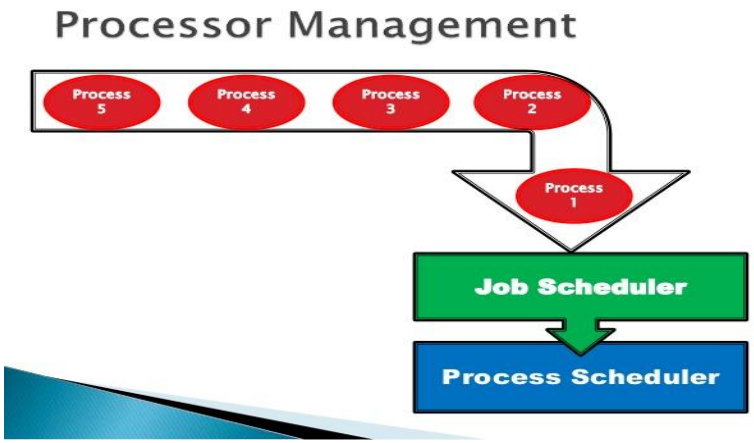
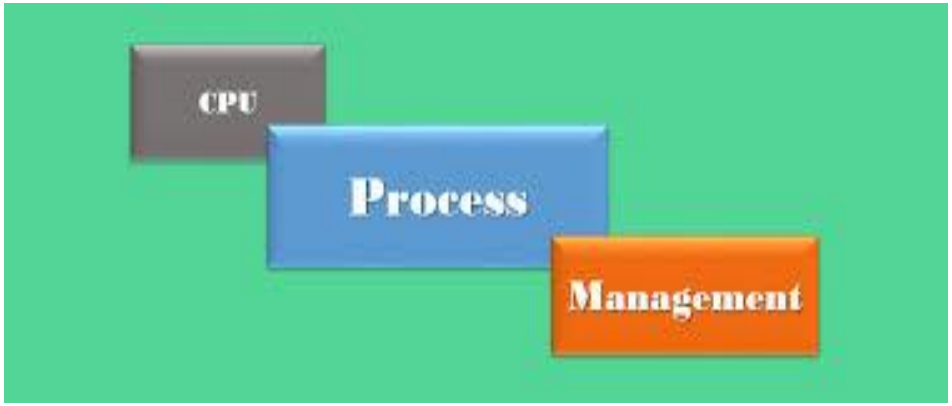
AMIRAJ

COLLEGE OF ENGINEERING & TECHNOLOGY

CHAPTER -2
PROCESS MANAGEMENT





Subject:- OS **Code:-3140702**	**Prepared by:** **Asst.Prof.Foram Patel** **(Computer Department,ACET)**	AMIRAJ COLLEGE OF ENGINEERING & TECHNOLOGY

