Part 1: Operating Systems (Processes and Memory)  
a) Describe in detail what process management is and detail how a Linux system manages processes using process id's, sessions and  
groups.  
b) “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”. Explain in detail the operation of swapping and paging when used  
with respect to memory management.

**PROCESS MANAGEMENT**

**1.PROCESS ID**

In UNIX, all child processes of a process share a process group. All processes fall under a tree of the special init process. All child processes will share user input simultaneously and react to the input separately. In POSIX, fork is the only system call to create a process, it will create a copy of original process. Fork will return a PID to the parent process and return 0 to child process. Child process can know its own PID using system call getpid.

**2.SESSION**

When a user logs out of a system, the kernel needs to terminate all the processes the user had running (otherwise, users would leave a slew of old processes sitting around waiting for input that can never arrive). To simplify this task, processes are organized into sets of sessions. The session's ID is the same as the pid of the process that created the session through the setsid() system call. That process is known as the session leader for that session group. All of that process's descendants are then members of that session unless they specifically remove themselves from it. The setsid() function does not take any arguments and returns the new session ID.

In most cases a session is related to the terminal process, and all process a user created will be given a SID the same as that terminal’s PID.

**3.PROCESS GROUP**

Process is a running program in memory. The concept of Process Management is to maintain the illusion of multiple processes running paralleled. This means to switch between processes. In order to switch between processes, processes are periodically suspended and invoked. A running process will switch between running state, ready state, and blocked state. For example, a process may be blocked and wait for a disk interrupt. While suspended, all context related to a process will be stored in what we call a process table. Multiprogramming can actually enhance the efficiency of CPU.

**MEMORY MANAGEMENT**

**Since the memory is far from big enough to store all the running programs, two methods dealing with memory overload were created.**

**1.SWAPPING**

Swapping is to store a complete program into memory and store it back to disk after sometime being active. Idle processes are stored in the disk. Most processes are changing between being active and idle and swapped between memory and disk. Swapping will generate holes in memory, therefore, a technique called memory compaction that moves processes to lower addresses was created.

**2.PAGING/VIRTUAL MEMORY**

Since program grows in size too fast, paging itself can’t suffice the need of memory, virtual memory was born to solve this. Sometimes, even a single program is too large to be all loaded into memory. The basic idea of virtual memory is to separate a program’s memory address space into small sections called page. Addresses generated during runtime are called virtual addresses, and virtual addresses are sent to MMU and then mapped into physical addresses on memory. A process doesn’t require all pages in the memory to run. Whenever an address is needed, system will go to look up the page table in MMU, if the virtual address is not mapped to the physical address, system will find a less used page frame and swap it to the disk if it is dirty, then load the desired content and change the mapping of page table.