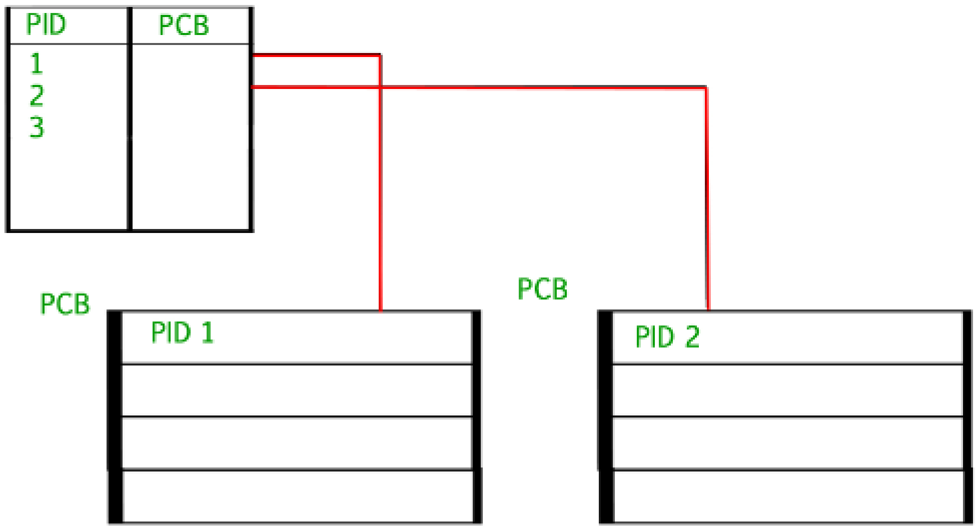
Process Table and Process Control Block (PCB):

Operating system is capable of running multiple processes at the same time, right. So it becomes necessary to manage each process in every cases. So how does an OS manage this?

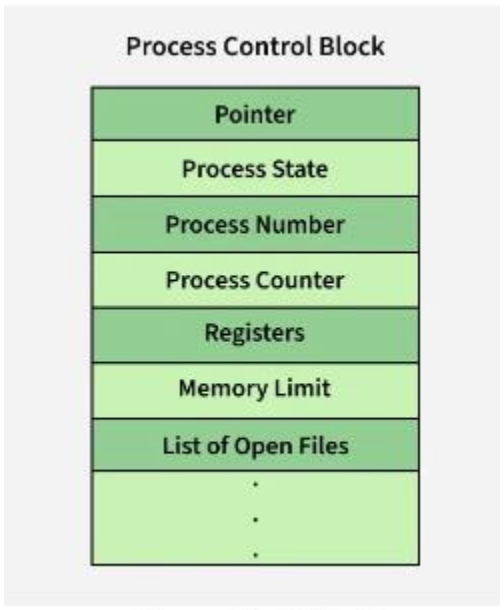
The question is through the Process Table and Process Control Block.
Process table is the table that contains all the processes running at a given time inside the OS. And Process Control Block is a data structure used by the operating system to manage information about a process. It keeps tracks of many important pieces of information needed to manage the process efficiently. Such as ProcessID (PiD), Process State, Program Counter, Stack Pointer, etc.



Process table and process control block

Structure of a Process Control Block:

There is several information about a process that the PCB stores. They are given in the following image:



Process Control Block

- **Pointer:**
  It is a stack pointer that is required to be saved when the process is switched from one state to another to retain the current position of the process. For example, in context switch, CPU may need to work on something else. During this switch, the Pointer is used to remember where the process left off, so when it resumes, it can pick up exactly where it stopped. So basically, it keeps track of **current state** of the process.

- **Process State:**
  It stores the respective state of the process in its lifecycle. It helps OS to keep track of what the process is doing at any given moment. A process can be in one of the several state during its execution.
  - **New:**
    The process is being created. Like when you double-click a program icon, the OS initializes the program and prepares it for execution.

  - **Ready:**
    The process is ready to run but is waiting for CPU to become available. Like when multiple programs are open, and they are all ready to execute but the CPU can only execute one at a time. Other programs wait in the "Ready" queue.

  - **Running:**
    The process is currently being executed by the CPU. Like if we are watching a video or playing a game, the process associated with that activity is in the "Running" state while it is actively performing its task.

  - **Blocked (or waiting):**
    The process is waiting for a specific event to occur before it can continue. A program that waits for user input, like typing text in a Word document, enters the "Blocked" state until you start typing.

  - **Terminated:**
    The process has completed its execution or has been stopped by the operating system. When you close a program, its process is moved to the "Terminated" state, and resources used by it are freed.

  - **Suspended (optional):**
    The process is temporarily paused and moved to secondary memory (like a hard drive) to free up resources. When you minimize a program or put your computer to sleep, some processes may enter the "Suspended" state to save system resources.

- **Process Number:**
  The Process Number or Process ID (PID) is a unique identification number that the operating system assigns to each process. It helps the OS distinguish one process from another, especially when managing multiple processes running at the same time.
  It is extremely important when a process needs to communicate with each other (IPC), the PID helps the process involved in the communication. Another importance is Tools like Task Manager in Windows or top/ps in Linux display the PID to help you identify processes for troubleshooting or monitoring system performance.

