CSCI 4061: Introduction to Operating System

Fall 2017 (Instructor: Anand Tripathi)

Assignment 4: Concurrent Programming with POSIX Threads

Due: November 27, 2017

This assignment can be done either individually or in a team of up to two students.

Instruction: This assignment contains 2 parts. Solution of part 1 must be saved in a PDF file and put in the folder named 'Theory'. Solution of part 2 must be put in the folder 'Project'. Create another file named teaminfo.txt in the same directory containing the name and student IDs of the team members. These 2 folders must be zipped (.zip or .tar or .tar.gz) in a single file and submitted using the Homework Submission link on Moodle.

PART - 1 : Theory (20 points)

Question 1: (5 points): Consider a system with 5 jobs in the ready-queue with service-time requirements as shown with their order in the queue as (20, 10, 5, 15, 30). The job with the service time requirement of 20 is at the head of the queue. What will be the average waiting-time and average turnaround time when FCFS scheduling is used?

Question 2: (5 points): For the above problem, what will the values for these two measures when round-robin scheduling with service quantum value of 5 is used?

Question 3: (5 points): For the same problem, now consider the Shortest Job First scheduling and determine the values for these two measures.

Question 4: (5 points): In a system, five jobs are waiting for execution. Their required service times are (9, 6, 3, 5, and X). In what order should they be executed to minimize the average turnaround time? Your answer will depend on the value of X.

PART – 2 : Programming Project (100 points)

1. General information

For this assignment, please study the example programs on course webpage under “Examples for POSIX Thread Programming”. It has several helpful examples.

2. Objectives

The objective of this assignment is to learn the basic concepts in how to program multi-threaded applications using the POSIX pthread library.

3. Group size

This assignment is to be completed either individually or in a group of two.

4. Assignment statement

In this assignment you are required to write a multi-threaded program in C using the pthread library. You will write a program that will sum the size of all regular files in a directory by recursively visiting all nested directories. The program will ask the user to specify a directory pathname and then it will compute the sum of the size of all the regular files inside that directory and its subdirectories.

You will write the program in a very specific way. For each directory, a thread will only look at the files that are in that directory. If there are any subdirectories, your program needs to create a new thread for each of the subdirectories. For example if the user specified directory “dir” has two files “a.txt” and “b.txt” and three subdirectories “dir1”, “dir2” and “dir3”, then one thread will handle the task of getting the sizes of the two files and three more threads will need to be created to get the sizes of each of the three subdirectories. The same logic applies at the lower level of the hierarchy. For instance, if “dir3” has more subdirectories, then the thread responsible for “dir3” will create new threads to handle the counts for each of the subdirectories of “dir3”.

The thread handling a directory will wait for the completion of the child threads that it created to handle the nested subdirectories. It will then collect the results (i.e. the sizes reported by each of these threads), add them to the sizes of the regular files in the directory, and then return the total sum to its creator thread.

Your program is also required to print intermediate totals for each of the directories in the directory hierarchy. Please note that we are not counting here the sizes of the directory entries.

You can assume that the directory tree given as input to your program will not contain any links (hard or soft) or any device files. Only regular files and nested directories will be contained in the tree.

For example, if the directory hierarchy and the sizes of each of the files look like the following,

```
dir/
|
----- a.txt (4)
----- b.txt (5)
----- dir1/
        |
        ----- a1.txt (40)
----- dir2/
----- dir3/
        |
        ----- a3.txt (400)
        ----- dir31/
                |
                ----- a31.txt (4000)
```

then your output should contain the following lines (need not be in the same order).

```
DEBUG: dir/dir1/ 40
DEBUG: dir/dir2/ 0
DEBUG:  dir/dir3/dir31/ 4000
DEBUG: dir/dir3/ 4400
DEBUG: dir/ 4449

Total size: 4449
```

Your program should do error checking to make sure that the user specified directory exists.

5. Things to submit

Submit your solution using the Homework Submission link on Moodle. Please include the information given in the guidelines on your submission. In particular, please supply a makefile and a README file. You will loose points if you do not supply those.

6. Requirements for a Makefile and Code Comments

You must include a properly working makefile. If your makefile fails to compile your program code correctly, 5% of the points will be deducted.

Please comment your code. It is a good practice to comment your code so that it is understandable to you and others. Proper commenting will save you lot of time while debugging. Also see to it that you do not have excessive comments. Comments should be crisp clear and to the point.

7. Grading guidelines

Grading would be broken up into the following main parts:

- 10% Correct final output in terms of the sum of the sizes of all files in the directory tree
- 20% Correct printing of the sizes of all nested subdirectories
- 20% Proper creation and initialization of threads for each of the subdirectories
- 10% Proper synchronization for the termination of child threads
- 20% Proper synchronization for collecting results from all child threads and returning results to the parent thread
- 10% Error handling and printing errors when calling library functions or system calls
- 5% Makefile to compile your program; it must be able to compile your code properly
- 5% Documentation and readability: Documentation of the code. Other documentation includes summary of the programs functionality, instructions if any for using the program, general assumptions in the program, corner cases in which the program would not work, and any other general comments.

8. Errors and omissions

If there are any errors/omissions in this document, the TAs will post about such errors/omissions or other changes to the class bulletin board. It is the responsibility of the students to check the FAQ page and the Moodle Forum regularly to make sure you do not miss such updates.

Moreover, if you have any questions or having difficulty understanding the assignments, please contact us by email (this is the preferred way).

9. Helpful Suggestion

You should consider first writing a sequential (non-threaded) program first, so you know that your basic logic for scanning the entire directory tree is correct. Once you have this sequential program, then derive from this a multi-threaded program. This is merely a suggestion and not a requirement.