
# Operating Systems – CSEN 602

# M1

## Team 91


| | |
|---|---|
| Abdallah Mahmoud | 55-23342 |
| Ali Mahmoud Shokry | 55-9658 |
| Ahmed Hawater | 55-24666 |
| Ahd Mostafa | 55-24623 |
| Youssef Elshinnawy | 55-9643 |
| Abdelrahman Mohamed Talaat | 55-23722 |

**Under Supervision of**

Dr. Eng. Catherine M. Elias

# Outline

I. **Introduction**
   - A. **Problem :** create our own OS demo
   - B. **Target :** skeleton code and test different scheduling policies

II. **Methodology**
   - A. Libraries
   - B. scheduling Algorithm
   - C. CPU cores
   - D. Threads

III. **Results**
   - A. First In First Out (FIFO)
   - B. Round Robin (RR)
   - C. Priority
   - D. Other

# I. Introduction

## The Problem

An operating system is system software that abstracts and arbitrates. An operating system acts as an intermediary between the user and the available hardware. It is also responsible for managing the available hardware. The best way for you to understand the concepts of an Operating System is to build an operating system and then to experiment with it to see how the OS manages resources and processes.

## The Target

In this project, the target is set to
Implement our **OS demo** by implementing the concepts taken in the course, such as the scheduler, memory management and synchronization

In this Milestone, we implemented a foundation for the upcoming milestones by implementing threads and experimenting with different predefined scheduling algorithms and observing how the outputs change

# II. Methodology

## Libraries

Start by including the needed libraries in our project

```
#define _GNU_SOURCE
#include <stdio.h>
//Provides functions for input and output operations, such as printf and
scanf.
#include <pthread.h>
//Defines functions, data structures, and constants for creating and
managing POSIX threads (threads in a UNIX-like operating system).
#include <sys/time.h>
//Contains definitions related to time, including structures like struct
timeval for representing time values.
#include <sys/wait.h>
//Declares functions and constants related to process management,
particularly waiting for child processes to change state.
#include <stdlib.h>
//Provides general utility functions, such as memory allocation (malloc,
free) and random number generation (rand).
#include <unistd.h>
//Defines symbolic constants and types and declares functions for various
system calls and POSIX operating system services, such as fork and exec.
#include <sched.h>
//Contains functions and definitions for scheduling policies and CPU
affinity management.
#include <time.h>
//Declares various functions for manipulating time and date information,
such as time, strftime, and clock.
```

These headers cover a wide range of functionalities, from basic input/output operations to threading, time management, process control, and system calls. They are commonly used in systems programming and multi-threaded applications on UNIX-like operating systems.

## Scheduling Algorithm

- A scheduling Algorithm is an algorithm that chooses the process (in this case the threads) that gets to execute
    - First In First Out (FIFO)
        - Non-preemptive, priority based on order of arrival
    - Round Robin (RR)
        - preemptive, time based on quanta
    - Priority
        - Non-preemptive, priority based on assigned priority and order of arrival
    - Other

- We set our scheduling policy by:
    - Initialize thread attribute
    - set its scheduling policy to the one used
    - give the scheduling priority to max to be the one used by the system in the implementation
    - Set scheduling inheritance to explicit from the parent

```
pthread_attr_init(&attr);

pthread_attr_setschedpolicy(&attr, SCHED_FIFO);
//First in First out(with or without priority)
pthread_attr_setschedpolicy(&attr, SCHED_RR); // Round Robin
pthread_attr_setschedpolicy(&attr, SCHED_OTHER);
```

## CPU cores

Set CPU affinity to one core so we limit our work only on one core to have the real effect of scheduling and cancel parallelism effect

```
   cpu_set_t cpuset;
   CPU_ZERO(&cpuset);
   CPU_SET(0, &cpuset);
pthread_attr_setaffinity_np(&attr, sizeof(cpu_set_t), &cpuset);
```

# Threads

Create **4 threads** using the pthreads library in C

```
pthread_create(&ptid1, &attr, &thread1, NULL);
```

Each thread run a function custom for it

```
void * thread1()
```

Inside it we

- Start timer to calc start, end time, response and wait time
- print statements :
    - indicating being inside this thread
    - the pid
    - current state (waiting, woke up)
    - KPIs e.g. start time, end time response time, wait time
- Each one has different delay time simulating actual task time :
    - Thread 1: 8 sec
    - Thread 2: 5 sec
    - Thread 3: 3 sec
    - Thread 4: 6 sec

