Lab 3 Scheduler

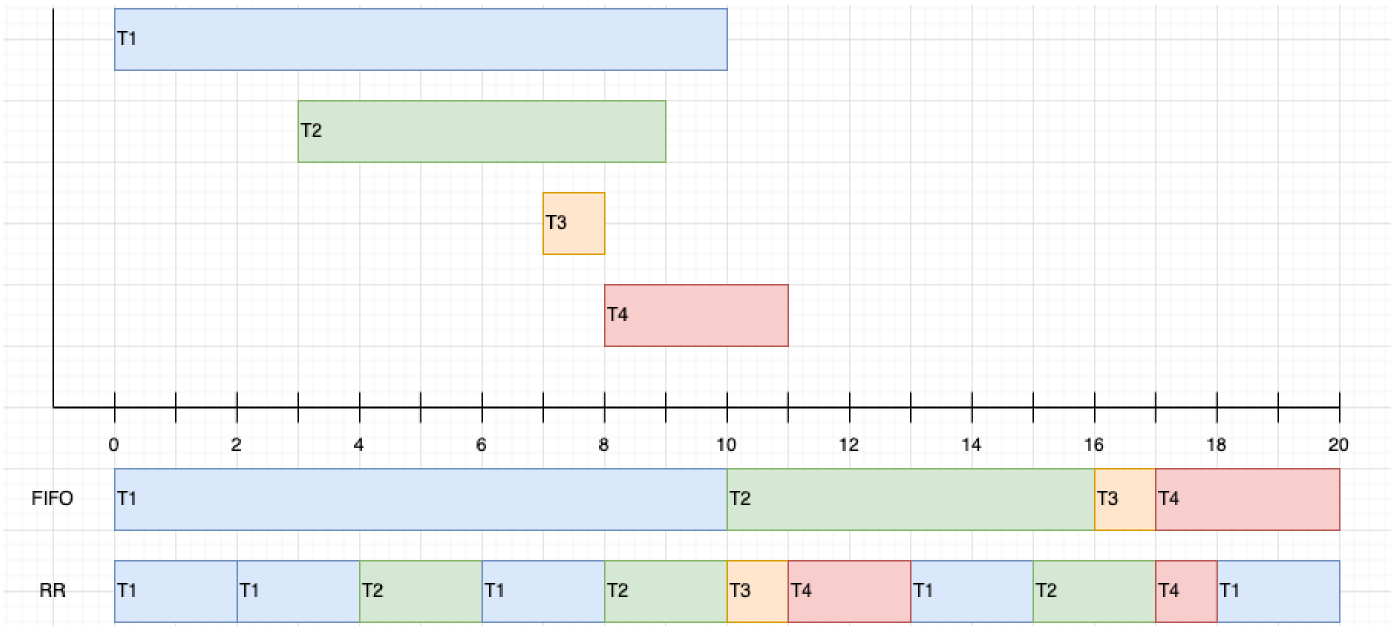
1 - Basic Theory

1.1 - Schedules

Burst Time is an expression for the runtime of a task representing the computing time and not including any wait times for I/O which may increase the actual completion time of the task.

Given the following task list, determine the FIFO and RR schedules. Assume a quantum (q) of 2.

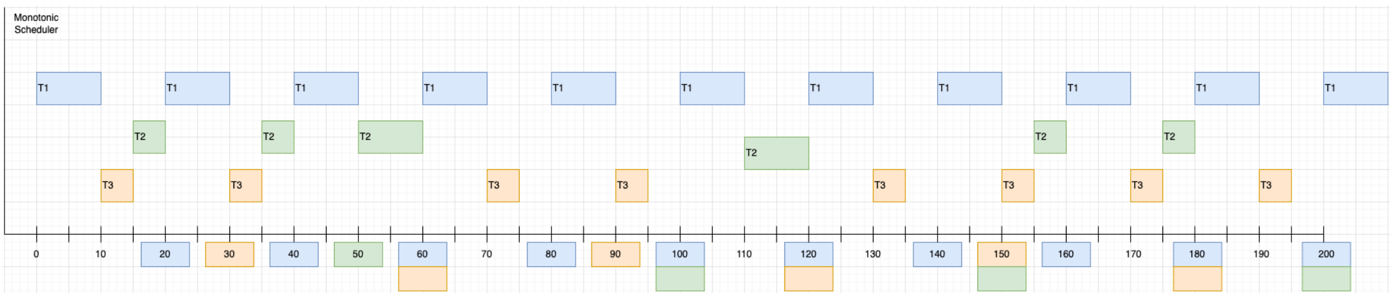
Task	Arrival Time	Burst Time
T1	0	10
T2	3	6
T3	7	1
T4	8	3



