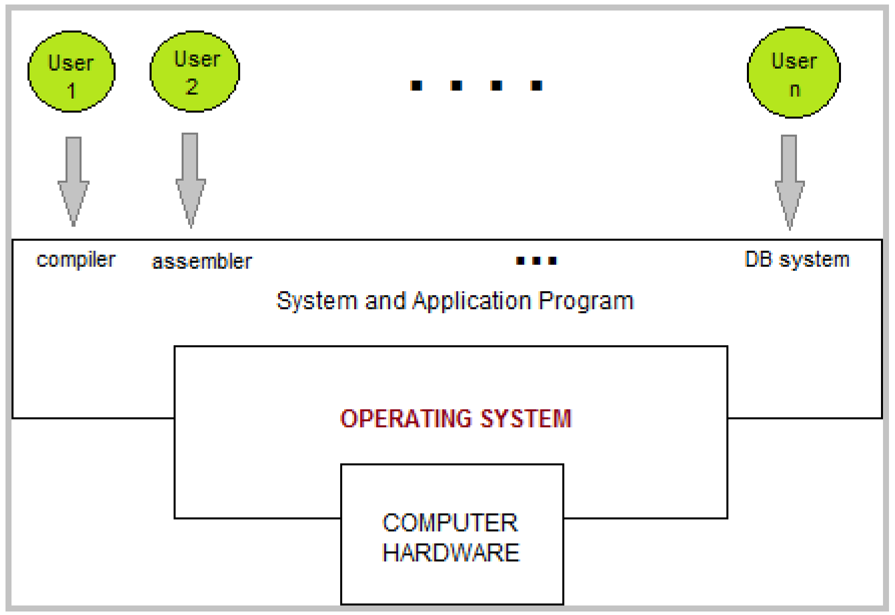
# Introduction to Operating Systems

## operating system

- operating system is the resource manager i.e. it can manage the resource of a computer system internally.
- The resources are processor, memory, files, and I/O devices.
- **In simple terms, an operating system is the interface between the user and the machine.**

Four Components of a Computer System



## Operating System Management Tasks

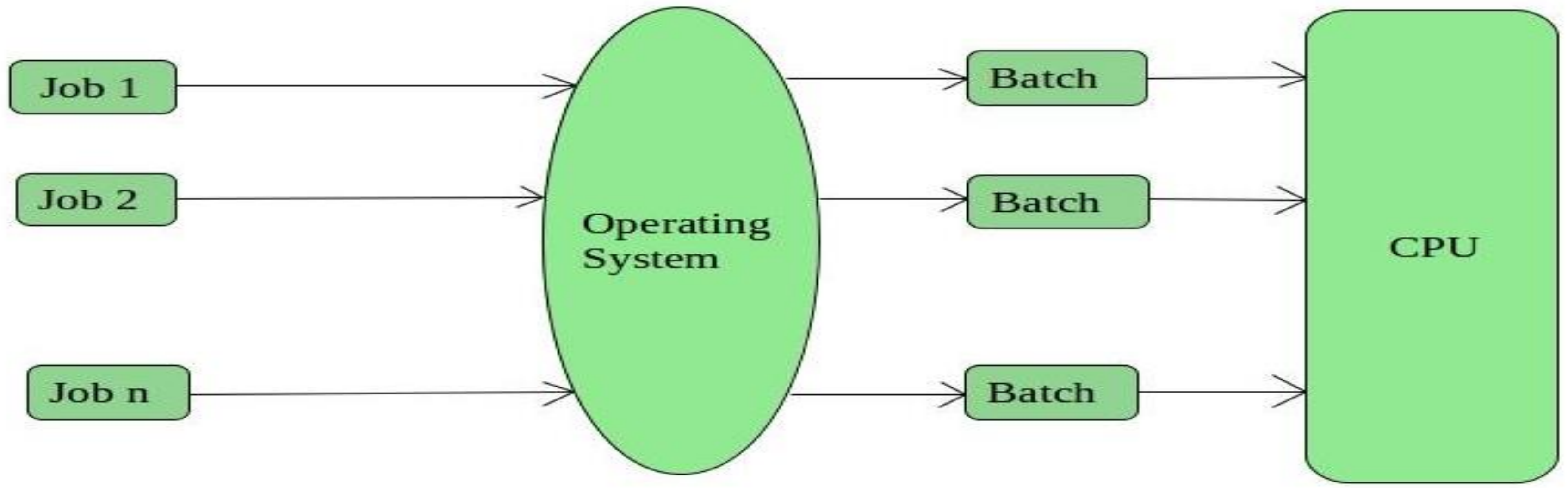
- **Processor management** --- which involves putting the tasks into order and pairing them into manageable size before they go to the CPU.
- **Memory management** --- which coordinates data to and from RAM (random-access memory) and determines the necessity for virtual memory.
- **Device management** --- which provides interface between connected devices.
- **Storage management** --- which directs permanent data storage.
- **Application** --- which allows standard communication between software and your computer.
- **User interface** --- which allows you to communicate with your computer.

