Homework 5

PART 1

UNIX – PROCESSES

Processes and Job Control

A. From the UNIX handout, read and cover the examples given in subchapters:

18. ***Foreground and Background Processes (fg, bg, suspend)***
19. ***nice*** and
20. ***Abnormal termination of processes***

B. More on these topics:
Sequential , Parallel execution

Execute:
date& who; whoami; uname; echo Hello, World!&

- **Show the output of your command.**

Explanation: the last "**&**" puts all the commands after the previous "**&**" in one process;, therefore the date command executes as one process and all the commands in **who**; **whoami**; **uname**; and **echo Hello, World!&** as another process. Note again that the output of the date command ppears after the output of the who command, owing to the scheduling of the two processes.
UNIX allows you to group commands using semicolons and enclose them in parentheses. All of the commands are executed sequentially.

nice command
The priority value for every process in the system is recalculated every second. When it is time for scheduling, the CPU is given to the process with the smallest priority number. If multiple processes have the same priority number, a decision is made based on FCFS order.

Priority value = Threshold priority + Nice value + (recent CPU usage/2)

Threshold priority: integer having a value of 40 or 60
Nice value: integer with a default value of 20
UNIX favors processes that have used less CPU time in the recent past.

- **To understand more about the nice command, look at the man page.**
Give the default value for adjustments.

The adjustment value (between 1 and 19) can be used to decrement the priority of a process. The **Superuser** can use the **nice** to increase the priority of a command by using a negative value.

PART 2

Give an outline of the scheduling algorithms currently used in Windows XP, Unix (Linux) and MacOS.

PART 3

JAVA THREADS

In Java, write a program that creates three concurrent threads.
Each thread will keep printing a statement and sleep for a random time. The run method should be a ***while(true)*** loop.

Try to use the ***age( )*** method to keep track of the time.

```
protected static final long age() {
return System.currentTimeMillis() - startTime;
}
```

Submit a copy of your java program together with a sample of its output.