**ReadMe**

Question 2:

In this project, we have developed three distinct processes, each executing on a single CPU core. These processes are tasked with counting from 1 to 2^32.

We've assigned them different scheduling policies: one employs the "sched\_other" policy with a nice value of 0, while the other two use "sched\_fifo" and "sched\_rr" policies with a priority value of 1.

To configure the scheduling policies for these processes, we've utilized the "sched\_setscheduler" function. Additionally, we've employed the "clock\_gettime" function to measure the time elapsed during the execution of each process. We've created three separate programs, each designed for a specific scheduling policy, to ensure that all processes are in the ready state simultaneously.

To achieve this, we've employed the '&' operator in our makefile, allowing multiple commands to run “simultaneously”. To ensure that all three processes execute on a single core, thereby enabling the CPU to make scheduling decisions, we've used the "sched\_setaffinity" function to bind them to the 0th core. Upon completion of each process, the elapsed time is recorded in a C file named "data.txt." Additionally, a Python script named "histogram.py" was made to generate a bar graph illustrating the time consumption of each process.

Explanation for the outcome we receive:

The three processes have been configured to run simultaneously on a single core, the outcomes therefore demonstrate how the CPU core makes scheduling decisions among these processes. Context switching takes place among processes. And the corresponding time for completion for each is printed.

Question 3)  
This code is designed as a Loadable Kernel Module (LKM) for Linux systems, created to count the number of actively running processes in the system. Its purpose is to provide information about the currently executing processes on your Linux-based computer.

Navigate to the directory containing the provided source code files and the Makefile. Compile the module by executing the make command in your terminal. Once compiled, load the module into the kernel with the command insmod OSq3.ko. One can check the count of running processes by viewing the kernel logs using the dmesg command. To remove the module from the kernel, simply use the rmmod OSq3 command. The module itself operates by iterating through all the processes in the system using the for\_each\_process macro provided by the Linux kernel. For each process encountered, it checks if that process is currently in the running state by using the task\_is\_running function. The count of running processes is then printed to the kernel log, providing you with an overview of the active processes on your system.

The Makefile provided alongside the source code simplifies the compilation and cleaning processes. To compile the module, execute make, and to clean up the compiled files, use make clean.

For question 3 we referred the following links:

<https://www.geeksforgeeks.org/linux-kernel-module-programming-hello-world-program/>

<https://elixir.bootlin.com/linux/latest/C/ident/task_struct>

<https://www.cs.fsu.edu/~baker/opsys/examples/task_struct.html>

<https://stackoverflow.com/questions/48458476/processing-entries-from-task-struct-why-some-tasks-next-have-0-how-to-go-to-n>

(task->state)

<https://tldp.org/LDP/lki/lki-2.html>

(TASK\_RUNNING corresponds to 0)

<https://tuxthink.blogspot.com/2011/03/using-foreachprocess-in-proc-entry.html>

(for each process macro, iterates over all the processes in the process list)

Linux Kernel Development by Robert Love (third edition)

Pages 25 to 30

<https://stackoverflow.com/questions/61886139/why-thread-info-should-be-the-first-element-in-task-struct>

(Thread\_info)

<https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/tree/kernel/signal.c?h=v6.5.3>

(Kernel source code, task\_is\_running)....path: root/kernel/signal.c

<https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/tree/kernel?h=v6.5.3>

(signal.c header)

<https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/tree/?h=v6.5.3>

(kernel)

<https://www.kernel.org/>

(info about kernel version 6.5.3, browse)