NYCU Operation System Homework 2

IOC PhD 徐浩哲 411551005

1. Describe how you implemented the program in detail.

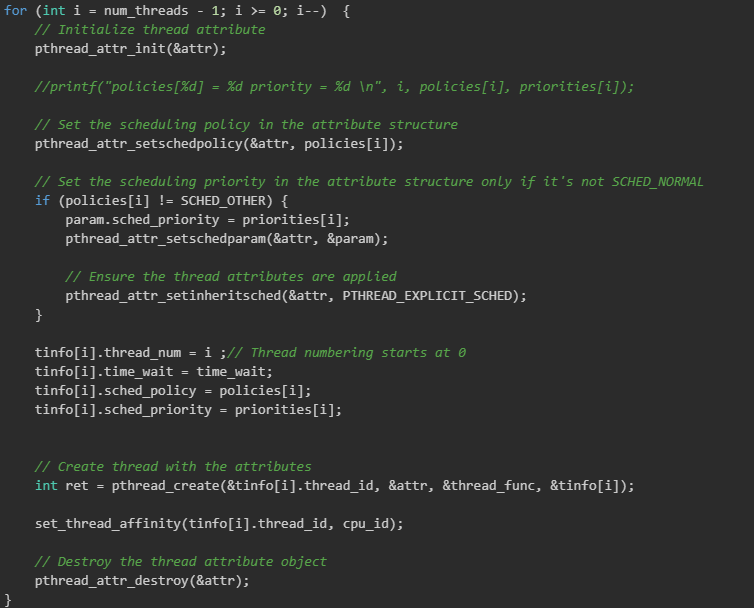
* Parsing Program Arguments

This task involves extracting and interpreting command-line arguments provided to the program, such as the number of threads, waiting time, scheduling policies, and priorities.



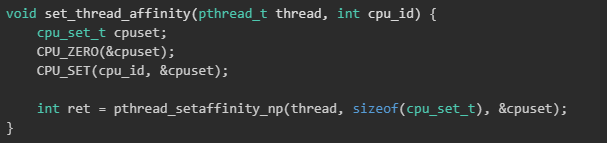
* Create worker threads

In this task, the code creates multiple threads (referred to as "worker threads") to perform concurrent tasks. The number and characteristics of these threads are determined based on the program's configuration.



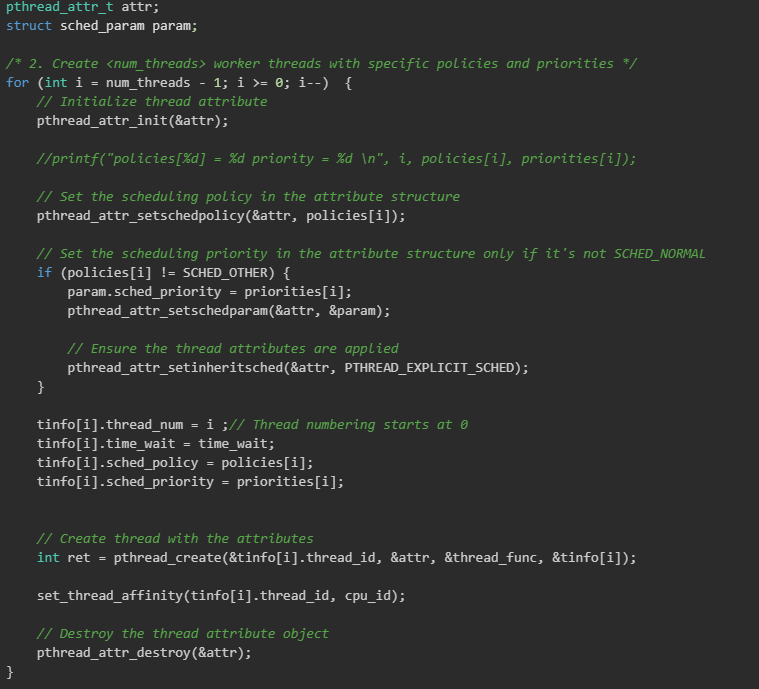
* Configuring CPU Affinity

This task involves setting the CPU affinity for each thread, specifying which CPU core each thread should execute on. This helps control the thread's execution on a specific processor.



* Assigning Attributes to Each Thread

Here, attributes like scheduling policy and priority are assigned to individual threads. These attributes define how threads are scheduled and their relative priorities.



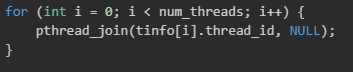
* Simultaneously Initiating All Threads

This step ensures that all threads start their execution at roughly the same time. A synchronization barrier is used to make sure that no thread proceeds until all are ready to begin.

