CMPUT 379 - Practice Questions

1. General Concepts

1.1 Briefly explain (in point-form) each of the following concepts/objects:

Time-sharing - Time-sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time.

Multiprogramming - In a multiprogramming system there are one or more programs loaded in main memory which are ready to execute. Only one program at a time is able to get the CPU for executing its instructions while all the others are waiting their turn.

Multiprocessing - Multiprocessing sometime refers to executing multiple processes (programs) at the same time. This might be misleading because we have already introduced the term “multiprogramming” to describe that before.

Threads - A thread is a basic unit of CPU utilization; it comprises a thread ID, a program counter, a register set, and a stack. It shares with other threads belonging to the same process its code section, data section, and other operating-system resources, such as open files and signals. A traditional process has a single thread of control. If a process has multiple threads of control, it can perform more than on task at a time.

Processes - A program in execution. A process is the unit of work in a modern time-sharing system.

Programs -

Jobs - The term job and process are used almost interchangeably in this text.

perror ( ) - The perror ( ) function produces a message on standard error describing the last error encountered during a call to a system or library function.

errno - The <errno.h> header file defines the integer variable errno, which is set by system calls and some library functions in the event of an error to indicate what went wrong.

fork ( ) - Fork system call use for creates a new process, which is called child process, which runs concurrently with process and this process is called parent process. After a new child process created, both processes will execute the next instruction following the fork ( ) system call. A chiles process uses the same program counter, same CPU registers, same open files which use in the parent process.

execl ( ) - The exec family of function replaces the current running process with a new process. It can be used to run a C program by using another C program. It comes under the header file unistd.h There are many members in the exec family which are shown below with examples. execl also serve the same purpose bu the syntax is a bit different.

signal ( ) - A signal is a software generated interrupt that is sent to a process by the OS because of when user press ctrl-c or another process tell something to this process.

sigaction ( ) - The sigaction ( ) system call is used to change the action taken by a process on receipt of a specific signal.

pipe - A pipe acts as conduit allowing two processes to communicate. Pipes were one of the first IPC mechanisms in early UNIX systems. They typically provide one of the simpler ways for processes to communicate with one another, although they also have some limitations.

Deadlock Prevention:
Mutual Exclusion - At least one resource must be non-sharable. Sharable resources, in contrast, do not require mutually exclusive access and thus cannot be involved in a deadlock.
Hold and Wait - We must guarantee that, whenever a process requests a resource, it does not hold any other resources.
No Preemption - There be no preemption of resources that have already been allocated. To ensure that this condition does not hold, we can use the following protocol. If a process is holding some resources and requests another resource that cannot be immediately allocated to it, then all resources the process is currently holding are preempted. In other words, these resources are implicitly released. The preempted resources are added to the list of resources for which the process is waiting. The process will be restarted only when it can regain its old resources, as well as the new ones that it is requesting.