DALHOUSIE UNIVERSITY
FACULTY OF COMPUTER SCIENCE

CSCI 3120 Operating Systems

Assignment 3: Multithreaded Sorting

Due: 16:00, Nov. 3, 2023

- **Teaching Assistants:**
  - Kamran Awaisi (km521977@dal.ca)
  - Hui Huang (huihuang@dal.ca)
  - Arka Ghosh (arka.ghosh@dal.ca)
  - Yitong Zhou (yt760204@dal.ca)
  - Shiyun Wang (sh776410@dal.ca)
- **Help Hours: FCS Learning Center (Goldberg 233) or via the Channel "Office Hour - TAs" on MS Teams:**
  - Monday: 1:00pm-2:00pm, Kamran Awaisi
  - Tuesday: 1:00pm-2:00pm, Hui Huang
  - Wednesday: 1:00pm-2:00pm, Arka Ghosh
  - Thursday: 1:00pm-2:00pm, Yitong Zhou
  - Friday: 1:00pm-2:00pm, Shiyun Wang

---

1. Assignment Overview

In this assignment, you need to design and implement a C program that utilizes multithreading to accomplish the task of integer sorting.

2. Important Note

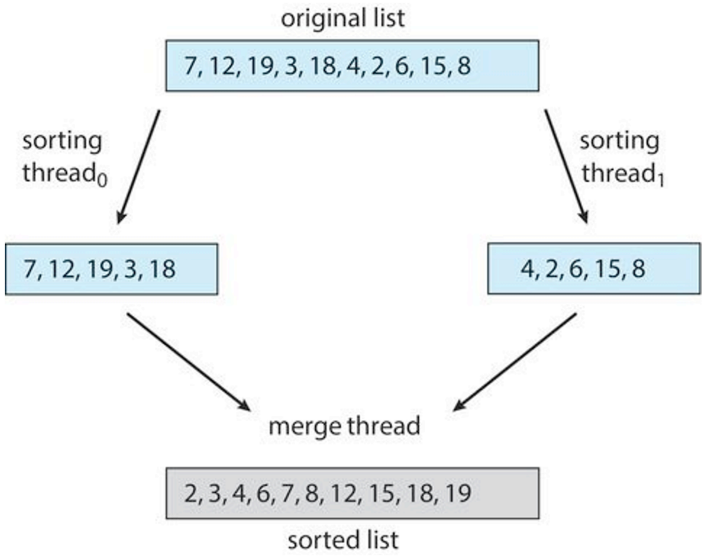
There is a zero-tolerance policy on academic offenses such as plagiarism or inappropriate collaboration. By submitting your solution for this assignment, you acknowledge that the code submitted is your own work. You also agree that your code may be submitted to a plagiarism detection software (such as 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 between your program and others' programs.
2) When you refer to some online resources to complete your program, you need to understand the mechanism, then write your own code. Your code should not be similar to the online resources. In addition, you should cite the sources via comments in your program.

3) You may use AI-driven tools to assist your learning, but you are not allowed to use them to produce work to be submitted for course evaluations. Your submitted program should not be highly similar to others' programs.

3. Detailed Requirements

1) Overview: In this assignment, you need to design and implement a C program that utilizes multithreading to accomplish the task of integer sorting. Specifically, a list of integers should be divided into two sub-lists of integers by your program. The size of the sub-lists should be the same (note that when the total number of integers is odd, the integer in the middle of the list should only be included one of the sub-lists). Then two separate threads (which are called "sorting threads") are created. Each of them sorts a sub-list using a sorting algorithm of your choice. Finally, the sorted sub-lists are merged into a single sorted list by a third thread (which is called "merging thread").

Because global data are shared across all threads, perhaps the easiest way to store the initial list of integers is to create a global array. Each sorting thread could work on one half of this array. A second global array (whose size is the same as that of the unsorted integer array) could also be established. The merging thread could then merge the sorted sub-lists into this second array. Graphically, the operation flow of your C program could be illustrated using the following figure:



2) Parameter Passing: As mentioned previously, the parent thread in your C program needs to create two worker threads (i.e. the sorting threads) to complete the sorting operation. When the worker threads are created, the parent thread needs to pass two unsorted sub-lists of integers to the worker threads. If you choose to use a global array to store the initial list of integers, then the parent only needs to pass the index number of the starting/ending element of the sub-lists (instead of the sub-lists themselves) to the worker threads. To pass the index

numbers, the easiest approach is to use "struct". For example, a struct to pass the starting and ending index number can be found below:

```
/* structure for passing data to threads */
typedef struct
{
  int starting_index;
  int ending_index;
} parameters;
```

Then the parent thread could create worker threads using a strategy similar to the following approach (assuming that N1 and N2 are the starting and ending index number; tid is the variable corresponding to thread identification; sorter() is the function for the worker threads):

```
/* create worker threads */
parameters *data = (parameters *) malloc(sizeof(parameters));
data->starting_index= N1;
data->ending_index = N2;
pthread_create(&tid, NULL, sorter, data);
```

Note that, for the merging thread, the required parameters can be passed in a similar manner.