## Types of Operating Systems

# Batch Operating System –

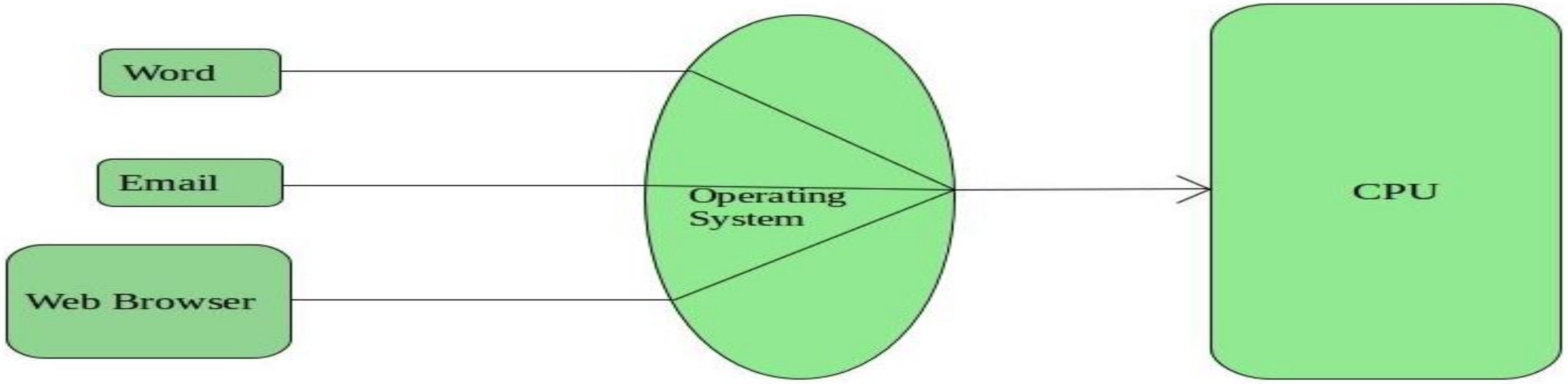
**Batch Operating System**

- This type of operating system do not interact with the computer directly.
- There is an operator which takes similar jobs having same requirement and group them into batches.
- It is the responsibility of operator to sort the jobs with similar needs.



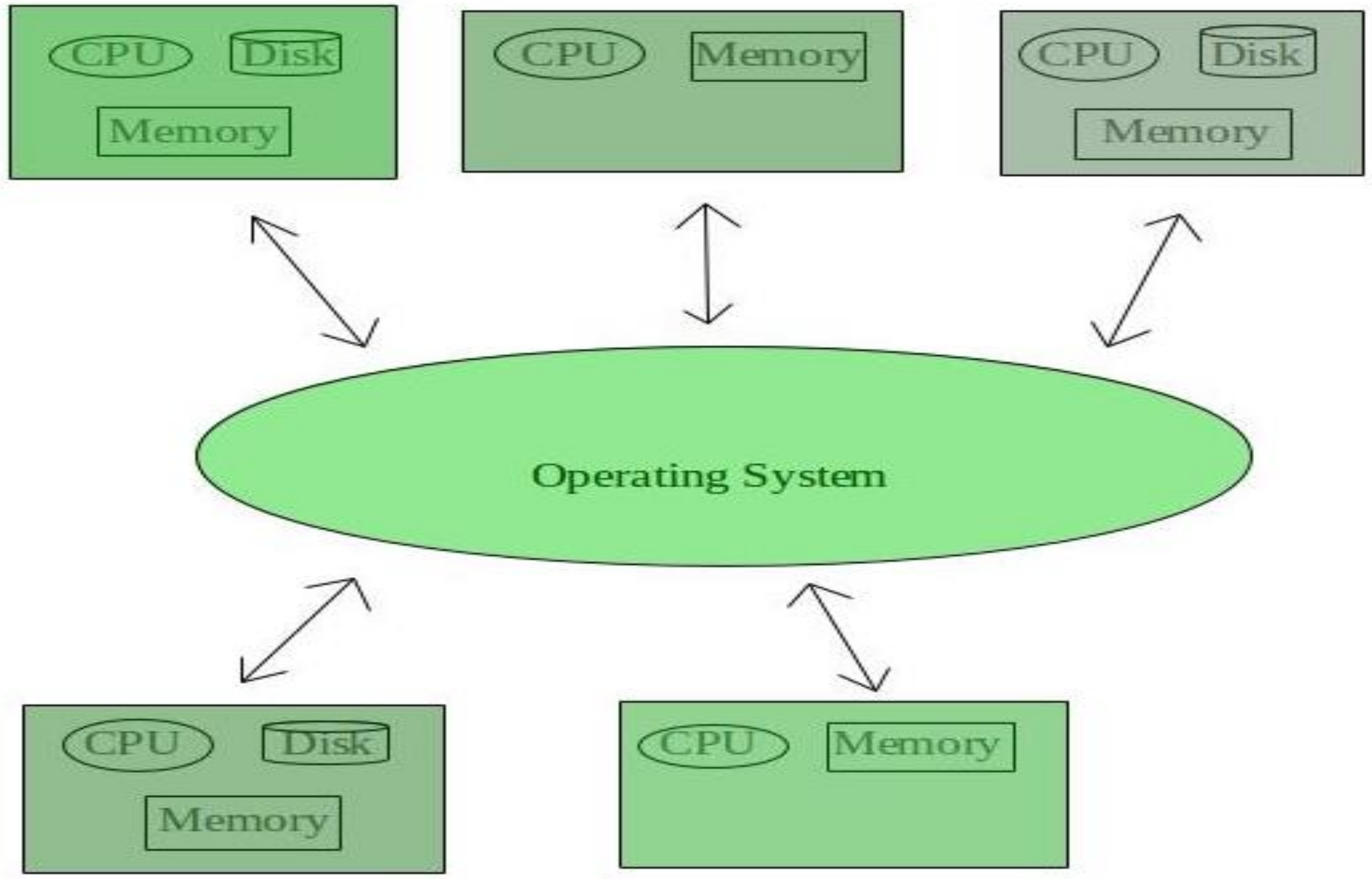
# Time-Sharing Operating Systems

- Each task has given some time to execute, so that all the tasks work smoothly.
- Each user gets time of CPU as they use single system.
- These systems are also known as Multitasking Systems.
- The task can be from single user or from different users also.
- The time that each task gets to execute is called quantum.
- After this time interval is over OS switches over to next task.



