DALHOUSIE UNIVERSITY
FACULTY OF COMPUTER SCIENCE

CSCI 3120 Operating Systems

Assignment 4: CPU Scheduling

Due: 16:30, Nov. 5, 2021

- **Teaching Assistants:**
  - Patricia Kibenge-MacLeod (p.kibenge@dal.ca)
  - Hui Huang (huihuang@dal.ca)
  - Kamran Awaisi (km521977@dal.ca)
  - Yitong Zhou (yt760204@dal.ca)
  - Felix Fei (felix.fei@dal.ca)
- **Help Hours via the Channel "Office Hour - TAs" on MS Teams:**
  - Monday: 11:00am-11:59am, Kamran Awaisi
  - Tuesdays: 11:00am-11:59am, Yitong Zhou
  - Wednesday: 11:00am-11:59am, Patricia Kibenge-MacLeod
  - Thursday: 11:00am-11:59am, Hui Huang
  - Friday: 11:00am-11:59am, Felix Fei

---

1. Assignment Overview

In this assignment, you need to design and implement a C program that simulates a CPU scheduler. This scheduler is capable of using varied scheduling algorithms to schedule a group of computation tasks.

2. Important Note

There is a zero-tolerance policy on academic offenses such as plagiarism or inappropriate collaboration. By submitting your solution for this assignment, you acknowledge that the code submitted is your own work. You also agree that your code may be submitted to a plagiarism detection software (such as MOSS) that may have servers located outside Canada unless you have notified me otherwise, in writing, before the submission deadline. Any suspected act of plagiarism will be reported to the Faculty's Academic Integrity Officer in accordance with Dalhousie University's regulations regarding Academic Integrity. Please note that:
1) The assignments are individual assignments. You can discuss the problems with your friends/classmates, but you need to write your program by yourself. There should not be much similarity in terms of coding.
2) When you refer to some online resources to complete your program, you need to understand the mechanism, then write your own code. In addition, you should cite the sources via comments in your program.

3. Detailed Requirements

1) Overview: In this assignment, you need to design and implement a C program that simulates a CPU scheduler. This scheduler is capable of using the following scheduling algorithms to schedule a group of computation tasks:

- First-Come First-Served (FCFS): FCFS schedules tasks in the order in which they arrive at the ready queue.
- Round-Robin (RR): Computation tasks are served in a round-robin fashion. When there are multiple tasks to be served, each task is executed for a time quantum, then the next task in the queue will be executed. When there is only one task to be served in the computer system, this single task will keep using CPU.
- Non-preemptive Shortest-Job-First (NSJF): NSJF schedules tasks in order of the length of the tasks' next CPU burst. No task could be preempted.
- Preemptive Shortest-Job-First (PSJF): NSJF schedules tasks in order of the length of the tasks' next CPU burst. When a new task arrives, if the next CPU burst of the new task is shorter than what is left of the currently-executing task, PSJF will preempt the currently-executing task.

2) Task Specification (i.e. Input File): The details of the tasks to be scheduled are included in a file named "TaskSpec.txt". A sample "TaskSpec.txt" is provided in this assignment. However, the TA will use a variety of different scenarios to test your program.

Each row of "TaskSpec.txt" includes the information of one task. Each row follows the format "[task name],[arrival time],[burst time]". The tasks in "TaskSpec.txt" are listed according to their arrival time. Namely, the task that arrives first is listed first, then the one that arrives next is listed, etc. Note that multiple tasks could arrive at the same time. In this scenario, the tasks that arrive at the same time are added to the ready queue according to their order in "TaskSpec.txt". Namely, the task appears first is executed first.

Here is the content of the sample "TaskSpec.txt":

```
T1,0,8
T2,1,4
T3,2,9
T4,3,5
```

Actually, the tasks in the sample "TaskSpec.txt" are the same ones used to illustrate PSJF in the lecture notes. You can use the lecture notes to understand how these tasks should be scheduled by PSJF.

Note that you can use fgets() to retrieve the information from "TaskSpec.txt". Here is a tutorial about fgets(): https://www.tutorialspoint.com/cprogramming/c_file_io.htm

3) Scheduling Results (i.e. Output File): Once your program is executed, your program needs to retrieve the details of the tasks to be scheduled from "TaskSpec.txt". Thereafter, your program schedules these tasks according to FCFS, RR, NSJF and PSJF respectively. The scheduling results are stored in a file named "Output.txt" in the order of FCFS, RR, NSJF and

PSJF. Namely, the scheduling result of FCFS should be the first component in "Output.txt", which is followed by the scheduling result of RR, NSJF, and PSJF.

