Name: M.Abbas ID#13063 Dep#CS(SE)

Q1:- Differentiate between a Process and thread with example.

Ans:- Process:- An instance of a Program running on a computer.

- ✯ Resource Ownership.
- ✯ Virtual address space to hold process image including Program, data, stack, and attributes.
- ✯ Execution of a Process follows a path through the Program.
- ✯ Process switch – An expensive operation due to the need to save the control data and register content.

Thread:-

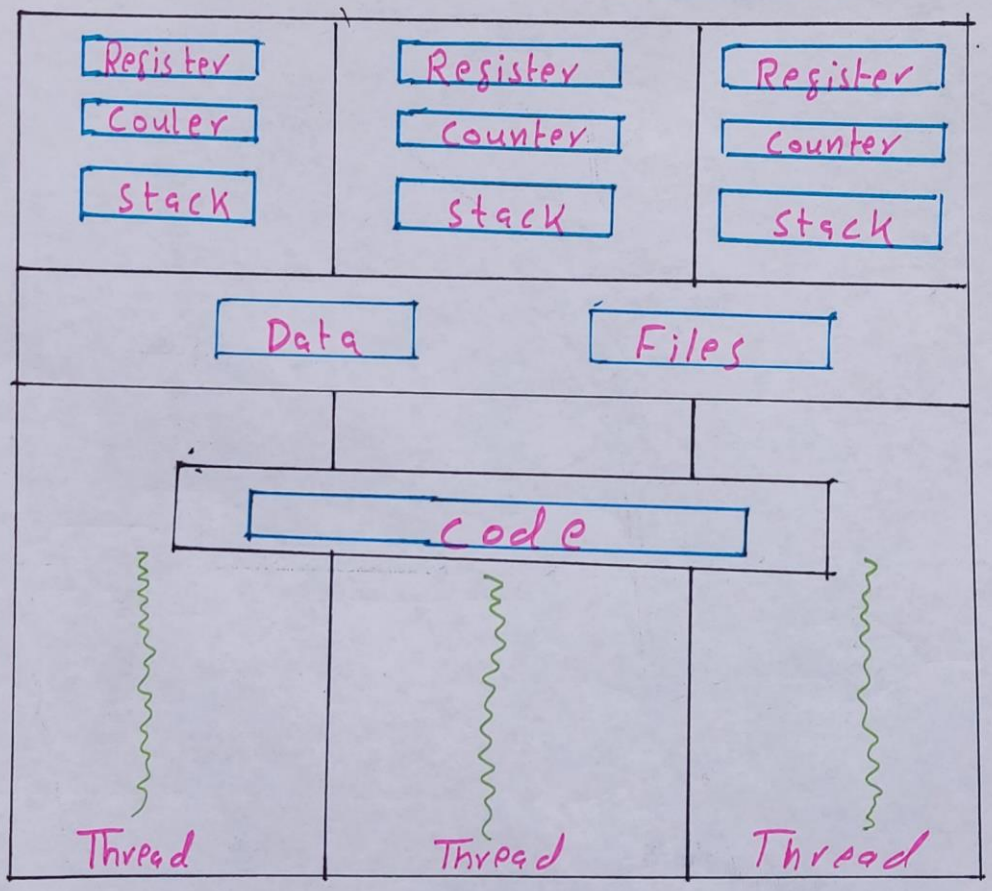
A dispatchable unit of Work within a Process.

- ✯ Interruptible: Processor can turn to another thread.
- ✯ All threads within a process have share code and data segments.
- ✯ Thread switch is usually much less costly than Process switch.

Example On next Page

(Q1) Diagram

Single Process with three threads.



Q2:- List and discuss Few types of thread.

Ans:- Thread:-

Thread is a single sequence stream within a process. Threads have same properties as of the processes so they are called as light weight Processes. Threads are excuted one after another. Feach thread has

1. Program Counter
2. A Register set
3. A stack space

TYPES OF THREAD

1. User Level thread (ULT):-

is implemented in the user level library, they are not created using the system calls, thread switching does not need to call OS and to cause interrupt to Kernal. Kernal doesn't know about the user level thread and manages them as if they were single-threaded Processes.

Advantages of ULT:-

- can be implemented on an OS that doesn't support multi threading.
- simple to create since no intervention of Kernal.

Q2)

Disadvantage of ULT:-

L> No or less co-ordination among the threads and Kernal.

L> If one thread causes a Page fault, the entire Process blocked.

② Kernal Level THREAD:-

~~Since~~ Kernal knows and manages the threads. Instead of thread table in each process, the Kernal itself has thread table (a master one) that keeps track of all threads in the system.