WHAT IS PROCESS

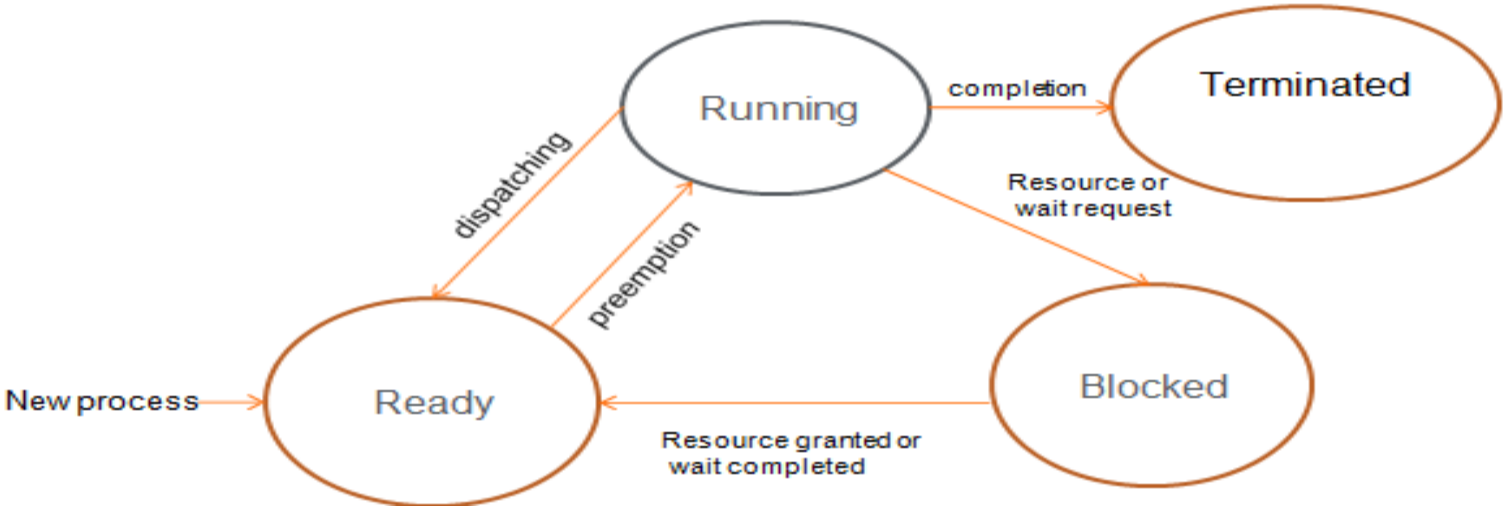
- Process is a Execution of program.
- Process is an active entity that requires a set of resource, including a processor, program counter , register to perform its function.
- Process execution perform in sequential order.
- Each process has its own address space.
- Address space divided into different region. 1) Text region 2) Data region 3) stack region
- A process can run to completion only when all requested hardware and software resources have been allocated to the process.

WHAT IS PROCESS MANAGEMENT

- The operating system is responsible for the following activities in connection with process management:
- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

PROCESS STATES

- The process state is an indicator of the nature of the current activity in a process.
- The notion of process state is introduced to simplify control of process by the operating system.

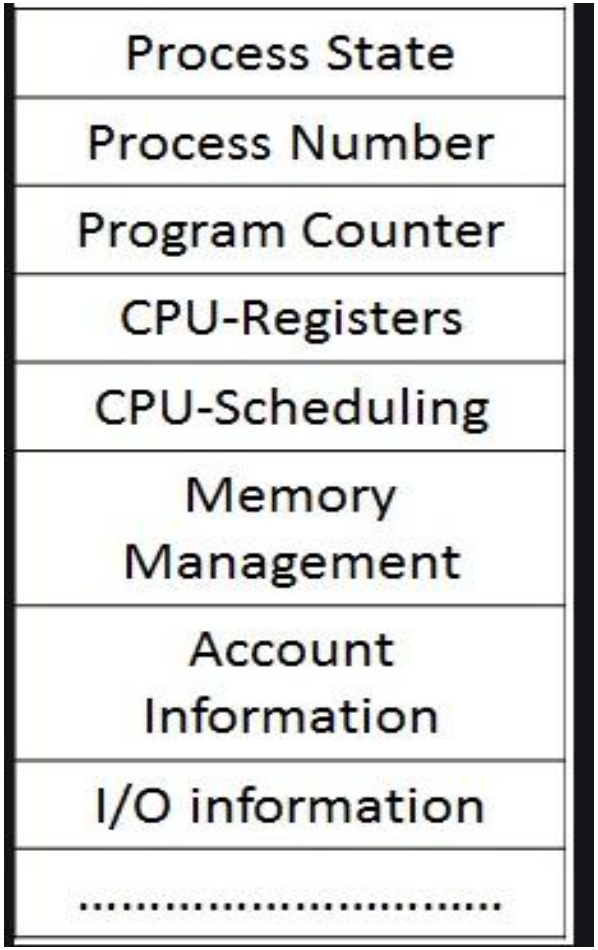


PROCESS STATE TRANSITION

- **Ready → Running**
- The process is dispatch able .
- **Blocked →Ready**
- The process is pre-empted because the OS decides to schedule some other process.
- **Running →Blocked**
- Process requests an I/O operation, memory or some other resources.
- Process wishes to wait for some action by another process.
- **Running →Terminated**
- Proper execution done by process.
- Termination by a parent.

