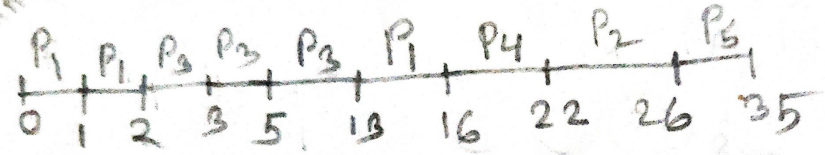
Process	A_T	B_T	Py	C_T	T_{AT}	W_T
P_1	0	5 ~~4~~ 3	2	16	16	11
P_2	2	4	4	26	24	20
P_3	2	11 ~~6~~ 8	1	13	11	0
P_4	1	6	3	22	21	15
P_5	5	9	5	35	30	21

Preemptive



$P_1 P_4 P_3 P_2 P_5$

Shortest Remain Time first / SJF with preemption

Process ID	A_T	B_T	C_T	T_{AT}	W_T
P_1	0	3	3	3	0
P_2	4	8	19	15	7
P_3	3	8	11	8	0
P_4	3	10	37	34	24
P_5	7	8	27	20	12

Avg T_{A_1} = 16

Avg W_T = 8.6

P_1 P_3 P_3 P_3 P_3 P_3 P_2 P_5 P_4

0 1 3 4 5 6 7 11 19 27 37

Process	A_T	B_T	C_T	T_{AT}	W_T
P_1	0	5	9	9	4
P_2	1	3	4	3	0
P_3	1	6	15	14	8
P_4	5	11	26	21	10
P_5	5	1	6	1	0

Avg T_{A_1} = 9.6

Avg W_T = 4.4

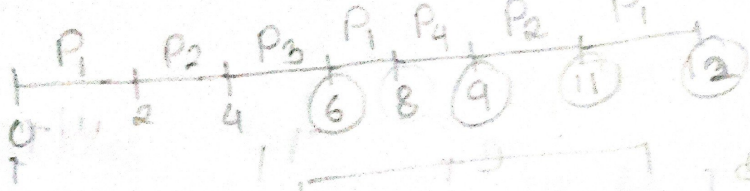
P_1 P_2 P_2 P_1 P_5 P_1 P_3 P_4

0 1 2 4 5 6 9 15 26

Round Robin

TQ = 2

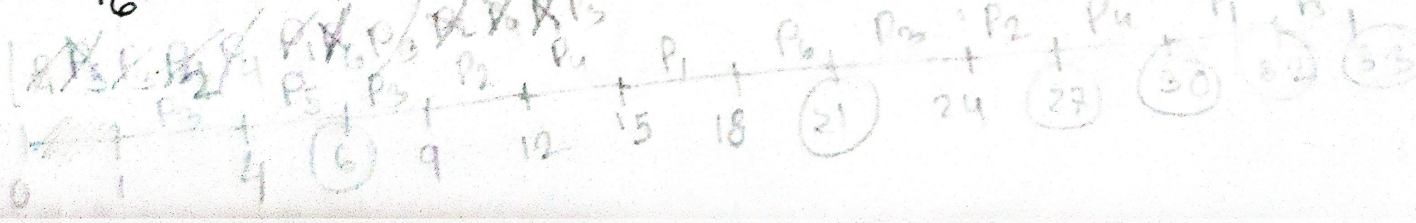
Process	A_T	B_T	C_T	T_A	W_T	R_T
→ P_1	0	5	12	12	7	0
P_2	1	4	11	10	6	1
P_3	2	2	6	4	2	2
P_4	4	1	9	5	4	4



TQ = 3

- Process

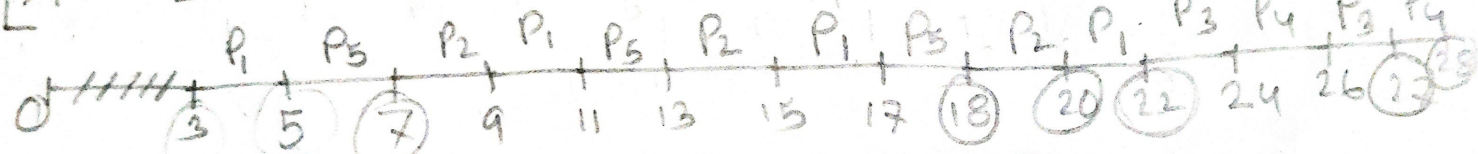
	A_T	B_T	C_T	T_T	R_T	W_T
P_1	5	5	32	27	10	22
P_2	4	6	27	23	5	17
P_3	3	7	33	30	2	23
P_4	1	9	30	29	0	20
P_5	2	2	6	4	2	2
P_6	6	3	21	15	12	12



