**CSCE 611 600: OPERATING SYSTEMS Homework #1**

**Name:** Rohan Chaudhury

**Email Address:** [rohan.chaudhury@tamu.edu](mailto:rohan.chaudhury@tamu.edu)

**UIN:** 432001358

**List of people I have worked with:** None

**Question 1:**

What is the difference between processes and threads?

**Answer:**

1. A process is a program under execution i.e., a program that is active whereas a thread is a lightweight process that a scheduler can manage independently. Threads are basically segments of a process.
2. For context switching, processes require more time than threads as processes are heavier than threads.
3. Processes are independent of other processes and don’t share any memory with them whereas a thread may share some memory with its peer threads.
4. Processes require more time for communication between them than for communication between threads.
5. If a process gets blocked, the remaining processes can continue execution whereas if a user-level thread gets blocked, all its peer threads also get blocked.
6. The processes have independent code and data segments whereas threads share the files, code segment, data segment, etc. with their peer threads.
7. All the different processes are treated separately by the operating system whereas all user-level peer threads are treated as a single task by the operating system.
8. More time is required for the creation of processes than for threads.

**References for this question:**

https://www.tutorialspoint.com/difference-between-process-and-thread

https://www.geeksforgeeks.org/difference-between-process-and-thread/

**Question 2:**

What is spooling? Do you think that advanced personal computers will have spooling as a standard feature in the future?

**Answer:**

SPOOL stands for simultaneous peripheral operations on-line. It is a process or a type of buffering mechanism in which data is temporarily held to be used and executed by a device, program or the system. Until the computer or program requests the data for execution, it is sent to and stored in memory or other volatile storage. Relative to the performance of the system, peripheral devices such as card readers, printers, etc. are very slow. Because of this reason, getting input and output from the system can be a bottleneck and that is why spooling was needed.

Spooling works like a request queue where data, processes, and instructions from several sources are accumulated to be executed later in a FIFO manner i.e. whichever instruction had entered first in the queue will be popped and executed first. It is generally maintained on the physical memory of the computer, buffers, or the I/O device-specific interrupts.

I think that in the future, advanced personal computers will have output spooling as a standard feature but not input spooling. Input spooling is the process where information about a job is taken from the input device, prepared for scheduling, and is placed as an entry in the job queue. It is the process of reading in jobs from peripheral devices such as printers onto the disk so that when the processes currently executing on the CPU are finished, it will get new jobs to execute. Whereas output spooling is the process of sending the job output from the system to disk storage instead of sending it directly to an output device to allow the job that produces the output to continue processing without considering the availability or the speed of the output devices. As advanced computers will be highly efficient in reading in and executing processes from peripheral devices, input spooling would not be required. However, peripheral devices will still be slower than computer systems so output spooling may be required and added as a standard feature in the future.

**References used for this question:**

https://www.geeksforgeeks.org/what-exactly-spooling-is-all-about/

https://www.ibm.com/docs/en/i/7.3?topic=concepts-input-spooling

https://www.ibm.com/docs/en/i/7.2?topic=concepts-output-spooling

https://studylib.net/doc/8982175/answer

**Question 3:**

What is priority inversion with respect to processes? Give an example.

**Answer:**

A scenario in which a higher priority process is pre-empted by a lower priority process in an operating system is known as priority inversion. So in this case, the priorities of the two processes are inversed.

To give an example:

We consider an application having 3 threads:

T1 thread has the highest priority.

T2 thread has a medium priority.

T3 thread has the lowest priority.

Now we assume that T1 and T3 are sharing the same critical section (CS) code.

At the beginning of our example scenario, we assume that T3 is running and has entered a critical section whereas T1 and T2 are sleeping or blocked.

Now, when T2 starts running, it pre-empts T3 because T2 has a higher priority than T3. Now in this case T3 is still owning the shared critical section.

Later, when T1 starts running, it pre-empts T2 and tries to enter the shared critical section which is still owned by T3. Since the critical section is owned by another thread, T1 gets blocked, and it waits for the critical section CS.

So now, T2 starts running since it has a higher priority than T3 and T1 is not running. T3 doesn’t release the critical section and T1 is kept waiting while T2 continues to run.

So, in this case, the thread with the highest priority in the system, T1, becomes blocked and it waits for lower priority thread T2 to run. This is an example showing priority inversion with respect to processes.

**References used for this question:**

https://www.tutorialspoint.com/priority-inversion

https://stackoverflow.com/questions/4252158/what-is-priority-inversion