Assignment:Real-time Programming with `pthreads`

17th February 2021

For this assignment, you will need the template code `critical.c` and `multithread.c`. Download this from the module's KEATS page and save it to your computer. Use the following command to compile the source code:

```
gcc filename.c -o filename -lrt -lpthread
```

where `filename` is the name of the source code file. The resultant executable should be run using `sudo`, *i.e.,* use the command:

```
sudo ./filename
```

Check that the above works for the source code provided, and complete the following exercises.

POSIX support for mutexes is illustrated in program `critical.c`. Examine this code and make sure that you understand what it does.

1. Compile and run the program `critical.c`. To run this program you need to use `sudo` (*i.e.,* use the command: `sudo ./critical`). Record the output. Briefly explain what this program does and how the scheduling gives rise to the observed behaviour. Include in your report a timing diagram; be sure to show the state of each task (including the main routine) at all times.

   [5 marks]

2. Modify the code so that the mutexes are no longer "commented-out". Run the program again (using `sudo`) and record the output. Briefly explain the execution of the modified program. Draw a timing diagram of the modified program.

   [5 marks]

Next, you will need the code for program `multithread.c`. Download this from the module's KEATS page before continuing with the below.

Examine the code for program `multithread.c` and make sure that you understand what it does.

3. Run the code and record the output. To run this program you need to use sudo (*i.e.,* use the command: `sudo ./multithread`). Briefly explain what this program does and how the scheduling gives rise to the observed behaviour. Include in your report a timing diagram; be sure to show the state of each task (including the main routine) at all times.

   [5 marks]

Modify the `threadA` function so that after printing half the letters it increases the priority of `threadB` using the instructions:

```
param.sched_priority = priority_min+2;
pthread_setschedparam(threadB_id,policy,&param);
```

Make sure that you understand what these changes do.

4. Run the modified program (using sudo) and record the output. Draw a timing diagram showing the state of each task (including the main routine) at all times. Explain the scheduling of the threads. Modify the code to decrease the priority of `threadB` after `threadA` has printed half its letters. Report what happens and comment on this result. Modify the priority of only the executing thread and report what happens when the priority is increased and when it is decreased. Comment on and explain these results.

   [5 marks]

5. Modify the original program so that `threadA` sleeps for 1 millisecond after printing half its letters (use the `nanosleep` command). Run the code (using sudo) and record the output. Draw a timing diagram and explain the scheduling of the threads. Include a print-out of your program code.

   [5 marks]

6. Change the scheduling policy used by `multithread.c` to be `SCHED_RR` rather than `SCHED_FIFO`. Perform experiments to determine how these scheduling policies differ. Describe the experiments that you performed and summarise the results. Describe in your report how both scheduling policies determine the scheduling of tasks both when threads have equal priority and when when threads have unequal priorities. Include a print-out of your program code.

   [5 marks]

Completed assignments should be submitted to KEATS on 5pm, 9th March 2021.

Your report must be **no longer than four A4-pages**. It should be submitted as a single PDF with a minimum font size of 11pt, and margins of no less than 2cm.

This assignment is worth 15% of the module mark.