Sydney Ng
804794021

Overview:

This project will use file descriptors that will either only open/ only write to, execute a command to a specific file descriptor, and possibly print out the commands you're executing. For this lab, we implemented simpsh to do this. Here below, we which hare measuring the difference in efficiency of the implementation for our solution in comparison to other types of benchmark types like Bash and Dash. Below are three examples that show how commands such as 'sort', 'wc', and 'cat' can be used to manipulate file descriptors and log the amount of time it takes the user, system, user's children, and system's children to complete its respective tasks.

Test Case #1:

Description: This command will first open 3 files: pg98.txt (which is a copy of the original file that we had to download for our sanity check. This file was chosen as it is a standard file that all students/instructors are familiar with). test1out.txt, which will have the final output of everything from this command, and also open 2 pipes that will allow us to read from pg98.txt and write it into test1out.txt. Then, we will run two commands. First, you will count the number of characters that in this file and output it into the pipe. Then, you will wait for this to finish. Then, you will read from the pipe and then output the number of lines the command to count the number of characters took. (this answer will always be 1, as the output for counting the number of characters is just one number & this takes one line. This will then get outputted into test1out.txt.
Bash Command: cat pg98.txt | wc -c | wc -l; times
Dash Command: cat pg98.txt | wc -c | wc -l; times
simpsh Command: ./simpsh --profile --rdwr pg98.txt --creat --trunc --rdwr test1out.txt --creat --trunc --rdwr test1err.txt --pipe --command 0 4 2 wc -c --wait --command 3 1 2 wc -l --close 0 --close 1 --close 2 --close 4 --close 3 —wait

Bash Times:
- System Time: 0m0.728s
- User Time: 0m0.260s
- Children System Time: 0m1.675s
- Children User Time: 0m1.296s

Dash Times:
- User Time: 0.00ms
- System Time: 0.00ms
- Elapsed Time: 0:00.00ms

simpsh Times:
- System Time: 0ms
- User Time: 1125ms
- System Children: 0ms
- Children User Time: 1931ms

Test Case #2:

Description: This command will first open 3 files: pg98.txt (which is a copy of the original file that we had to download for our sanity check. This file was chosen as it is a standard file that all students/instructors are familiar with). test1out.txt, which will have the final output of everything from this command, and also open 2 pipes that will allow us to read from pg98.txt

and write it into test1out.txt. Then, we will run two commands. First, you will count the number of characters that in this file and output it into the pipe. Then, you will wait for this to finish. Then, you will read from the pipe and then output the number of lines the command to count the number of characters took. (this answer will always be 1, as the output for counting the number of characters is just one number & this takes one line. This will then get outputted into test1out.txt.
Bash Command: cat pg98.txt | wc -w | wc -c; times
Dash Command: cat pg98.txt | wc -w | wc -c; times
simpsh Command: ./simpsh --profile --rdwr pg98.txt --creat --trunc --rdwr test1out.txt --creat --trunc --rdwr test1err.txt --pipe --command 0 4 2 wc -w --wait --command 3 1 2 wc -c --close 0 --close 1 --close 2 --close 4 --close 3 —wait

Bash Times:
- System Time: 0m0.730s
- User Time: 0m0.262s
- System Children Time: 0m1.676s
- User Children System Time: 0m1.300

Dash Times:
- User Time: 0.00ms
- System Time: 0.00ms
- Elapsed Time: 0:00.00ms

simpsh Times:
- System Time: 0ms
- User Time: 1887ms
- System Children: 0ms
- Children User Time: 15810ms

Test Case #3:

Description: This command will first open 3 files: pg98.txt (which is a copy of the original file that we had to download for our sanity check. This file was chosen as it is a standard file that all students/instructors are familiar with). test1out.txt, which will have the final output of everything from this command, and also open 2 pipes that will allow us to read from pg98.txt and write it into test1out.txt. Then, we will run two commands. First, you will count the number of characters that in this file and output it into the pipe. Then, you will wait for this to finish. Then, you will read from the pipe and then output the number of lines the command to count the number of characters took. (this answer will always be 1, as the output for counting the number of characters is just one number & this takes one line. This will then get outputted into test1out.txt.
Bash Command: cat pg98.txt | sort | cat; times
Dash Command: cat pg98.txt | sort | cat; times
simpsh Command: ./simpsh --profile --rdwr pg98.txt --creat --trunc --rdwr test1out.txt --creat --trunc --rdwr test1err.txt --pipe --command 0 4 2 sort --command 3 1 2 cat --close 0 --close 1 --close 2 --close 4 --close 3 --wait

Bash Times:
- System Time: 0m0.726s
- User Time: 0m0.259s
- System Children Time: 0m1.674s
- User Children System Time: 0m1.292s

Dash Times:
- User Time: 0.00

- System Time: 0.00
- Elapsed Time: 0:00.02

simpsh Times:

- System Time: 0ms
- User Time: 1043ms
- System Children: 0ms
- Children User Time: 49272ms

Conclusion:

These processes did not require many system calls, or reading and writing to memory. We were simply looking at data stored in a local directory, manipulating it, and returning what the user wanted. Therefore, the system time and system's children time were close to 0. Instead, the majority of the time was spent in the user time and user's children times. Test Case 3 would need to use the registers to hold the different values while it is sorting the file, which can explain why we see that the amount of system time is 0:00.02 for Dash (while every other test case has 0ms for the system time). In addition, all of the processes were quite complex. All three had processes with subprocesses that required us to fork the processes into children ones. When the second command was more complex in nature, we can see the increase in the amount of time that it took to execute the child process. This is exemplified in Test Case 1 vs Test Case 2, where counting the numbers of characters instead the number of words, the time it takes to execute the child process increases as this is a harder task).