Operating Systems–1: CS3510

Programming Assignment -1: Computation of Execution Time

Deadline: December 20, 2020, 21:00 hrs

Problem Statement: This assignment requires you to develop a multi-process program involving IPC to find the time needed to execute an instruction from the command line.

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 an 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.

The execution needs the processes to share variables using a concept called shared memory. The child process has to 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 (10th Edition of the textbook) 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.

The following pseudo-code can be used as an outline for the method you have to develop.

```
//structure declarations

int main(){

char *ptr;        //shared memory pointer

int FD = shm_open(...)          // create shared memory object
ptr = /*map the shared memory*/

pid_t pid;
pid = fork();

if( /*condition for fork failure*/ ){
….
….
}
else if( /*condition for child process*/ ){
…
…
}
else    // for parent process
{
…
}

shm_unlink(/*...*/)

}
```

Deliverables:

1) You have to prepare a report for the above assignment, where you have to explain the working details of the program you have submitted. In addition to this, you also have to explain the output you have obtained. This file should be named as Asgn1-Report.pdf

2) Prepare a **README** file that contains the instructions on how to execute your submitted file. The file should be named as Asgn1-README.txt

3) Name the source code file in the following format: Asgn1-<Roll Number>.c

4) Upload all the above three as a zip archive file and name it as Assgn1-<RollNo>.zip