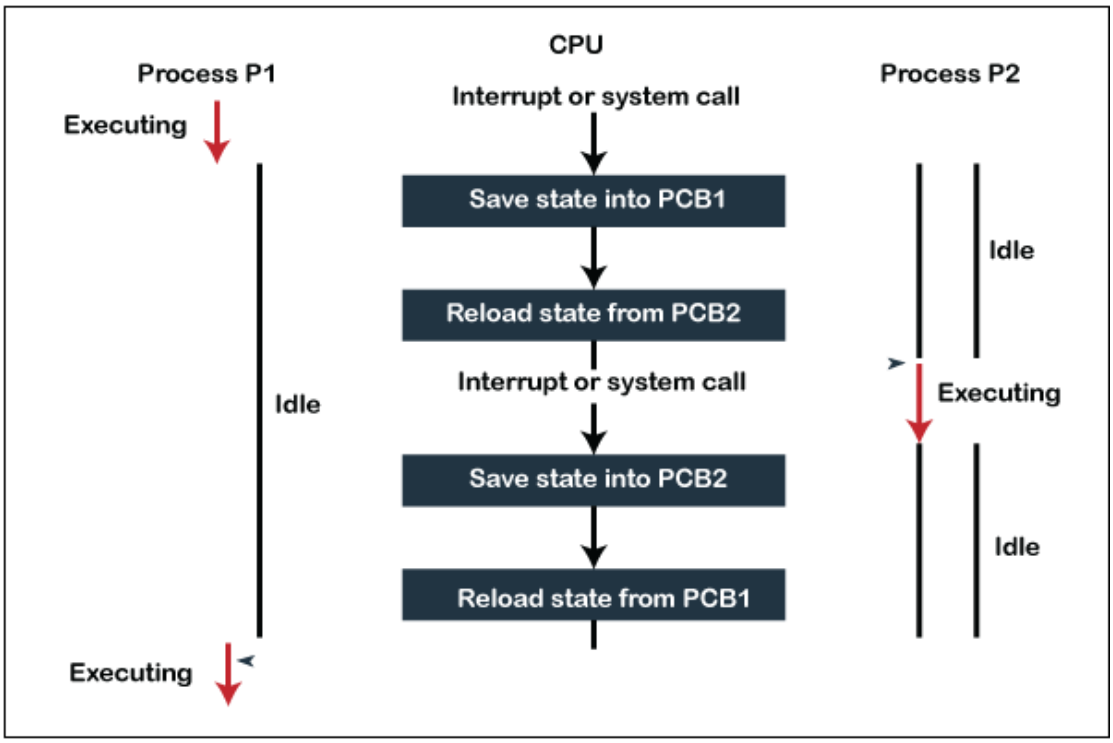
The switching of one process to another process is not directly in the system. **A context switching helps the operating system that switches between the multiple processes to use the CPU's resource to accomplish its tasks and store its context**. We can resume the service of the process at the same point later. If we do not store the currently running process's data or context, the stored data may be lost while switching between processes.

If a **high priority process** falls into the ready queue, the currently running process will be shut down or stopped by a high priority process to complete its tasks in the system.

If any running process requires I/O resources in the system, the current process will be switched by another process to use the CPUs. And when the I/O requirement is met, the old process goes into a ready state to wait for its execution in the CPU. Context switching stores the state of the process to resume its tasks in an operating system. Otherwise, the process needs to restart its execution from the initials level.

If any interrupts occur while running a process in the operating system, the process status is saved as registers using context switching. After resolving the interrupts, the process switches from a wait state to a ready state to resume its execution at the same point later, where the operating system interrupted occurs.

A context switching allows a single CPU to handle multiple process requests simultaneously without the need for any additional processors.



Context switching steps

Context switching triggers

Following are the three types of context switching triggers as follows.

Interrupts: A CPU requests for the data to read from a disk, and if there are any interrupts, the context switching automatic switches a part of the hardware that requires less time to handle the interrupts.

Multitasking: A context switching is the characteristic of multitasking that allows the process to be switched from the CPU so that another process can be run. When switching the process, the old state is saved to resume the process's execution at the same point in the system.

Kernel/User Switch: It is used in the operating systems when switching between the user mode, and the kernel/user mode is performed.

