COMPSCI 3SH3 Winter, 2021
February 5, 2021

Lab report due date: February 12th, 2021, 11:59:59 pm

Lab Assignment 2 - Processes

You need to create two different IPC mechanisms. The implementation has to be completed during lab time.

a) Show your solution to TA

b) Answer all questions related to the implementation details

Your lab report should contain a short description the implementation and source code. The report should be submitted on Avenue.

3.19 Question

Write a C program that determines the amount of time necessary to run a command from the command line. This program will be run as

$$./time - xxx < command >$$

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 two different IPC mechanisms. 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-xxx ls
time-xxx.c
time-xxx

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. You will write two versions of this program, each representing a different method of IPC.

1) The first version called *time-shm.c* will have the child process that 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. 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.

2) The second version called *time-pipe.c* will use a **pipe**. The child will write the starting time to the pipe, and the parent will read from it following the termination of the child process. 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 `tv_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:

```c
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. When pipes are used, a pointer to a `struct timeval` can be written to - and read from - the `pipe`.

Lab Report 2 - Your lab report should contain:

- Full source code with comments of both implementations (*time-shm.c*, *time-pipe.c*).
- Short description of your implementation and advantage/disadvantage of each IPC mechanism. The description should not be longer than half of the page.