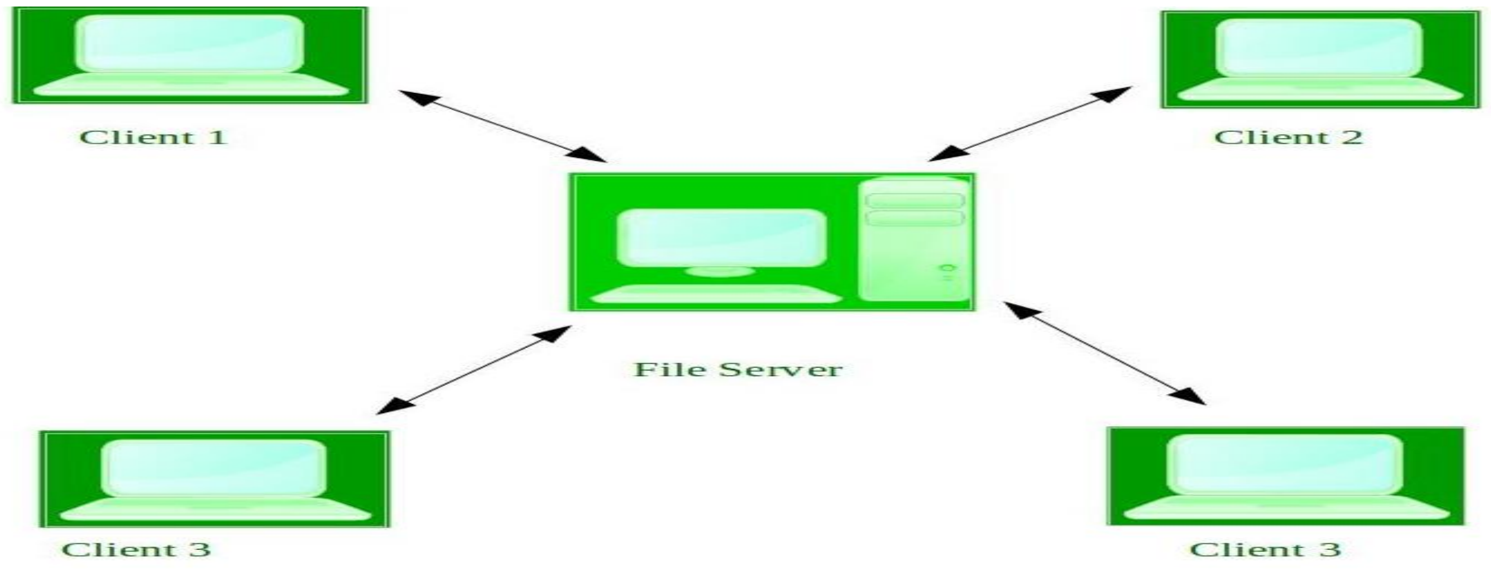
## Distributed Operating System

- These types of operating system is a recent advancement in the world of computer technology and are being widely accepted all-over the world
- Various autonomous interconnected computers communicate each other using a shared communication network.
- Independent systems possess their own memory unit and CPU.
- These are referred as **loosely coupled systems** or distributed systems.
- These systems processors differ in sizes and functions.
- The major benefit of working with these types of operating system is that it is always possible that one user can access the files or software which are not actually present on his system but on some other system connected within this network i.e., remote access is enabled within the devices connected in that network.


CPU
Disk
Memory
CPU
Memory
CPU
Disk
Memory
Operating System
CPU
Disk
Memory
CPU
Memory

# Network Operating System

- These systems runs on a server and provides the capability to manage data, users, groups, security, applications, and other networking functions.
- These type of operating systems allows shared access of files, printers, security, applications, and other networking functions over a small private network.
- Important aspect of Network Operating Systems is that all the users are well aware of the underlying configuration, of all other users within the network, their individual connections etc. and that's why these computers are popularly known as **tightly coupled systems**.



## Real-Time Operating System

- These types of OSs serves the real-time systems.
- The time interval required to process and respond to inputs is very small.
- This time interval is called **response time**.
- **Real-time systems** are used when there are time requirements are very strict like missile systems, air traffic control systems, robots etc.