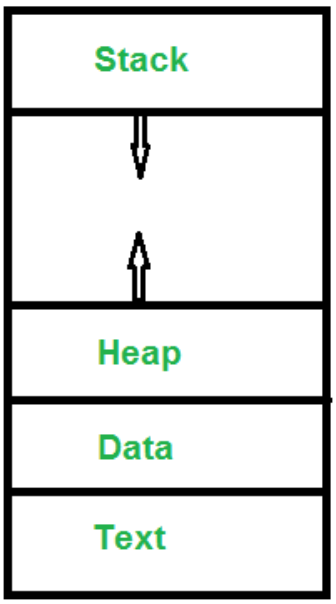
What is the PCB?

A PCB (Process Control Block) is a data structure used in the operating system to store all data related information to the process. For example, **when a process is created in the operating system, updated information of the process, switching information of the process, terminated process in the PCB.**

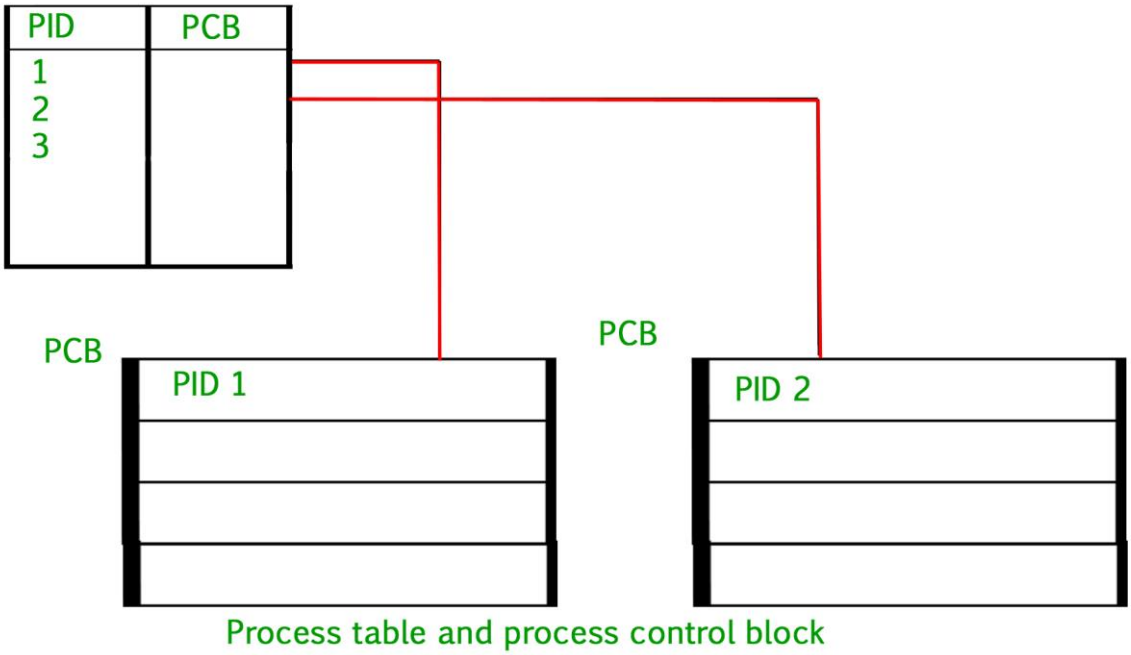
Memory management Stack & heap

The Attributes of the process are used by the Operating System to create the process control block (PCB) for each of them.

1. **Text Section**: The text area contains the program's machine instructions (i.e., the executable code).
2. **Stack**: The stack contains temporary data, such as function parameters, returns addresses, and local variables.
3. **Data Section**: Contains the global variable.
4. **Heap Section**: Dynamically allocated memory to process during its run time.



The functional memory organization of a running program



Process table and process control block

What is Program in OS?

A Program is an executable file which contains a certain set of instructions written to complete the specific job or operation on your computer. For example, Google browser chrome.exe is an executable file which stores a set of instructions written in it which allow you to open the browser and explore web pages.

Programs are never stored on the primary memory in your computer. Instead, they are stored on a disk or secondary memory on your PC or laptop. They are read from the primary memory and executed by the kernel.

What is Process?

A Process is an execution of a specific program. It is an active entity that actions the purpose of the application. Multiple processes may be related to the same program. For example, if you double-click on Google Chrome browser, you start a process that runs Google Chrome and when you open another instance of Chrome, you essentially create a second process.

Program vs Process: 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).