# III. Results

FIFO:

```
Inside FIRST thread.
Thread Start Time: 2024-04-19
17:48:12.873546
--------------------------------
Thread 1 is waiting for 8 seconds
Thread 1 woke up
--------------------------------
First Thread ID is 140277303494208
Thread 1 is running on CPU 0
End of THREAD 1

Thread End Time: 2024-04-19 17:48:20.011082

Response Time for Thread 1: 7137536.000000
microseconds
Wait Time for Thread 1: 103.000000
microseconds

---------------------------------------
```

```
Inside SECOND thread.
Thread Start Time: 2024-04-19
17:48:20.011322
--------------------------------
Thread 2 is waiting for 5 seconds
Thread 2 woke up

--------------------------------
Second Thread ID is 140277295101504
Thread 2 is running on CPU 0
End of THREAD 2

Thread End Time: 2024-04-19 17:48:25.011075

Response Time for Thread 2: 4999753.000000
microseconds
Wait Time for Thread 2: 7137879.000000
microseconds

---------------------------------------
```

```
Inside THIRD thread.
Thread Start Time: 2024-04-19
17:48:25.011194
--------------------------------
Thread 3 is waiting for 3 seconds
Thread 3 woke up

--------------------------------
Third Thread ID is 140277286708800
Thread 3 is running on CPU 0
End of THREAD 3

Thread End Time: 2024-04-19 17:48:28.011062

Response Time for Thread 3: 2999868.000000
microseconds
Wait Time for Thread 3: 12137751.000000
microseconds

---------------------------------------
```

```
Inside FOURTH thread.
Thread Start Time: 2024-04-19
17:48:28.011201
--------------------------------
Thread 4 is waiting for 6 seconds
Thread 4 woke up

--------------------------------
Fourth Thread ID is 140277278316096
Thread 4 is running on CPU 0
End of THREAD 4

Thread End Time: 2024-04-19 17:48:34.011043

Response Time for Thread 4: 5999842.000000
microseconds
Wait Time for Thread 4: 15137758.000000
microseconds

---------------------------------------
```

```
Time for the program to execute: 21137718.000000 microseconds
Usefull work is 0.999966
```

# Round Robin (RR)

```
Inside FIRST thread.
Thread Start Time: 2024-04-19
19:04:29.127956

--------------------------------
Thread 1 is sleeping for 8 seconds

Inside SECOND thread.
Thread Start Time: 2024-04-19
19:04:29.144704

--------------------------------
Thread 2 is sleeping for 5 seconds

Inside THIRD thread.
Thread Start Time: 2024-04-19
19:04:29.154650
--------------------------------
```

```
Thread 3 is sleeping for 3 seconds

Inside FOURTH thread.
Thread Start Time: 2024-04-19
19:04:29.164644
--------------------------------
Thread 4 is sleeping for 6 seconds
Thread 3 woke up

--------------------------------
Third Thread ID is 139770672952896
Thread 3 is running on CPU 0
End of THREAD 3

Thread End Time: 2024-04-19 19:04:32.024725

Response Time for Thread 3: 2870075.000000
microseconds
Wait Time for Thread 3: 26883.000000
microseconds
```

```
Thread 2 woke up

--------------------------------
Second Thread ID is 139770681345600
Thread 2 is running on CPU 0
End of THREAD 2

Thread End Time: 2024-04-19 19:04:34.024721

Response Time for Thread 2: 4880017.000000
microseconds
Wait Time for Thread 2: 16937.000000
microseconds
--------------------------------------
Thread 4 woke up
--------------------------------
Fourth Thread ID is 139770664560192
Thread 4 is running on CPU 0
End of THREAD 4
Thread End Time: 2024-04-19 19:04:35.014719

Response Time for Thread 4: 5850075.000000
microseconds
```

```

Wait Time for Thread 4: 36877.000000
microseconds
Thread 1 woke up

--------------------------------
First Thread ID is 139770689738304
Thread 1 is running on CPU 0
End of THREAD 1

Thread End Time: 2024-04-19 19:04:37.014694

Response Time for Thread 1: 7886738.000000
microseconds
Wait Time for Thread 1: 189.000000
microseconds

---------------------------------------

Time for the program to execute:
7887019.000000 microseconds
```

