OS Project 2

Shared Memory Management -Number Comparison

Deadline: 2014/12/7 23:59, No late submission accepted

Description

The goal of this project is to practice how to use multi-thread/multi-process mechanism to speed up your program, and to manage the shared memory to protect critical section. You will be given 4 files with lots of numbers on it, you need to compare the numbers on those files, and output the number whenever your program gets a number larger than the maximum number you have obtained so far. More details will be given in the following sections.

Input files:

You will be given 4 files: testdata1.txt ~ testdata4.txt
Each input file (text file) contains lots of integer numbers (unsigned). See the testdata for more detail.

Implement Details

Your program should create 4 threads (or 4 processes). Each of them reads the numbers from one of the 4 input files and then writes to the shared memory concurrently with other threads (processes). While the 4 threads (processes) are writing to the shared memory, there is a fifth entity which concurrently gets the numbers from shared memory, and do the comparison. For the fifth entity (called comparison entity), it needs to produce 2 output files:

Output Files : share.txt & result.txt

Every time the comparison entity gets a number from the shared memory, it writes the number to the share.txt file, and specifies where the number from (it may come fromThread1~Thread4). For example, when comparison entity gets a number "990" from thread 1, it should output "Thread 1 : 990" to share.txt. Next, do the comparison. If current number is larger than all the previous numbers, write this number to the result.txt.**(Notice: 0 is initialized in share memory at the beginning of program)** So you can see the history of temporary largest number through this file, and the last number will be the largest number in all the 4 input files. The example is given as the testdata.

Noted that when a reading entity reads a number from the input file, it writes the number to shared memory first, and continues to read the next number. Since four reading entities may write numbers to the shared memory at the same time, you need to guarantee mutual exclusion of accessing the shared memory among them and the comparison entity.

Environment

The environment of this project is limited to the CSIE workstation. All the other environment is forbidden.

Hand in

You should upload your code, execution program and makefile to e3. Please make sure the correctness of your code and the makefile. You may get **zero** if we cannot compile your code.

A report of this project is also needed. **Do not copy and paste the code directly.** You can include your thoughts or the difficulties encountered in this project.

All the files should be put into one single compressed folder and the folder name should be the same as your student ID (ex: 00XXXXX.rar or 00XXXXX.zip).

Scoring

The scoring of this project is as follows:

- Program correctness (60%) and time consumed (30%)
- Report (10%)

Reference:

Pthread API: http://cursuri.cs.pub.ro/~apc/2003/resources/pthreads/uguide/users-gu.htm#301162