Advantages of KLT:-

L> Since Kernal has full knowledge about the threads in the system, scheduler may decide to give more time to processing having large number of threads.

L> Good for applications that frequently block.

Disadvantages of KLT:.

L> Slow and inefficient.

L> It requires thread control block so it is an overhead.

Q3:- What is a deadlock? In what situation it occurs in OS.

Ans:- **Deadlock:-**

Deadlock is a situation that occurs in OS when any process enters a waiting state because another waiting process is holding the demanding resource. Deadlock is a common problem in multi-processing where several processes share a specific type of mutually exclusive resources known as a soft lock or software.

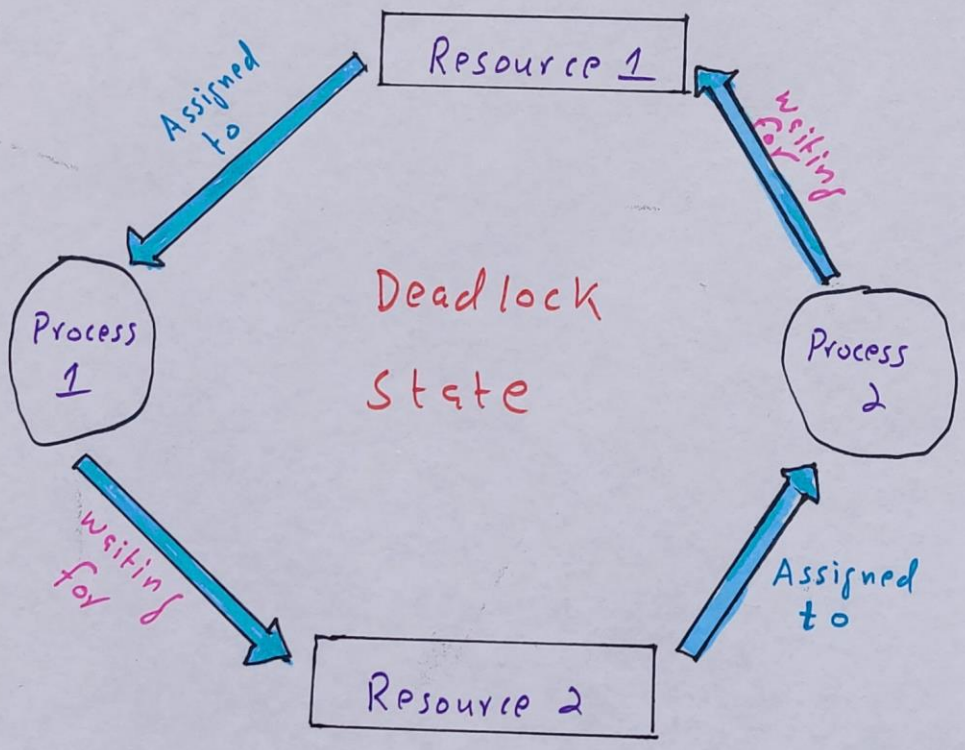
Example of Deadlock:-

- A real word example would be traffic going only in one direction.
- Here, A bridge is considered A resource.
- So, when deadlock happens, it can be easily resolved if one car backs up (Preempt resources and roll back).
- several cars may have to be backed up if a deadlock situation occurs.
- So starvation is possible.

Deadlock Diagram is on Next Page.

Q3)

Deadlock in OS



Q4:- Discuss a solution to the critical-section Problem must satisfy the three requriements.

Ans:- A solution to the critical-section problem must satisfy the following threee requirements.

1)- Mutual Exclusion:- If process P_i is exculing in its critical section, then no other processes can be executing in their critical sections.

2)- Progress:- If no process is executing in its critical section and some processes wish to enter their critical sections, then only those processes that are not executing in their remainder sections after a process which will enter its critical section next, and this selection cannot be postponed indefinitely.

