1 Purpose
This assignment primarily focuses on process status structures and the Linux scheduler.

2 Readings
Please see assignment web page.

3 Assignment Description

1. Nice Values – Part 1 (10 points)
   On the assignment web page you will find a program called cpuTimeWaste.c. By default the nice value assigned to cpuTimeWaste.c is 0. You will study the effect of the nice command by running instances of cpuTimeWaste.c with different nice values. You can use the nice command at the command line to assign a nice value e.g.,

   nice –n 5 cpuTimeWaste

   You should use these values of nice values 10, -5, 0, 5, 10, 15, 19. You can use the top command to see the impact on the percentage of CPU used by each process. Provide an explanation for the behaviour observed.

2. Nice Values – Part 2 (10 points)
   Assume that you start two instances of cpuTimeWaste.c with the default nice value of 0. Both processes should have similar values for the percentage of CPU used. Assume that one of the processes will have the default nice value.

   • What should the nice value be of the other process to ensure that the process with the default nice value receives approximately 10% of the CPU?
   • What should the nice value be of the other process to ensure that the process with the default nice value receives approximately 95% of the CPU?

3. Compare System Call and function (10 points)
   You are to compare the time it takes to do a minimal function call in C and a minimal system call in C. The minimal function call is one that neither takes any parameter nor does anything inside the function. For the system call use getpid(). The name of this program should be compare.c.
4. **Kernel Information** (15 points)

Using the /proc file system you are to write a program that tells the user the processor type and the kernel version. The information is found in /proc/cpuinfo and /proc/version. Please note that the processor type consists of vendor name and model. The name of this program should be `kernelInfo.c`.

5. **Process Monitoring** (20 points)
The purpose of this part of the assignment is to give you insight on how to monitor processes. You are to write a program that has as its input (through the command line) the name of the executable to be observed. The program you write will start the named executable and report the amount of time this process has been scheduled in user mode and the amount of time that the process has been scheduled in kernel mode. This should be done every second until the monitored process terminates. The data needed for this assignment are found in the file /proc/pid/stat. Please note that the process being observed may have an infinite loop and may be terminated using the `kill` command. You must be able to handle this.

The name of this program should be `observer.c` and it accepts one argument, which is the name of the executable. Test your program thoroughly by observing the behaviour of different types of programs.

6. **Impact of Scheduling Policies - Part 1** (20 points)

There are processes that basically sit idle until an event occurs. We will simulate this by having a process sleep for a second and upon waking up calls a function. This function should be CPU intensive. Basically you should have an infinite loop that alternates between the `sleep` function and calling the CPU intensive function. You should terminate the program with a CTRL-C. You should sleep for one second in each iteration. The name of this program should be `comparison1.c`. You should investigate the behaviour of this program for each scheduling policy.

To investigate the behaviour try the following for each of the scheduling policies:
- On the web page you will find a program called `cpuTimeWaste.c`. Run this program during your experiments. Use tools like `top` to determine the impact that the different scheduling policies have.
- Look at the impact of `comparison1.c` on interactive programs e.g., command line prompt on the shell.

Provide a summary of your observations. What can you conclude from the results?

7. **Impact of Scheduling Policies - Part 2** (15 points):

Modify `comparison1.c` so that the function to be executed after the sleep is I/O intensive. This program should be `comparison2.c`. Repeat the experiments you did for part 6.
4 Report
Write a report with a section for each enumerated item listed in the assignment description. The title of each section should correspond to the title used for each question. Make sure you answer everything requested for each question. This should be backed up by the results of your experiments.

5 Submission
Please submit your any code you developed and your report.
6 Report
Write a report with a section for each enumerated item listed in the assignment description. The title of each section should correspond to the title used for each question. Make sure you answer everything requested for each question. This should be backed up by the results of your experiments.

7 Submission
Please submit your any code you developed and your report.