NAME Muhammad Ilyas

Class B.S (software Engr)

ID No 15392

Paper O.S

Teacher Sir - M. Daud

QUESTION - No - 1

In deadlock prevenention strategey do you think it is necessary to check that either safe state exists or not? Give reason to support your answer

ANSWER:-

A state is safe if the system can allocate all resources requested by all process up to their stated maximums without entering a deadlock state. If a safe sequence does not exist, then the system is an unsafe state, which may lead to deadlock. All safe states are deadlock free, but not all unsafe states lead to deadlock.

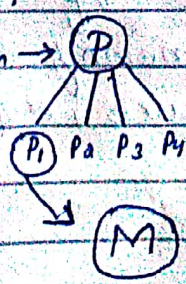
QUESTION - No - 2

Difference between Dynamic Loading and Dynamic Linking with the help of examples.

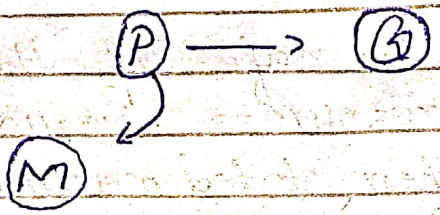
ANS:-

Dynamic loading:-

Computer program is loaded into memory but sometimes a certain part of program is loaded only when it is called by program.



Dynamic Linking:- Intially only (P) gets loaded and CPU Links the dependent program (Q) for main exicuting program (P) when it is needed.



$P_1 \rightarrow P_2 \rightarrow P_3 \cdots$

QUESTION - NO - 3

ANSWER:-

The most suitable component of an operating system that suited to ensure fair, secure, orderly and efficient use of memory is Memory Management system.

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 managment is to ensure fair, secure, orderly and efficent use of memory.

QUESTION - NO - 4

Differentiate between Symmetric and Asymmetric.

ANSWER:-

ASymmetric Encryption:-

1- Different keys are used

2- Encyption is done with the help of user's public key.
Decryption with private key.

3- Ciphertext may be of large size.

4- No problem of key Exchange.

5- It is used for confidentiality, Authentication Non-repudiation.

6- Process is slow

RSA, ECC, DSA etc are the examples.

Symmetric Encryption.

1- Same key are used.

2- Both encryption and Decryption are done with user's private / secret key.

3- Ciphertext same or less in size.

4- Problem.

5- Confidentiality

6- Process is fast.

Blowfish, AES, RC4, DES and RC6 are the examples.

QUESTION - No - 5

ANSWER:-

Internal fragmentation:

The space wasted internal to the allocated region. Allocated might be slightly larger than requested memory.

External fragmentation.

is unused space between allocated regions of memory. Typically external fragmentation result in memory regions that are too small to ~~sta~~ satisfy a memory request, but if we were to combine all the regions of external frementation, we have enough memory to satisfy a memory request. External fragmentation is also avoided by using paging techniques.

QUESTION - No - 6

ANSWER:-

Following are the list of four Memory allocation algorithms.

1- First - fit:

first - fit allocate into the first avalible gap found of adequate size, starting from the beginning of memory. Scan memory region list from start for first fit. Must always skips over potentially many regions as the start of list.

2- **Next-fit:-**

allocate into the first avalible gap found, resuming the search from the last allocation.

Scan memory regim list from point of last allocation to next fit. Breaks up large block at the end of memory.

3- **Best-fit:**

Search all of memory to find the smallest valid gap and allocate into it, on the basis that it all leave the smallest unusable gap.

Pick the closest free region on the entire list. Leaves small unusable regions and slower due to searching of entire list.

4- **Worst-fit**

Search ~~all~~ and place into the largest area, on the basis that is likely to leave a gap big enough for something else to use.

Finds the worst fit in the entire list slower as it searches entire list. Fragmentation still in issue.

First-fit and next-fit most commonly used as it is easier to implement and works out better.

* —— * —— *

QUESTION - No - 7

ANSWER:-

User - Level threads implement in user - level libraries, rather than via system calls. So threads switching does not need to call the operating system and to cause interruption to the kernal. In fact, the kernal knows nothing about user - level threads and manages them as if they were single - threaded process. Threads are very inexpensive to create and destroy, and they are inexpesive to represent. For example They require space to store, the PC, the SP ~~store~~ and the general - purpose registors, but they do not require space to share memory information. Information about open files of I/O devices in use etc with so little content it is much faster to switch between threads. In other words, it is relatively easier for content switch using threads.

The most obvious advantages of this technique is that a user - level threads package can be implemented on an operating system that does not support threads.