Final Term Assignment

Course Name: Operating System

Submitted By:

Muhammad Safeer (13033)

BS (SE-8) Section: A

Submitted To:

Sir Daud Khan

Dated: 22nd June 2020

Department of Computer Science,
IQRA National University, Peshawar Pakistan

Final Term Assignment

Operating System Concepts

Time Allowed: 6 hours ***Marks: 50***

Note: Attempt all questions. Copying from Internet and one another is strictly prohibited. Such answers will be marked zero.

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

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

Q3. Which component of an operating system is best suited to ensure fair, secure, orderly, and efficient use of memory? Also identify some more tasks managed by that component.

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

Q5. Describe the difference between external and internal fragmentation. Why should they be avoided?

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

Q7. Why is the context switch overhead of a user-level threading as compared to the overhead for processes? Explain.

Q1
Answer:

Yes, in deadlock prevention strategy it's necessary to check that either safe state exists or not because some deadlock prevention strategy check all the request created by processes for resources, it checks for the safe state, if after granting request system remains within the safe state it permits the request and if there's no safe state it doesn't permit the request created by the process.

A state is safe if the system can allot all resources requested by all processes (up to their declared maximums) while not getting into a deadlock state.

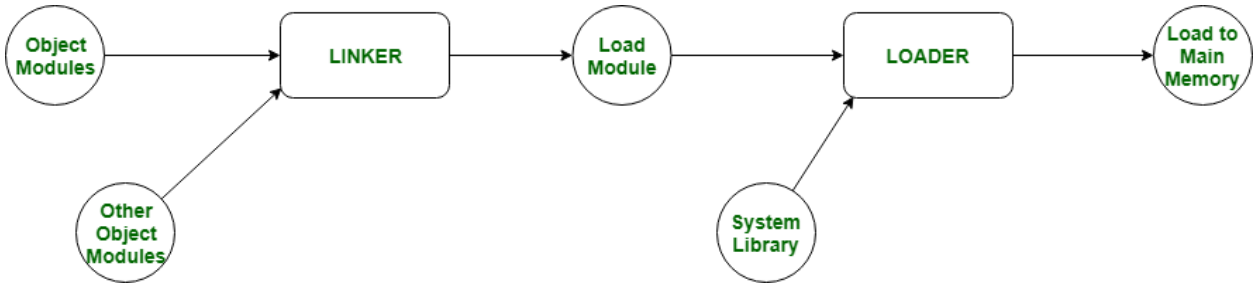
More formally, a state is safe if there exists a safe sequence of processes such that all of the resource requests can be granted using the resources presently allotted. (I.e. if all the processes prior to Pi (P0, P1, P2, P3… PN) end and release their resources, and then Pi is able to end too, using the resources that they have freed up.)

If a safe sequence doesn't exist, then the system is in unsafe state, which can result in deadlock. (All safe states are deadlock free, however not all unsafe states result in deadlocks).

Q2
Answer:

Dynamic loading:
Dynamic loading are the utility programs that play an important role in the execution of a program. The function of dynamic loading is to loads this executable module to the main memory for execution. In other words bringing the program from secondary memory to main memory is called loading.

Dynamic Linking:
It also plays an important role in the execution of a program which intakes the object codes generated by the assembler and combines them to generate the executable module. Connecting the link between all the modules or all the functions of the program to continue the program execution is called linking.



Q3
Answer:

Memory Management is the component of an operating system that is best suited to ensure fair, secure, orderly, and efficient use of memory. Memory Management is the method of controlling and coordinate storage, assigning portions referred to as blocks to numerous running programs to optimize the general performance of the system.
It is the foremost necessary function of OS that manages primary memory. It helps processes to maneuver back and forward between the main memory and execution disk. It helps OS to keep track of each memory location, regardless of whether or not it's allotted to some process or it remains free.

Task managed by Memory Management:

Memory management task is the allocation (and constant reallocation) of memory blocks to different programs as user demands modification. At software level, memory management ensures the provision of adequate memory for the objects and information structures of every running program in any respect times. Application memory management combines 2 connected tasks, called allocation and recycling.
When the program requests a block of memory, a section of the memory manager known as the allocator assigns that block to the program.
When a program no longer wants the information in previously allotted memory blocks, those blocks become accessible for reassignment. This task will be done manually (by the programmer) or automatically (by the memory manager).

Q4
Answer:

Symmetric Encryption:

Symmetric encryption suffers from key exhaustion issues because without proper maintenance of a key hierarchy or effective key rotation. In this encryption it's chance that every usage can leak information that can be leveraged by an attacker. In symmetric encryption it is faster and functions without a lot of overheads on network. A single key shared that can be used for encryption and decryption in symmetric encryption. It doesn't scale very well because the key must not be lost or they can read the messages.