## Two types of Real-Time Operating System are

### Hard Real-Time Systems:

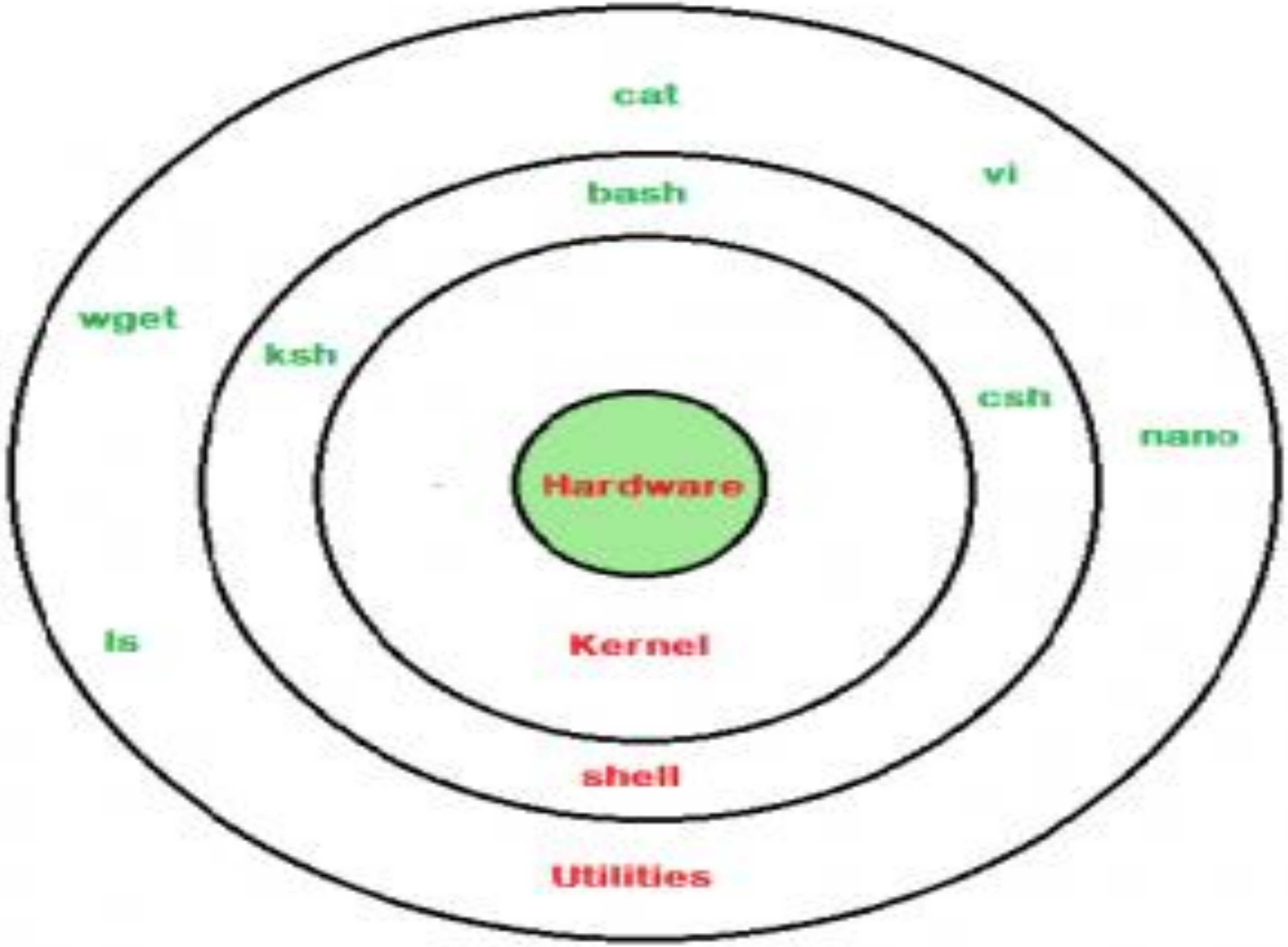
- These OSs are meant for the applications where time constraints are very strict and even the shortest possible delay is not acceptable.
- These systems are built for saving life like automatic parachutes or air bags which are required to be readily available in case of any accident.
- Virtual memory is almost never found in these systems.

### Soft Real-Time Systems:

- These OSs are for applications where for time-constraint is less

strict.

