Thatyana Morales

# **Outcome**

**Timer1**

1. What are the mean and standard deviation values for your observations?
   1. **Mean: 868.419 milliseconds**
   2. **Standard Deviation: 175.0451**
2. What is the 95% confidence level interval?
   1. **Confidence Interval: 0.347108**
3. Why shouldn't it always take the same amount of time to perform this simple action of program startup?
   1. There might be slight differences in the amount of time it takes for the process to perform because of how each operating system and cpu handles forks and process creations. It could be that the cpu is also dealing with other stuff that could affect the time.

**Timer2**

1. What are the mean and standard deviation values for your observations?
   1. **Mean: 2023.255**
   2. **Standard Deviation: 31551.83**
2. What is the 95% confidence level interval?
   1. **Confidence Interval: 62.56609**
3. Why shouldn't it always take the same amount of time to perform this simple action of program startup?
   1. Since the CPU is switching between both processes, sometimes the second process will start before the first process is even finished. Running this program on a loop will cause a greater disparity between process runtimes.

# **Testing and Documentation**

**Q1.** What do you need to test to ensure that the program works properly (that the results are correct)?

The things I would need to test would be the following:

Application

* Make sure the program can create and open a file and deletes it successfully.
* Need to make sure the recordsFile function can successfully write 10 records of 120 random characters to one file.
* Need to make sure the compareRecords function can compare what was written to the file in a specific record to what was read back for that specific record.
* Also need to make sure compareRecords randomly picks a record from 0-9 to analyze.

Timer1 and Timer2

* Need to make sure it correctly keeps track of the start and end time in the right places
* Need to make sure the process is successfully forked into a parent and child process that run simultaneously.
* Need to make sure the time differences are correct when they are printed to the file.

**Q2.** Discuss how you have implemented testing.

* For testing in the application, I would initially use print statements and have files for all the data I needed (ex, I have a file for comparing the records, to make sure the comparisons are correct). I also have the files for the timers to write the end time into. Once those are completed, I have a separate file for the differences in the start and end times depending on the timer program that executed application
* For timer 1, I have a file that holds the start and end times for the process that executes the application program. I also had originally included some print statements to make sure I was in the right process, but I’ve either commented them out or removed them.
* For timer 2, the testing methods are the same as in timer 1. I have separated timer files to keep track of the start and end times form each child process run in the program. In parent, I wait until the other processes are done before finishing parent process. I write the start times to their respective times files before closing the files. After the entire for loop is finished (about 1000 iterations), I have a 2-dimensional array that holds the start and end times respectively. After scanning the appropriate times and storing them in the buffer, I create a file that holds the differences and use the array to print the time differences to the file (end – start). Note, this is done for separate difference files, and for the histogram I used the data from Difference 2 (which is the 2nd child process from timer 2).

**Q3.** Describe how the parameters you have chosen to use in executing your program, and to examine as output will demonstrate your testing objectives.

Your written program and testing documentation should be submitted as separate documents. Your source code and executables should be packaged in a file (e.g., ZIP file). All your files should then be submitted via the assignment in Canvas.

* For running application from timer1 and timer2, I passed to exec 4 different arguments. I passed the name of the executable, the name of the file where the records are written into, the name of the times file which includes the start and end times, and an ID number which tells application which timer file it is dealing with. For example, if the ID is 2, then application will write to the Times 2 file.

Describe the purpose and objectives of the program

* The purpose of application is for the program to create a file, write records to the file, compare records from the file, and then delete the file. In addition, it also needs to catalogue the end time from the respective timer functions and write that to the file.
* The purpose of timer 1 is for the program to fork only once (to create a parent and child process), and execute the application program in the child process. It is also supposed to time how long it takes for the program to fork and execute application, and write that to a file. It then needs to take the differences of all those start and end times and write that to a file as well for easy access to make an excel histogram.
* Timer 2’s purpose is to fork twice, thus creating 2 child processes. The first child process is supposed to execute the application, and while that happens, the second child process is supposed to execute the application as well. Each process is supposed to take a start and end time before going back to the parent process so the parent process can print the times to the files and finish up. Outside of the processes and for loop, timer 2 then takes the differences of those start and end times and writes it to the file so I can easily make a histogram later.

