Name:- M. Musa

ID:- 14531

Sem:- Summers - 20.

Question N.O (1).-

Answer:-

i) Process means a program is in execution, whereas thread means a thread means a segment of a process.

ii) A process is not lightweight, whereas threads are light weight.

iii) A process takes more time to terminate, and the thread takes less time to terminate.

iv) Process likely takes more time for creation, whereas Threads takes less time for creation.

v) Process likely takes more time for context switching, whereas threads takes less time for context switching.

vi) A process is mostly isolated, whereas threads share memory.

vii) Process does not share data, whereas threads share data with each other.

Example :-

The Java virtual machine (JVM) is a process in your OS. And inside the JVM you can have multiple threads running concurrently.

Question N.O (2) :-

Answer:-

1) User Level Thread (ULT):-

It 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 interupt to kernal. Kernal doesn't know about the ULT and manages them as if they were single-threaded processes.

— Advantages of ULT:-

i) can be implemented on an OS that doesn't support multi threading.

ii) Simple representation since

thread has only program counter, register, set stack space.

iii) Simple to create since no intervention of kernal.

iv) Thread switching is fast since no OS calls need to be made.

2) Kernal level Thread (KLT) :- 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 the threads in the systems. In addition kernal also maintains the traditional process table to keep track of the processes. OS

kernal provides system call to create and manage threads.

Advantages of kLT:-

i) Since kernal has full knowledge about the threads in the systems scheduler may decide to give more time to processes having large number of threads.

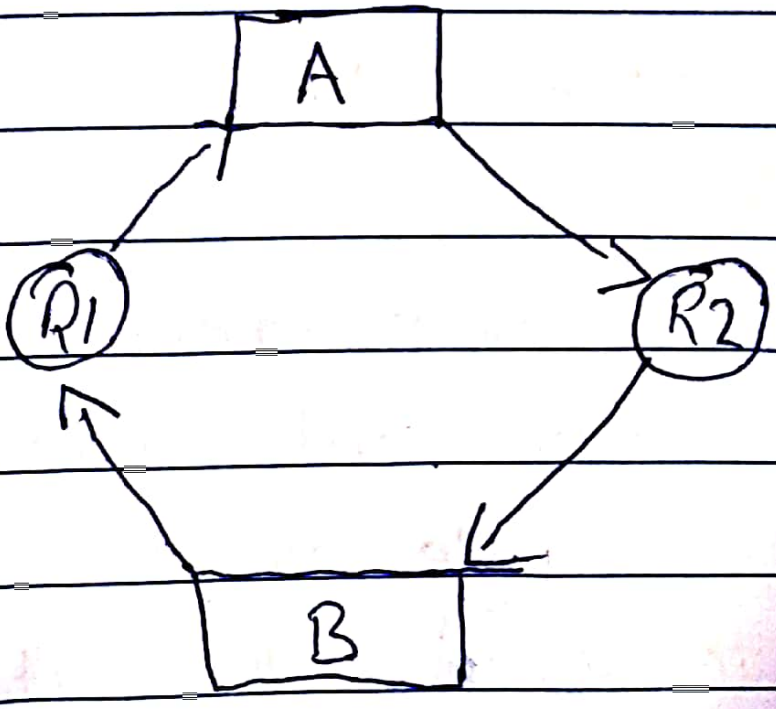
ii) Good for applications that frequently block.

Question N.o (3):-

Answer:-

There was a time where operating systems were only able to execute a single process at a time, thus giving full system

resource and attention to that one single ~~resource~~ process. Nowadays OS can hadle multiple tasks at once, but sometimes they have to deal with a problem known as deadlock. A dead lock occurs when there is atleast one process which is waiting for resource to be released by another pro-cess inorder to finish task properly



In this graph, Process A is waiting for resource 2 R2 to be released by Process B to finish a task. At the same time Process B is waiting for Resource 1 to be released by PA to finish a task. PB can't begin until PA finishes, which would then prevent the whole system from moving forward if deadlocked. This is something OS have to deal with, and many techniques can be implemented to prevent deadlock from occurring altogether such as :-

No Mutual exclusion, Circular wait, Resource holding, No preemption.

Question N.o (4):-

Answer:-

i) Mutual Exclusion:-

Out of a group of cooperating processes, only one process can be in its critical section at a given point of time.

ii) Progress:-

If no process is in its critical section, and if one or more threads want to execute their critical section then any one of threads must be allowed to get into its critical section.

iii) Bounded waiting:-

After a process makes a

request for getting into its critical section, there is a limit for how many others processes can get into their critical section before this process's request is granted. So after the limit is reached, system must grant the process permission to get into its critical section.

Question N.O (5):-

Answer:-

Dynamic Loading:-

Suppose our program that is to be executed consist of various modules. Of course its not wise to load all the modules into main memory together at

once (in some cases it might not be even possible because of limited main memory). So basically what we do here is to load the main module first and then during execution we load some other module only when its required. and the execution can not process further without loading it.

Dynamic linking:-

Suppose our program has some function whose defination is present in some system library. We do know the header file only consists of declarations of functions and not definations. So

during executions when the functions gets called we load that system library into main memory and link the function call inside our program with the function defination inside system library.

Question N.o (6):-

Answer:-

logical Address is generated by CPU while prgogram is running.

Physical Address identifies a physical location of required data in a memory. The user never directly deals with the physical address but can access

by its corresponding logical address.