Programming Assignment 1 Report

Gizem Topsakal, 27924

October 31, 2022

1 Command

For this programming assignment we were supposed to pick a command and write a simulation that executes the command. The command I chose is:

```
man diff | grep -A 1 -e "-u" -m 1
```

I picked the `diff` command and `-u` option. To fix the special characters issue, I used `-e`, and to display the description wanted in the homework document, I used `-A 1` and `-m 1`. This way the only sentence printed in the shell is

```
-u  -U NUM  --unified[=NUM]
              Output NUM (default 3) lines of unified context.
```

`diff` basically stands for difference. It finds the differences line by line in a file, and displays it to the user. I picked this command because I think it is one of the most useful commands in Linux. It aims to make two different files identical. This feature of `diff` is really helpful when it comes to really long files and big data. It can help data scientists or engineers to catch mistakes easily. The reason I picked `-u` option is I think it is displays the most easy to understand output. I reduces the unnecessary outputs and generates the basic form of the differences in unified form. It indicates the differences with + and - signs so what is missing or what is residual can be comprehended easily.

2 Process Hierarchy

In my process hierarchy, first I printed out the display message of shell process because the main function will be my shell. Then in my shell I created **pipe**. Pipe takes two integers. This two integers are the file descriptors for this pipe. So I created an array for the pipe called **fd**. Then to start my first child process I used fork. This child process is for the **MAN** command. If fork is successfully created, I displayed the man process message with the PID of the process. After that, I used **dup2()** function to duplicate and write the standard output of the man process to fd[1]. fd[1] is the writing part of the file descriptor. Then, I closed the read and write ends of the file descriptor. Later, I created the arguments for man command and executed them with **execvp**. Then, the shell process will wait for this execution to end. After these processes were successfully executed, I created a second child process with fork for the **GREP** command then printed out the PID of the process. Then, I used **dup2()** function again so that I can duplicate the standard input to the read end of the file descriptor so that pipe can read the input of the grep command. I again closed the read and write ends of the file descriptor. Then I added an open statement to write the grep command's output to the output.txt. After that, I executed the grep arguments. Then in the else statement I closed the read and write ends of the file descriptor for the last time. Then shell process waited for the child processes to finish and executed the shell process message with it's PID.