NAME: AMIT SINGH

ID: 15478

SUBJECT: OPERATING SYSTEM

SEMESTER: 3RD

PROGRAMME: BS (SOFTWARE ENGINEERING)

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.

ANS) Yes, it's necessary to check whether the system is in safe state or not which means there exist safe sequence of processes. A safe sequence exists when all the process finish or finished their execution successfully. A deadlock can be prevented when the system is in safe state.

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

ANS) Examples:

Dynamic Loading	Dynamic linking
Suppose our program that is to be executed consist of various modules. Of course it's not wise to load all the modules into main memory together at onceSo basically what we do here is we load the main module first and then during execution we load some other module only when its required and the execution cannot proceed further without loading it	IF our program has some functions whose definition is present in some system library. We do know the header file only consists of declarations of functions and not definitions. So during execution when the function gets called we load that system library into main memory and link the function call inside our program with the function definition inside system 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) Memory management is suited best. Memory management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution. Memory management keeps track of each and every memory location, regardless of either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes.

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

ANS) Examples:

SYMMETRIC ENCRYPTION	A-SYMMETRIC ENCRYPTION
It is used when a large amount of data is required to transfer.	It is used to transfer small amount of data.
it only provides confidentiality	It provides confidentiality, authenticity and non-repudiation
3DES, AES, DES and RC4	Diffie-Hellman, ECC, El Gamal, DSA and RSA

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

INTERNAL FRAGMENTATION	**EXTERNAL FRAGMENTATION**
IT OCCURS WHEN MEMORY IS DIVIDED INTO FIXED SIZED PARTITIONS IT CAN BE CURED BY ALLOCATING MEMORY DYNAMICALLY OR HAVING PARTITIONS OF DIFFERENT SIZES	IT OCCURS WHEN MEMORY IS DIVIDED INTO VARIABLE SIZED PARTITIONS BASED ON THE SIZES OF PROCESSES IT CAN BE CURED BY COMPACTION, PAGING AND SEGMENTATION

External fragmentation can be reduced by compaction or shuffle memory contents to place all free memory together in one large block. To make compaction feasible, relocation should be dynamic .External fragmentation is also avoided by using paging technique

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

ANS)

Fixed partitioning the main memory is divided into partitions of equal or different sizes.

Variable partitioning the system of dividing memory into non-overlapping but variable sizes

Dynamical memory allocation We can dynamically allocate storage space while the program is running, but we cannot create new variable names

First fit In the first fit the partition is allocated which is first sufficient from the top of main memory.

Next fit, best fit or worst fit. First and next fit are generally preferred as they reduce external fragmentation and have low overheads

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

ANS) A user-level thread is a thread within a process which the OS does not know about. In a user-level thread approach the cost of a context switch between threads can be made even lower since the OS itself does not need to be involved–no extra system calls are required. A user-level thread is represented by a program counter, registers, stack, and small thread control block (TCB).. This provides an interface for creating and stopping threads, as well as control over how they are scheduled. When using user-level threads, the OS only schedules processes, which in turn are responsible for scheduling their individual threads. Unfortunately, since the threads are invisible to the OS, the OS can make poor decisions such as scheduling a process with idle threads or giving unbalanced CPU shares between processes that have different numbers of threads.