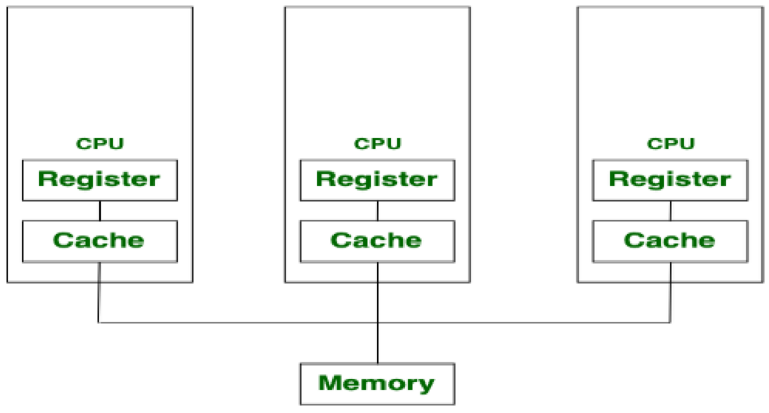
Multiprocessing, Process coordination
By
G. M. AL-Arafat. Shaown.

Multiprocessing:

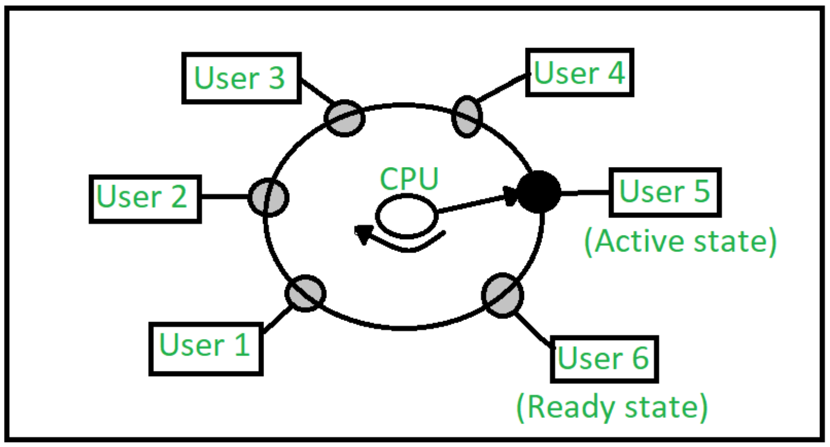
Multiprocessing is a system that has two or more than one processors. In this, CPUs are added for increasing computing speed of the system. Because of Multiprocessing, there are many processes that are executed simultaneously. Multiprocessing are further classified into two categories: Symmetric Multiprocessing, Asymmetric Multiprocessing.



Multiprocessing

Time sharing:

A time shared operating system allows multiple users to share computers simultaneously. Each action or order at a time the shared system becomes smaller, so only a little CPU time is required for each user.



In above figure the user 5 is active state but user 1, user 2, user 3, and user 4 are in waiting state whereas user 6 is in ready state.

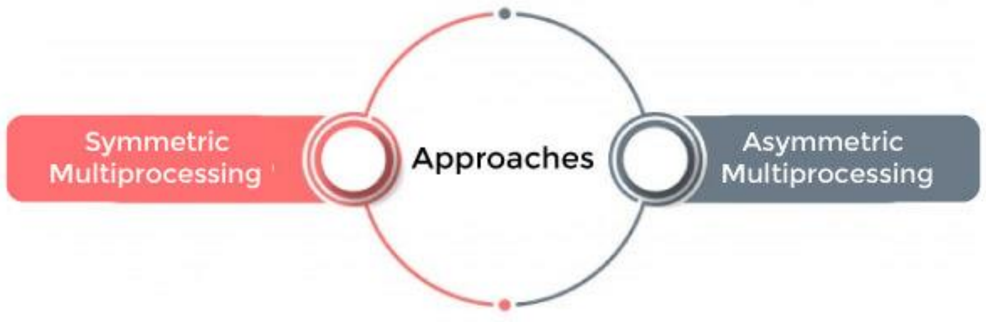
- **Active State –** The user’s program is under the control of CPU. Only one program is available in this state.
- **Ready State –** The user program is ready to execute but it is waiting for it’s turn to get the CPU. More than one user can be in ready state at a time.
- **Waiting State –** The user’s program is waiting for some input/output operation. More than one user can be in a waiting state at a time.

Advantages:
- Each task gets an equal opportunity.
- Less chances of duplication of software.
- CPU idle time can be reduced.

Disadvantages:
- Reliability problem.
- One must have to take of security and integrity of user programs and data.
- Data communication problem.

