The German University in Cairo (GUC)
Faculty of Media Engineering and Technology
Computer Science and Engineering
Operating Systems - CSEN 602

Threads and Scheduling in Operating Systems

Team Members :

Jomana Mahmoud, 58-1034
Yehia Hassan, 58-2751
Rawan Hossam, 58-25160
Nada Yasser, 58-3703
Aesha Anwar, 58-0464
Sara Elsawy, 58-1857
Walid Moussa, 58-1001

Team Number :
Team 64

Under Supervision of :

Dr. Eng. Catherine M. Elias and Dr. Aya Salama

Introduction

This report will analyze the use of multi-threading in C using pthreads and scheduling mechanisms the program executes 3 tasks concurrently. All 3 tasks run on the same processor (Unicore), but different threads.

It explores the behavior of threads under different scheduling algorithms, focusing on First-Come First-Served (FCFS) and Round Robin (RR).

The execution time of each thread is measured with a monotonic built-in clock (clock_gettime()) to measure the time efficiency of each scheduling mechanism, additionally CPU utilization and memory consumption are measured to evaluate the efficiency of each scheduling strategy.

Used Scheduling Mechanisms

1- Round Robin (RR) Policy

Preemptive scheduling algorithm designed for time-sharing systems. It ensures fair CPU time allocation by giving each process a fixed time slice (quantum) before moving to the next process in a cyclic order.

2- First in First out (FIFO) Policy

This scheduling mechanism relies on first in first serve policy meaning if thread A starts executing first, the CPU will keep executing thread A until it's done and then start with thread B.
However the issue with this scheduling mechanism is that it might cause starvation, if thread A keeps running for a really long time or infinitely threat B would be ready to execute but it never gets the chance to do so causing what's caused as starvation.

Scheduling and Affinity Rules

1. **CPU Affinity**:

- The threads are pinned to CPU core 0 using `CPU_SET(0, &cpuset)` to ensure deterministic scheduling behavior.

2. **Scheduling Policy**:

- `SCHED_RR` (Round Robin) is used to distribute CPU time fairly among the threads.
- `SCHED_FIFO` (First In, First Out) is used to schedule threads based on their priority, where a thread runs until it finishes or is preempted by a higher-priority thread. Lower-priority threads must wait until higher-priority ones complete.

3. **Thread Prioritization**:

- The scheduling parameters are set using `pthread_attr_setschedparam(&attr, &param)`, though priority values are not explicitly assigned.

Performance Metrics Rules

4. **Time Measurement**:

- `clock_gettime(CLOCK_MONOTONIC, &ts)` is used to measure the release, start, and finish times of each thread.

5. **Execution Time Calculation**:

- `time_diff(start, end)` function calculates execution time for each thread based on `start_times` and `finish_times`.

6. **Response Time**:

- Computed as the difference between the release time and the start time of each thread.

7. **Turnaround Time**:

- Defined as the time taken from thread release to thread completion (`finish_time - release_time`).

8. **Waiting Time**:

- Derived from turnaround time minus execution time (`waiting_time = turnaround_time - execution_time`).

9. **CPU Utilization**:

- The ratio of execution time to the sum of execution and waiting time (`execution / (execution + waiting)`).

10. **Overall CPU Utilization**:

- The average of individual CPU utilizations across all threads.

Memory Usage Measurement

11. **Memory Consumption Tracking**:

- Reads `/proc/self/status` to obtain `VmRSS` (Resident Set Size) and display memory consumption.

CPU Load Averages (from /proc/loadavg via getloadavg)

12. The function `print_cpu_load()` retrieves the system's load average using `getloadavg()`, which provides the average number of processes in the system run queue over the last **1, 5, and 15 minutes**.

Analyzing Individual Thread Behavior

Before testing different scheduling algorithms, we will first compare the execution of all threads using the **same scheduler**. This will establish a baseline for analyzing differences in execution time and resource usage before introducing scheduling variations.

