**Importance of PCB for OS to control and manage process: -**

The work of process control block (PCB) in process management is that it can access or modified by most OS utilities including those are involved with memory, scheduling, and input / output resource access. It can be said that the set of the process control blocks give the information of the current state of the operating system. Data structuring for processes is often done in terms of process control blocks. For example, pointers to other process control blocks inside any process control block allows the creation of those queues of processes in various scheduling states.

**Some points of PCB’s importance are:**

1. At any instance, a process will be having various information associated with it like id, state, priority, program counters, memory pointers, accounting information etc. Such information are stored in a data structure called as Process Control block (PCB),Created by kernel.
2. It is an important tool that helps the OS support multiple processes and provide for multiprocessing.
3. It contains sufficient information such that if an interrupt occurs, the process can begin from the point where it left later as if nothing had happened.
4. The blocks are read and/or modified by every module in the OS including
   1. Scheduling
   2. Resource allocation
   3. Interrupt processing
   4. Performance monitoring and analysis
5. It can be said that the process control block defines the state of OS.
6. However, there comes the issue of Security; a bug in a single routine say the interrupt handler could damage the PCB which in turn can make it difficult for OS to manage the affected process.
7. Changes made to the Process control block (like semantics or design changes) can affect various modules.

PCB is mostly stored as per-process kernel stack which is in the kernel space and kernel has access to this which is kept protected from any users.

PCB store the following information for a process.

**Process state: -**

A process can be new, ready, running, waiting, etc.

**Program counter: -**

The program counter lets us know the address of the next instruction, which should be executed for that process.

**CPU registers:** -This component includes accumulators, index and general-purpose registers, and information of condition code.

**CPU scheduling information: -**

This component includes a process priority, pointers for scheduling queues, and various other scheduling parameters.

**Accounting and business information: -**

It includes the amount of CPU and time utilities like real time used, job or process numbers, etc.

**Memory-management information: -**

This information includes the value of the base and limit registers, the page, or segment tables. This depends on the memory system, which is used by the operating system.

**I/O status information: -**

This block includes a list of open files, the list of I/O devices that are allocated to the process, etc.

**Context switching: -**

Context Switching is the switching of CPU from one process to another process. Context switching means storing the process state so that we can reload the process when needed, and the execution of the process can be resumed from the same point later. Context Switching is the characteristic of a multitasking operating system. In context switching, one CPU can be shared among several processes. In other words, context switching is the mechanism that permits a single CPU to handle several threads or processes without the need for extra processors.

In context switching, processes are switched so quickly that the user gets the myth that all processes are running simultaneously.

**Relations with PCB: -**

Some information are store in PCB for context switching for reloading of processes, it is happened whenever CPU switch another process and then process is reload in a time slice.

The information stored in PCB for context switching are:

1. Program counter
2. Information related Scheduling
3. Accounting information
4. Base and limits registers
5. Changed state

**Critical section problem**: -

Each process has a segment of code(instructions), that part or section is the critical section (CS) of that process in which process may changing resources. Now the critical section problem is to implement such a solution, which can be used by the processes to cooperate when they share common resources.

One process at a time can be executed in its critical section no others processes are allowed, To execute its critical section, a process must take care of the three properties

i): mutual exclusion ii) progress iii) bounded wait.

Possessing these three properties a process can execute its critical section successfully.

**Peterson Solution: -**

A commonly used solution to critical section problems is Peterson's solution. This was an algorithm which was developed by a computer scientist Peterson that's why it is named as a Peterson's solution.

In this solution, when a process is executing in a critical state, then the other process only executes the rest of the code, and the opposite can happen. This method also helps to make sure that only a single process runs in the critical section at a specific time.

**Semaphore Solution: -**

It is another algorithm or solution to the critical section problem. Semaphore is a simple variable that is non-negative and shared between threads, It is a signal -ing mechanism and a thread that is waiting on a semaphore, which can be signaled by another thread.

It uses two operations

1)wait, and 2) signal for the process synchronization.

**Limitation of both solution: -**

* Peterson's solution works for two processes, but this solution is best scheme in user mode for critical section.
* Peterson's solution is a time wasting solution so CPU time is wasted. And this problem can come in any of the busy waiting solution.
* Semaphores solution is a complex solution so the wait and signal operations must be implemented in the correct order to prevent deadlocks.
* Semaphores solution might be on priority based, where low priority processes may access the critical section first and high priority processes later.