## Operating-System Structure


cat
vi
bash
wget
ksh
csh
nano
Hardware
ls
Kernel
shell
Utilities

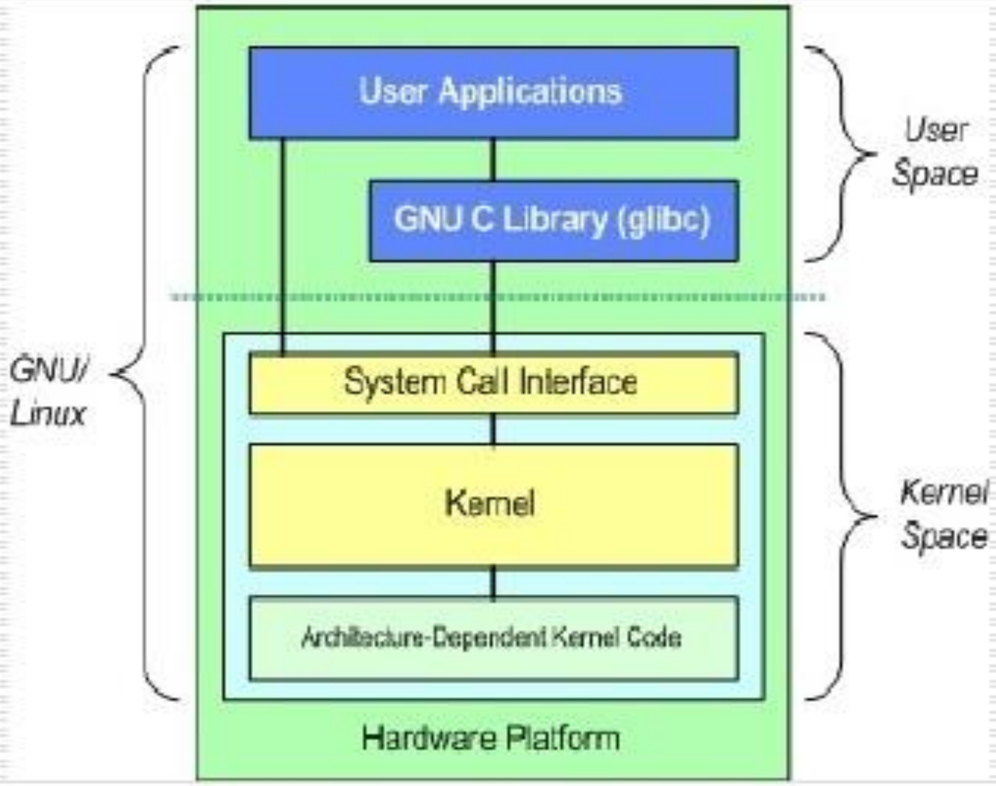
## User Space vs Kernel Space

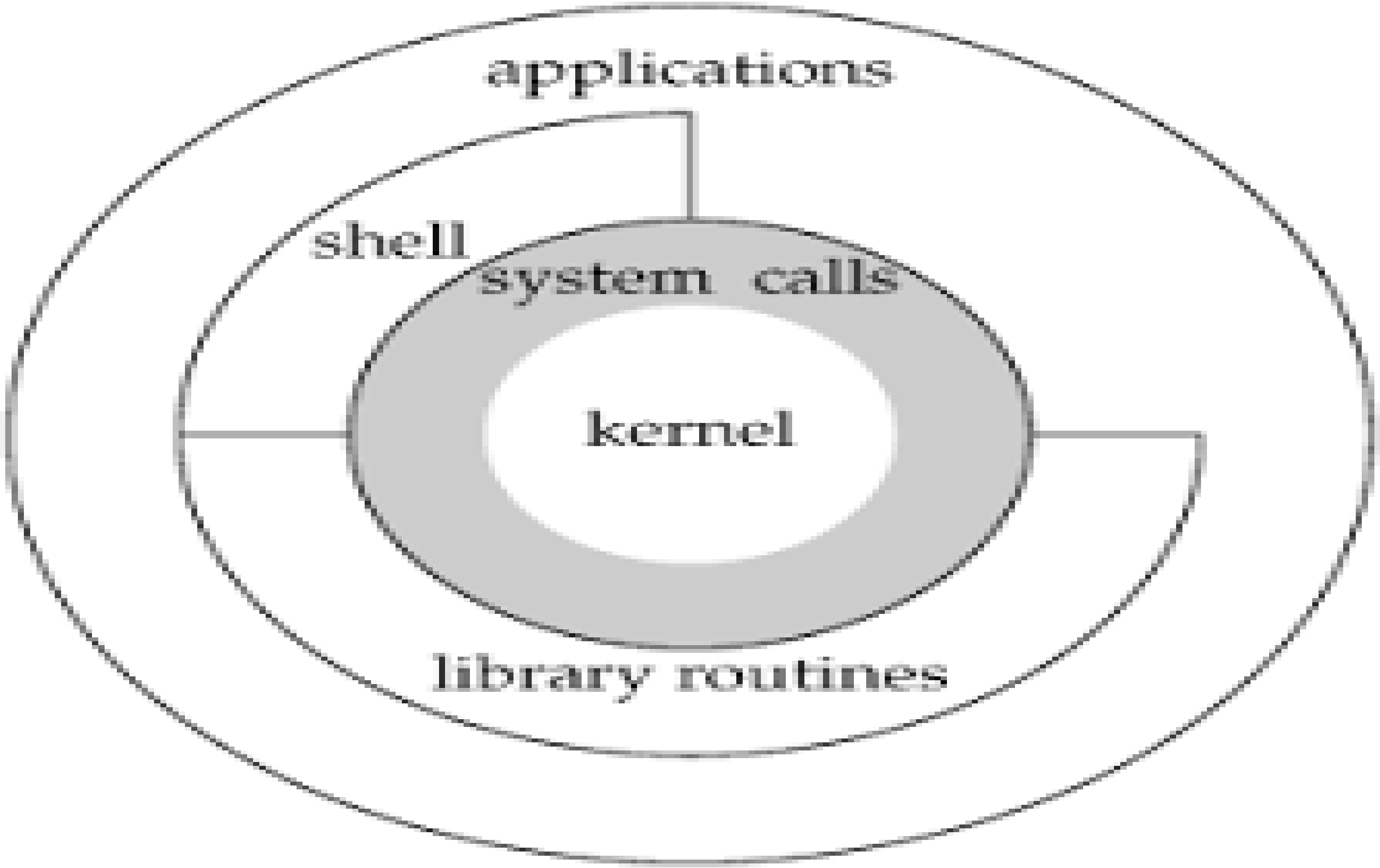
### User Space

- User space is the memory area where all user mode application work and this memory can be swapped out when necessary
- User space process normally runs in its own virtual memory space and unless explicitly requested, cannot access the memory of other processes.

### Kernel Space

- Kernel Space us strictly reserved for running the kernel (OS background process), kernel extensions and most device drivers
- Linux kernel space gives full access to the hardware, although some exceptions runs in user space. (The graphic system most people use with Linux does not run in kernel in contrast to that found in Microsoft Windows)




applications
shell
system calls
kernel
library routines

## Process

## What is a Process?

- A process is a program in execution.
- Process is not as same as program code but a lot more than it.
- A process is an 'active' entity as opposed to program which is considered to be a 'passive' entity.
- Attributes held by process include hardware state, memory, CPU etc.

