**Process Management**

* **Process:** aprogram in execution.
* **Process in memory is divided into four sections:**

1. **Text section:** comprises the compiled program code.
2. **Data section:** stores global and static variables.
3. **Heap section:** is used for dynamic memory allocation.
4. **Stack section:** is used for local variables.

|  |
| --- |
| Stack |
|  |
| Heap |
| Data |
| Text |

**Figure (1): Process in Memory**

**Process State:**

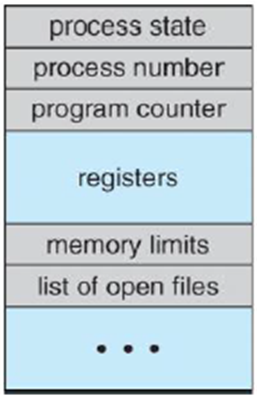
1. **New:** The process is being created.
2. **Ready:** The process is waiting to be assigned to a processor.
3. **Running:** Instructions are being executed.
4. **Waiting:** The process is waiting for some event to occur.
5. **Terminated:** The process has finished execution.

**Figure (2): Process State Diagram**

**Process Control Block (PCB)**

The Information associated with each process are the following:

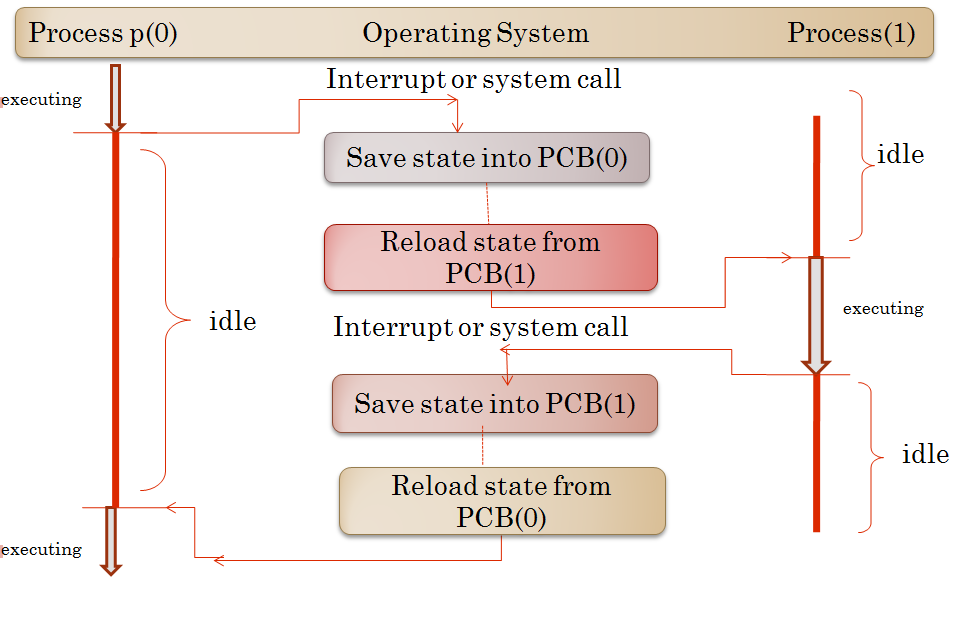
* 1. Process State.
  2. Process ID.
  3. Program Counter.
  4. CPU Registers.
  5. CPU Scheduling Information.
  6. Memory – Management Information.
  7. Accounting Information.
  8. I/O Status Information.



**Figure (3): Process Control Block (PCB)**

**Context Switch:**

* When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process via a context switch.
* Context of a process represented in the PCB.
* Context-switch time is overhead; the system does no useful work while switching.