TQ = 2

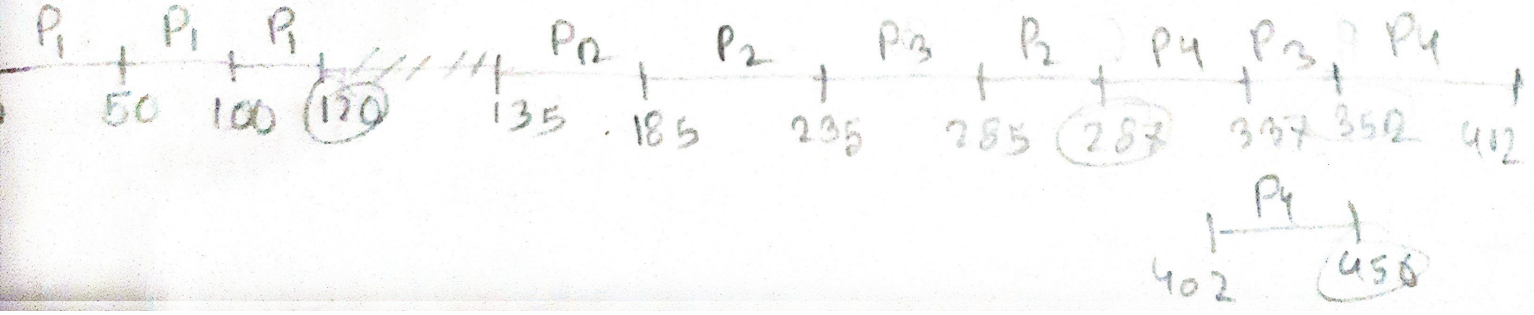
Process	A_T	B_T	C_T	T_A	W_T	R_T
P_1	3	8 ~~6 4 2 0~~	22	19	11	0
P_2	5	~~6 4 2 0~~	20	15	9	2
P_3	18	~~3~~ 1 0	27	9	6	4
P_4	20	~~3~~ 1	28	8	5	4
P_5	4	5 ~~3 1 0~~	18	14	9	1

[~~P1 P5 P2 P1 P3 P2 P1 P3 P2 P1 P3 P4 P3 P4~~ P4



Process	A_T	B_T	C_T	T_T	W_T
P_1	0	120 ~~70 20~~	120	120	0
P_2	135	102 ~~52 2 0~~	287	152	50
P_3	200	65 ~~15 0~~	352	152	87
P_4	300	148 ~~98 48~~	450	150	2

[~~P1 P2 P3 P2 P1~~ P3 P4



q=3

Process	A_T	B_T	C_T	T_{AT}	W_T
P_1	0	~~4~~ 1	9	9	5
P_2	4	~~8~~ 5	24	20	12
P_3	1	~~5~~ 2	17	16	5
P_4	1	~~2~~	5	4	2
P_5	4	~~5~~ 2	19	15	10

~~P1 P4 P3 P1 P5 P2 P3 P5 P2~~

q = 3

Process	A_T	B_T	C_T	T_{AT}	W_T
P_1	0	~~8~~ ~~5~~ 2	22	22	14
P_2	5	~~2~~	11	6	4
P_3	1	~~7~~ ~~4~~ 1	23	22	15
P_4	6	~~3~~	14	8	5
P_5	8	~~5~~ 2	25	17	12

Avg T_{AT} = 15

Avg W_T = 10.

~~P1 P5 P1 P2 P4 P5 P5 P1 P5~~ P5

• Difference between Short Term and Long-Term Scheduler

Long-Term Scheduler	Short-Term Scheduler
1. Long-Term Scheduler is also known as Job scheduler	1. Short-Term Scheduler is also known as CPU scheduler
2. Long-Term scheduler is Changing the process from New to Ready.	2. Short-Term scheduler is Changing the process from Ready to Running.
3. It controls Multi-Programming	3. It controls multi-tasking.
4. Speed is less than short-Term schedulers	5. Speed is faster than Long-Term scheduler.

