cp ~ cis620s/pub/prod_cons.c
cp ~ cis620s/pub/ buffer.c
gcc prod.cons.c buffer.c -l pthread
./a.out

CIS 620 Sec. 50 Project 1 Fall 2018

(Due: Sep. 26)

The aim of this warm-up project is to give you experience using threads, processes, and make. You are asked to evaluate the creation/deletion cost for threads and processes. You also need to measure/compare the performance of a thread-based benchmark program on multi-core machines.

You are asked to write five C files: `crt.c et.c ep.c mm_thr.c etime.c`. The file `etime.c` contains a function `etime()` which return the elapsed time since the last call to `etime()`. The `etime()` function can be implemented by using `gettimeofday()` and a static local variable.

The files `crt.c`, `ep.c`, `et.c` and `etime.c` are compiled and linked together to build the executable file `pcrt`. Then, create `tcrt` which is a hard link to `pcrt`. If a user runs `pcrt`, the `main()` function in `crt.c` checks `argv[0]` and then calls the function `ep()` in `ep.c`. In `ep()`, you need to use `fork()/waitpid()` to measure the process creation/deletion time. Similarly, when a user runs `tcrt`, the `main()` function in `crt.c` checks `argv[0]` and then invokes the function `et()` in `et.c`. You have to use `pthread_create()/pthread_join()` in `et()` to evaluate the time for thread creation/deletion.

To see how the dynamic memory affects the performance of thread/process, your programs need call the function `calloc()` to allocate memory before starting the timer. The argument `argv[1]` determines the size of the dynamic memory (in K-Bytes) to be allocated. For example,

```
spirit% pcrt 1024
```

requests 1024K bytes from the heap area using `calloc()`. Note that your child process (or thread) needs to update the dynamic memory to see how copy-on-write affects the performance. For comparison purpose, run both of your programs (`pcrt` and `tcrt`) with at least five different sizes (e.g. 0, 1024, 2048, 4096, 8192).

The files `mm_thr.c` and `etime.c` are compiled and linked together to build the executable `mm_bench`. In `mm_thr.c`, you need to measure the computation time for multiplying two 160x160 matrices. Your program can split the task to a few threads and each thread just handles a portion of the matrix. The number of threads is given through the argument `argv[1]`. For example,

```
spirit% mm_bench 4
```

uses 4 threads to perform the matrix multiplication. You need to run your program using the following different number of threads : 1, 2, 4, 8.

Write a makefile to describe the dependency among these files and to build the three executable files `pcrt`, `tcrt`, and `mm_bench`.

Turnin

Each student has to submit this project electronically using the following turnin command (on grail):

```
turnin -c cis620s -p proj1 crt.c et.c ep.c mm_thr.c etime.c makefile
```

Each student also needs to hand in a hard-copy document which includes the description of your code, experiences in debugging, and explanation of the experimental results. The cover page should contain your picture, name, and your login id. Start on time and good luck. If you have any questions, send e-mail to `sang@cis.csuohio.edu`.