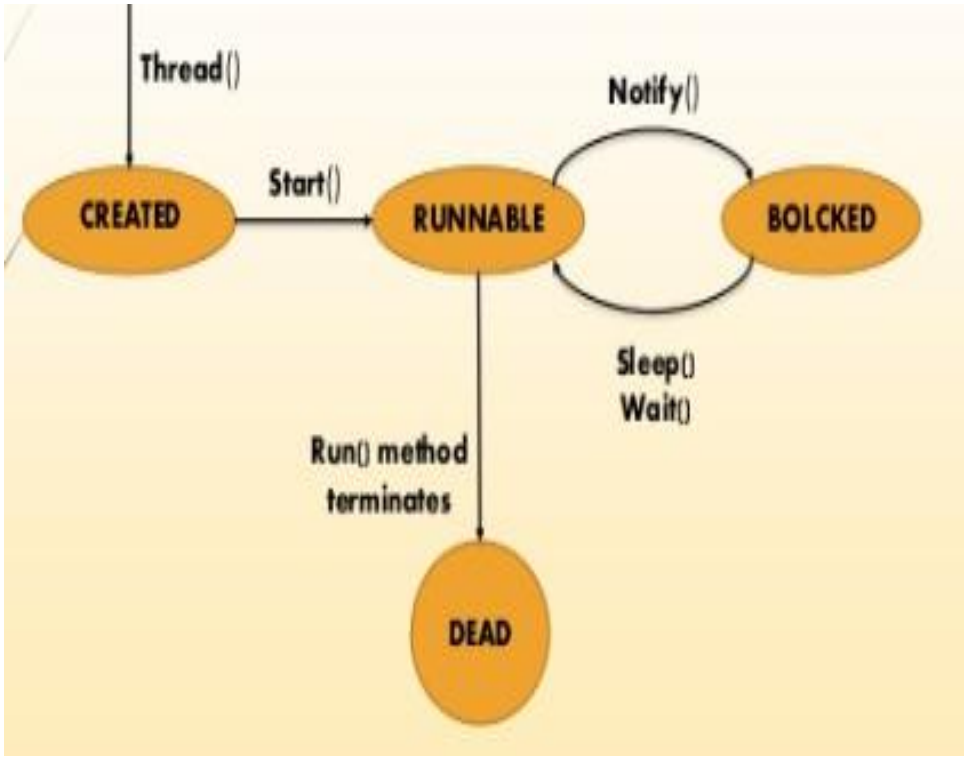
PROCESS CONTROL BLOCK

- **Process state –** It stores the respective state of the process.
- **Process number –** Every process is assigned with a unique id known as process ID or PID
- **Program counter –** It stores the counter which contains the address of the next instruction.
- **Register –** These are the CPU registers which includes: accumulator, base, registers.
- **Memory limits –** This field contains the information about memory management.
- **Open files list –** List of files opened for a process
- **I/O information:** Store information about I/O devices.