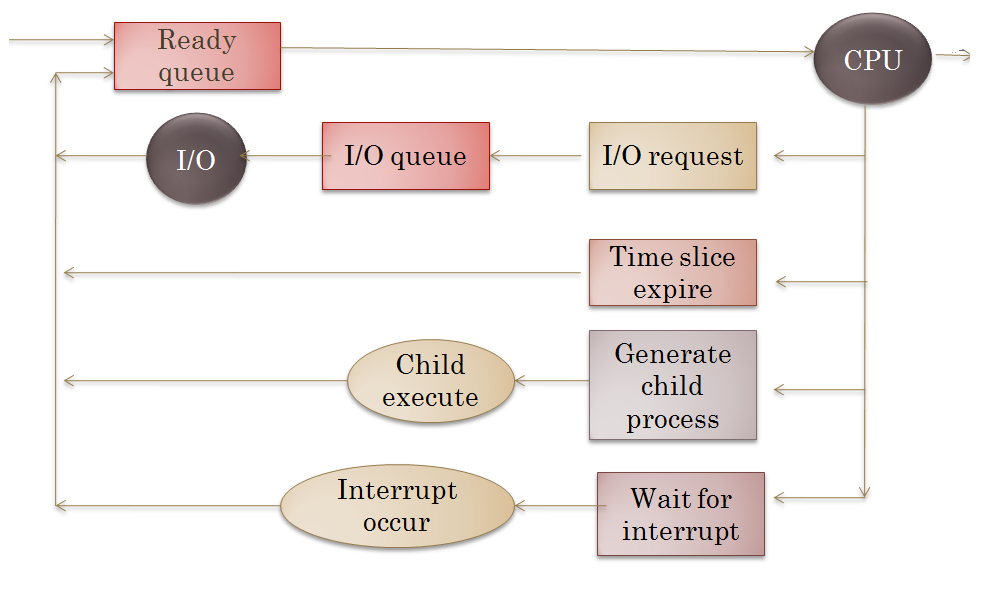


**Figure (4): CPU Switch from Process to Process**

**Process Scheduling Queues**

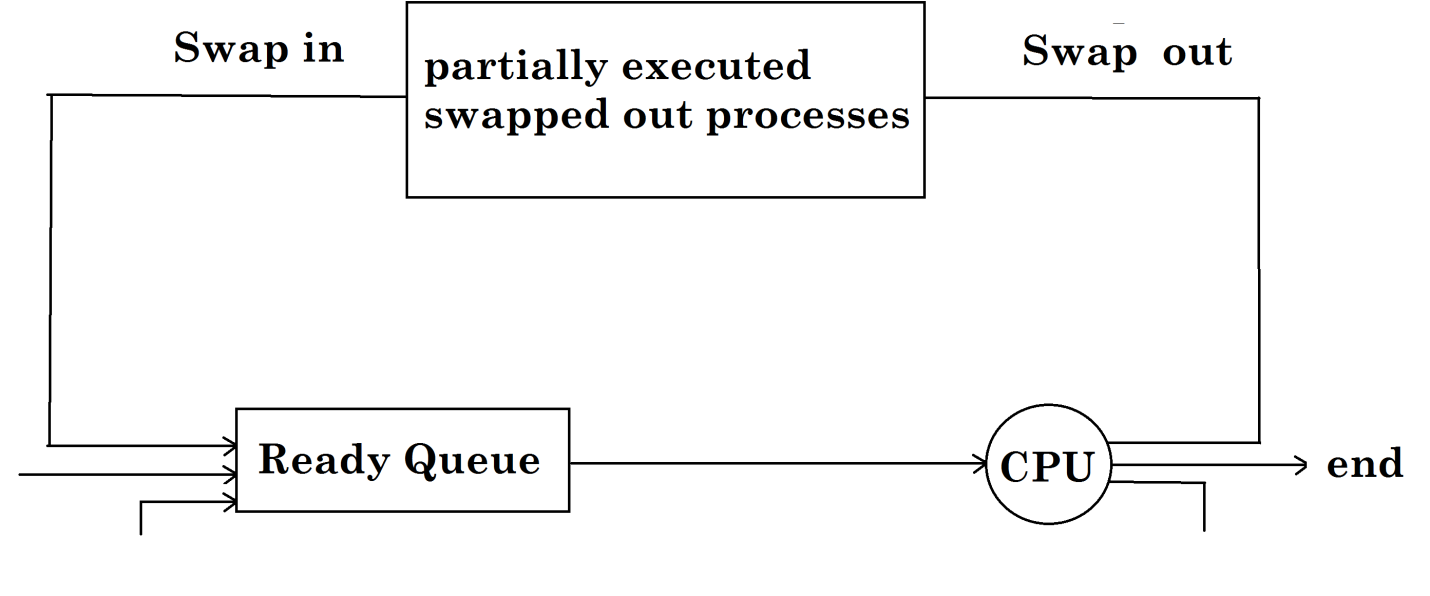
1. **Job queue:** set of all processes in the system.
2. **Ready queue:** set of all processes residing in main memory, ready and waiting to execute.
3. **Device queue or waiting queue:** set of processes waiting for an I/O device.

Note: The processes migrate among the various queues.

**Figure (5): Representation of Process Scheduling**

**Schedulers**

* **Long-term scheduler (or job scheduler)**: selects which processes should be brought into the ready queue.
* **Short-term scheduler (or CPU scheduler)**: selects which process should be executed next and allocates CPU.