Include a high level or pseudo language implementation of the program

* Application:
  + Take the end time
  + Open a file
  + Void writeTheRecord(file name, buffer)
    - Create a random character generator
    - For (int I = 0; I < records; i++){
    - For (int j = 0; j < characters; j++){
    - Get random character
    - Put random character into buffer
    - Put random character into file name
  + Void compareTheRecords(file name, buffer)
    - Generate random number to pick record
    - Read record from file name into array
    - For(int I = 0; I < characters; i++){
    - Read record into array}
    - Compare record with buffer and print yes if the same
  + Delete file
  + Write end time to file.
* Timer 1:
  + Gettimeofday for start
  + Fork
  + Child Process
    - Execute application
  + Parent Process
    - Wait for child to finish
    - Write start time to times file
  + Get differences of start and time
  + Write differences to array
  + Write differences from array to file
* Timer 2:
  + Gettimeofday for start time 1
  + Fork
  + Child Process 1
    - Execute application
  + Parent Process
    - Gettimeofday for start time 2
    - Fork
      * Child Process 2
        + Execute application
  + Same Parent Process
    - Write start time 1 and start time 2 to files
    - Close files
    - Wait for child processes to finish
  + Get differences of start and time
  + Write differences to array
  + Write differences from array to file

Include a list of the functions used and describe their relationship to the problem solution

* Application
  + void recordsFile(FILE\* fp, char records[10][120]);
    - This function creates the main file with which to write the records in. Not only that, but it also generates random characters and writes them to the file for each record. There is no return value for this function since it is a void type function. The inputs are the name of the file that is to have the records, and a buffer array which will hold the written records to be compared to later.
  + void compareRecords(FILE\* fp, char records[10][120]);
    - This function takes the record file as input as well as the buffer array mentioned earlier with the previous function I explained. It is with this function that the random record number is chosen and read from the file to be compared to the record that is in the buffer array that corresponds to that specified record. Again, there is no return value since this is a void function. One interesting thing to note is that, when comparing the functions, I used garbage read lines since I used \n to separate the records for easy reading.

Document each function –

May be included with the code; NOT just as comments next to statements. State its purpose. Describe any algorithms used in the function. Specify and describe the input and output variables of the function. Describe key local variables and their relationship to the algorithms or purpose of the function. Choose names for variables such that the name describes the purpose or use of the variable.

* All of this is documented in each program (application, timer 1, timer 2).

Testing

Overall: Detail the specific testing objectives, including potential critical points of failure

* My testing objects with application was to ensure the file was created properly, written to the file properly, deleted properly, and that the end time was correctly written to the file.
* For Timer1 and Timer2, some of the issues I came across was successfully having the for loop run on the entire program. For some reason, the files did not delete correctly in the early stages of running the program, but I was able to patch it up later. The only other possible fault with timer1 (and timer 2 in this case) is the start and end times being written in the right order in each timer file (for instance, I checked the first few instances which correctly listed them as start end, start end, etc. but I don’t know if that is the case for all the numbers in the file). This may affect the difference times in the histogram, but from what I have seen, the data looks normal. Other than that, there could be possible disparities with the time difference (even though the code correctly executes and the execution happens right after calling the fork), so I don’t know.

List and describe the specific tests to be performed (and why these are important for evaluating functionality)

* The only thing that needs to be done is for the user to run either timer1 or timer2. If after running it there are no error messages, then the programs ran correctly. All the testing output is in the respected text files included.

Describe testing of modules (functions)

* Within application I included two helper functions that create the file, write the records to the file, and compare records within the file. A print statement is made if the file is successfully removed after running the application program. Files are passed to the functions from the arguments detailed in timer1 and timer2. If there are not enough arguments for application that was listed in timer1 and timer2 (or if the file is the wrong one), it will print “No arguments passed/not enough arguments passed”.

Present test data and the resulting test output. Describe and interpret the test results.

* The test results are on the excel file and the rest of the raw data can be found in the text files included.