***Final Term Assignment***

***Operating System Concepts***

***Time Allowed: 6 hours 11775 Talha Sajjad 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, it is important to check that safe state exists. In a deadlock prevention process safe state has to exists as it make sure the process takes place in step by step order. System is in safe state if there exists a sequence for all processes. Sequence <P1, P2, Pn> is safe for each Pi. Since is j<i. Without safe state existence there will be deadlock since the processes would collapse.   
There are 4 type of conditions if we avoid that we can prevent deadlock to happen.  
1) Eliminate Mutual Exclusion: It is not possible to dissatisfy the mutual exclusion because of some of the resources or things that are attached to the system e.g Tape drive and printer.  
2) **Eliminate Hold and wait:** Allocate all required resources to the process before the start of its execution, this way hold and wait condition is eliminated but it will lead to low device utilization.

1. **Eliminate No Preemption:** Preempt resources from the process when resources required by other high priority processes.
2. **Eliminate Circular Wait: in this process a process can request the resource increase and decreasing. Order of numbering each resource will be assigned with a numerical number.**

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

ANS) Dynamic loading is basically the type of loading when a certain part of the program is loading into main memory only when it is called by the program. This type is call Dynamic loading as it is used to increase performance. This is useful in efficient memory usage since many of it’s subs may not be considered calling out. Program > compiler > linker > loader > in execution. The dynamic loading for example can be created using LoadLibrary call in C or C++.

While Dynamic linking refers to the linking that can be done during load time and not when the exe is being created. In case of dynamic linking the the exe does the minimal work. It is also called dynamic loader. E.g to call functions in a shared library we use dynamic linking but u must specify the shared library.

**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.

**ANS) The kernel is a computer program which is a part of operating system and it has a complete control over everything in the system. It is an important and integral part of the operating system. It interacts between both software and hardware components. The critical code of kernel is often loaded in seperate part of the memory. It includes various other processes that are task managed by it. E.g It handles the input/output requests from software, translating them into data processing for CPU. It also includes to handle like printers, speakers, Keyboard and monitors. When a process make a request to the kernel it is called system call. It resets the operating states of the CPU for the best operations at all times. It loads first in start and remain in the main memory.**

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

ANS) Symmetric uses a single key that is used between people who wants to receive messages while in in A-symmetric encryption we use a pair of private key and public that are used for encrypting and decryption the messages when communicating.A-symmetric encryption is relatively new method compared to symmetric method which is an old method. E.g AES, DES, RC4 are examples of symmetric while A-symmetric encryption AES-128, AES-192, AES-256.

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

#### Ans) Internal fragmentation happens when the memory is split into mounted size blocks. Whenever a method requests for the memory the mounted sized block is allotted to the method. Internal fragmentation is the wasted space within each allocated block because of rounding up from the actual requested allocation. While External is the free space holes that are generated in your disk space or your memory. External fragmentation happens when there’s sufficient area within the memory to satisfy the memory request of a method.

External Fragmention:

Fragment 15kb

Assigned space=>

Fragment 10kb

Assigned space=>

Fragment 5kb

Internal Fragmentation:

Assigned space=> Fragment

USED SPACE

Assigned space> Fragment

USED SPACE

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

Answer) First Fit: it is used in start to allocate free space or hole large enough which can accommodate the process. It finishes after it finds a free partition first. It is fastest algorithm considering as it searches as little as possible.

Best Fit: It is used to to meet the requirement of the requesting process by allocating smallest free partition. It memory allocation is better than first fit as it looks for smallest free space.

Worst Fit: In worst fit approach is to locate largest available free portion so that the portion left will be big enough to be useful. It is the reverse of best fit.

Next Fit: Next fit is basically an advanced version of first fit. It starts free space and when it’s called again it continues from the place where it left off.

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

Ans) In User-level threading generally if we want threads to be fast and cheap we make sure it is on user level implemented. It is entirely managed by run time system. The kernel knows nothing about user level threads and handle them as single threaded processes. They’re represented by stack, register and thread control block. In other words, it is relatively easier for a context switch using Threads. Sharing Treads allow the sharing of a lot resources that cannot be shared in process for example sharing code section, data section, Operating System resources like open file etc. The most common and obvious advantage of this technique is it can be implemented on an operating system that does not support threads.   
Advantages: User level threads does not require modifications to the operating system.

Each threads is simply represented by PC, registers, stack and a small control block.

It is often fast and efficient and cheaper than Kernel level. They do not need inter process communication they can simply share data. They can take advantage of the multiprocessors cause of very nature. Threads use very little resources of operating system.