For each of the scheduling algorithms, the scheduling result is composed of four parts:

a) Algorithm Name: It should be FCFS or RR or NSJF or PSJF.
b) Execution Sequence: This part includes a number of rows to indicate how the tasks are executed. Each row consists of three field: task name, starting time, and ending time. These fields are separated using tabs.
c) Waiting Time of Each Task: The waiting time of each task should be listed. And the waiting times should be listed according to the arrival time of the tasks.
d) Average Waiting Time: The resulting average waiting time should be placed in a separate line.

For example, here is the scheduling result of PSJF when the sample "TaskSpec.txt" is supplied to your program:

```
PSJF:
T1      0       1
T2      1       5
T4      5       10
T1      10      17
T3      17      26
Waiting Time T1: 9
Waiting Time T2: 0
Waiting Time T3: 15
Waiting Time T4: 2
Average Waiting Time: 6.50
```

Furthermore, there should be an empty line between the scheduling results of the algorithms under investigation.

4) Additional Requirements:

a) "TaskSpec.txt" and "Output.txt" are in the same directory as your program.
b) There are different approaches to get the input from a file. One of the approaches is based on fgets(). Here are two useful links about fgets():
   https://www.tutorialspoint.com/cprogramming/c_file_io.htm
   https://man7.org/linux/man-pages/man3/fgets.3p.html
c) You can assume that there are at least 2 rows and there are at most 50 rows in "TaskSpec.txt". Namely, at least 2 tasks need to be scheduled and at most 50 tasks need to be scheduled.
d) You can assume that the tasks in "TaskSpec.txt" are ordered according to their arrival time. Namely, the task that arrives first appears first in "TaskSpec.txt".
    a. Note that multiple tasks could arrive at the same time. In this scenario, the tasks that arrive at the same time are added to the ready queue according to their order in "TaskSpec.txt".
e) You can assume that the arrival time and burst time in "TaskSpec.txt" are always integers.

f) For simplicity:
   a. Starting time, ending time, and waiting time should be listed as integers.
   b. The average waiting time is always rounded down to the nearest hundredth. In addition, we always keep two digits after the decimal point (even these two digits are zeros).
      i. For example, rounding 6.509 down to the nearest hundredth would lead to 6.50 and rounding 6.501 to the nearest hundredth would lead to 6.50 too. And if the average waiting time is 5, is should be listed as 5.00 in the "Output.txt". There are a few examples on the following webpage, illustrating how to format the result properly:
      https://www.geeksforgeeks.org/g-fact-41-setting-decimal-precision-in-c/
g) The time unit in this assignment is millisecond.
h) For Round-Robin:
   a. The time quantum is fixed at 4 milliseconds.
   b. If a task continues to use the CPU for 2 time quanta (or more than 2 time quanta) because it is the only task in the system during the corresponding period, each of these time quanta should correspond to one row in "Output.txt".
   c. If a new task arrives at the moment when an existing task's time quantum expires, and both of them need to be placed in the ready queue, the new task should be ahead of this existing task in the ready queue (if there are only two tasks in the system at this moment, the new task will start to use the CPU and the existing task will be put in the ready queue).
i) A document named "Sample-Input-Output.pdf" is provided to illustrate what "Output.txt" should include for the sample "TaskSpec.txt". A detailed analysis is also included in this file.
j) Compiling and running your program on timberlea.cs.dal.ca should not lead to errors or warnings. To compile and run your program on timberlea, you need to be able to access the command-line interface of timerlea. In addition, you need to be able to upload a file to or download a file from timberlea.
   a. To access command-line interface of timerlea, you can use the software tool "putty" on MS Windows computers. "putty" can be downloaded here: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html . On Mac and Linux computers, you can use the command "ssh" to access timberlea via the program called "Terminal".
   b. To transfer files between your computer and timberlea, several different methods could be used. Here are two methods for MS Windows and macOS/Linux computers.
      i. MS Windows Computer: WinSCP is popular tool used to transfer files between two computers. You can download WinSCP from the following webpage: https://winscp.net/eng/download.php . The documentation for WinSCP can be found here: https://winscp.net/eng/docs/start. Specifically, you can focus on the "Uploading Files" and "Downloading Files" section of this document to understand how to transfer files.

ii. Mac and Linux Computer: On Mac and Linux computers, you can use the command "scp" to transfer files. Here is a tutorial on the command "scp": https://www.linuxtechi.com/scp-command-examples-in-linux/.

5) Readme File: You need to complete a readme file named "Readme.txt", which includes the instructions that the TA could use to compile and execute your program on timberlea.