# Process memory

**Process memory** is divided into four sections for efficient working :

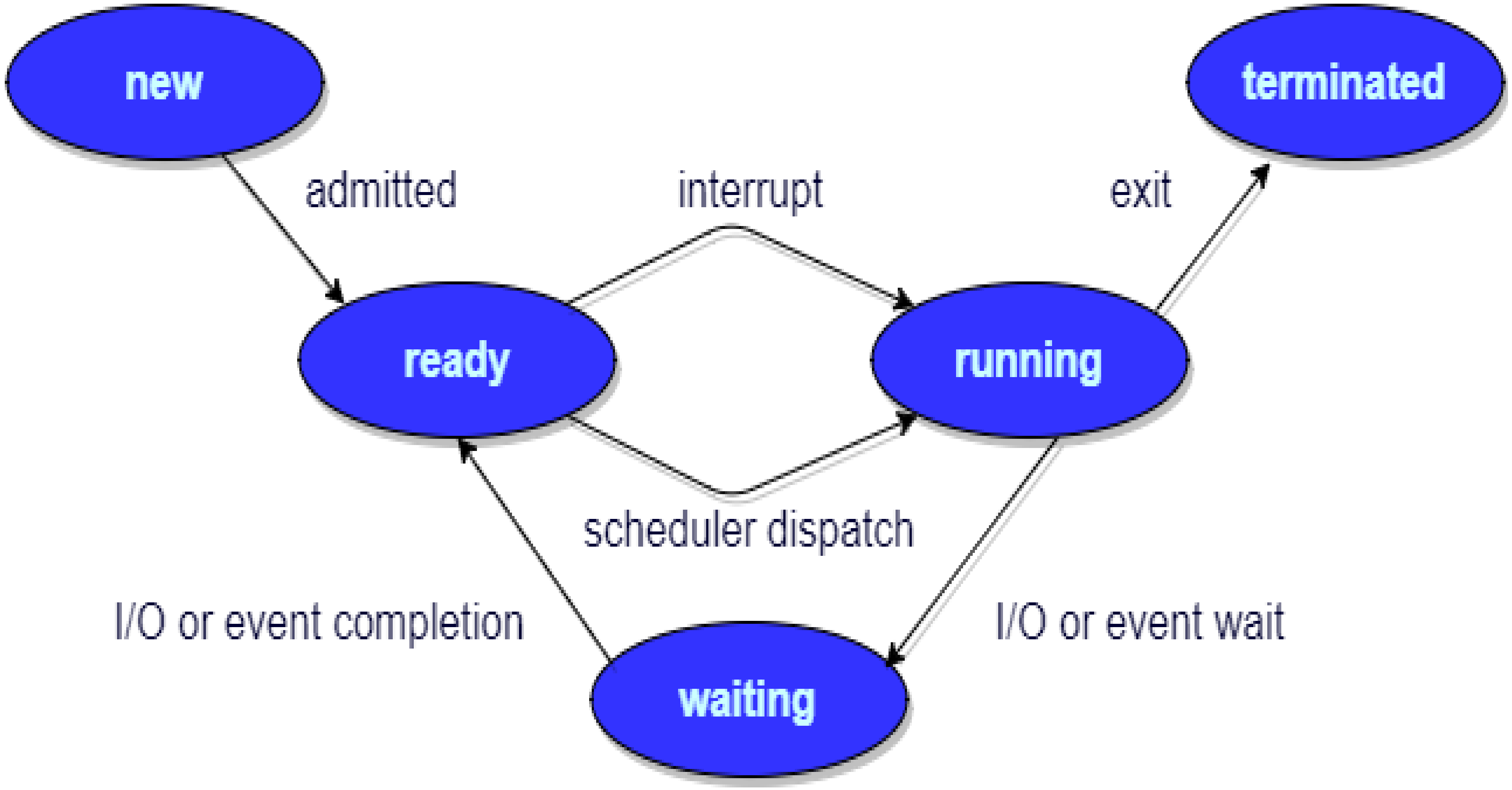
- **Text section** is made up of the compiled program code, read in from non-volatile storage when the program is launched.
- **Data section** is made up the global and static variables, allocated and initialized prior to executing the main.
- **Heap** is used for the dynamic memory allocation, and is managed via calls to new, delete, free, etc.
- **Stack** is used for local variables. Space on the stack is reserved for local variables when they are declared.

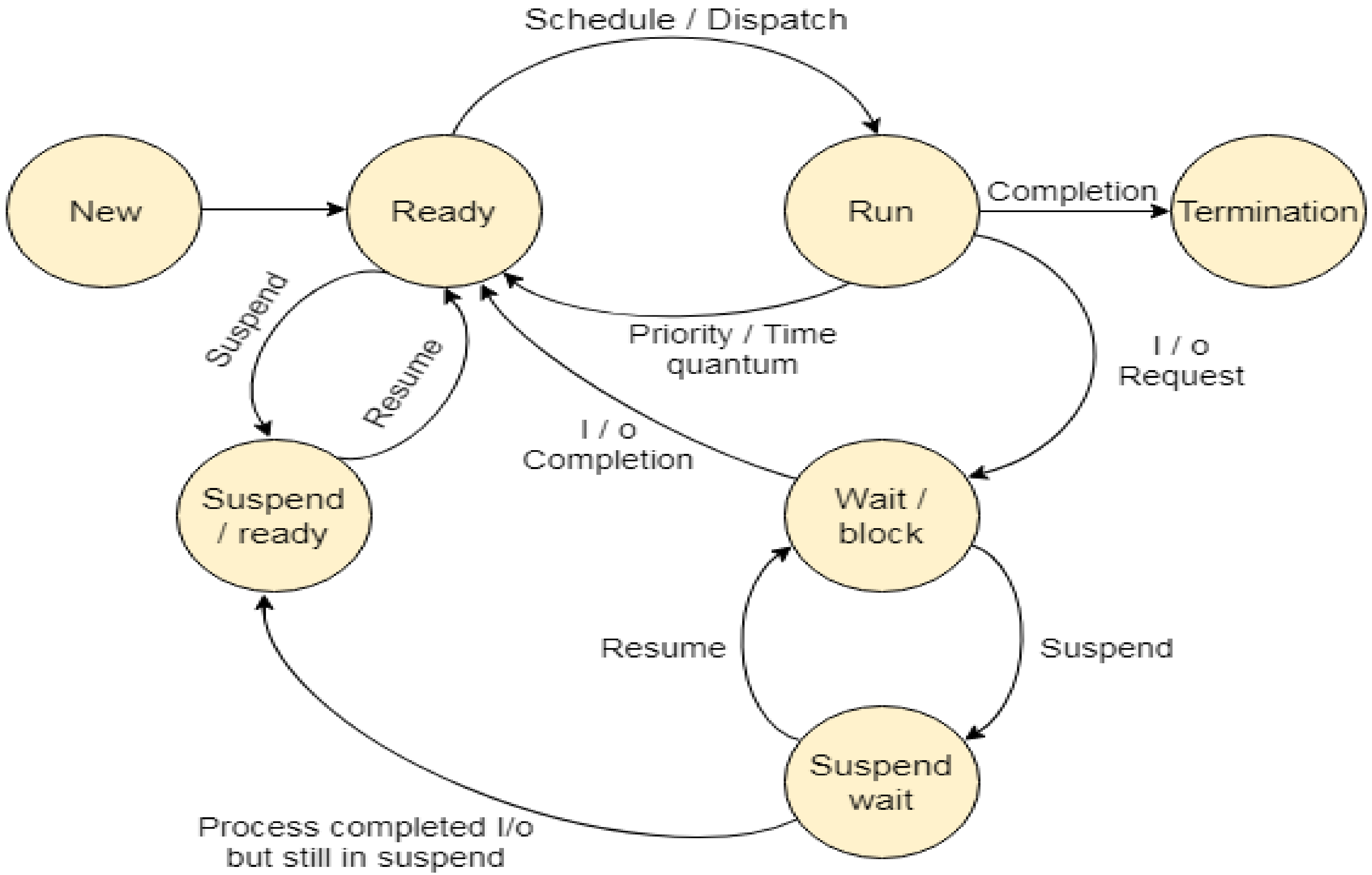
## Different Process States

Processes in the operating system can be in any of the following states:

