Course : Operating Systems **Instructor:** 黃能富

Student name: Tuguldur Tserenbaljir **Assignment:** Assignment 1

Student ID: 109006271

3.9 Describe the differences among short-term, medium-term, and long-term scheduling.

In the operating system view, the timing is an essential tool for providing resources equally to processes and it is divided into short-term, medium-term, and long-term, with their unique functions.

Scheduling in the long-run is an instrumental factor as it enables selecting the programs to be executed and in this way effectively controls the multiprogramming levels and system load. Here we schedule strategically at this level of system optimization where we decide resource deployment not frequently but for a long time and that determines the efficiency of the channel.

In the medium run, what is called swapping regulates the active work during learning and memory optimization, which helps CPU and memory use. It can dynamically change the count of active processes according to system load by swapping processes out to backing storage, if there is need of, to ensure that the load on the system is managed evenly

Short-term scheduling makes a selection of the processes that are ready to run next, and as a result, impacts the replies to users of the system. With the use of algorithms like first-come, first-served or round-robin, it aims at giving the finest performance based on the optimization of response time, and CPU utilization as its performance criteria.

The layers of scheduling operate either concurrently in parallel, or one after the other in each category to minimize resource use, maintain performance, and achieve the overall goal of providing better system and user services.

3.10 Including the initial parent process, how many processes are created by the program shown below?

Course : Operating Systems **Instructor:** 黃能富

Student name: Tuguldur Tserenbaljir **Assignment:** Assignment 1

Student ID: 109006271

In the process shown below, the fork() function is called 3 times. There are 8 processes created in total. Because whenever a fork() function is called, it creates processes as child nodes of a growing binary tree. Therefore, the formula to find how many processes are created by the program we can use the formula 2^n where n is the number of times the fork() function is called. So, in our case it will be 2^3 where it is 8 where there will be 1 parent alongside 7 children.

3.11 Using the program below, identify the values of pid at lines A, B, C, and D. (Assume that the actual pids of the parent and child are 2600 and 2603, respectively.)

The child process returns a value of 0 after the fork() call. The same process is true for the parent process in which the child process returns the PID of the parent. In this scenario:

NOTE that at point a (right within the child process), pid equals 0 because it is in other words the return value of fork() in the child which is seen at this point.

The childs PID at point B (myself also being the child process) is 2603 which is the value of pid1 inside the child process (getpid() obtains this value).

In the next point (at C) the process number of the parent is 2603, which is the PID of the child that exist in fork() of the parent process.

By arriving at D (the process that forks into), we know that PID of the parent process is 2600, which is also got from the parent process with getpid().

So, the values are as follows: A: 0; B: 2603; C: 2603; D: 2600.