Tasks Scheduling in OS :

Process : A program that is in running state.

Eg. File->Save - This program has some processes.

- Opening the dialog box
- Opening the default folder (Documents folder of Windows)
- Waiting for the file name
- Saving the data on disk with the given name

Thread : A single sequence of a process

- Opening the dialog box(Thread 1)
- Opening the default folder (Documents folder of Windows (Thread 2)
- Waiting for the file name (Thread 3)
- Saving the data on disk with the given name (Thread 4)

At any point in time, some threads are running on the processor.

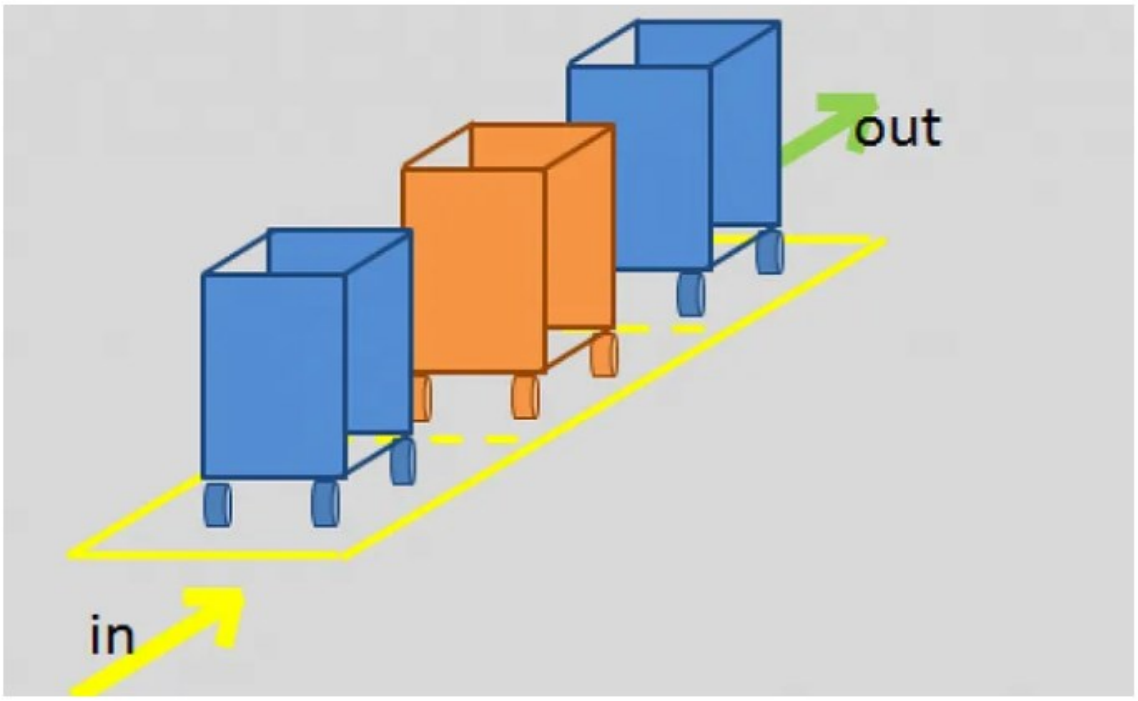
Others are waiting their turn for the processor.

Still, other threads are blocked waiting for I/O to complete, a condition variable to be signalled.

When there are more than one runnable thread, the processor scheduling policy determines which thread to run next.

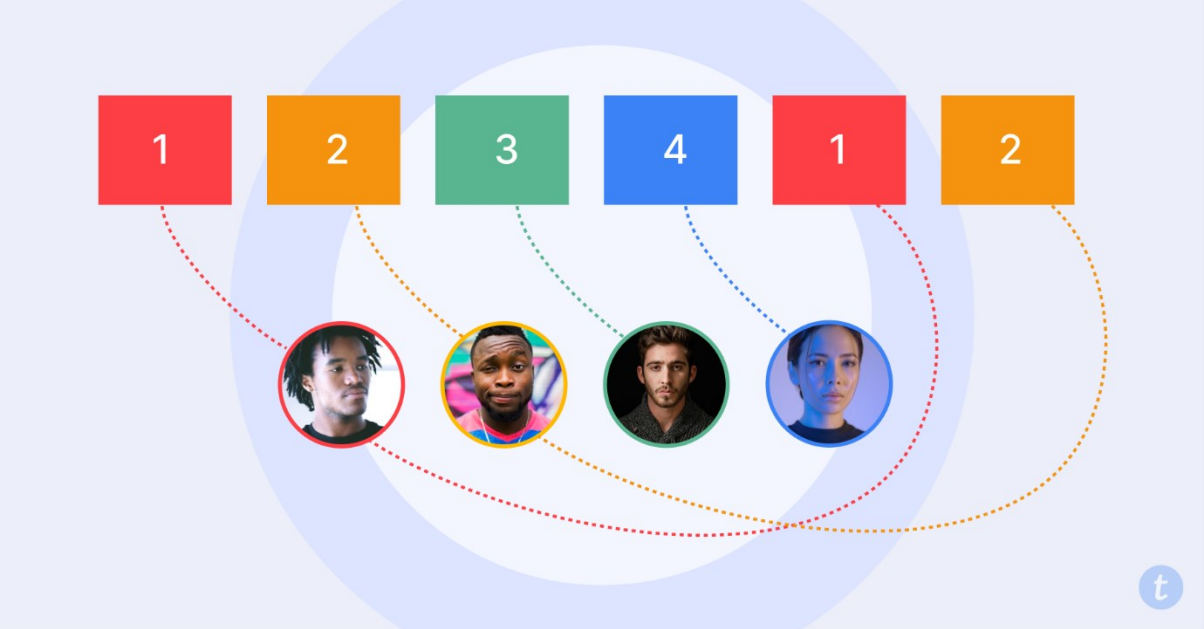
Scheduling Algorithms:

