Operating Systems–1: CS3510 Autumn 2018

Programming Assignment 1: Multi-Process Computation of Execution Time

Submission Date: 16th November 2017, 9:00 pm

Goal: Solve a simpler variant of the programming problem 3.19 given in the 10th Edition of the book. In this assignment, you will develop a multi-process program that determines the amount of time necessary to run a command from the command line.

Details: You have to write a C program called time.c that determines the amount of time necessary to run a command from the command line. This program will be run as " ./time <command> " and will report the amount of elapsed time to run the specified command. This will involve using fork() and exec() functions, as well as the gettimeofday() function to determine the elapsed time. It will also require the use of a IPC mechanism.

The general strategy is to fork a child process that will execute the specified command. However, before the child executes the command, it will record a timestamp of the current time (which we term "starting time"). The parent process will wait for the child process to terminate. Once the child terminates, the parent will record the current timestamp for the ending time. The difference between the starting and ending times represents the elapsed time to execute the command. The example output below reports the amount of time to run the command ls :

```
./time ls
time.c
time
Elapsed time: 0.25422
```

As the parent and child are separate processes, they will need to arrange how the starting time will be shared between them.

For this you have to develop a mechanism for the programs to share variables using shared memory. The child process will write the starting time to a region of shared memory before it calls exec(). After the child process terminates, the parent will read the starting time from shared memory. Refer to Section 3.7.1 for details using POSIX shared memory. In that section, there are separate programs for the producer and consumer. As the solution to this problem requires only a single program, the region of shared memory can be established before the child process is forked, allowing both the parent and child processes access to the region of shared memory.

You will use the gettimeofday() function to record the current timestamp. This function is passed a pointer to a struct timeval object, which contains two members: tv_sec and t_usec . These represent the number of elapsed seconds and microseconds since January 1, 1970 (known as the UNIX EPOCH). The following code sample illustrates how this function can be used:

```
struct timeval current;

gettimeofday(&current,NULL);
// current.tv sec represents seconds
// current.tv usec represents microseconds
```

For IPC between the child and parent processes, the contents of the shared memory pointer can be assigned the struct timeval representing the starting time.

Report Details: As a part of this assignment you have to prepare a report which will describe the low-level design of your program and give an analysis of its output.

Submission Format
You have to upload: (1) The source code in the following format: Assgn1Src-<RollNo>.cpp (2) Readme: Assgn1Readme-<RollNo>.txt, which contains the instructions for executing the program. (3) Report: Assgn3Report-<RollNo>.pdf. Name the zipped document as: Assgn1-<RollNo>.zip Upload all this on the drive by **16th November 2018, 9:00 pm**.

Please follow this naming convention. Otherwise, your assignment will not be graded.

Grading Policy: The policy for grading this assignment will be -
(1) Design as described in report and analysis of the results: 50%; (2) Execution of the tasks based on description in readme: 40% (3) Code documentation and indentation: 10%.