6) Submission: Please pay attention to the following submission requirements:
   a) You should place "Readme.txt" in the directory where your program files are located.
   b) Your program files (not including TaskSpec.txt and Output.txt) and "Readme.txt" should be compressed into a zip file named "YourFirstName-YourLastName-ASN4.zip". For example, my zip file should be called "Qiang-Ye-ASN4.zip".
   c) Finally, you need to submit your zip file for this assignment via brightspace.

Note that there is an appendix at the end of this document, which includes the commands that you can use to compress your files on timberlea.

4. Grading Criteria

The TA will use your submitted zip file to evaluate your assignment. The full grade is 20 points. The details of the grading criteria are presented as follows.

- "Readme.txt" with the correct compilation/execution instructions is provided [1 Point]
- FCFS correctly schedules the tasks in "TaskSpec.txt"
  - Task execution sequence is correct [2 Points]
  - Waiting Time and Average waiting time are correct [1 Point]
- RR correctly schedules the tasks in "TaskSpec.txt"
  - Task execution sequence is correct [3.5 Points]
  - Waiting Time and Average waiting time are correct [1.5 Point]
- NSFJ correctly schedules the tasks in "TaskSpec.txt"
  - Task execution sequence is correct [3.5 Points]
  - Waiting Time and Average waiting time are correct [1.5 Point]
- PSFJ correctly schedules the tasks in "TaskSpec.txt"
  - Task execution sequence is correct [3.5 Points]
  - Waiting Time and Average waiting time are correct [1.5 Point]
- Proper programming format/style (e.g. release the memory that is dynamically allocated when it is not needed any more; proper comments; proper indentation/variable names, etc.). [1 Point]

Please note that when "Readme.txt" is not provided or "Readme.txt" does not include the compilation/execution instructions, your submission will be compiled using the standard command `gcc -o A4 A4.c (or your filename)`, and you will receive a zero grade if your program cannot be successfully compiled on timberlea.

5. **Academic Integrity**

At Dalhousie University, we respect the values of academic integrity: honesty, trust, fairness, responsibility and respect. As a student, adherence to the values of academic integrity and related policies is a requirement of being part of the academic community at Dalhousie University.

1) What does academic integrity mean?

Academic integrity means being honest in the fulfillment of your academic responsibilities thus establishing mutual trust. Fairness is essential to the interactions of the academic community and is achieved through respect for the opinions and ideas of others. Violations of intellectual honesty are offensive to the entire academic community, not just to the individual faculty member and students in whose class an offence occur (See Intellectual Honesty section of University Calendar).

2) How can you achieve academic integrity?

- Make sure you understand Dalhousie's policies on academic integrity.
- Give appropriate credit to the sources used in your assignment such as written or oral work, computer codes/programs, artistic or architectural works, scientific projects, performances, web page designs, graphical representations, diagrams, videos, and images. Use RefWorks to keep track of your research and edit and format bibliographies in the citation style required by the instructor. (See http://www.library.dal.ca/How/RefWorks)
- Do not download the work of another from the Internet and submit it as your own.
- Do not submit work that has been completed through collaboration or previously submitted for another assignment without permission from your instructor.
- Do not write an examination or test for someone else.
- Do not falsify data or lab results.

These examples should be considered only as a guide and not an exhaustive list.

3) What will happen if an allegation of an academic offence is made against you?

I am required to report a suspected offence. The full process is outlined in the Discipline flow chart, which can be found at:
http://academicintegrity.dal.ca/Files/AcademicDisciplineProcess.pdf and includes the following:
a. Each Faculty has an Academic Integrity Officer (AIO) who receives allegations from instructors.
b. The AIO decides whether to proceed with the allegation and you will be notified of the process.
c. If the case proceeds, you will receive an INC (incomplete) grade until the matter is resolved.
d. If you are found guilty of an academic offence, a penalty will be assigned ranging from a warning to a suspension or expulsion from the University and can include a notation on your transcript, failure of the assignment or failure of the course. All penalties are academic in nature.

4) Where can you turn for help?

- If you are ever unsure about ANYTHING, contact myself.
- The Academic Integrity website (http://academicintegrity.dal.ca) has links to policies, definitions, online tutorials, tips on citing and paraphrasing.
- The Writing Center provides assistance with proofreading, writing styles, citations.
- Dalhousie Libraries have workshops, online tutorials, citation guides, Assignment Calculator, RefWorks, etc.
- The Dalhousie Student Advocacy Service assists students with academic appeals and student discipline procedures.
- The Senate Office provides links to a list of Academic Integrity Officers, discipline flow chart, and Senate Discipline Committee.

Appendix: How to Use Zip and Unzip on Timberlea

To compress:

```
zip squash.zip file1 file2 file3
```

To uncompress:

```
unzip squash.zip
```

this unzips it in your current working directory.