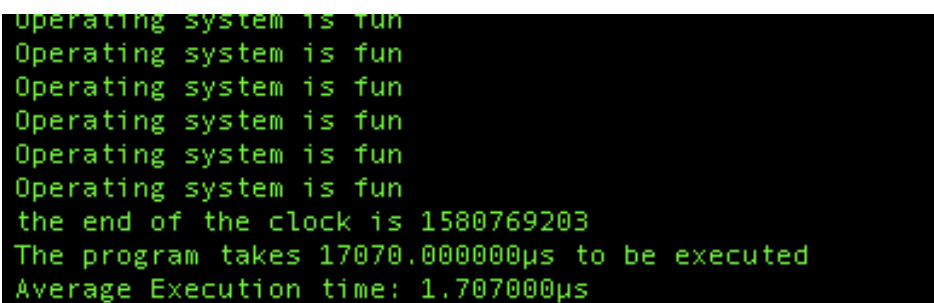


Project1:
System Calls and Context Switch Measurements

Part 1: System Call

In this first part we created a program that measures the cost of a system call. To implement this program, we first create a system call function. Moreover, to create our system call function, we used the `write ()` function which is part of File Manipulation. We use the `write ()` function to make our system call, this function will run as much as the number of iterations that the loop has and calling the `write` function repeatedly creating a simple system call. To measure the how long the program takes, we used the `gettimeofday ()` function which returns the approximate processor time of the program. We subtract time at the end of the program with the one at the beginning and divide it to the number of iterations to give us how long the program takes to execute. In this first part we learned how system called are used and to calculate how long a program takes to execute. The principal challenge encounter in this part is the `gettimeofday ()` function. it is the first time we use this function in C, we did not know how this function works and what library it belongs to. This part took us around 3 hours.



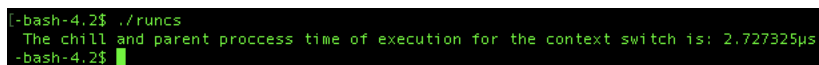
```
Operating system is fun
Operating system is fun
Operating system is fun
Operating system is fun
Operating system is fun
Operating system is fun
the end of the clock is 1580769203
The program takes 17070.000000µs to be executed
Average Execution time: 1.707000µs
```

System Call Program Output Figure 1

This output shows how long the program takes to execute and the average execution time of the program.

Part 2: Measure the cost of a context switch

On the Second part of this project, our goal was to measure the time of a context switch, to do so we used the pipe and the fork functions to implement the algorithm. The principal problem in this second part is to know what pipe and fork functions work and how we can use it to measure the cost of a context switch. This second part took us days just to do search and study these functions. We implement the pipe algorithm to be able to run two processes on a single CPU, this second part was tricky and took us much of our time to implement it. The hardest part was to understand how a process works using pipes and how to connect those pipes. In this part we created three pipes, p1, p2 and p3. P1 and P2 are used to connect to each other, to implement and create the context switch. P1 is used to write on the first pipe, and wait for read from the second pipe (P2). The Operating system basically puts the first pipe in the block state, and switches to the other process, this switching will now read from the first pipe (P1) and then writes to the second part (P2). Due to these pipes are connect to each other, when the second process(P2) tries to read from P1 it begins the block state again and the cycle goes back and forth communicating through the pipes. Pipe 3 (P3) was used to show the time spend between P1 and P2. In addition, another challenge we encounter was to understand how to implement the use of `sched_setaffinity()`, thanks to the help we found with our Ta who explained us how it works and which parameters it takes to make it work and be able to execute the instructions only in one processor. Something we found interesting was the fact that the larger the size for the sample we use for looping, the more accurate our reading was going to be. In this case, with a sample size of one hundred thousand we found the time of execution for the context switch was 2.727us. giving us a better idea how costly it is for the system and why it takes longer to execute than other types of execution.



```
[-bash-4.2$ ./runctxs  
The child and parent process time of execution for the context switch is: 2.727325us  
-bash-4.2$ █
```

Measure the cost of a context switch Figure 2

This output shows the measure cost of the context switch Printing the child and parent process time of execution for the context switch.

Conclusion

In this project we learn how pipes, fork, clock, gettimeofday functions work and how to implement it, it was a long research that took us around 3 days to learn and implement it due to all this function were new for us. This project was challenging for us because it took us most of our time just doing research and try a lot of things first in order to understand the concept of it. Now we know what a pipe is and how we use it to connect between two processes. We learn that the fork () function is a system call that is used to create a new process, fork call the child process by calling parent process using fork (). This project teach give us a better idea of how the cost of the processing time and the data of a context switch cost work in a computer. This give us a better knowledge of how the architecture on a computer works and how the processor of a computer works respect to the cost of a processing time using system call and processor states. Finally, thanks to this assignment we were able to get a better idea on how the states interacts to maintain and run two processes almost simultaneously.