CS 5523: Operating Systems
Fall 2012

Project 1: Performance Measurement for Processes & Threads
(**Due: Oct. 9, 2012, Tuesday before class**)

- **Objectives**
  - learn and practice the usage of system timers;
  - learn and practice performance measurement with system timers
  - practice system calls for using processes
  - practice system calls for using threads
  - practice the usage of synchronization mechanisms
  - learn how to write scientific project report;

- **Project Description** (this is an **INDIVIDUAL** project; you may discuss with your classmates on the usage of system calls, but should write your own codes)

  Part I: Choose a system timer (for example, *gettimeofday*( ) for C/C++ or *currentTimeMillis*() of System class for Java) and measure the **resolution** of the timer. **How**: the program should get the reading of the timer **multiple times consecutively without other statements in between**. Then, the smallest (positive) value between two consecutive readings should be its resolution that you can observe.

  Part II: For the following two parts, you should first create a data file containing one million randomly generated double values.

  Then, in your program, create one child process. The parent process should repeatedly do the following: read in X data values; pass the X values to the child process; and wait for the result data from the child process; once the parent process receives the result X data, write them to another result data file.

  In the child process, it first waits for the new batch of X data values. Then, for each of the X data value, it should perform 1 million rounds of random operations (such as, +1, -1, log() and exp(); you may need to adjust the data value for certain operations). The idea is to do some CPU-intensive work in the child process. Once the computation is done, send the result X values back to the parent process.

  You can use any IPC mechanism (e.g., shared memory or message) for communicating the data values between parent and child process. You should use appropriate **synchronization** mechanisms between the parent and child process.

  For X=1, 10, 1000 and 10,000, measure how much time it takes for these two processes to complete the work (i.e., read the data from a file, pass data between processes, do the computation and write out the results to another data file). You should measure the total time needed in the parent process.

Part III: Repeat the above steps defined in Part II using threads.

Part IV: Write a report for this project. You should include at least the following materials and discussions.

- What is the system timer did you use and information about your working machine (such OS and architecture etc);
- Discuss what you find out about the system timer;
- Discuss how do you address the synchronization problem between processes and threads;
- Compare the measured performance of processes/threads using a figure or table; and discuss your results.
- List the references (such as books and links for sample codes) that help you finish this project.

Extra Credits: to gain some extra credits, you can create more than one child threads (such as 2, 5, 10), measure and report their performances (i.e., how much time it takes to finish the needed computation). Note that you need appropriate synchronization among the parent and child threads.

- **Project submission (what and how):**

    - For program codes and data files: zip them to a single file and name them as XXX-proj-1.zip (for example, Dakai-proj-1.zip) and email it as an attachment to me (dzhu@cs.utsa.edu) and the TA (Rehana Begam: rehan.sheta@gmail.com) **No hardcopy is needed for the program codes**;
    - For the report: print a **hardcopy** and hand in before the class **on the due day.**