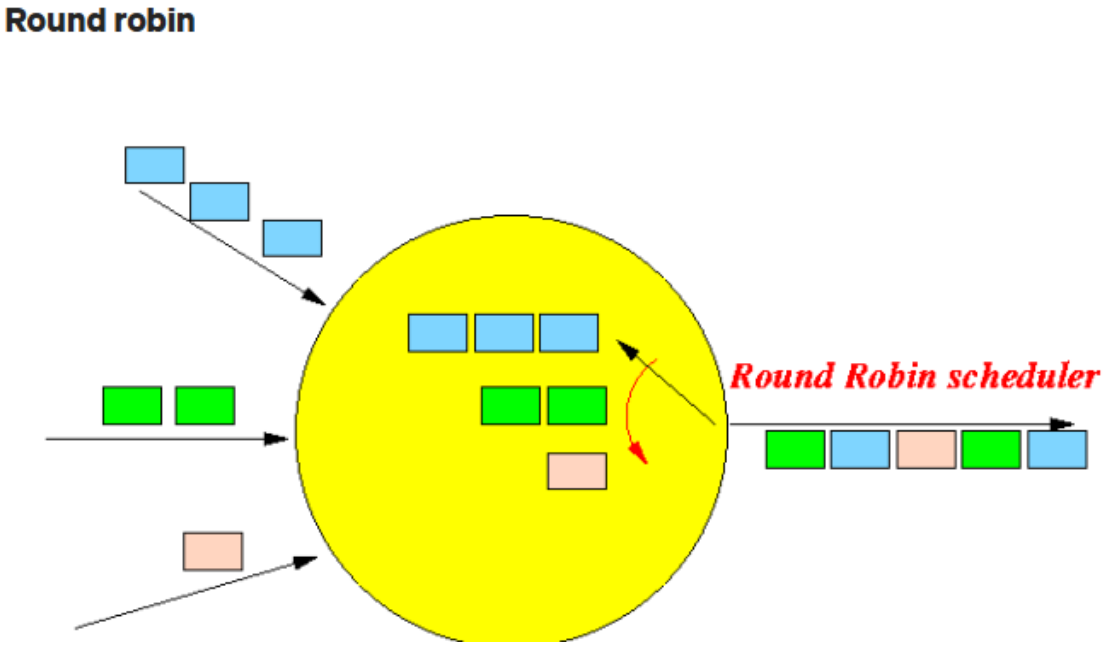
First In First Out (FIFO)



Shortest Job First (SJF)