- NEW- The process is being created.
- READY- The process is waiting to be assigned to a processor.
- RUNNING- Instructions are being executed.
- WAITING- The process is waiting for some event to occur(such as an I/O completion or reception of a signal).
- TERMINATED- The process has finished execution

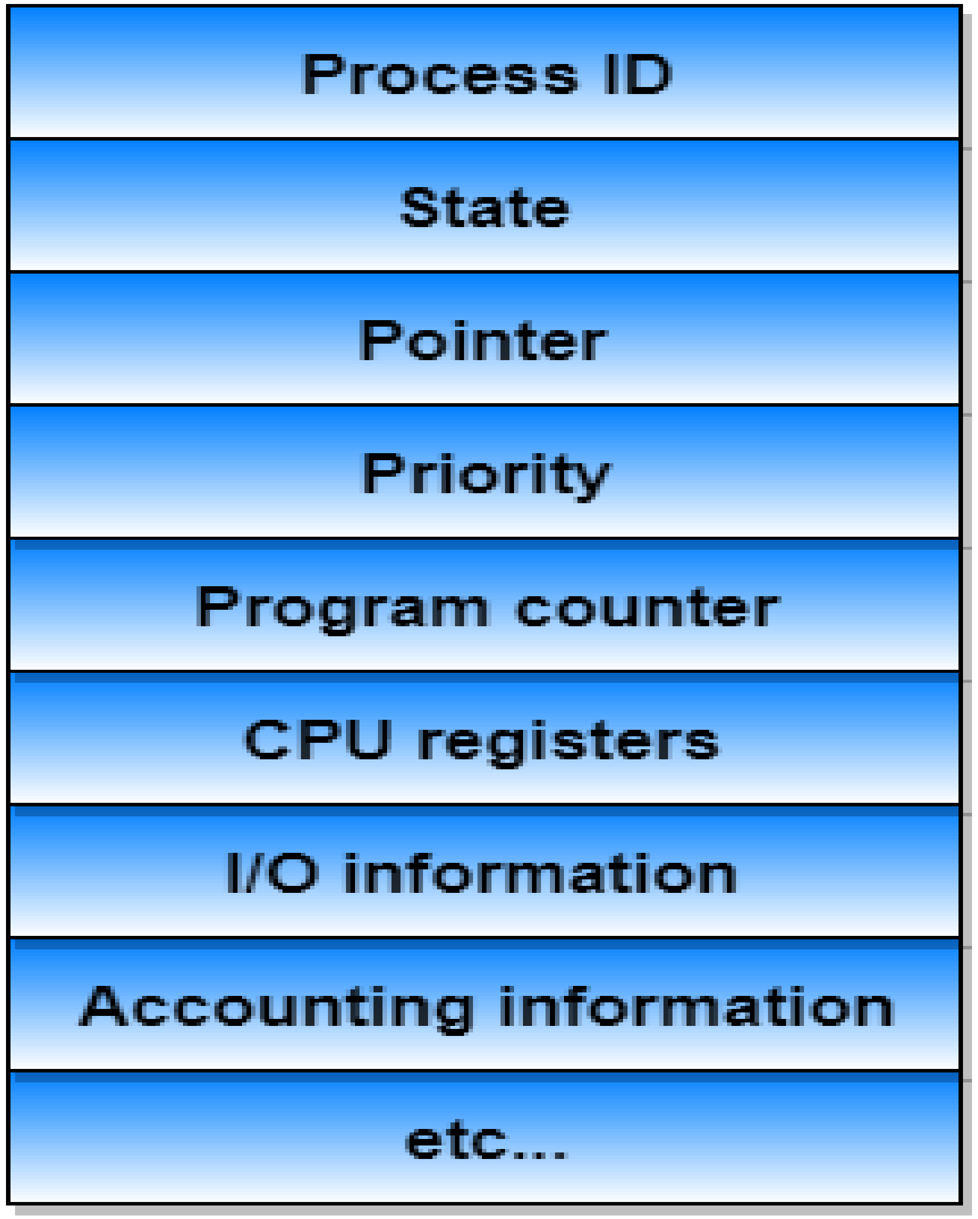
## Different Process States




Schedule / Dispatch
New
Ready
Run
Completion
Termination
Suspend
Resume
Priority / Time quantum
I / o Request
I / o Completion
Suspend / ready
Wait / block
Resume
Suspend
Suspend wait
Process completed I/o but still in suspend

## Process Control Block

- There is a Process Control Block for each process, enclosing all the information about the process. It is a data structure, which contains the following:
- **Process State**: It can be running, waiting etc.
- **Process ID** and the **parent process ID**.
- CPU registers and Program Counter. **Program Counter** holds the address of the next instruction to be executed for that process.
- **CPU Scheduling** information: Such as priority information and pointers to scheduling queues.
- **Memory Management information**: For example, page tables or segment tables.
- **Accounting information**: The User and kernel CPU time consumed, account numbers, limits, etc.
- **I/O Status information**: Devices allocated, open file tables, etc.


Process ID
State
Pointer
Priority
Program counter
CPU registers
I/O information
Accounting information
etc...