# Priority

```
Inside SECOND thread.
Thread Start Time: 2024-04-19
19:14:39.785331

Second Thread ID is 140646584133184
Thread 2 is running on CPU 0
End of THREAD 2

Thread End Time: 2024-04-19 19:14:39.785434

Response Time for Thread 2: 103.000000
microseconds
Wait Time for Thread 2: 188.000000
microseconds

---------------------------------------
```

```
Inside THIRD thread.

Inside FOURTH thread.
Thread Start Time: 2024-04-19
19:14:39.785504

Fourth Thread ID is 140646567347776
Thread 4 is running on CPU 0
End of THREAD 4

Thread End Time: 2024-04-19 19:14:39.785525

Response Time for Thread 4: 21.000000
microseconds
Wait Time for Thread 4: 361.000000
microseconds
---------------------------------------
```

```
Thread Start Time: 2024-04-19
19:14:39.785476

Third Thread ID is 140646575740480
Thread 3 is running on CPU 0
End of THREAD 3

Thread End Time: 2024-04-19 19:14:39.785585

Response Time for Thread 3: 109.000000
microseconds
Wait Time for Thread 3: 333.000000
microseconds

---------------------------------------
```

```
Inside FIRST thread.
Thread Start Time: 2024-04-19
19:14:39.785234

First Thread ID is 140646592525888
Thread 1 is running on CPU 0
End of THREAD 1

Thread End Time: 2024-04-19 19:14:39.785669

Response Time for Thread 1: 435.000000
microseconds
Wait Time for Thread 1: 91.000000
microseconds

---------------------------------------
Time for the program to execute: 567.000000
microseconds
```

# Other

```
Inside FIRST thread.
Thread Start Time: 2024-04-19
19:17:22.543580

--------------------------------
Thread 1 is sleeping for 8 seconds

Inside SECOND thread.
Thread Start Time: 2024-04-19
19:17:22.564673

--------------------------------
Thread 2 is sleeping for 5 seconds
Inside THIRD thread.
Thread Start Time: 2024-04-19
19:17:22.574673
--------------------------------
Thread 3 is sleeping for 3 seconds
```

```

Inside FOURTH thread.
Thread Start Time: 2024-04-19
19:17:22.584733
--------------------------------
Thread 4 is sleeping for 6 seconds
Thread 3 woke up
--------------------------------
Third Thread ID is 139842567849536
Thread 3 is running on CPU 0
End of THREAD 3

Thread End Time: 2024-04-19 19:17:25.044675

Response Time for Thread 3: 2470002.000000
microseconds
Wait Time for Thread 3: 31210.000000
microseconds
```

```
---------------------------------------
Thread 2 woke up
--------------------------------
Second Thread ID is 139842576242240
Thread 2 is running on CPU 0
End of THREAD 2
Thread End Time: 2024-04-19 19:17:27.014675

Response Time for Thread 2: 4450002.000000
microseconds
Wait Time for Thread 2: 21210.000000
microseconds
---------------------------------------
Thread 4 woke up

--------------------------------
Fourth Thread ID is 139842559456832
Thread 4 is running on CPU 0
End of THREAD 4

Thread End Time: 2024-04-19 19:17:28.014675
```

```
Response Time for Thread 4: 5429942.000000
microseconds
Wait Time for Thread 4: 41270.000000
microseconds
---------------------------------------
Thread 1 woke up
--------------------------------
First Thread ID is 139842584634944
Thread 1 is running on CPU 0
End of THREAD 1

Thread End Time: 2024-04-19 19:17:30.014669

Response Time for Thread 1: 7471089.000000
microseconds
Wait Time for Thread 1: 117.000000
microseconds
---------------------------------------
Time for the program to execute:
7471326.000000 microseconds
```

# Output comment

All the output was as expected, suggesting a successful execution of the multi-threaded program using different policies, with each thread completing its task within expected timeframes and contributing to the overall workload efficiently.

**FIFO**

Was executed in the order of creation

**RR**

Due to time based property each thread was given a fixed quanta and go through threads in sequence, it was seen clearly in the output as thread 1 finished last as it was the longest time taken thread

**Priority**

It's seen in the second frame of the output that although thread 3 started exec but it was preempted by thread 4 for having higher priority

**Other**

The output cannot be expected as it's the OS decision to choose which thread to be executed first