Multiple Processors Scheduling:
Multiple processor scheduling or multiprocessor scheduling focuses on designing the system's scheduling function, which consists of more than one processor. Multiple CPUs share the load (load sharing) in multiprocessor scheduling so that various processes run simultaneously. In general, multiprocessor scheduling is complex as compared to single processor scheduling. In the multiprocessor scheduling, there are many processors, and they are identical, and we can run any process at any time. Multiprocessor systems may be ***heterogeneous*** (different kinds of CPUs) or ***homogenous*** (the same CPU). There may be special scheduling constraints, such as devices connected via a private bus to only one CPU.

Approaches to Multiple Processor Scheduling:
There are two approaches to multiple processor scheduling in the operating system: Symmetric Multiprocessing and Asymmetric Multiprocessing.

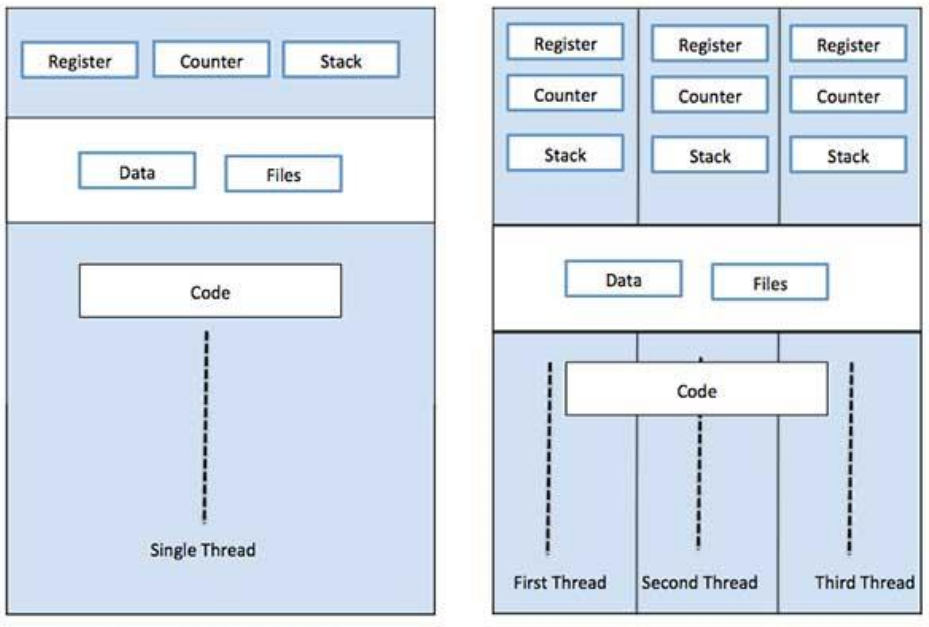


Symmetric Multiprocessing: It is used where each processor is self-scheduling. All processes may be in a common ready queue, or each processor may have its private queue for ready processes. The scheduling proceeds further by having the scheduler for each processor examine the ready queue and select a process to execute.

Asymmetric Multiprocessing: It is used when all the scheduling decisions and I/O processing are handled by a single processor called the Master Server. The other processors execute only the user code. This is simple and reduces the need for data sharing, and this entire scenario is called Asymmetric Multiprocessing.

Threads in Operating System:

A thread is a single sequential flow of execution of tasks of a process so it is also known as thread of execution or thread of control. There is a way of thread execution inside the process of any operating system. Apart from this, there can be more than one thread inside a process. Each thread of the same process makes use of a separate program counter and a stack of activation records and control blocks. Thread is often referred to as a lightweight process.



Single Process P with single thread   Single Process P with three threads

Advantages of Thread:

- Threads minimize the context switching time.
- Use of threads provides concurrency within a process.
- Efficient communication.
- It is more economical to create and context switch threads.
- Threads allow utilization of multiprocessor architectures to a greater scale and efficiency.

Difference between Process and Thread:

S.N.	Process	Thread
1	Process is heavy weight or resource intensive.	Thread is light weight, taking lesser resources than a process.
2	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
3	In multiple processing environments, each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
4	If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.
5	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
6	In multiple processes each process operates independently of the others.	One thread can read, write or change another thread's data.

Types of Thread:
Threads are implemented in following two ways −
User Level Threads − User managed threads.
Kernel Level Threads − Operating System managed threads acting on kernel, an operating system core.

User Level Threads:
In this case, the thread management kernel is not aware of the existence of threads. The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The application starts with a single thread.
Examples: Java thread, POSIX threads, etc.

Kernel Level Threads:
In this case, thread management is done by the Kernel. There is no thread management code in the application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.
The Kernel maintains context information for the process as a whole and for individuals threads within the process. Scheduling by the Kernel is done on a thread basis. The Kernel performs thread creation, scheduling and management in Kernel space. Kernel threads are generally slower to create and manage than the user threads.
Example: Window Solaris.