- In this encryption only one key is used, which is also used to encrypt and decrypt the message.
- It is a simple technique by which the encryption will be fast.
- Length of keys is short according to security requirement.
- Key is shared which indicate that risk is higher.

Examples are 3DES, AES, DES and RC4

Asymmetric Encryption:

Asymmetric encryption uses a pair of keys. Public key and private key, Public key is accessible to everyone which is used by others to encrypt the messages they send to you, but to decrypt and read these messages you need to access the private key. These both keys are mathematically related but the private key cannot be derived from it.

- Two different keys are used in this encryption, Public key and Private Key.
- It involves two keys which make it slower than symmetric.
- The length of the key is much larger, recommended size is 2048 or maybe higher.
- Key is not shared which make the process more secure as compared to the symmetric encryption.

Examples are ECC, El Gamal, DSA and RSA.

Q5
Answer:

The difference between internal and external fragmentation is given below:

Internal Fragmentation:
It happens when the memory is split into mounted sized blocks. Method request for memory the mounted sized block is allotted to the method, but if the memory allotted to the method is larger than the memory requested, then the distinction between allotted and requested memory is internal fragmentation.

- Fixed sized Memory, blocks unit appointed to method.
- It happens when the method or process is larger than memory.
- Best-fit is the solution of internal fragmentation.
- It occurs when the memory is divided into fixed sized partitions

External Fragmentation:
It happens when there is a enough quantity of area within the memory to satisfy the memory request of a method. The process memory request cannot be fulfilled because the memory offered is during a non-contiguous manner. If you apply first- fit allocation or best-fit allocation strategy it will cause external fragmentation. External fragmented blocks are accessible for allocation, but may be too small to be of any use.

- Variable Sized memory blocks appointed to method.
- It happens when the method or process is removed.

- Paging and segmentation is the solution of external fragmentation.

- It occurs when memory divided into variable size partitions.

The problem of internal fragmentation is resolved by assigning memory to the programs in dynamic portions of memory blocks at their wish and frees it when there is no need for it during the execution of a program. The solution is compaction but the problem is expensive to implement therefore the paging and segmentation is the best way for it.

Q6
Answer:

The four memory allocation algorithms are:
First-Fit: In the linked list of obtainable memory addresses, we have a tendency to place information within the 1st entry that may fit its data. Its aim is to minimize the number of looking out, however results in external fragmentation afterward. This approach is to allot a hole massive enough which may accommodate the process. It finishes after finding the first appropriate free partition.
Next-Fit: Just like 1st fit, however rather than looking out from the start every time, it searches from the last successful allocation. Greatly reduces the number of searching however leaves external fragmentation at the start of memory. Next fit is a modified version of 1st work. It begins as 1st fit to find a free partition. When called next time it starts looking out from where it left off, not from the start.
Worst-Fit: Traverses the memory and provides the partitions as massive spaces as possible to depart usable fragments left over. Has to search the entire list and such could be a poor performer. Worst fit approach is to find largest obtainable free portion so the portion left are sufficiently big to be useful. It's the reverse of best fit.
Best-Fit: Fastidiously scours the memory for spaces that completely fit the RAM we would like. However, the search is probably going to require a really long time. The best fit deals with allocating the smallest free partition that meets the need of the requesting process. This algorithm 1st searches the whole list of free partitions and considers the smallest hole that's adequate. It then tries to search out a hole that is close to actual process size required.
We most commonly use first-fit and next-fit in practice. They're faster to use and easier to implement.

Q7
Answer:

To give every method on a multi-programmed machine a good share of the C.P.U., a hardware clock generates interrupts sporadically. This permits the package to schedule all processes in main memory to run on the C.P.U. The interrupt handler checks what quantity time this running method has used. If it's burnt up its entire time slice, then the CPU programming formula (in kernel) picks a special process to run. Every switch of the CPU from one process to a different is termed a context switch.

The values of the C.P.U. registers area unit saved within the process table of the method that was running simply before the clock interrupt occurred. The registers area unit loaded from the method picked by the C.P.U.

In a multi-programmed uniprocessor ADPS, context switches occur oft enough that everyone processes seem to be running at the same time.

If a method has over one thread, the package will use the context switch technique to schedule the threads so that they seem to execute in parallel. This can be the case if threads area unit enforced at the kernel level.

Threads may also be enforced entirely at the user level in run-time libraries.

Since during this case no thread programming is provided by the package, it's the responsibility of the engineer to yield the CPU enough in every thread therefore all threads within the process will build progress.

The most obvious advantage of this method is that a user-level threads package are often enforced on Associate in nursing package that doesn't support threads.

User-level threads don’t need modification to operative systems. Every thread is painted just by a computer, registers, stack and a little management block, all hold on within the user method address house. This merely means making a thread, switch between threads and synchronization between threads will all be shunned intervention of the kernel.