ID #16950 Muhammad Asif

ID # 16950 : Muhammad Asif

Paper# Operating System.

Q1 In deadlock prevention strategy do you think it is necessary to check that either Safe state exists or not ? Given reason to Support your answer.

Ans Deadlocks can be prevented by preventing at least one of the four required conditions.

Mutual Exclusion-

Shared resources such as read-only files do not lead to deadlocks.

unfortunately some resources such as printers and tape drives, require exclusive access by a single Process.

Hold and wait:-

To prevent this condition processes must be prevented from holding one or more resource while simultaneously waiting for one or more others.

- Require that all processes request all resources at one time.
- Require that processes holding resources must release them before requesting new resources.
- Either of the methods described above can lead to starvation ; if a process requires one or more popular resources.

ID#16950 Muhammad Asif

NO Preemption:-

Preemption of Process resources allocation can prevent this condition of deadlocks, when it is possible

- one approach is that if a process is forced to wait when requesting a new resources.
- Another approach is that when a resource is requested and not available than the system looks to see what other processes currently have those resources and are themselves blocked waiting for some other resource.
- Either of these approaches may be applicable for resources whose state are easily save and restored.

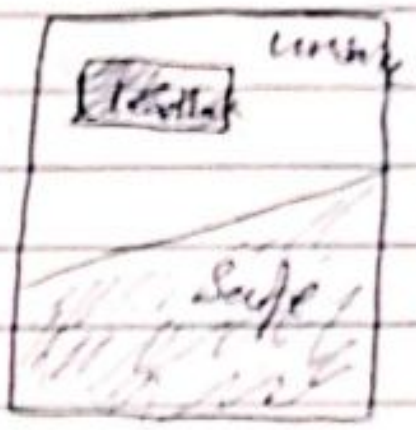
Circular wait:-

- one way to avoid circular wait is to number all resources, and to require that processes request resources only in strictly increasing order.
- In other words, in order to request resource Rj, a process must first release all Rj such that i=j.
- one big challenge in this scheme is determine the relative ordering of the different resources.

Safe state:-

A state is safe if the system can allocate all resources requested

by all processes (up to their stated maximums) without entering a deadlock state.



Consider a system with 12 tape drives allocated as follows. Is this is safe what is safe sequence?

	Maximum Needs	Current Allocation
P_0	10	5
P_1	4	2
P_2	9	2

What happens to the above table if process P_2 requests and is granted one more tape drives.

key to the safe state approach is that when a request is granted only if the resulting allocation state is a safe one.

id # 16950 Muhammad Asif

Q2 Differentiate between Dynamic loading and Dynamic linking with the help of examples.

Ans • Dynamic loading refers to mapping an executable or library into a process memory after is has started.

• Dynamic linking refers to resolving symbols associating their names with addresses or offsets after compile time.

The reason it's hard to make a distinction is that the two are often done together without recognizing the subtle distinctions arounds the parts, put in bold perhaps the clearest way to explain is to go through what the different combinations would mean in practice

• Dynamic loading. static linking. The executable has an address/offsets table generated at compile time. but the actual code/data aren't loaded into memory at process start. This is not the way things tend to work in most systems nowadays, but it would describe some old-fashioned overlays system: i'd also be utterly unsurprised if some current embedded systems work this way too. in either case the goal is to give the programmer control over

memory, use while also avoiding the overhead of linking at runtime.
* Static loading, dynamic linking. This is how dynamic libraries specified at compile time usually work. The executable contains a reference to the dynamic/shared library. but the symbol table missing or incomplete. Both loading and linking occur at process start which is considered "dynamic for linking but not not for loading.
* Dyamic loading, dynamic linking. This is what happens when you call dlopen or its equivalent on other systems. The object file is based loaded dynamically under program and in the library are resolved based on the process' possibly-unique memory layout at that time.

Q3 Which Component of an operating system is best suited to ensure fair, secure, ordely and efficient use of memory? Also identify some more task managed by that Component?

Ans The most suitable component of an operating system that suited to ensure fair, secure, ordely and efficient use of memory is memory management system.

ID# 16950 Muhammad Asif

The task of memory management includes keeping track of used and free memory space as well as when, where, and how much memory to allocate and deallocate it is also responsible for swapping process in and out of main memory. The purpose of memory management is to ensure fair, secure orderly and efficient use of memory.

Q7 why is the Context Switch overhead of a user level threading as compared to the overhead for processes? Explain.

Ans One might that in general processes are more flexible than threads. for example processes are controlled independently by the OS, meaning that if one crashes it will not affect other Processes, However, Processes require explicit communication using either message passing or shared memory which may add overhead since it requires support from the OS kernel, using threads within a process allows them all to share the same address space simplifying communication between threads However, threads have their own problems because they communicate through shared memory they must exist on same machine. and Cave

must be taken to create thread-safe code that function correctly despite multiple threade actings on the same set of shared data.
when comparing processes and thread, we can also analyze the switch cost. whenever it is needed to switch between two processes, we must invalidate the TLB cache which can be a slow operation. When we switch between two threads, on the other hand, it is not needed to invalidate the TLB because all threads share the same address space, and thus have the same contents in the cache. Thus the cost of switching between threads is much smaller than the cost of switching between processes

ID # 16950 Name : Muhammad Asif

Q4 : Differentiate between Symmetric and A-Symmetric encryption with the help of example.

Ans : Symmetric encryption uses a single key that needs to be shared among the people who need to receive the message while asymmetrical encryption uses a pair of public key and a private key to encrypt and descript when communicating.

- symmetric encryption is an old technique while asymmetric encryption is relatively new.

- Asymmetric encryption was introduced to complement the inherent problem of the need to share key in symmetrical encryption model eliminating the need to share the key ~~and~~ by using a pair of public private keys.

- Asymmetric encryption takes relatively more time than the symmetric encription.

Q6 List and describe the four memory allocation algorithm covered in lectures. which two of the four are more commonly used in practice?.

Ans The four memory allocation are

(i) **first fit:-** in the linked list of available memory addresses. we place the data in the first entry that will fit its data. its aim is to minimise the amount of Searching but leads to external fragmentation later on.

(ii) **Next fit:-** Similar to first fit. but instead of searching from the beginning each time. it Searches from the last Successful allocation Greatly reduces the amount of Searching but leaves external fragmentation at the beginning of memory.

(iii) **worst - fit:-** Traverses the memory and gives the partitions as large Spaces as possible. to leave usable fragments left over Needs to Search the Complete list and Such is a poor performer

(iv) **Best - fit:-** CareJully Scours the memory for spaces that perfectly

fit the Ram we want However the Search is likely to take a very long time.

we most Commonly use fast fit and next fit in Practise. They're to implement and are faster to boot.

Describe the difference between external and internal fragementation why should they be avoided?

Internal fragmentation is the wasted space within each allocated block because of rounding up from the actual requested allocation to the allocation granularity.

External fragmentation is the various free spaced holes that are generated in either your memory or disk space
External fragmented blocks are available for allocation but may be too small to be of any use.