IPC: Inter-process Communication.

• IPC helps to achive the communication among the ~~among~~ ~~the~~ processes or threads in a system.

Attributes.

1. Pipes
2. Named Pipes
3. Message Queuing
4. Semaphores
5. Shared Memory.

BIOS : Basic Input Output System

- It is a computer firmware that directs many basic functions of operating system, as booting and keyboard controls.

Paralled System

- It is system that can process the data simultaneously and increase the computational speed of a system.

It is a type of computer processing platform that breaks large task into the smallers pieces which done at the same time in different places and by different mechanisms

Multiprogrammed System

- A multiprogramming Operating System runs multiple programs on a single processor. If a program waits for simple I/O operations, others program utilize the CPU in the meantime

Difference between Multiprograming, Multitasking, Multiprocessing

- Multiprograming
  - It is based on concept of context switching
  - It utilizes single CPU for execution process
  - It takes more time to execution process.
- Multitasking
  - It is based on concept of time sharing
  - It utilizes multiple CPU for execution process
  - It takes less time to execution process
- Multiprocessing
  - In multiprocessing, multiple processing units are used.
  - A large amount of work can be done in a short period of time.
  - Multiprocessing can divided between two unit.
    1. Symetric Multiprocessing
    2. Asymetic Multiprocessing.

time sharing.

- It is an interface between the system hardware and users. It allows to user perform more than one tasks at a same time. And each task will get same priority. Therefore, all the tasks run very smoothly. It is extention of multiprograming system.

Microkernel and Monolithic Kernel

Microkernel	Monolithic Kernel
1. Os complex to Design	1. Os is easy to Design
2. Size is small	2. Size is large
3. Execution speed is low.	3. Execution speed is high
4. Debugging is simple	4. Debugging is hard.
5. Simple to maintain.	5. Difficult to maintain.

- Goal of Multiprogramming
  - ⟶ Maximize CPU Utilization
  - ⟶ Increase throughput.
  - ⟶ Run multiple program in Single processor.

- System call Interface.
  - System call Interface is a set of functions for requesting a service from the kernel on the operating system. It provides the essential interface between process and operating system.

  for example Open( ) is a system call used to provide get access to a file in file system.

Threads

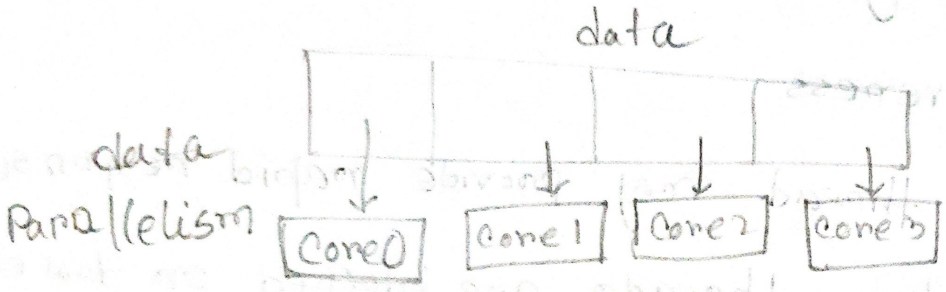
- There are four major categories of benefits to multi-threading

1. Responsiveness
   - One thread may provide rapid response while other threads are blocked or slow down.
2. Resource Sharing
   - By default threads share common data, code other resources which allows multiple task to be perform simultaneously.
3. Economy
   - Creating and manging theads is much faster
4. Scalability
   - A single threaded process can only run on one CPU. no matters how many are available.

