**Chapter Three**

**Process**

A process is a program in execution. A process is more than the program code. It also includes the current activity, as represented by the value of the program counter and the contents of the processor's registers.

**3.1 Process state**

The state of a process is defined in part by the current activity of the process. Each process may be in one of the following states:

* + **New:** The process is in the stage of being created.
	+ **Running:** The CPU is working on this process's instructions.
	+ **Waiting:** The process cannot run at the moment, because it is waiting for some resource to become available or for some event to occur. For example the process may be waiting for keyboard input and disk access request.
	+ **Ready:** The process has all the resources available that it needs to run, but the CPU is not currently working on this process's instructions.
	+ **Terminated:** The process has completed.



Figure ( 3.1) process state

**3.2 Process Control Block**

Each process is represented by a process control block (PCB). A PCB contains many pieces of information associated with a specific process, such as:

* **Process State** - Running, waiting, etc., as discussed above.
* **Process ID**, and parent process ID.
* **CPU registers and Program Counter** - this need to be saved and restored when swapping processes in and out of the CPU.
* **CPU-Scheduling information** - Such as priority information and pointers to scheduling queues.
* **Memory-Management information** - E.g. page tables or segment tables.
* **Accounting information** - user and kernel CPU time consumed, account numbers, limits, etc.
* **I/O Status information** - Devices allocated, open file tables, etc.



Figure ( 3.2 ) PCB process control block



Figure (3.3) Diagram showing CPU switch from process to process

**3.3 Process Scheduling**

A uniprocessor system can have only one running process. If more processes exist, as in a multiprogramming system, there will be only one process running and the rest must wait until the CPU is free and can be rescheduled.

* **Scheduling Queues**
* All processes are stored in the **job queue.**
* Processes in the Ready state are placed in the **ready queue.** A new process as entering the system is put in a queue called a ready queue. It waits in the ready queue until it is selected for execution
* Processes waiting for a device to become available or to deliver data are placed in **device queues**. There is generally a separate device queue for each device.



Figure (3.4) Queueing-diagram representation of process scheduling.

Some systems also employ a**medium-term scheduler**. When system loads get high, this scheduler will swap one or more processes out of the ready queue system for a few seconds, in order to allow smaller faster jobs to finish up quickly and clear the system



Figure (3.5 ) Addition of a medium-term scheduling to the queueing diagram

* **Scheduler**

A process migrates between the various scheduling queues throughout its lifetime. The operating system must select processes from these queues in some fashion. The selection process is carried out by the appropriate scheduler. There are two types of scheduling algorithms categorized according to the frequency of their execution.

* **Long-term scheduler** **(job scheduler**) which selects a process from the job pool and loads them into the MM.
* **Short-term scheduler** (**CPU scheduler**) selects a process from the ready queue and allocate it to the CPU.
* **Context Switch**

Switching the CPU to another process requires saving the state of the old process and loading the saved state for the process. This task is known as a context switch.

**3.4 Operation on Processes**

The process in the system can execute concurrently, and they must be created and deleted dynamically.

* **Process Creation**

A process may create several new processes during the course of execution. The creating process is called a parent process, whereas the new processes are called the children.

When a process is created it obtains various resources and initialization values that may be passed along from the parent process to the child process.

* **Process Termination**

A process terminates when it finishes executing its final statement and asks the operating system to delete it. At that point, the process may return data to its parent process and the OS deallocates all the physical and logical resources that are previously allocated to that process.

**3.5 Cooperating Processes**

The concurrent process executing in the operating system may be either independent processes that do not share any data or cooperating that affects each other.

**3.6 Inter process Communication**

The cooperating processes can communicate in a shared memory environment. The scheme requires that these processes share a common buffer pool. Another way to achieve the same effect for the operating system is provided via interprocess communication (IPC).

**IPC provides a mechanism to allow processes to communicate and synchronize their actions without sharing the same address space**. This technique is useful for distributed systems. IPC is provided by a message-passing system.