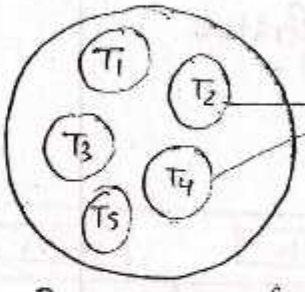
Threads

Definition

- Light weight process. (less no of ins^{n})
- Thread is just like a process having light weight.



process is divided into multiple threads.

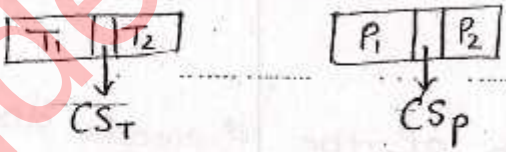
Big process (means containing more number of instruction)

Advantages-

1) Responsiveness-

- If the process is divided into multiple threads, then if one thread is completed its execution, then the o/p will be responded immediately. This response will be very faster compare to the response of process. Hence, the threads will improve the responsiveness.

2) Faster Context switches-



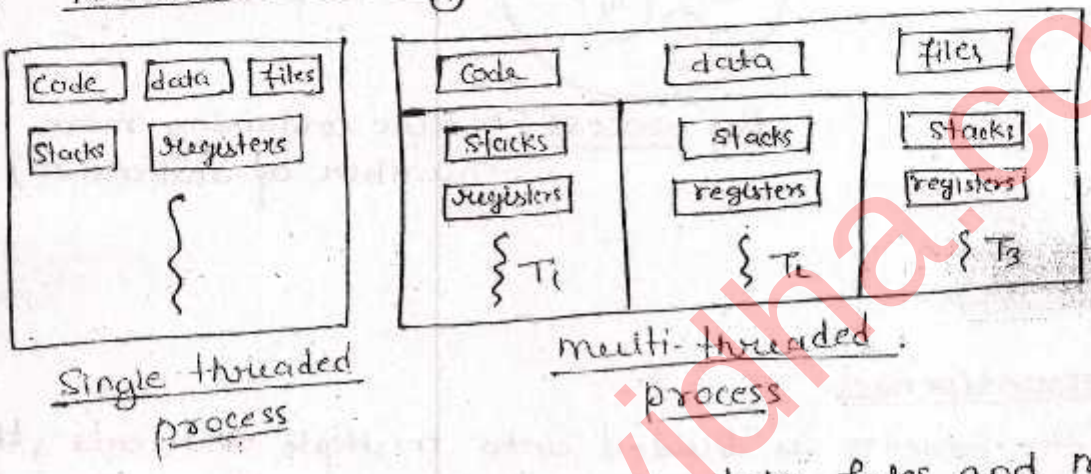
$$CS_T << CS_P$$

- The context switching time b/w the threads will be very less compare to the context switching time b/w the processes bcz the threads will have less content compare to the context of processes.

3) Effective Utilization of multiprocessor systems.

If the process is divided into multiple threads then the different threads will can be scheduled onto the different processor, so that the process execution will be faster.

4) Resource Sharing



Single threaded process

Multi-threaded process

- The resources like the code, data, files and memory will be shared among all the threads within the process.
- The stacks and registers cannot be shared b/w the threads, every thread will have its own stacks and registers.

5) Economical

- The implementation of the threads doesnot require any cost, there are various programming language API's will support implementation of the threads

Eg: JAVA API

6) Enhanced Throughput of the System-

If the process is divided into multiple threads and if we consider one thread as one job, then the no. of jobs completed for unit time will increase, and hence the throughput of the system will be enhanced.

The THREADS are categorised into two types-

→ User level threads
→ Kernel level threads

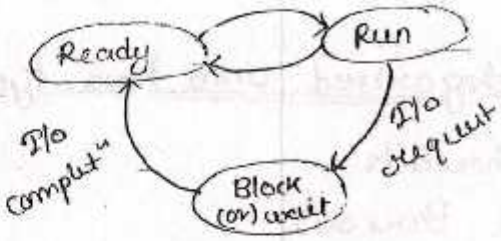
User level threads	Kernel level threads
1) User level threads are implemented by user or programmer.	1) Kernel level threads are implemented by OS.
2) OS doesn't know about the user level threads & OS can't recognise the user level threads & OS views the user level threads as a process only	2) Kernel level threads are recognised by the OS.
3) If one user level thread is performing blocking system call, then the entire process will be blocked.	3) If one kernel level thread is performing blocking system call, then another thread will continue the execution.
4) User level threads are dependent	4) Kernel level threads are independent.
5) The implementation of user level threads is easy.	5) The implementation of kernel level threads is complicated
6) U.L.T will have less context.	6) K.L.T will have more context
7) No h/w support is required for the U.L.T.	7) Scheduling of K.L.T requires h/w support (GANG) GANG scheduling

Note

The I/O of the processes is categorised into two types -

→ Synchronous I/O (by default)

→ Asynchronous I/O



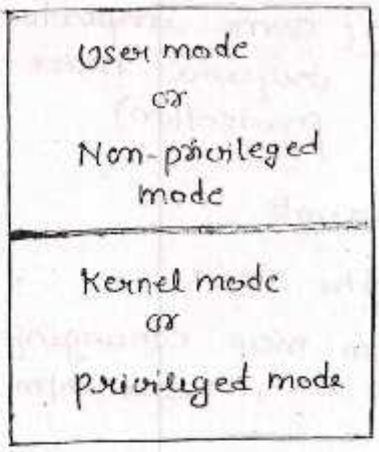
Synchronous I/O -

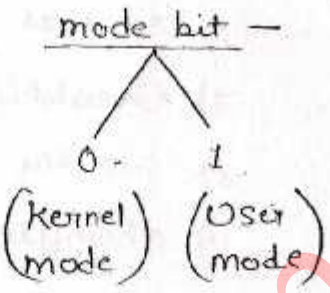
- In synchronous I/O, the process performing I/O operation will be placed in the block state, till the I/O operation is completed.
- At the point of time, the I/O operation will be completed & interrupt service routine will be initiated, which places the process from block state to ready state.

Asynchronous I/O -

- In asynchronous I/O, while initiating the I/O request, a handler func" will be registered.
- The process is not placed in the blocked state & it will continue to execute the remaining code after initiating the I/O request.
- At the point of I/O request is completed, a signal mechanism is used to notify the process that the data is available and the registered handler function will be asynchronously invoked.

all ins" are not dependent on I/O oper". Those (who are independent, continue its exec"





Dual mode operation

- In the h/w level, the two different modes are used in order to execute the instructions
  - User mode
  - Kernel mode
- Depending on the type of ins", the OS will decide in which particular mode, the ins" has to be executed.
- Generally the privileged ins" are executed in the kernel mode, and the non-privileged ins" will be executed in the user mode.
- Dual mode of operation is require to provide the security & protection to the user programs & to the OS from the errant user. (Corrupted user)
- In which particular mode, the current ins" is executing will be identified by the using mode bit.
- At the boot time, the system will always start in the kernel mode.
- The OS always runs only in the kernel mode.

privileged instruction (some important inst" which require more security & protection).

1) context switching
2) Disabling Interrupts
3) set the time of the clock
4) changing the m/m map (changing the process from one m/m location to another)
5) "I/o oper" (reading the data from the file of the hard disk).

Non privileged instruction

1) Reading the time of the clock
2) Reading the states of the processor.
3) Sending the final print o/p to the printer.