NAME - Daniyal Alam

I.D : 15385.

Subject: Operating Systems

Teacher : Daud Khan.

Q: 1

Ans:

Yes, it is necessary to check that either safe state exists or not in deadlock prevention because when we disallow deadlock by setting "safe states", it means process completion is always guaranteed.

Q: 2

Ans:

Dynamic Loading:

Dynamic loading is a process/mechanism by which a computer program can, at run time, load a library (or other binary) into memory, retrieve the addresses of functions and variables contained in the library,

execute those functions or access those variables, and unload the library from memory.

Loading the Library Example:

Loading the library is accomplished with LoadLibraryEx on windows. Example follows:

```
HMODULE sdl_library = LoadLibrary(TEXT("SDL.dll"));
if (sdl_library == NULL) {
    // report error ...
} else {
    // use the result in a call to GetProcAddress
}
```

Dynamic linking:

Dynamic linking refers

to the linking that is done during load or run-time and not when the exe is created.

In case of dynamic linking the linker while creating the exe does minimal work. For the dynamic linker to work it actually has to load the libraries too. Hence it's also called linking loader.

Programs that are linked dynamically are linked against shared objects that have the extension.

We use a command-line option to the compiler driver gcc to tell the tool chain whether we are linking statistically or dynamically. This command line option then determines the extension used (either .a or .so).

Q:3

Ans:

Memory management is the best suited to ensure fair, secure, orderly and efficient use of memory in operating 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 processes in and out of main memory.

Q.4
Ans:

Comparison Factor	Symmetric Encryption	Asymmetric Encryption.
=> Number of Cryptographic Keys.	It incorporates only one key for encryption.	It consists of two cryptographic keys. These keys are regarded as Public & Private key.
=> Complexity.	It is a simple technique compared to the other as only one key is employed to carry both operations.	Contribution from seperate keys for encryption and decryption make it a complex process.
=> Swiftness of Execution.	Due to its simplistic nature both the operations can be carried out pretty quickly.	Because of encryption and decryption by two seperate keys & the process of comparing them make it a tad slow procedure.
=> Algorithms Employed/Examples.	RC4, AES DES, QUAD	RSA, ECC Diffie-Hellman

Q:5
Ans:

Internal Fragmentation:

It happens when the memory is split into mounted sized blocks. Whenever a method request for the memory, the mounted sized block is allotted to the method. If the memory allotted to the method is somewhat larger than the memory requested, then the distinction between alloted and requested memory is that the internal fragmentation.

External Fragmentation:

It happens when there's a sufficient quantity of area within the memory to satisfy the memory request of a method. However the process's memory request cannot be

fulfilled because the memory offered is during a non-contiguous manner. Either you apply first-fit or best-fit memory allocation strategy it'll cause external fragmentation.

Why should be avoided?

The main reason of internal and external fragmentation is memory wastage and inflexibility. As the memory is allocated to a file or a process it will grow during the run. But unit a file grows many blocks allocated to it remains unutilized.

Q: 6

Ans:

First fit:

allocate into the first available gap found of adequate size, starting from the beginning of memory.

Next fit:

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

Best fit:

search all of memory to find the smallest valid gap and allocate into it on the basis that it'll leave the smallest unusable gap.

Worst fit:

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

=> First and Next fit are used in practise as they are faster, with comparable performance.

Q: 7

Ans: User-level threads implement in user-level libraries rather than via system calls, so thread switching does not need to call the operating system and to cause interruption to the kernel. In fact kernel knows nothing about user-level threads and manages them as if they were single-threaded processes. Threads are very inexpensive to create and destroy, and they are inexpensive to represent. For example they require space to store, the PC, the SP, and the general purpose registers, but they do not require space to share memory and formation about open files of I/O devices in use, etc.

The most obvious advantage of this technique is that a user-level threads package can be implemented on a OS that does not support threads.

=> User-level threads do not require modification to OS.
=> Each thread is represented simply by a PC, registers, stacks, all stored in the user process address space.
=> Simple management:
This simply means that creating a thread, switching between threads and synchronization b/w threads can all be done without intervention of the kernel.
=> Thread switching is not much more expensive than a procedure call.