## What is Process Scheduling?

- The act of determining which process is in the **ready** state, and should be moved to the **running** state is known as **Process Scheduling**.
- The prime aim of the process scheduling system is to keep the CPU busy all the time and to deliver minimum response time for all programs.
- For achieving this, the scheduler must apply appropriate rules for swapping processes IN and OUT of CPU.

Scheduling fell into one of the two general categories:

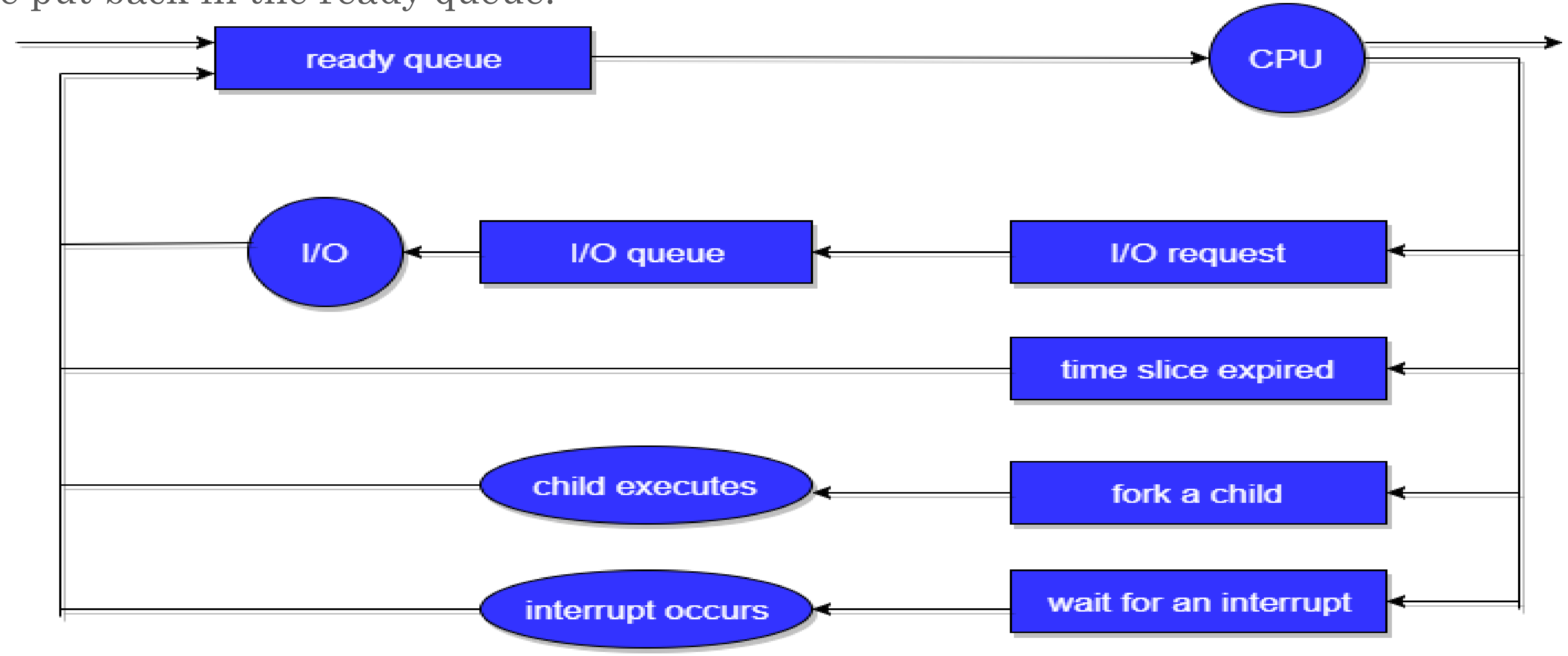
- **Non Pre-emptive Scheduling:** When the currently executing process gives up the CPU voluntarily.
- **Pre-emptive Scheduling:** When the operating system decides to favour another process, pre-empting the currently executing process

# What are Scheduling Queues?

- All processes, upon entering into the system, are stored in the **Job Queue**.
- Processes in the Ready state are placed in the **Ready Queue**.
- Processes waiting for a device to become available are placed in **Device Queues**.
- A new process is initially put in the **Ready queue**.
- It waits in the ready queue until it is selected for execution (or dispatched).

Once the process is assigned to the CPU and is executing, one of the following several events can occur:

- The process could issue an I/O request, and then be placed in the **I/O queue**.
- The process could create a new subprocess and wait for its termination.
- The process could be removed forcibly from the CPU, as a result of an interrupt, and be put back in the ready queue.



## Operations on Process

### Process Creation

- Through appropriate system calls, processes may create other processes.
- The process which creates other process, is termed the **parent** of the other process, while the created sub-process is termed its **child**.
- Each process is given an integer identifier, termed as process identifier, or PID. The parent PID (PPID) is also stored for each process.
- A child process may receive some amount of shared resources with its parent depending on system implementation.

There are two options for the parent process after creating the child :

- Wait for the child process to terminate before proceeding.
- Run concurrently with the child, continuing to process without waiting.
- It is also possible for the parent to run for a while, and then wait for the child later, which might occur in a sort of a parallel processing operation.

## Process Termination

Processes may also be terminated by the system for a variety of reasons, including :

- The inability of the system to deliver the necessary system resources.
- In response to a KILL command or other unhandled process interrupts.
- A parent may kill its children if the task assigned to them is no longer needed
- If the parent exits, the system may or may not allow the child to continue without a parent.

**Zombie Process:**

- A process which has finished the execution but still has entry in the process table to report to its parent process is known as a zombie process.
- A child process always first becomes a zombie before being removed from the process table.
- The parent process reads the exit status of the child process which reaps off the child process entry from the process table.

**Orphan Process:**

A process whose parent process no more exists i.e. either finished or terminated without waiting for its child process to terminate is called Orphan process