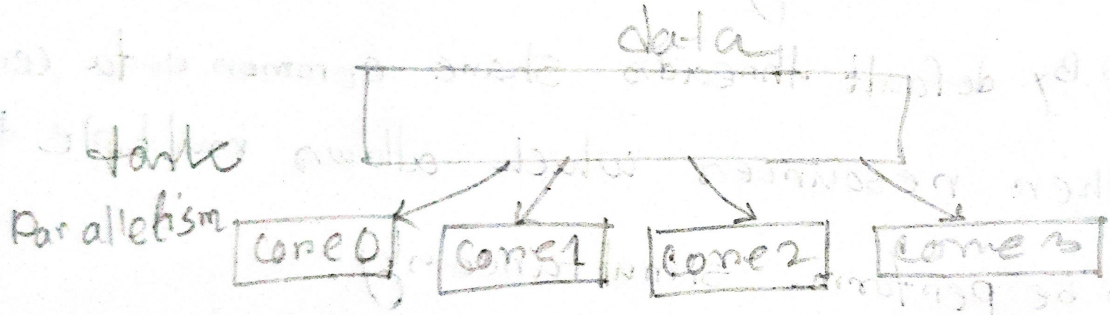
Multicore Programming

- Two types of Multicore Programming

→ Data Parallelism



→ Task Parallelism



Amdahl's Law

$$\text{speed up} \leq \frac{1}{S + \frac{1-S}{N}}$$

S	.25
P	.75

- $N = 1 \Rightarrow$

$$\text{speed up} \leq \frac{1}{0.25 + \frac{1-0.25}{1}}$$

$$\leq 1$$

- $N = 2 \Rightarrow$

$$\text{speedup} \leq \frac{1}{0.25 + \frac{1-0.25}{2}}$$

$$\leq 1.6$$

$$\text{ratio} = \frac{1.6}{1} = 1.6$$

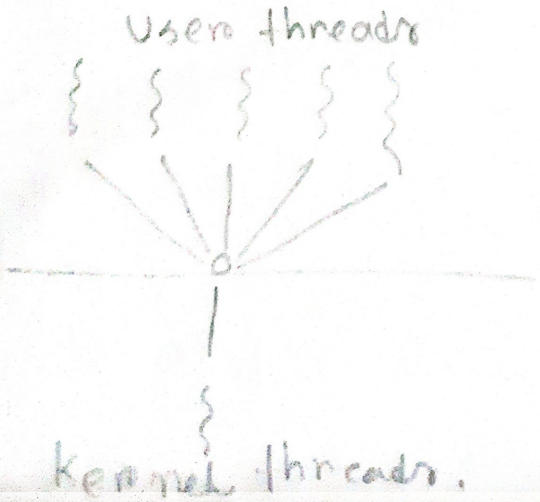
Difference between User Thread and Kernel Thread.

User Thread	Kernel Thread
1. User thread are implemented by user.	1. Kernel thread are implemented by operating system
2. Implementation of user threads is easy	2. Implementation of kernel threads is complicated.
3. Context switch is time less	3. Context switch is more
4. Context switch requires no hardware supports	4. Hardware support is needed.

Multithreading Models.

Many to One.

- In the many to model maps many users levels threads to one kernel threads.

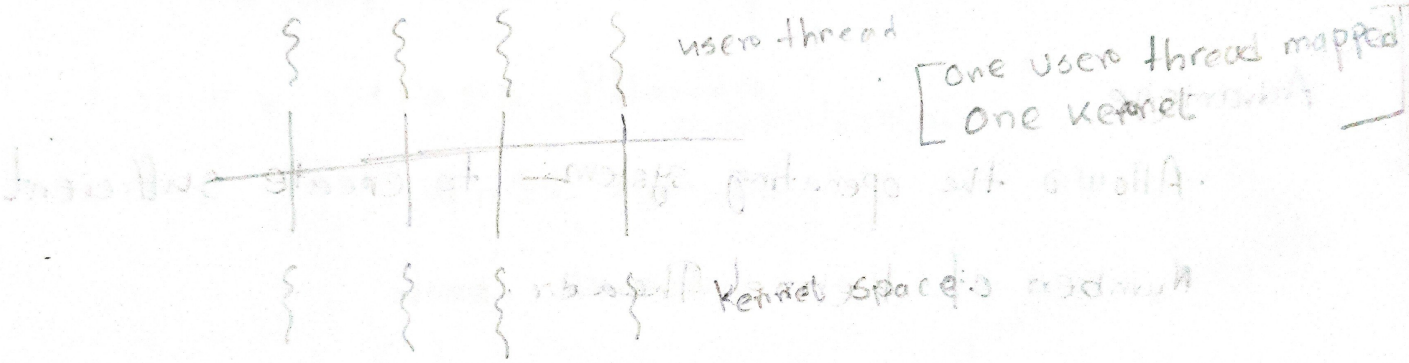


[Many user threads mapped one kernel]

- Advantage.
  - One thread block cause all to block.
- Disadvantage.
  - Can not run in parallel because only one kernel at a time

One to One.

- Each user level thread maps to kernel thread.