WCET stands for Worst Case Execution Time. It represents the maximum possible runtime of a piece of code or a process/thread independently of the runtime environment and any scheduling factors. WCET generally includes wait times for I/O. It is used in realtime systems and embedded systems where I/O access times tend to be deterministic - i.e completed in constant-time. I/O may include disk access times but rarely depends on I/O from a (non-deterministic) user.

Given the following task table, answer the following questions and complete the exercise for a Rate Monotonic Scheduler

Task	WCET	Period
	(C)	(T)
T1	10	20
T2	10	50
T3	5	30



a) Which task has the highest priority?

In RM scheduling, the tasks with shorter periods are assigned higher priorities and executed first. This is because tasks with shorter periods have stricter timing constraints and require more frequent execution, so they must be given higher priority to ensure that their deadlines are met. Priority: (T1 > T3 > T2) -> defined by lowest period.

b) Is there a guaranteed schedule?

If the sum of the WCETs of all tasks in the system exceeds the total available processing time within a period, then the system is not schedulable and cannot guarantee that all deadlines will be met. Also, the schedule is guaranteed if following equation is true.

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{\frac{1}{n}} - 1)$$

For the tasks T1 - T3 the sum of the fractions results in 0.86.

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} = \frac{10}{20} + \frac{10}{50} + \frac{5}{30} = 0.86$$

With the right side of the equation resulting in 0.78.

$$n(2^{\frac{1}{n}} - 1) = 3 \cdot (2^{\frac{1}{3}} - 1) = 0.78$$

$$0.86 > 0.78$$

This means that the premise is not fullfilled. Hence the schedule of the tasks is not guaranteed.

2 - Setup & Basic Tasks

Setup your virtual machine with at least 4 cores.

Check the number of CPUs and the number of online-cpus (using which command?)

```
bash lscpu
cat /proc/cpuinfo
```

Check the compiler installation (using which command?)

```
dpkg --list | grep compiler
gcc --version
```

2.1 - Process scheduling and manipulation on keyboard using nice and chrt

Inspect the code in the file processes_SCHED.c and answer the following questions

1.) Predict what it does

The code forks the program and starts a child process. Child flushes "Child" to the console, while Mother flushes "Mother" to the console.

```
top - 10:56:11 up 7 min,  2 users,  load average: 1.44, 0.53, 0.21
Tasks: 116 total,   3 running, 113 sleeping,   0 stopped,   0 zombie
%Cpu(s): 24.8 us,  0.8 sy,  0.0 ni, 74.4 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
st
MiB Mem :   7961.6 total,   7270.7 free,    165.8 used,    525.1 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.   7549.5 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+
COMMAND
   1624 ubuntu    20   0    2488    704    640 R  50.2   0.0   0:37.98
processes_SCHED
   1625 ubuntu    20   0    2488     76      0 R  49.8   0.0   0:37.97
processes_SCHED
```

6.) You she the print of "Mother" and "Child" in seemingly random fashion. What do you learn from this behaviour?

Since the priority and the nice are equal for both processes, they are equally likely to be run next. This means that the order is dependent on the process queue where both have equal possibility of running.

7.) Open a third terminal and get the PIDs of the two processes. Using the command chrt -p display the current priority and scheduler

```
ubuntu@boy-lab:~$ chrt -p 1624
pid 1624's current scheduling policy: SCHED_OTHER
pid 1624's current scheduling priority: 0
ubuntu@boy-lab:~$ chrt -p 1625
pid 1625's current scheduling policy: SCHED_OTHER
pid 1625's current scheduling priority: 0
```

8.) What is the nice value of the shell process? Look ath the manpages for nice and renice

nice - run a program with modified scheduling priority
renice - alter priority of running processes

```
PID       PPID    CMD    NI
18342   18341   -bash    0
```

9.) Change the nice of one process to -20 and that of the other to 20. What do you see now?

```
ubuntu@boy-lab:~$ sudo renice -n -20 -p 1625
1625 (process ID) old priority 0, new priority -20
ubuntu@boy-lab:~$ sudo renice -n 20 -p 1624
1624 (process ID) old priority 0, new priority 19
```

Only process with nice -20 is printing.