In Round Robin:

```
jomana@JomanasLaptop:~/OS$ ./ProjectCode
Please enter 2 letters: Running with Thread ID: 140430302344896
This is print statement 1
This is print statement 2

a g
Please enter 2 integers: Thread 1 Output: a b c d e f g
1 9

Sum: 45
Average: 5.00
Product: 362880

Performance Metrics:

Thread 1:
Release Time: 159.466973 sec
Start Time: 159.467134 sec
Finish Time: 162.497520 sec
Execution Time: 3.030386 sec
Response Time: 0.000162 sec
Turnaround Time: 3.030548 sec
Waiting Time: 0.000162 sec
CPU useful work: 0.999947

Thread 2:
Release Time: 159.467088 sec
Start Time: 159.467259 sec
Finish Time: 159.467266 sec
Execution Time: 0.000007 sec
Response Time: 0.000171 sec
Turnaround Time: 0.000178 sec
Waiting Time: 0.000171 sec
CPU useful work: 0.040639

Thread 3:
Release Time: 159.467150 sec
Start Time: 159.467865 sec
Finish Time: 164.568139 sec
Execution Time: 5.100274 sec
Response Time: 0.000714 sec
Turnaround Time: 5.100988 sec
Waiting Time: 0.000714 sec
CPU useful work: 0.999860
CPU Average Utilization: 68.014854%
CPU Load Averages: 1min=0.05, 5min=0.08, 15min=0.03
Memory Consumption: VmRSS:          1764 kB

Main Thread: All threads finished execution.
```

	Release Time (r_i)	Start Time	Finish Time	Execution Time (e_i)	Response Time	Turnaround Time	Waiting Time	CPU Utilization
Thread 1	159.466973 sec	159.467134 sec	162.497520 sec	3.030386 sec	0.000162 sec	3.030548 sec	0.000162 sec	0.999947
Thread 2	159.467088 sec	159.467259 sec	159.467266 sec	0.000007 sec	0.000171 sec	0.000178 sec	0.000171 sec	0.040639
Thread 3	159.467150 sec	159.467865 sec	164.568139 sec	5.100274 sec	0.000714 sec	5.100988 sec	0.000714 sec	0.999860
Avg					0.00034899 sec	2.7105713 sec	0.00034899 sec	0.68014854

1. Response Time

- The response time for all three threads is very low, averaging 0.00034899 seconds. This indicates that Round Robin ensures fast initial access to the CPU for all threads.

2. Turnaround Time

- The turnaround time varies across the threads. The average turnaround time is 2.7105713 sec, meaning tasks generally complete within a reasonable time frame.
- Since RR splits execution into time slices, longer tasks (like Thread 3) take more time compared to shorter ones (Thread 2).

3. Waiting Time

- The waiting time is almost negligible, with an average of 0.00034899 sec.This suggests that the RR scheduler ensures tasks do not experience excessive delays before execution.

4. CPU Utilization

- The CPU utilization for Thread 1 and Thread 3 is close to 100% (0.999947 and 0.999860, respectively).
- However, Thread 2 has very low CPU utilization (0.040639), indicating that it either completed very quickly or did not use CPU resources effectively.

The overall CPU utilization averages 0.68014854 (68.01%), meaning the system still has some idle time, potentially due to context switching overhead .

In FIFO:

```
jomana@JomanasLaptop:~/OS$ ./ProjectCode
Please enter 2 letters: Running with Thread ID: 140682836887232
This is print statement 1
This is print statement 2

a g
Please enter 2 integers: Thread 1 Output: a b c d e f g
1 9

Sum: 45
Average: 5.00
Product: 362880

Performance Metrics:

Thread 1:
Release Time: 1532.887268 sec
Start Time: 1532.887425 sec
Finish Time: 1534.371522 sec
Execution Time: 1.484097 sec
Response Time: 0.000157 sec
Turnaround Time: 1.484254 sec
Waiting Time: 0.000157 sec
CPU useful work: 0.999894

Thread 2:
Release Time: 1532.887380 sec
Start Time: 1532.887439 sec
Finish Time: 1532.887521 sec
Execution Time: 0.000083 sec
Response Time: 0.000059 sec
Turnaround Time: 0.000141 sec
Waiting Time: 0.000058 sec
CPU useful work: 0.586510

Thread 3:
Release Time: 1532.887424 sec
Start Time: 1532.887707 sec
Finish Time: 1535.538892 sec
Execution Time: 2.651185 sec
Response Time: 0.000284 sec
Turnaround Time: 2.651468 sec
Waiting Time: 0.000284 sec
CPU useful work: 0.999893
CPU Average Utilization: 86.209900%
CPU Load Averages: 1min=0.02, 5min=0.02, 15min=0.00
Memory Consumption: VmRSS:          1884 kB
```