Difference between Program and Process

- Process is an executing part of a program whereas a program is a group of ordered operations to achieve a programming goal.
- The process has a shorter and minimal lifespan whereas program has a longer lifespan.
- Process contains many resources like a memory address, disk, printer while Program needs memory space on the disk to store all instructions.
- When we distinguish between process and program, Process is a dynamic or active entity whereas Program is a passive or static entity.

What is the stack?

The stack is a segment of memory where data like your local variables and function calls get added and/or removed in a last-in-first-out (LIFO) manner. When you compile a program, the compiler enters through the main function and a stack frame is created on the stack. A frame, also known as an activation record is the collection of all data on the stack associated with one subprogram call. The main function and all the local variables are stored in an initial frame

What is the heap?

The heap is the segment of memory that is not set to a constant size before compilation and can be controlled dynamically by the programmer. **Think of the heap as a "free pool" of memory you can use when running your application.** The size of the heap for an application is determined by the physical constraints of your RAM (Random access memory) and is generally much larger in size than the stack.

Example:

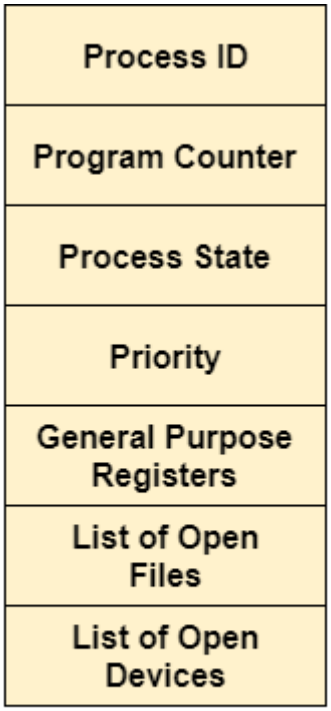
Global, static data types.

primitive data types.

Reference data types.

Attributes of a process

The Attributes of the process are used by the Operating System to create the process control block (PCB) for each of them. This is also called context of the process. Attributes which are stored in the PCB are described below.



Process Attributes

1. Process ID

When a process is created, a unique id is assigned to the process which is used for unique identification of the process in the system.

2. Program counter

A program counter stores the address of the last instruction of the process on which the process was suspended. The CPU uses this address when the execution of this process is resumed. Register the line number of execution instruction.

3. Process State

--We will discuss about them later in detail.

4. Priority

Every process has its own priority. The process with the highest priority among the processes gets the CPU first. This is also stored on the process control block.

5. General Purpose Registers

Every process has its own set of registers which are used to hold the data which is generated during the execution of the process.

6. List of open files

During the Execution, Every process uses some files which need to be present in the main memory. OS also maintains a list of open files in the PCB.

7. List of open devices

OS also maintain the list of all open devices which are used during the execution of the process.

States of Process: A process is in one of the following states:

1. New State

Newly Created Process (or) being-created process.

When a program in secondary memory is started for execution, the process is said to be in a new state.

2. Ready State

After creation process moves to Ready state, i.e. the process is ready for execution. After being loaded into the main memory and ready for execution, a process transitions from a new to a ready state. The process will now be in the ready state, waiting for the processor to execute it. Many processes may be in the ready stage in a multiprogramming environment.

3. Run State

After being allotted the CPU for execution, a process passes from the ready state to the run state (only one process at a time can be under execution in a single processor).

4.Terminate State

When a process's execution is finished, it goes from the run state to the terminate state. The process completed its execution. The operating system deletes the process control clock (or PCB) after it enters the terminate state.

5. Block or Wait State

If a process requires an Input/Output operation or a blocked resource during execution, it changes from run to block or the wait state.

The process advances to the ready state after the I/O operation is completed or the resource becomes available.

6. Suspend Ready State

When the ready queue becomes full, some processes are moved to suspended ready state. If a process with a higher priority needs to be executed while the main memory is full, the process goes from ready to suspend ready state. Moving a lower-priority process from the ready state to the suspend ready state frees up space in the ready state for a higher-priority process. Until the main memory becomes available, the process stays in the suspend-ready state. The process is brought to its ready state when the main memory becomes accessible.

7. Suspend Wait State

If a process with a higher priority needs to be executed while the main memory is full, the process goes from the wait state to the suspend wait state. Moving a lower-priority process from the wait state to the suspend wait state frees up space in the ready state for a higher-priority process.

The process gets moved to the suspend-ready state once the resource becomes accessible. The process is shifted to the ready state once the main memory is available.