Advantage

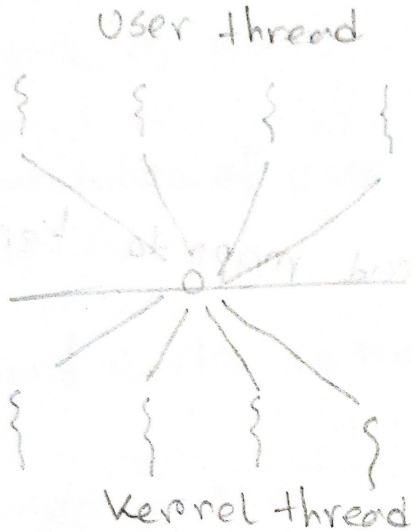
- Creating a user-level threads creats a kernel threads
- More concurrency than many to many

Disadvantage

- Number of threads per process restricted due to Overhead.

Many to many

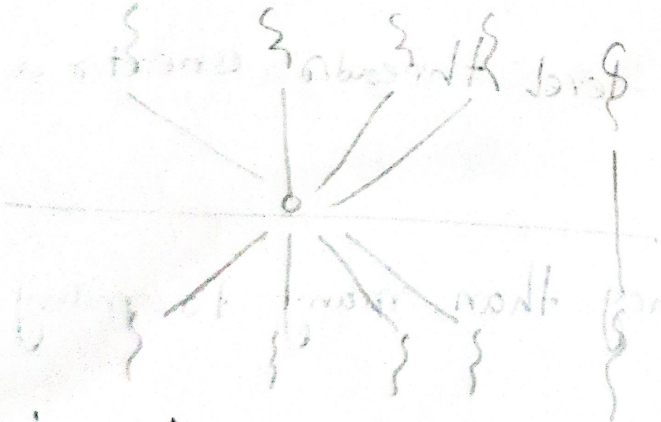
- many users threads mapped to many kernel threads.



Advantage

- Allows the operating system to create sufficient Number of kernel Theads.

Two level models.



Similar to M:M

except except that it allows a users threads to be bound to kernel threads.

Thread Library.

- It provides with an API for creating and managing threads.
- Implemented either in user space or in kernel space.
- Three main threads libraries in use today.
  - Posix Pthreads
  - Win 32 threads.
  - Java threads.

Pthreads

- Available on Solaris, Linux, Mac, OSX and public domain shareware for windows.
- Global variables are shared among all threads.
- One thread can wait others threads to rejoin before continuing.
- Common in Unix operating system.

Java Threads.

- managed by the JVM
- Typically implemented using the threads model provided by underlying OS.

→ It creates.

- Extending Threads class
- Implementing the Runnable interface.

```
public interface runnable
{
    public abstract void run();
}
```

Process Synchronization.

Race Condition.

A race condition occurers when two threads using same variable at same time.

- Access and manipulate several process same data concurrently
- Outcome of the execution depends on the particulars orders in which the access takes places.

Against Condition

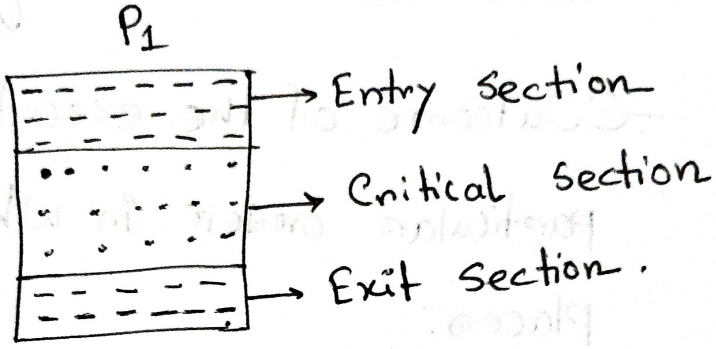
- One process at a time can manipulate the counters variable.
- The process should be synchronized.

- How "Critical Section" concept helps to solve race condition?

→ This critical section problem is to make sure that only one process should be in critical section at a time. When a process is in critical section, no others process are allowed to enter the critical section. This is how it solves the race condition.

Critical Section

- It is a segment of code of each process which may change common variables, update a table, write a file and so on.



Solution of Critical Section Problem.

1. Mutual execution.
   - If a process executing in critical section. then no other process will be execute in same time.
2. Progress
   - No process will execute their in critical section
   - some process wish enter their critical section
   - Those process will participate which are not executing in their reminder.
   - This selection can not be postponed indefinitely.