Área de Arquitectura y Tecnología de Computadores

Universidad Carlos III de Madrid

Computer Architecture

Lab 4. Free-lock programming

Bachelor in Informatics Engineering

2013/2014

1. Objective

The objective of this assignment is to apply the concepts seen in theory to code a C++ program using free-lock programming and atomic data types.

2. Introduction

In this section we describe the method used to compile the programs and the functions that can be used to acomplish the programming task.

Compiling C++ programs

In this assignment, we are going to use the GCC compiler for C++ named g++ (version 4.6). Be aware that lower versions of the compiler are not compatible with ISO threads and therefore cannot be used for the assignment. To compile the programs you can use the Makefile include in the initial code. Alternatively, you can manually compile programs using:

```
    g++-4.6   -Wall -std=c++0x -O3 -o <output> <input>   -
lpthread -lm
```

Allowed functions

In this assignment you must use the C++ threads and the atomic data types. You can find more information about those concepts in:

http://en.cppreference.com/w/cpp/thread

http://en.cppreference.com/w/cpp/atomic

Basic functions

As the assignment is based on atomic data types, you can use **exclusively** the following related functions:

- atomic_store
- atomic_load
- atomic_exchange
- atomic_compare_exchange_weak
- atomic_compare_exchange_strong
- atomic_fetch_add
- atomic_fetch_sub
- atomic_fetch_and
- atomic_fetch_or
- atomic_fetch_xor
- atomic_flag_test_and_set
- atomic_flag_clear

Explicit functions

The previous functions impose a memory access order that assures sequential consistency. That order can be altered explicitly using the following functions:

- atomic_store_explicit
- atomic_load_explicit
- atomic_exchange_explicit
- atomic_compare_exchange_weak_explicit
- atomic_compare_exchange_strong_explicit
- atomic_fetch_add_explicit
- atomic_fetch_sub_explicit
- atomic_fetch_and_explicit
- atomic_fetch_or_explicit
- atomic_fetch_xor_explicit
- atomic_flag_test_and_set_explicit
- atomic_flag_clear_explicit

Auxiliar commands

To compare to files use the diff command (diff –y file1 file2).

3. Part I: C++ Atomics

This part of the assignment must be done during the lab hours and must be submitted within the time frame established in Aula Global for the submission links.

3.1. Atomics vs Mutex

The objective of this part of the assignment is to compare the performance of a program when it has been programmed using atomics with a classical mutex approach. Compile the atomic_counter and mutex_counter programs and answer the following questions:

1. Execute both programs varying the number of threads to: 32, 64, 128, and 256 with 1000 iterations. Plot and comment the results.
2. Execute both programs varying the number of iterations to: 100, 1000, and 10000 with 32 threads. Plot and comment the results.
3. Execute both programs with 128 threads and 1000 iterations 20 times. Obtain the average and standard deviation and comment the results.

To execute the program use the following syntax:

$ ./atomic_counter <threads> <iterations>

$ ./mutex_counter <threads> <iterations>

4. Part II: Free-lock programming

You are given an initial code with a program that calculates the histogram of pixel values of an image in gray scale. Based on the values of the histogram, the program writes a new output image. To load and store the images, the program uses auxiliary functions found in img_aux. To test the functionality of your program you are given a set of input images: small.pgm, medium.pgm, and large.pgm.

You are asked to:

1. Write a new version of the program (histogram_atom.cpp) that provides the same functionality as the mutex version (histogram_mutex.cpp). Be aware that both programs must produce the same output given the same input files. The atomic version cannot use any element of the threads library except from the threads themselves. Elements from the pthread library are also forbidden.
2. Execute both programs with the parameters <input> output 1000 2, changing input for each of the input files. Plot and justify the obtained results.
3. Execute both programs with the following parameters: medium.pgm output 100 <n_threads>. Change the number of threads to: 1, 2, 4, 8, 16, 64, 128, and 256. Plot and justify the obtained results.
4. Modify the atomic version to use the explicit versions of the atomic functions. Employ a relaxed version whenever possible and justify its use. Repeat the tests of the third exercise comparing both versions.

5. Submission

You must submit your work using Aula Global. Part I must be submitted within the same day in the allocated time window (check Aula Global for details). Part II must be submitted before December, 13rd, 2013. You may also submit your assignment in the day following the deadline with a 25% penalty. The submission must be done separately for the code and the report.

Part I

You must submit a **pdf** file with the report named **infoac_p4_atomic_AAAAAAAAA_BBBBBBBBB_p1.pdf** where A…A and B…B are the student identification numbers of the group members.

Part II

You must submit a zip compressed file named **infoac_p4_atomic_AAAAAAAAA_BBBBBBBBB_p2.pdf** where A…A and B…B are the student identification numbers of the group members. The file must contain:

- Makefile
- Histogram_atom.cpp

You must submit a **pdf** file with the report named **infoac_p4_atomic_AAAAAAAAA_BBBBBBBBB_p2.pdf** where A…A and B…B are the student identification numbers of the group members.

The report must contain the following sections:

- Title page with the name and student identification number of the authors.
- Index of contents.
- Answer to the different questions found in the report. Please copy first the question before answering it. All answers must be adequately justified.
- Conclusions

NOTE: DO NOT NEGLECT THE QUALITY OF THE REPORT AS IT IS A DETERMINING FACTOR FOR YOUR GRADES.

- The submission of code and the reports will be done using AULA GLOBAL using the appropriated links.
- Only the last version of the submitted files will be reviewed.

Rules

1. **Programs that do not compile or do not satisfy the requirements will receive a grade of zero.**
2. **Students are expected to submit original work. In case plagiarism is detected between two assignments both groups will fail the assignment. Additional administrative charges of academic misconduct may be filled.**
3. **Programs without comments will receive a grade of 0.**
4. **The assignment must be submitted using the available links in Aula Global. Submitting the assigments by mail is not allowed without prior authorization.**