	Release Time (r_i)	**Start Time**	**Finish Time**	**Execution Time (e_i)**	**Response Time**	**Turnaround Time**	**Waiting Time**	**CPU Utilization**
Thread 1	1532.887 268 sec	1532.8874 25 sec	1534.371522 sec	1.484097 sec	0.000157 sec	1.484254 sec	0.000157 sec	0.999894
Thread 2	1532.887 380 sec	1532.8874 39 sec	1532.887521 sec	0.000083 sec	0.000059 sec	0.000141 sec	0.000058 sec	0.586510
Thread 3	1532.887 424 sec	1532.8877 07 sec	1535.538892 sec	2.651185 sec	0.000284 sec	2.651468 sec	0.000284 sec	0.999893
Avg					0.0001667 sec	1.3789210 sec	0.0001663 sec	0.8620990

Analysis of FIFO Scheduling Results

1. Response Time

- The response time for all three threads is very low, averaging **0.0001667 seconds**.
- This indicates that the FIFO scheduling method ensures quick initial access to the CPU, as tasks start execution as soon as the previous one completes.

2. Turnaround Time

- The turnaround time varies across the threads. The average turnaround time is 1.3786210 seconds, meaning tasks generally complete within a reasonable time frame.
- Since FIFO processes tasks in the order they arrive, longer tasks (such as Thread 3) increase the turnaround time for subsequent threads.

3. Waiting Time

- The waiting time is minimal, with an average of 0.0001663 seconds.
- This suggests that the FIFO scheduler allows tasks to begin execution with minimal delays, as long as they arrive in a balanced manner. However, tasks arriving after long-running threads may experience increased waiting times.

4. CPU Utilization

- The CPU utilization for Thread 1 (99.9894%) and Thread 3 (99.9893%) is very high, indicating that the processor is efficiently utilized when executing longer tasks.
- Thread 2 (58.651%) has significantly lower CPU utilization, likely due to its short execution time.
- The overall CPU utilization averages 86.2099%, suggesting that the system is being used effectively, but may still experience some idle time due to task variations.

Comparing Scheduling Algorithms

After analyzing individual metrics, we compared the **overall performance** of both scheduling algorithms. This comparison helped identify trade-offs between efficiency, fairness, and resource utilization.

	Response Time	Turnaround Time	Waiting Time	CPU Utilization
Round Robin	0.00034899 sec	2.7105713 sec	0.00034899 sec	0.68014854
FIFO	0.0001667 sec	1.3789210 sec	0.0001663 sec	0.8620990

Analysis of Comparing Scheduling Results

Response Time:

- FIFO has a slightly higher response time (0.000167 sec) compared to Round Robin (0.00034899 sec), indicating that RR provides quicker initial access to the CPU.

Turnaround Time:

- Both scheduling methods have similar turnaround times (~1.6 sec), meaning they complete tasks in nearly the same timeframe.

Waiting Time:

- For **Round Robin (0.00034899 sec)**: The waiting time is higher due to frequent context switching, leading to increased process wait times.
- For **FIFO**, the waiting time is **0.00016633 sec**, which is much lower than Round Robin. This indicates that FIFO minimizes waiting time since processes are executed in the order they arrive without interruption.

CPU Utilization:

- FIFO achieves better CPU utilization (86.2%) than Round Robin (68.01%), indicating fewer idle periods and better resource efficiency.

Summary:

- Round Robin provides faster response and lower waiting times, making it suitable for interactive tasks.
- FIFO achieves better CPU utilization but may lead to longer waiting times for some processes.