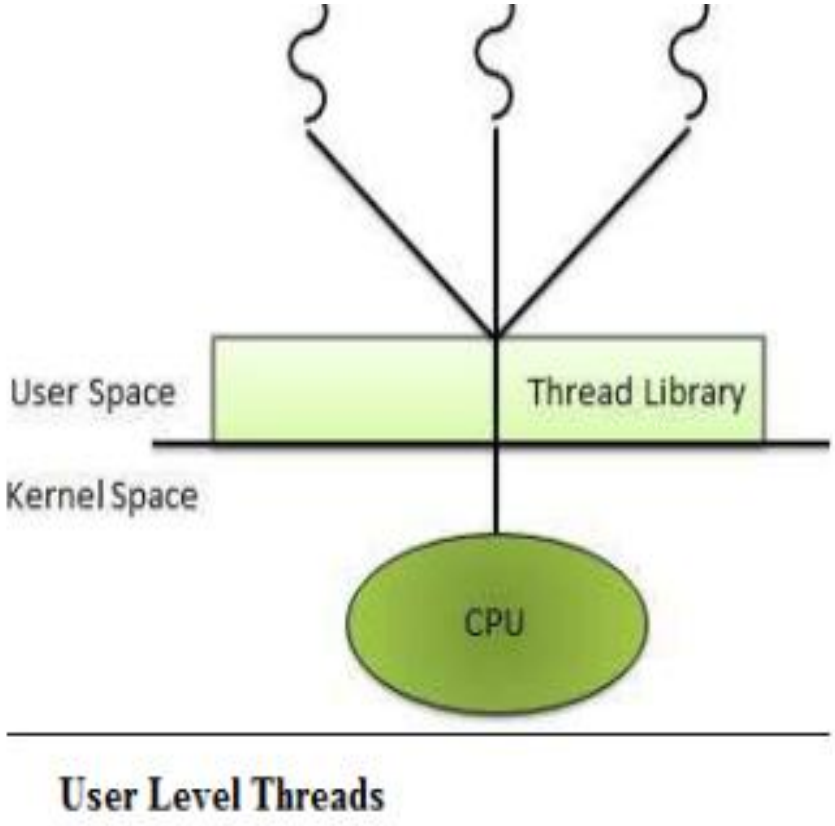
LIFECYCLE OF THREAD

- A process is divided into several light-weight processes, each light-weight process is said to be a thread.
- **The life cycle of thread is:**
- **New:** A thread that has just created.
- **Runnable:** The System assigns the processor to the thread means that the thread is being executed.
- **Blocked:** The thread is waiting for an event to occur or waiting for an I/O device.
- **Waiting:** A sleeping thread becomes ready after the designated sleep time expires.
- **Terminated/Dead:** The execution of the thread is finished.



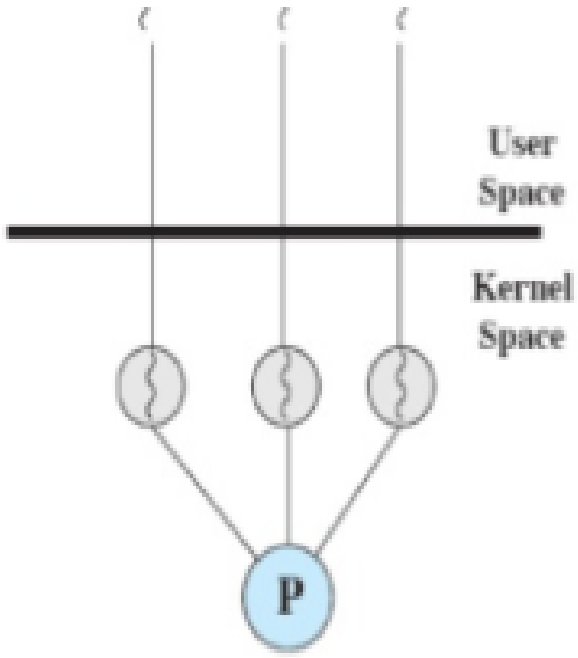
TYPES OF THREAD:

- **User level thread**: User managed threads
- The thread library contains code for creating and destroying threads.
- User level thread can run on any operating system.
- Scheduling can be application specific in the user level thread.
- User level threads are fast to create and manage.
- Kernel doesn't know about the user level thread



TYPES OF THREAD:

