Lab BSY SCHED

Introduction & Prerequisites

This laboratory is to learn how to:

- Create some basic schedules
- Find and understand the command line setting of task scheduler
- Find and understand the command line setting of task priority
- (*Find and understand the command line setting of task affinity*)
- Understand the programming settings of task scheduler
- Understand the programming task scheduler inheritance
- Find and understand the command line setting of I/O scheduler
- (Find and understand the programming setting of I/O scheduler)

The following resources and tools are required for this laboratory session:

- A ZHAW VPN session
- Any modern web browser
- Any modern SSH client application
- OpenStack Horizon dashboard: https://ned.cloudlab.zhaw.ch
- OpenStack account details
  - See Moodle
- Username to login with SSH into VMs in ned.cloudlab.zhaw.ch OpenStack cloud from your laptops
  - **`Ubuntu`**
- Ubuntu VM with 4 cores
- Installed C compiler, and tools (gcc/make)

Time

The entire session will require 90 minutes.

Assessment

No assessment foreseen

Task 1 - Basic Theory

Subtask 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 real-time 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?

b.) Is there a guaranteed schedule?

Task 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?)

Check the compiler installation (using which command?)

Subtask 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.

2.) Open a second shell and run top or htop
3.) Let the code run (./process_SCHED.e)
4.) Does the code do what you predicted?
5.) How many threads/units of execution are being run?

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

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

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

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

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

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

12.) What do you notice on the 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?

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

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)`.

Subtask 2.2 – Kernel-managed thread scheduling and manipulating on keyboard, including overriding inheritance rules

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

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

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

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

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

Cleanup

IMPORTANT: At the end of the lab session:

- **Delete** all -unused - OpenStack VMs, volumes, security group rules that were created by your team.

Additional Documentation

OpenStack Horizon documentation can be found on the following pages:

- User Guide: https://docs.openstack.org/horizon/latest/