10.) Change the priority for the scheduler to 1 using sudo `chrt -o -p 1 <PID>` what do you get and why?

```
ubuntu@boy-lab:~$ sudo chrt -o -p 1 1625
chrt: unsupported priority value for the policy: 1: see --max for valid
range

ubuntu@boy-lab:~$ chrt --max
SCHED_OTHER min/max priority    : 0/0
SCHED_FIFO min/max priority     : 1/99
SCHED_RR min/max priority       : 1/99
SCHED_BATCH min/max priority    : 0/0
SCHED_IDLE min/max priority     : 0/0
SCHED_DEADLINE min/max priority : 0/0
```

11.) Read the manpage for chrt and set one of the processes to priority 1 scheduler SCHED_RR and check that this has been done

```
ubuntu@boy-lab:~$ sudo chrt -r -p 1 1625
ubuntu@boy-lab:~$ chrt -p 1625
pid 1848's current scheduling policy: SCHED_RR
pid 1848's current scheduling priority: 1
```

12.) What do you notice on the output?

Only the process with changed priority and scheduler is printing to output.

13.) read the RR quantum using cat /proc/sys/kernel/sched_rr_timeslice_ms what does this tell you and what does it mean?

```
ubuntu@boy-lab:~$ cat /proc/sys/kernel/sched_rr_timeslice_ms
100
```

Reads the Round Robin (RR) quantum, which is the time slice allocated to each process when the Linux kernel is using the SCHED_RR scheduling policy. The output of the command represents the time slice in milliseconds. By default, the value of the sched_rr_timeslice_ms parameter is 100 milliseconds, which means that each process scheduled under the SCHED_RR policy is given a time slice of 100 milliseconds to run on the CPU before being preempted.

14.) Change the priority and scheduler of the other process to 80 and RR what do you see now?

```
ubuntu@boy-lab:~$ sudo chrt -r -p 80 2310
PID 2310 - PR: 81; NI: 19
PID 2311 - PR: 0; NI: -20
```

15.) Change the priority and scheduler of one process to 80 and FIFO. both processes still run, why? Read the man page for sched(7).

If one changes the priority and scheduler of one process to 80 and FIFO, it will not necessarily stop other processes from running. The priority and scheduling policy determine how the operating system schedules the processes, but they do not guarantee that a process will be the only one running at any given time.

2.1 - Kernel-managed thread scheduling and manipulating on keyboard, including overriding inheritance rules

1.) Inspect the code in threads_SCHEDULE.c and run the pre-compiled code threads_SCHED.e. What are the thread priorities?

-51

2.) Read the man pages for sched() - why do the child processes/threads have the same scheduler priority

When a child process or thread is created using the fork() or pthread_create() functions, it inherits the scheduling policy and priority of its parent. This means that by default, the child process or thread will have the same scheduler priority as its parent. The reason for this is that the child process or thread is typically created to perform a similar task as the parent, and thus it makes sense for it to have the same priority as the parent. However, the priority of a child process or thread can be changed using the sched() function, if desired. This allows the child to have a different scheduling policy and priority than its parent, if needed.

3.) Using htop look at the comparative CPU/core usage what do you see?

The parent thread uses 380% of the CPU and three of the children use approximately 95% of a CPU. Two threads use roughly 45% of a CPU. Therefore, the CPU usage for the parent thread is kind of an aggregation of all CPU core usages. Two threads have to share one core because there are only 4.

4.) Change the nice value of one of the threads with a core to itself - does anything change?

No

5.) What ways have we seen to change the scheduler/priorities of a thread? What would be more elegant?

nice

This command is used to adjust the priority of a running process. By default, it increases the "nice" value of the process, which lowers its priority

```
nice -n 5 -p 1234
```

chrt

This command is used to set the scheduling policy and priority of a process or thread.

```
chrt -f -p 10 1234
```

sched_setscheduler() & sched_setparam()

These functions can be used to set the scheduling policy and priority of a thread from within a program.

6.) Modify the code so that the threads (thread 5 has been prepared) start in their required priority and scheduler. Hint - use the following calls, in that order - and the man pages