Resource allocation:

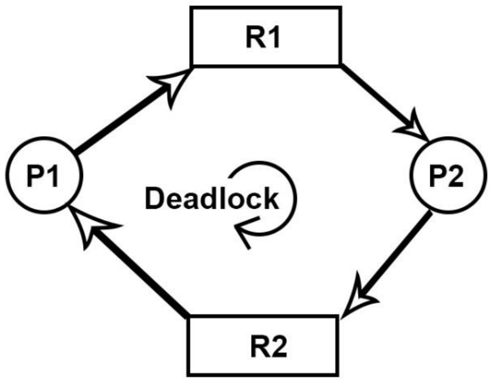
Resource allocation is the process by which a computing system aims to meet the hardware requirements of an application run by it. Computing, networking and energy resources must be optimized taking into account hardware, performance and environmental restrictions.

Types of resource allocation:

Two types of resource allocation occur in the operating system:

<u>Resource allocation with deadlock:</u>

Suppose there are two processes P1 and P2. And two resources R1 and R2.



Resource Allocation

P1 is holding R2 and P1 is requesting R1
P2 is holding R1 and P2 is requesting R2
As shown in the diagram it is making a circle and if a circle is made between processes and resources then there are chances that deadlock will occur.

<u>Resource allocation without deadlock:</u>

If there are multiple instances of the resource, there are chances that deadlock will not occur. Also if the circle is not made between processes and resources then deadlock will not occur.

Example of resource allocation:

An example of resource allocation is MS word which is a process and it is requesting a printer that is a resource. If a printer is held by another process then MS Word has to wait.

Dispatcher:

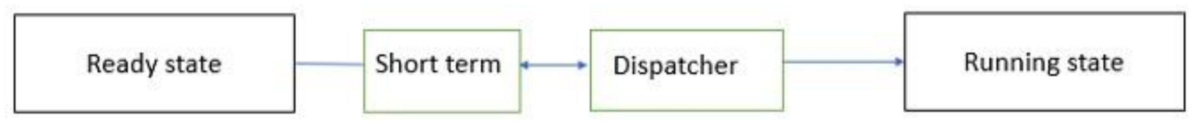
The dispatcher is done after the scheduler. It gives control of the CPU to the process selected by the short-term scheduler. After selecting the process, the dispatcher gives CPU to it.

Functions:

The functions of the dispatcher are as follows −

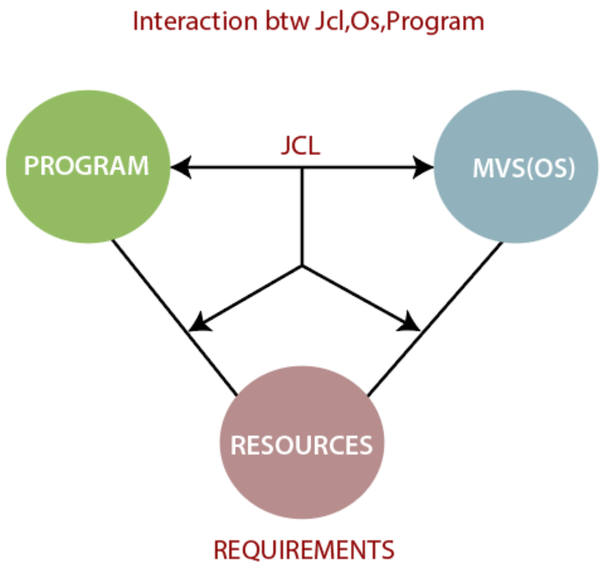
- Switching context.
- Switching to user mode.

The dispatcher is given below **−**



Job Control Language:

JCL stands for Job Control Language. JCL is a scripting language used on IBM mainframe operating system to instruct the system for the batch job. It is a set of statements that you code to tell the operating system about the task you want to perform**.**



Explain the JOB statement in JCL:

JOB statement gives the job identity to the Operating System (OS) in the spool and the scheduler. It is the first control statement in a JCL. The available parameters in the JOB statement help the OS in allocating the right scheduler. It is also useful for analyzing the required CPU time and issuing notifications to the user.
Syntax:
The basic syntax of a JCL JOB statement is below.

//Job-name JOB Positional-param, Keyword-param.

What is the use of symbol // in JCL?

It's an important symbol used in JCL statements. Each JCL statement must begin with this symbol. It is a predefined rule that is used to execute the JCL statements; otherwise, the JCL statement throws an error.
The JCL execution system first checks for the **symbol (//)** at the beginning of JCL statements. It avoids runtime exceptions.

What is condition checking in JCL? Is this possible?

JCL supports condition checking. Condition checking is possible at both the job level and the code level. It is done through the **COND** keyword with a return code and operand as predefined in JCL. So it is feasible in JCL.

Explain the hierarchy levels in JCL:

The level is the steps that describe the JCL statements according to their actions.

Every statement of JCL consist of the following keywords:

- NAME
- FIELDS
- OPERATIONS
- OPERANDS
- PARAMETERS
- POSITIONAL
- KEYWORD
- COMMENTS IF ANY