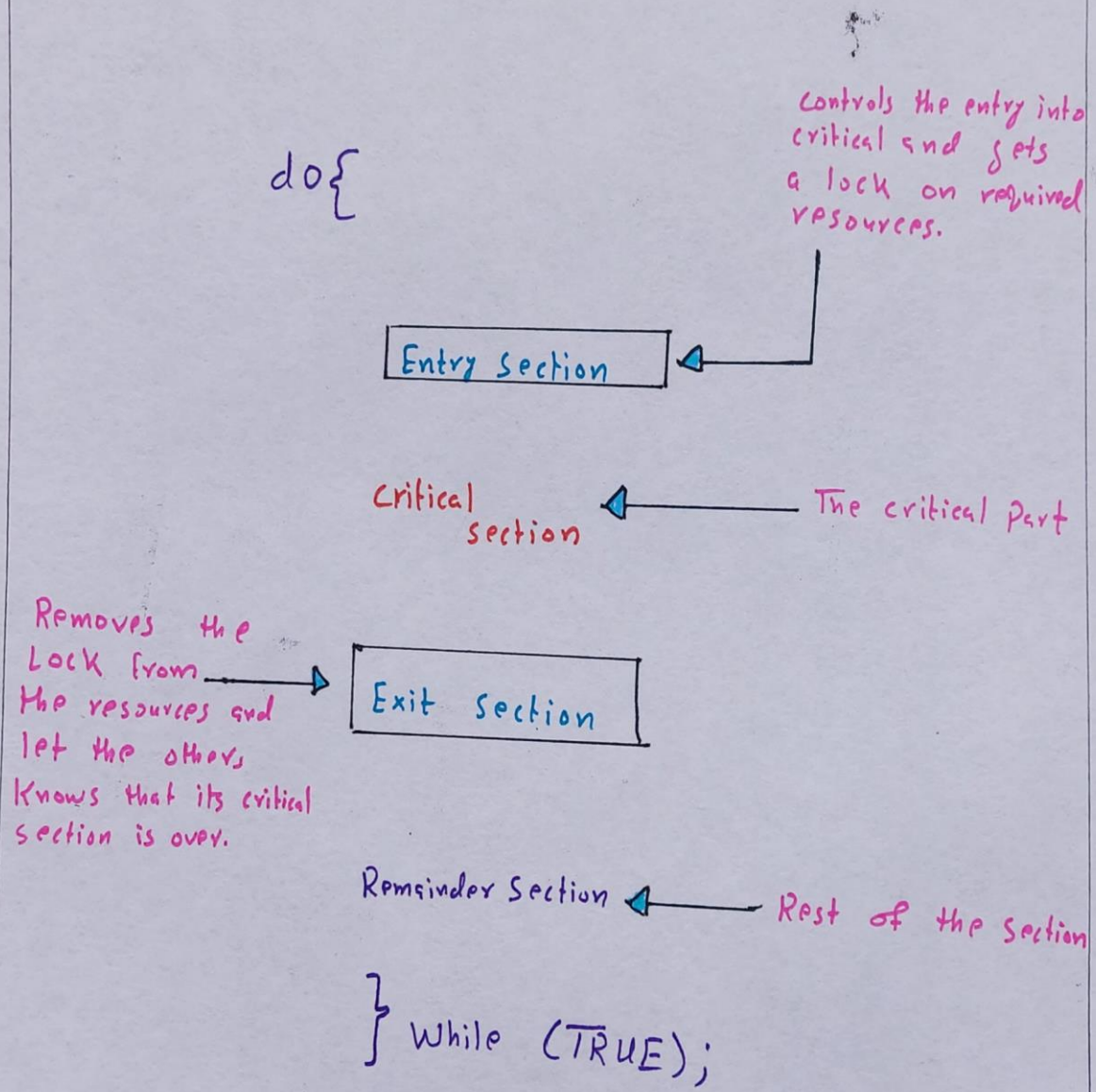
3)- Bounded Waiting:-

There exists a bound, or limit, on the number of times that other processes are allowed to enter their srytical sections after a Process has made a request to enter its critical section before that request is granted.

Example on Next Page

(4)

Example of Critical-Section



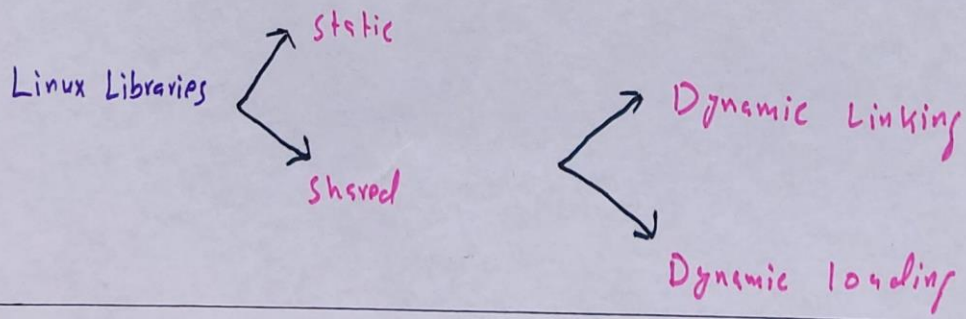
Q5:- Differentiate between dynamic loading and dynamic linking with example.

Ans:- Dynamic Loading:-

- Routine is not loaded until it is called.
- Better memory-space utilization; unused routine is never loaded.
- Useful when large amount of code are needed to handle infrequency occuring cases.