- **Program Counter:**
  The Program Counter (PC), also known as the Instruction Pointer, is a register in the CPU that plays a crucial role in process execution. It keeps track of the address of the next instruction to be executed in a program.
  In most cases, the PC increments automatically, after CPU executes an instruction, so the CPU processes the instructions one by one in order. For certain instructions like loops or conditionals, the Program Counter may be updated to point to a different address instead of incrementing, enabling the CPU to "jump" to another part of the program.

- **Registers:**
  Registers are very high-speed memory locations inside the CPU that temporarily hold data, instructions, and addresses needed for processing. The sole purpose of the register is a fast retrieval of data for processing by the CPU. Though accessing data from RAM is comparatively faster than HDD, it is still not enough for CPU. For even better processing, there are memories in CPU that can get data from RAM which are about to be executed beforehand. After registers, we have cache memory, which is faster but less than registers.

  There are different types of CPU Registers. They are:

1. **Accumulator (AC):**
   - **Purpose:** Holds the result of arithmetic or logical operations.
   - **Example:** If the CPU adds two numbers, the result is stored in the accumulator.
2. **Program Counter (PC):**
   - **Purpose:** Points to the memory address of the next instruction to be executed.
   - **Example:** It keeps track of where the CPU should fetch the next instruction.
3. **Instruction Register (IR):**
   - **Purpose:** Holds the instruction currently being executed.
   - **Example:** If the CPU is executing "ADD A, B," this instruction is stored in the IR.
4. **Memory Address Register (MAR):**
   - **Purpose:** Holds the memory address of the data or instruction that the CPU needs to access.
   - **Example:** If the CPU wants to fetch data from memory address 1024, the MAR will store "1024."
5. **Memory Data Register (MDR):**
   - **Purpose:** Temporarily stores data that is being transferred to or from memory.
   - **Example:** If the CPU fetches a value from memory, it is stored in the MDR before processing.
6. **General-Purpose Registers:**
   - **Purpose:** Used to temporarily store data during processing.
   - **Example:** These registers hold variables, intermediate results, or values that will be used in calculations.
7. **Stack Pointer (SP):**
   - **Purpose:** Points to the top of the stack in memory (used for managing function calls and local variables).
   - **Example:** If a program pushes data onto the stack, the SP keeps track of its position.
8. **Flag Registers (or Status Registers):**
   - **Purpose:** Contain flags that indicate the status of the CPU after an operation (e.g., zero, carry, overflow, etc.).
   - **Example:** After adding two numbers, a "Zero Flag" might be set if the result is zero.

How Registers Work

1. **Fetching Instructions:** The CPU fetches an instruction from memory. The **Program Counter (PC)** points to the instruction, which is then loaded into the **Instruction Register (IR).**

2. **Decoding:** The CPU decodes the instruction, and any required data (e.g., numbers to add) is fetched into **general-purpose registers.**
3. **Execution:** The CPU performs the operation (e.g., addition), and the result is stored in a register like the **Accumulator (AC).**
4. **Storing Results:** If needed, the CPU writes the result back to memory or keeps it in a register for future use.

So, in conclusion, when a process is running and it's time slice expires, the current value of the process specific registers would be stored in PCB and the process would swapped out. When the process is again scheduled to run, the registers values is read from PCB and written to the CPU registers. This is the main purpose of the registers in PCB.

- **Memory Limits:**
  his field contains the information about memory management system used by the operating system. This may include page tables, segment tables, etc.

- **List of Open Files:**
  This information includes the list of files opened for a process.

Advantages of Process Table:

1. Helps to keep track of all the processes which are either running, waiting, or completed.
2. It provides information needed to decide which process should run next i.e. Process Scheduling.
3. Organizes all details about process in one place which is simple for OS to manage them.

Disadvantages of Process Table:

- Extra work for OS to maintain it.
- Takes some space or memory.
- Can slow the system because it needs some time to maintain as well.

Advantages of Process Control Block:

1. Keeps all information about the process like ID, and resources it uses.
2. When a process is paused, the PCB saves it current states so that it can continue later.
3. By storing all necessary details, it helps OS to run processes run efficiently without interruptions.

Disadvantages of Process Control Block:

1. Uses more memory.
2. Slows context switching because the system has to update the PCB of the old process and load new one.
3. If PCB gets compromised, then someone could access and modify it.

Frequently Asked Questions on Process Table and Process Control Block – FAQ's

What information does a Process Control Block (PCB) contain?
A process control board (PCB) stores various information about a process so that the operating system can manage it properly. A typical printed circuit board contains the following components: Process ID (PID), Process Status, CPU Registers, Memory Management Information, I/O Information, etc.

Can processes share the same PCB?
No, each process has its own ***unique PCB****. Sharing would lead to confusion as the OS would not be able to differentiate between processes.*

What happens to a PCB when a process finishes?
When a process terminates:

- *Its PCB is removed from the Process Table.*
- *The resources held by the process are released.*

How does the OS prioritize processes using the Process Table?
The Process Table includes priority information in each PCB. The OS uses this priority to decide which process gets CPU time first, especially in a priority-based scheduling system.