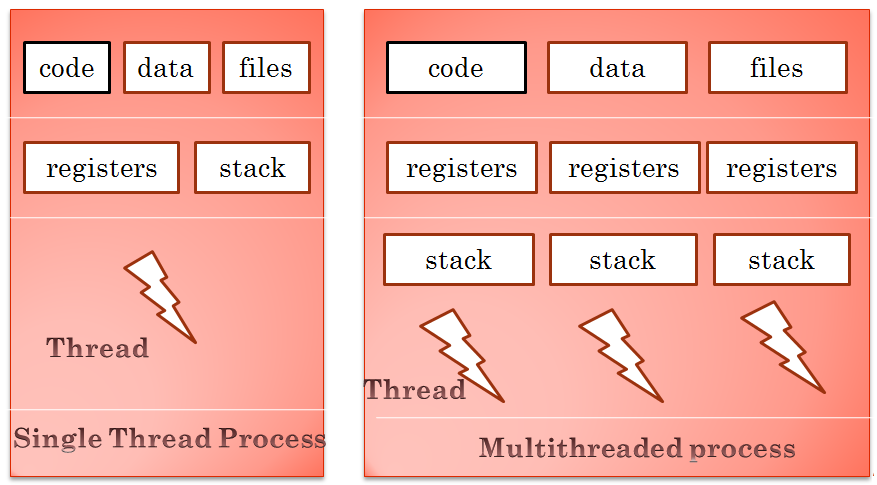
****

**Figure (6): The Addition of Medium Term Scheduling**

* Short-term scheduler is invoked very frequently (milliseconds) => (must be fast)
* Long-term scheduler is invoked very infrequently (seconds, minutes) => (may be slow)
* The long-term scheduler controls the degree of multiprogramming
* Processes can be described as either:
* **I/O-bound process:** spends more time doing I/O than computations, many short CPU bursts. Where the CPU burst is the amount of time the process uses the processor before it is no longer ready.
* **CPU-bound process:** spends more time doing computations; few very long CPU bursts.

**Models of Process**

* Two models of process
  + Single thread:
  + Multiple threads:
* Thread: Can be defined as a smaller unit within process, which can be scheduled and executed. or it is a basic unit of CPU utilization.

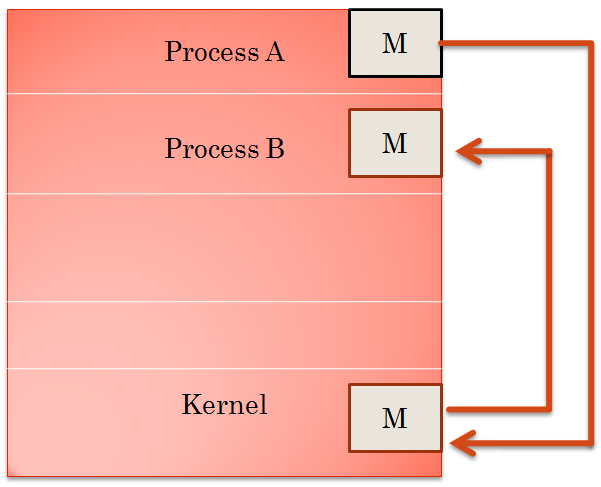


**Figure (7): Single thread and Multithreaded Processes**

**Inter Process Communication**

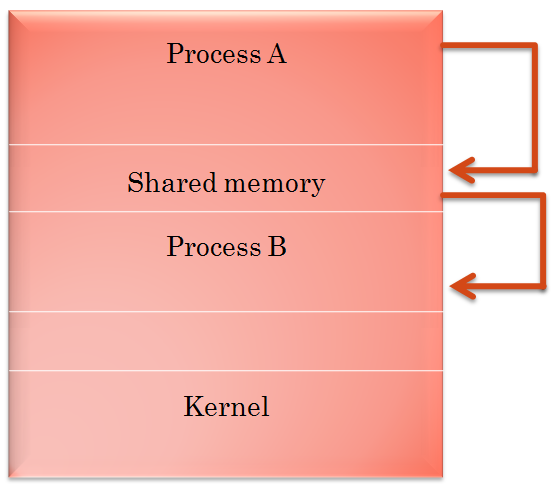
* Processes executing in operating system independent processes or cooperating processes.
* A process is independent if it cannot affect or be affected by the other processes executing in the system.
* A process is cooperating if it can affect or be affected by the other processes executing in system.
* Cooperating processes require an inter-process communication (IPC):
  + Message Passing.
  + Shared Memory.

**Message Passing:**

****

**Figure (8): Message Passing**

**Shared Memory:**

****

**Figure (9): Shared Memory**

**Operation on Process**

**Process Creation**

* + Address Space
* UNIX examples (POSIX API)
  + Fork system call create new process
  + Exec system call used after a fork to replace the process memory space with new program
* Windows examples (Win32 API)
  + Create Process system call to create new process.
  + Zero Memory system call. To allocate memory.
* Java API
  + Process Builder system call to create new process.
  + Buffered Reader system call to allocate memory.

**Process Termination**

* Process executes last statement and asks the operating system to delete it (exit).
  + Output data from child to parent.
  + Process’ resources are de-allocated by operating system.