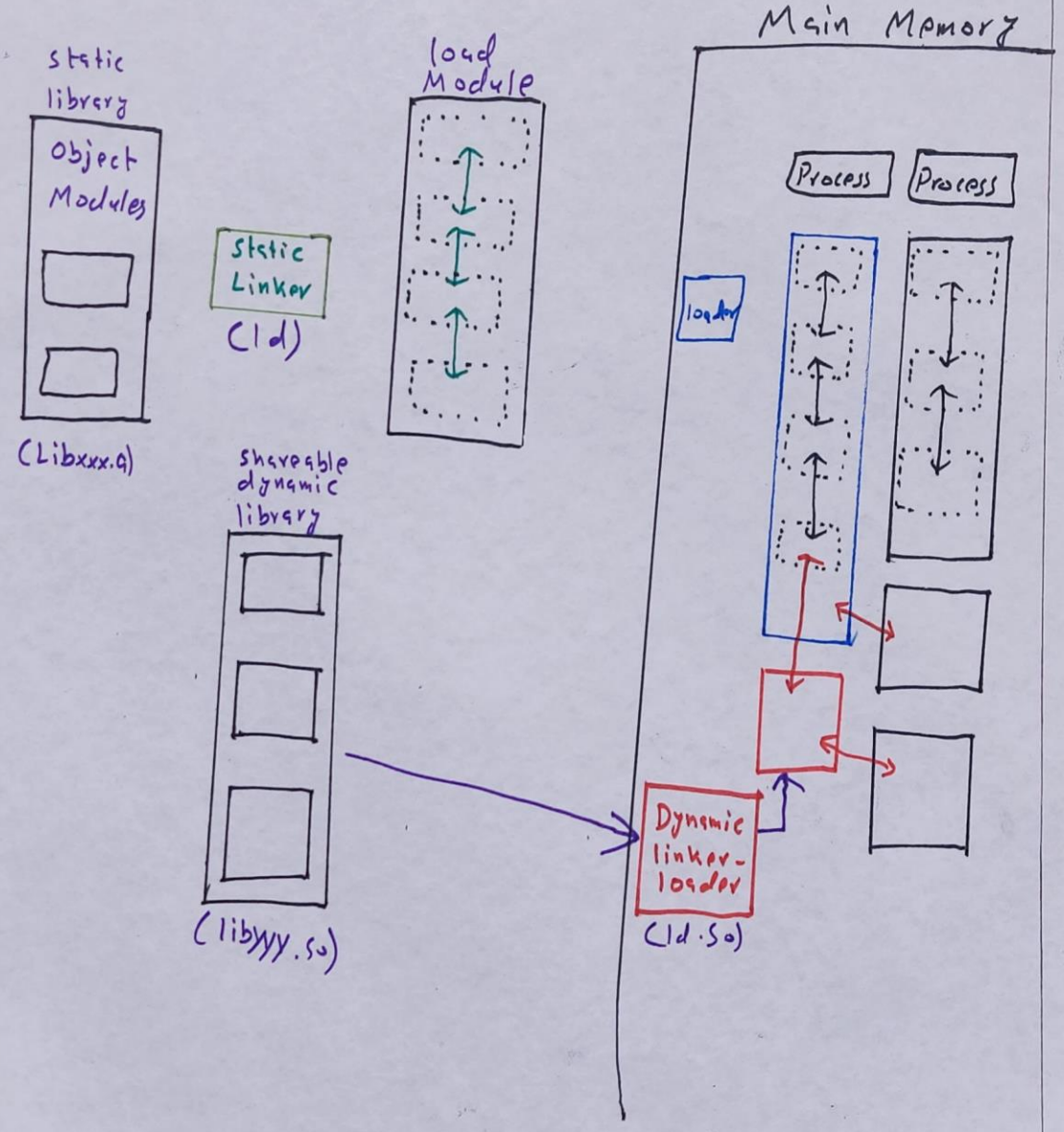
Dynamic Linking:-

- Linking Postponed until execution time.
- Small piece of code, stub, used to locate the appropiate memory-resident library routine.
- Stub replaces itself with the address of the routine and executes the routine.
- Operating system needed to check if routine is in processes' memory addresses.



Q5)

Diagram Dynamic loading and Dynamic Linking.



Q6 write your understanding about logical Vs Physical address space.

Ans:- **Logical vs. Physical Address Space.**

- The concept of a logical address space that is bound to a seperate physical address space is central to Proper Memory management.
  - L> Logical Address – generated by the CPU; also refferred to as virtual address.
  - L> Physical address – Address seen by memory unit.
- Logical and Physical addresses are the same in compiler-time and load-time address-binding schemes; logical (virtual) and physical addreses differ in execution-time address-binding scheme.