**CIS 657 Programming Assignment 1**

Submit two files:

* Create your document -- readme (doc or pdf) including the **disclosure form**, **screenshot(s) of your output**, **your solution code *– Must be text (image cannot be process with TurnItIn) --* (filename including full path, and your own code (created or modified by you), not Nachos original code) with description (you need to explain your solution)**  to the **Turnitin link** on the blackboard (under Assignments).
  + All of your code must be well documented.
* Do a “make clean” in code/build.linux, then compress your Nachos folder on the VM, download the compressed file into your local machine and name the compressed file **Ass1\_yourNetId.zip (Note: Zip only!!)**. Submit the zipped file to the **Assignment Code Submission link** of Blackboard.

Due: Nov. 2 (Friday, end of the day)

Late submission: you have total 72 hours throughout the semester.

*Follow the Lab1 instruction and create a new fresh Nachos folder.*

**Overview**

You are going to be adding multi-threading capabilities to Nachos. From the Unix point of view, Nachos is a process with user space threads. From the Nachos point of view, Nachos is an operating system that has processes without multi-threading. This assignment requires that you add multi-threading in Nachos as follows:

1. Create a Process class
2. Integrate the Process class with the Nachos Threads and System
3. Change the scheduler

You need to make some extensions to the Nachos operating system.

**Task 1 (20pts) :** Create a new class called **Process** in files (process.cc|h). Put the class in the threads directory and make sure everything compiles. At a minimum the Process class will need only constructor, destructor and some setter/getter functions.

* Determine what the member variables of the Process class should be and which member variables, if and, need to be pulled out of the Thread class (Nachos existing class)
  + You need to explain why you decided the member variables in Process class
  + If you changed something in the Thread class, explain what you changed in both the Thread class and Process class

**Task 2 (30pts):** Make your Nachos system run with processes. Create Process functions and Thread Fork function.

* Process::Fork – Forks off a child process
* Process::Terminate – Terminates the running process
* Process::Join – Makes a parent process block until all of the listed child processes have terminated. (**BONUS – 10 pts**)
* Thread::Fork – Forks off a new thread within this process. You can assume that this thread is a thread in user space. Change the existing Thread class.

You may need to change other files to run your NEW processes and threads.

**Task 3 (40pts):** Processes now will be added a priority. The Process constructor takes a priority as a parameter. Each process will take the CPU according to its priority. In this assignment, priorities will be between 1 and 5, 5 being the highest.

* **Priority-based scheduling scheduler**
  + For example, when two processes have the same priority, Process A has three user threads and Process B has one thread. All three threads of Process A must share the CPU time of Process A. The only one thread of Process B takes the CPU time for Process B.
  + What is your solution to schedule user threads in the same process?
* Add a Nachos flag **-quantum xxxx** that sets how long the time quantum will be for each process.

**Task 4 (10pts):** Write your own threadtest.cc to prove your multi-threading implementation working correctly. Quality of your testing cases will be graded.

* You need to create meaningful number of Processes and Threads in your testing routine.
* You need to display meaningful information to prove your multi-threading: Which process is running, which thread is running, Process Fork, Terminate and Join, and Thread Fork calls are made.

**Testing:**

We will build and run your Nachos on the VM. *The TAs will create their own copy of threadtest.cc which will test your multi-threading. Make sure that your solutions are general and robust so they will not break.*