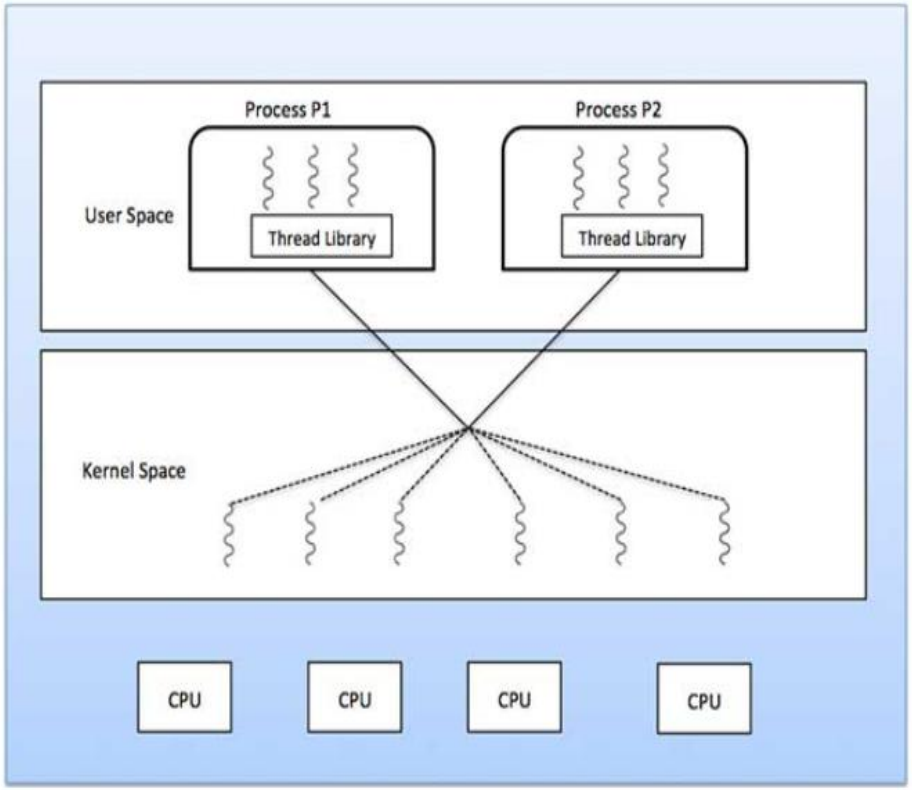
- **Kernel level thread**: Kernel knows and manages the threads.
- OS kernel provides system call to create and manage threads.
- The Kernel maintains information for the process.
- The Kernel performs thread creation, scheduling and management in Kernel space.
- Kernel threads are slower to create and manage.
- If one thread in a process is blocked, the Kernel can schedule another thread of the same process.



Kernel Level Threads

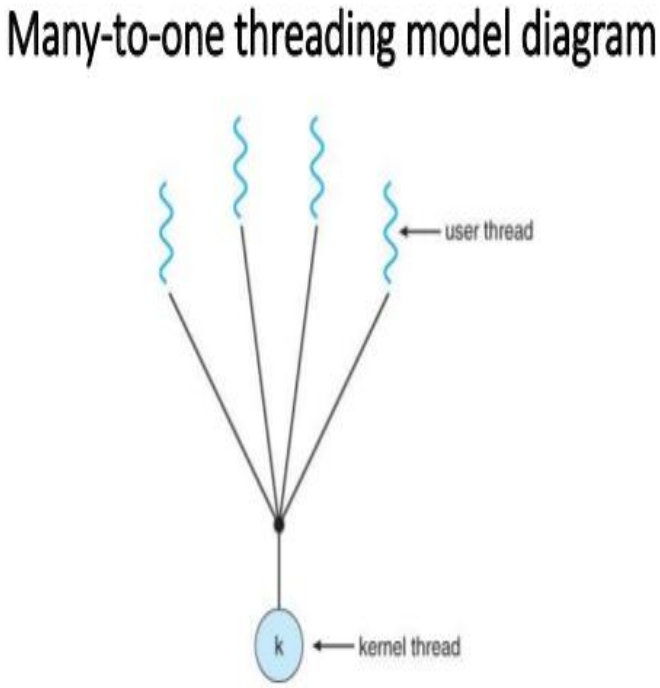
MULTITHREADING MODELS

- **Many to many:**
- Multiplexes any number of user threads onto an equal or smaller number of kernel threads.
- Users can create any number of the threads.
- Blocking the kernel system calls does not block the entire process.
- Processes can be split across multiple processors.



MULTITHREADING MODELS

- **Many to One:**
- Many-to-one model maps many user level threads to one Kernel-level thread.
- When thread makes a blocking system call, the entire process will be blocked.
- Only one thread can access the Kernel at a time.
- Multiple threads are unable to run in parallel on multiprocessors.



MULTITHREADING MODELS

- **One to one:**
- This model provides more concurrency than the many-to-one model
- One to one relationship between kernel and user thread. kernel thread.
- The one to one model maps each of the user threads to a kernel thread.
- Windows 2000 use one to one relationship model