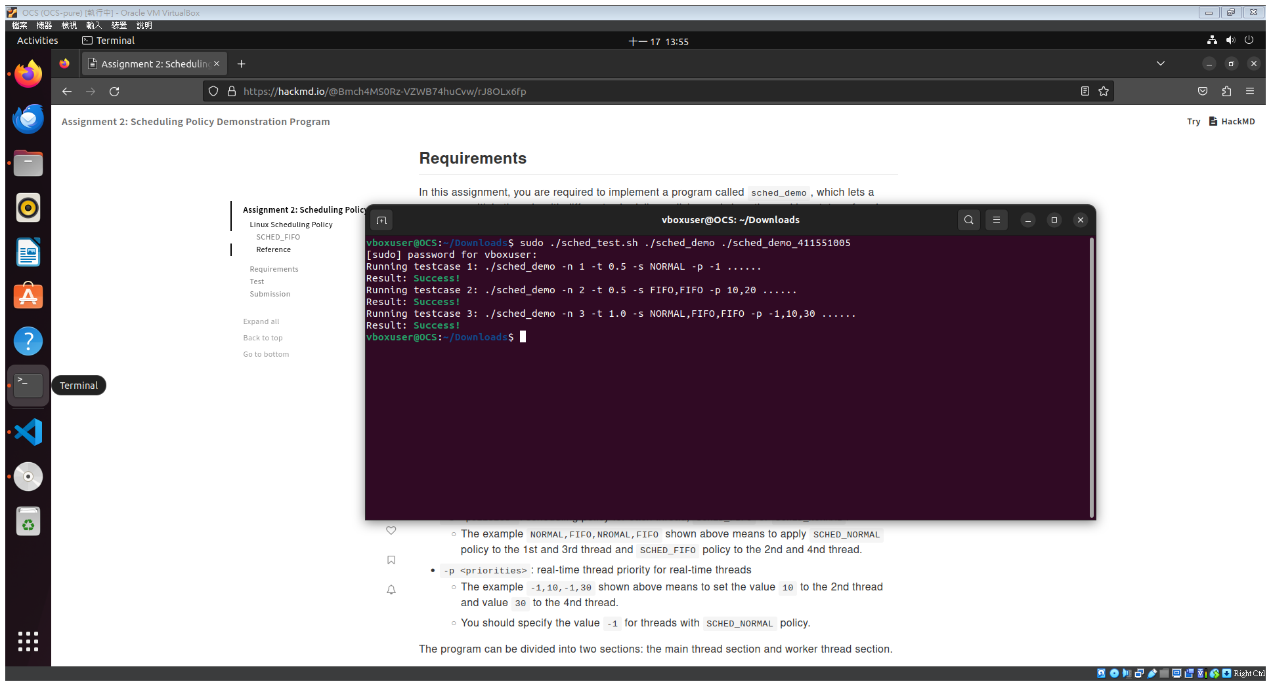


* Waiting for All Threads to Complete Execution

After starting the threads, the program waits for each thread to finish its work before proceeding further. This ensures that all threads complete their tasks before the program exits.



* Test Result



1. Describe the results of ./sched\_demo -n 3 -t 1.0 -s NORMAL,FIFO,FIFO -p -1,10,30 and what causes that.

|  |
| --- |
| Thread 2 is running  Thread 2 is running  Thread 2 is running  Thread 1 is running  Thread 1 is running  Thread 1 is running  Thread 0 is running  Thread 0 is running  Thread 0 is running |

There are three defined threads: Thread 1 and Thread 2 are configured as real-time tasks using the FIFO scheduling policy. Thread 2, having a higher priority than Thread 1, is executed before Thread 1. Meanwhile, Thread 0 remains suspended until all real-time tasks are completed.

1. Describe the results of ./sched\_demo -n 4 -t 0.5 -s NORMAL,FIFO,NORMAL,FIFO -p -1,10,-1,30, and what causes that.

|  |
| --- |
| Thread 3 is running  Thread 3 is running  Thread 3 is running  Thread 1 is running  Thread 1 is running  Thread 1 is running  Thread 2 is running  Thread 0 is running  Thread 2 is running  Thread 0 is running  Thread 2 is running  Thread 0 is running |

There are four threads designated for execution. Among them, Threads 1 and 3 are configured as real-time tasks with the FIFO scheduling policy. Thread 3, being of higher priority than Thread 1, takes precedence in execution over Thread 1. On the other hand, Threads 0 and 2 are kept in a pending state until all real-time tasks have finished their execution. These two threads operate under the SCHED\_OTHER policy, commonly referred to as the Completely Fair Scheduler (CFS). Consequently, Threads 0 and 2 alternate in executing for specified durations before yielding to other threads.

1. Describe how did you implement n-second-busy-waiting?

This busy\_wait function essentially blocks the program's execution until the desired time has passed, and it does so by repeatedly checking the current time and comparing it to the start time until the desired waiting period has elapsed. It's a simple form of a busy-wait or sleep function that can be used for short delays in a program.