***Final Term Assignment***

***Operating System Concepts***

***Name # Muhammad Bilal khan***

***ID # 12945***

***Subject # Operating System***

***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.

**ANSWER**

Yes , in deadlock the prevention strategy it is necessary to check that either safe exists or not because some deadlock prevention strategy check all the request created by process for resources it check for the safe state , if after granting request system remaining within the safe states it permits the requests if thee is no safe state it does not permit the request created by the process.

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

More formally a state is safe if there exists a safe sequence of process such that all of the resources request can be granted using the resources presently allocated .(I .E If all the process 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 sequences does not exist then the system is an safe .

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

**Answer #**

**Difference Between Dynamic loading and Dynamic Linking**

**1 Dynamic Loading**

Dynamic loading are that which a function or programe is not loaded until is known is dynamic loading.

In dynamic loading all the routine are help on disk in relocatable load format that means that if the need of requirement in future that are saved in disk .If we call that requirement they loud into memory .

The main program is loaded into memory is execute .The main function or program that are execute loud into main memory suppose the main program are call sum and also call sub that are routine are saved in disk .

When a routine need to call another one the calling routine the first check to see weather the other routine has been loaded.

If not the relocated linker loader is called to load the desired routine into memory .

Then the cantrol is passed newly loaded routine .

#2 **Dynamic Linking**

Linking postponed until execution time .That means that the program are not execute that the external program they have not connected to the link .

Small piece of codes stubs used to locate to appropriate memory resident library routine are used to library if the routine is not already in memory

Stubs are used into program that stubs are replace itself with the adress of of the routine execute the routine .

Dynamic linking lets several program we are a simple copy of the library codes .

**Example #** You have a program that will read a file ,write a file or give status of the file like length.

**Q3.W**hich 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.

**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.

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

**ANSWER**

The difference between symmetric and A- Symmetric encryption are given below .

**Symmetric Encryption**

1 **)** In symmetric encryption only one key are used in encryption and description .

2 ) In symmetric is also called private key or secret key .

3 ) symmetric encryption are faster in execution .

4 ) They are less complex and less computational power is required.

5 )It is used for transfer bulk data between executes faster .

**A - Symmetric encryption**

1) A - symmetric encryption is also called public encryption

2) In A- symmetric encryption are two keys (public key and also private key ) is used for encryption and decryption .

3) Thay are slower execution .

4) They have more complex and more computional and power needed .

5)They used for secretly exchanging the secret key .

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

**ANSWER No # 5**

**Difference between external and internal fragmentation**

**1) Internal Fragmentation**

* In external fragmentation total memory space is enough to satisfy a request On reside of the process in it but it is not contiguous so it cannot be used .

2) It occurs when the fixed sized memory blocks are allocated to the process.

3) When the memory are assigned to the process is slightly larger than the memory requested by the process that create free space in the allocated block causing internal fragmentation .

4) Paging suffers from internal fragmentation .

5) Solution for internal fragmentation the memory must be partitionad into variable size block and assigned the best bit block to the process .

**External Fragmentation**

**1)** It occurs when veriable size memory block are allocated to the process dynamically

2) When there is sufficient of amount of space in memory to satisfy they request of the process .But the process memory request can not be satisfied because the memory veriables is non contiguous manner .

3) Segmentation suffers from external fragmentation .

4) The solution for external fragmentation is paging and segmentation in that one it allow a process that accure physical memory space in a non contigous manner .This can be achiveved by paging and segmentation

**T**he problem of internal fragmentation can be solved by assigining memory to the programs in dynamic partition of memory block at their wish and free it when there is no need for its during the execution of a program.

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

**ANSWER**

The four memory allocated algorithems are that

**First -Fit**

In the linked list of obtainable memory adresses we have tendancy to a place information within the 1st entry that may fit its data .it aims is to minimaize 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 finish after finding the first appropriate free partition .

Next -Fit

Just like first fit however rather than looking out from the start everytime it searches from the last successful allocation .greatly reduce 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 is start looking out from where it left off not from the start .

Worst-Fit

Traverses the memory and provides the partition as massive space as a possibles to depart as usable fragments left over. Has to search the entire list such could be a

poor performer .worst fit approach is to find largest obtained free partition so the partition left are sufficiently .

**Best -Fit**

Fastidiously scours the memory for space that completely fit the RAM we would like .however the search is probarly going to require a really long time . This algorithms 1st searches the whole list of free partitions and considers the smallest hole that adequate .

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

**ANSWER # 7**

To give every method on a multiprogrammed 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|central methoding unit|CPU|C.P.U.|central processor|processor|mainframe|electronic equipment|hardware|computer hardware} programming formula (in kernel) picks a special process to run.

every switch of the {cpu|central methoding unit|CPU|C.P.U.|central processor|processor|mainframe|electronic equipment|hardware|computer hardware} from one process to a different is termed a context switch.

The values of the {cpu|central methoding unit|CPU|C.P.U.|central processor|processor|mainframe|electronic equipment|hardware|computer hardware} 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 multiprogrammed uniprocessor ADPS, context switches occur oft enough that every one 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.

every thread therefore all threads within the process will build progress.

User-level threads implement in user-level libraries, instead of via systems calls, therefore thread switch doesn't ought to decision package and to cause interrupt to the kerne

In fact, the kernel is aware of nothing concerning user-level threads and manages them as if they were single-threaded processes.

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 doesn't need modification to operative systems.

every thread is painted just by a computer, registers, stack and alittle management block, all hold on within the user method address house.

There is an absence of coordination between threads and package kernel.