3) Program Structure: With the background knowledge mentioned above, the overall structure of your program should look like the following skeleton:

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

void *sorter(void *params); /* thread that performs basic sorting algorithm */
void *merger(void *params); /* thread that performs merging of results */

int main (int argc, const char * argv[])
{
    /* create the first sorting thread */
    /* create the second sorting thread */
    /* now wait for the 2 sorting threads to finish */
    /* create the merge thread */
    /* wait for the merge thread to finish */
    /* output the sorted array */
}

void *sorter(void *params)
{
    /* sorting algorithm implementation */
}

void *merger(void *params)
{
    /* merging algorithm implementation */
}
```

4) Additional Requirements: Your program should also satisfy the following requirements:

a) Sorting Algorithm: You can use any sorting algorithm to sort a sub-list of integers. However, in order to make marking easy, you need to add a comment at the beginning of sorter(), which clearly states the name of the sorting algorithm used in your program. Here is a list of potential sorting algorithms:
https://en.wikipedia.org/wiki/Sorting_algorithm
Note that:
   a. You cannot reuse your previous code to implement the sorting algorithm.
   b. The GNU C library provides a sorting function, qsort(). You can use this function to sort integers if you prefer. If you choose to use this function, in the comment at the beginning of sorter(), please state "The sorting function qsort() in glibc is used to sort integers". Here are two webpages on qsort():
https://www.tutorialspoint.com/c_standard_library/c_function_qsort.htm
https://man7.org/linux/man-pages/man3/qsort.3.html

b) Sorting Result: Your program should sort the initial list of integers in ascending order (i.e. from the smallest integer to the largest integer).

c) Merging Algorithm: You should implement an efficient method to merge two sorted sub-lists. The following webpage (specifically, the section titled "Merging two lists") includes the pseudo code of an example method. You can use this method or a different method. However, you SHOULD NOT merge the sorted sub-lists by combining the sub-lists of integers into a set of integers and thereafter applying a sorting algorithm to the set of integers (because creating multiple sorting threads is designed to use parallel processing to speed up sorting).
https://en.wikipedia.org/wiki/Merge_algorithm#Merging_two_lists

d) Input and Output Files:
   a. Input file: The initial list of integers are provided via a file named "IntegerList.txt".
      i. In this file, the integers are separated by commas.
      ii. There is NO new line character (i.e. \n) at the end of the list.
      iii. The integers in the list are positive. The range of the integers is [1, 999].
      iv. There are at least 2 integers in the list. And there are at most 500 integers in the list. Note that each integer has a unique value.
      v. Your program should assume that this file is located in the same directory as your program.
      vi. An example "IntegerList.txt" is available via brightspace.
   b. Output file: The sorted list of integers should be placed in a newly-created file named "SortedIntegerList.txt". This file should be located in the same directory as your program. The integers in this file should be separated by commas.
   c. There are different methods to scan a file and get the integers from it. One of the methods consists of the following steps:
      i. Use fgets() to read the content of the file and store the content using a string. 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
      ii. Use strtok() to separate the string into sub-strings, each sub-string corresponds to an integer. Here is a tutorial:

http://www.cplusplus.com/reference/cstring/strtok/

iii. Use atoi() to convert each sub-string into an integer.

e) Total Number of Integers: When the total number of integers is even, each of the sub-lists includes half of the integers. When the total number of integers is odd, the integer in the middle of the list should only be included in one of the sub-lists.

f) Total Number of Threads: For simplicity, there should be four threads in your program in total. These threads include the parent thread (i.e. the thread corresponding to main()), two sorting threads, and one merging thread.

g) Your program must use the GNU C library (i.e. glibc) to implement the required features. Here are a few links about glibc:
   a. https://www.gnu.org/software/libc/documentation.html
   b. https://www.kernel.org/doc/man-pages/
   c. http://man7.org/linux/man-pages/index.html

h) 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 timberlea. In addition, you need to be able to upload a file to or download a file from timberlea.
   a. To access the command-line interface of "csci3120.cs.dal.ca", several different methods could be used. Here are two widely used methods:
      i. MS Windows Computer: You can use the software tool "putty" on MS Windows computers. With "putty", the connection type should be set to "SSH". Note that "putty" can be downloaded via the following link: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html .
      ii. Mac and Linux Computer: On Mac and Linux computers, you can use the command "ssh" to access "csci3120.cs.dal.ca" via the program called "Terminal".
   b. To transfer files between your computer and "csci3120.cs.dal.ca", several different methods could be used. Here are two widely used methods:
      i. MS Windows and Mac Computer: Cyberduck is popular tool used to transfer files between two computers via SFTP (i.e. SSH File Transfer Protocol). You can download Cyberduck from the following webpage: https://cyberduck.io/ . This document includes an appendix that provides an introduction to Cyberduck.
      ii. Linux Computer: On 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/.
   c. You should upload your program to "csci3120.cs.dal.ca" to verify that it works well on "csci3120.cs.dal.ca". After verifying that your program works on "csci3120.cs.dal.ca". You should submit your assignment in brightpace. For submission details, please proceed to the submission section of this document. Note that the TA will NOT look at your files on "csci3120.cs.dal.ca". Instead, the TA will retrieve your submission from brightspace.

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 and "Readme.txt" should be compressed into a zip file named "YourFirstName-YourLastName-ASN3.zip". For example, my zip file should be called "Qiang-Ye-ASN3.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 16 points. Your submission will be evaluated from the following perspectives. The specific rubric used by the marker is included in a separate file.

a) "Readme.txt" with the correct compilation/execution instructions is provided [1 Point]
b) main():
   a. Retrieve the initial list of integers from "IntegerList.txt" [2 Points]
   b. Create the first sorting thread [1 Point]
   c. Create the second sorting thread [1 Point]
   d. Wait for sorting threads to finish [1 Point]
   e. Create the merge thread [1 Point]
   f. Wait for the merge thread to finish [1 Point]
   g. Output the sorted array: The program creates a new file named "SortedIntegerList.txt", which includes the sorted list of integers and is located in the same directory as your program. The integers are separated using commas. [1 Point]
c) sorter(): It sorts the integers that are passed to the function. There is a comment at the beginning of sorter(), which clearly indicates the name of the sorting algorithm implemented in your program. [3 Points]
d) merger(): It merges two lists of sorted integers into a sorted list. [3 Points]
e) 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 A3 A3.c (or your filename) -lpthread`, 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 1: How to Use Zip and Unzip on "csci3120.cs.dal.ca"

To compress:

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

To uncompress:

```
unzip squash.zip
```

this unzips it in your current working directory.

Appendix 2: How to View/Kill Your Processes on "csci3120.cs.dal.ca"

Due to programming mistakes, you might leave a number of running processes on "csci3120.cs.dal.ca". When you leave too many running processes "csci3120.cs.dal.ca", your CS ID could be locked temporarily (then you need to talk to CS Help to unlock your account). To view/kill your processes on "csci3120.cs.dal.ca", you can use the following commands. Please keep an eye on your processes on "csci3120.cs.dal.ca".

1) Command to display the processes belonging to a specific ID (i.e. UID): **ps -u UID**
2) Command to kill a process using process ID (i.e. PID): **kill -9 PID**

3) Command to kill all processed belonging to a specific user ID (i.e. UID): **pkill -u UID**

Appendix 3: Introduction to Cyberduck

1. Optional Registration: Cyberduck is free software that supports file transfers via SSH File Transfer Protocol (i.e. SFTP). You can choose to pay for the registration to avoid seeing the donate/buy window when you quit the program. But it is perfectly ok to use Cyberduck without the registration. The only drawback is that you will see the donate/buy window when it is terminated (in this case, you can click on “Later” to completely terminate the program).
2. Set Up Connection: SFTP (SSH File Transfer Protocol) is used to set up the connection
    a. Start Cyberduck
    b. Click on the icon “Open Connection”
    c. Set Protocol to “SFTP (SSH File Transfer Protocol)”
    d. Set Server to “csci3120.cs.dal.ca”
    e. Enter your CSID and password
    f. Check “Add to keychain” on macOS or check “Save Password” on MS Windows
    g. Click on “Connect”
3. Upload/Download Files: “drag-and-drop" is supported
    a. Uploading: select a file on your computer and thereafter drag the file into the Cyberduck interface
    b. Downloading: select a file in the Cyberduck interface and thereafter drag the file to the proper destination