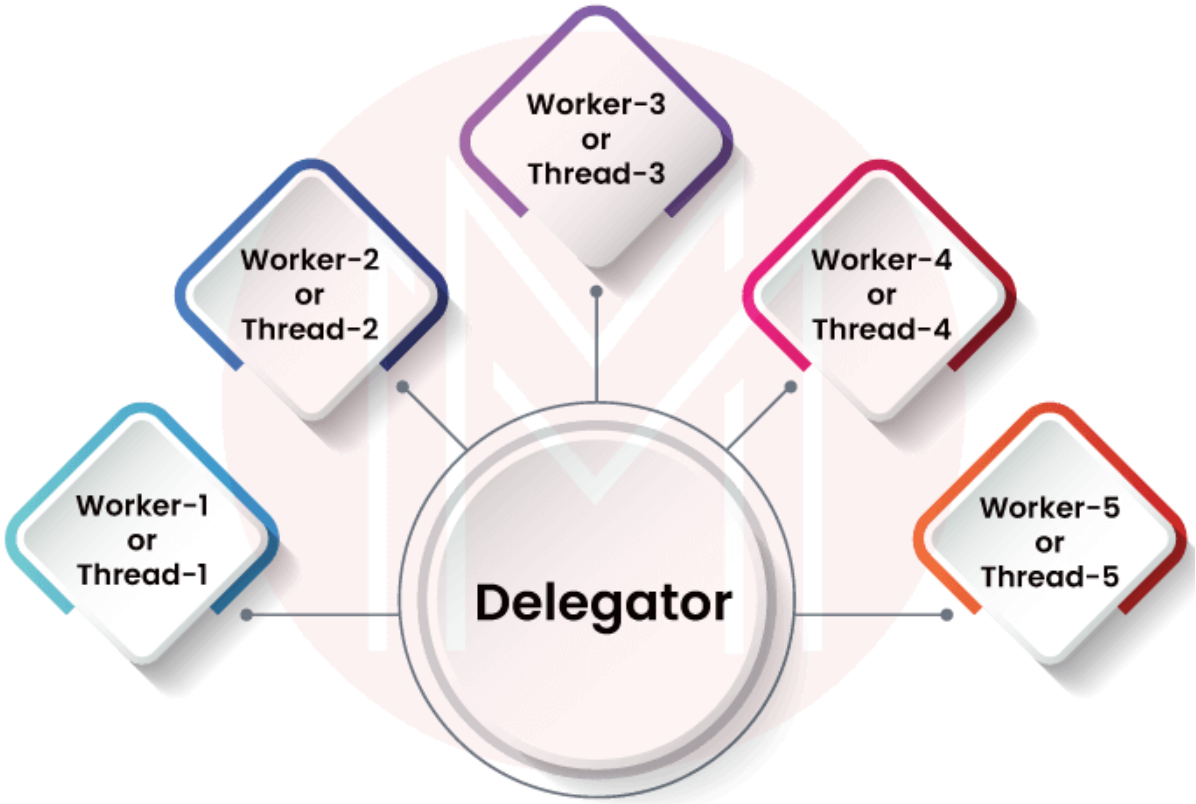
Round Robbin Scheduler :

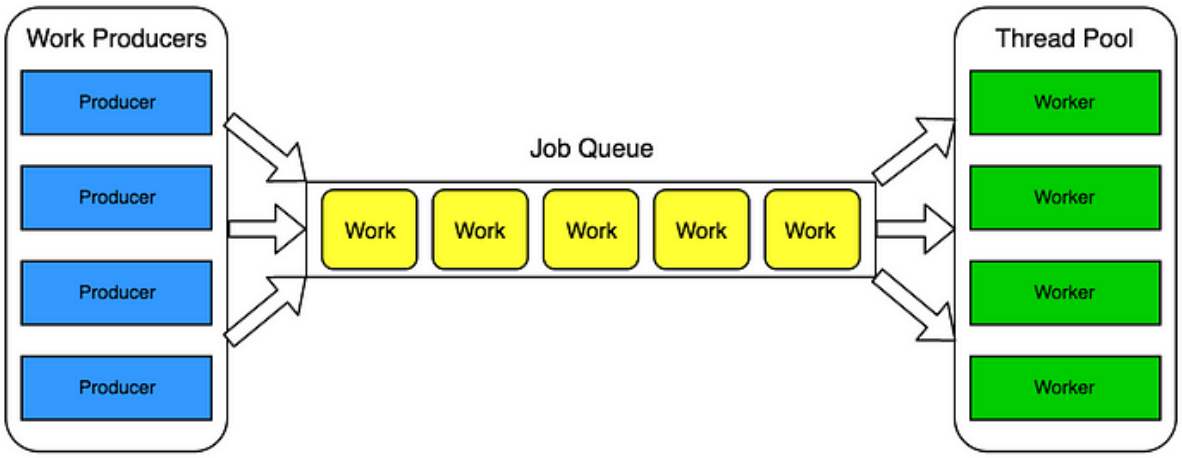




Worker Threads ;

- The idea is to create a lot of threads which don't do anything at first.
- Instead, they "wait for work".
- When work arrives (in the form of code), some kind of executor service (the reactor) identifies idle threads from the pool and assigns them work to do.
- That way, all the threads are created once (and not every time some work has to be done).
- At the same time, these threads are generic; they will do whatever work is assigned to them.





In multi-threaded programming, we often encounter scenarios where tasks need to be executed concurrently, but they don't necessarily need to return any values.

Eg. Chefs in a restaurant work simultaneously to execute their tasks, and they don't need to return any specific information.

In such cases, Runnables are used.

Thread implementation in Java

Extending Thread class or implementing Runnable Interface.

public void run() { …. }

** run() does not return any value.

Some times valuable data is needed upon completion of task.

In this scenario, Callables are used.

Eg. Imagine you are an online shopper placing multiple orders. You want to track the status of each order and get confirmation when each order is successfully processed.

Introduction to Callables

- **Callable** is like each online order you've placed. These are the tasks you care about because you want to track the status of your orders.
- **Future** is like the shipping confirmation email you receive for each order. When an order is shipped (the task is completed), you get an email **(Future)** that tells you the order's status and provides tracking information.