CS307-PA1-Report

Mert Coşkuner – 29120

Main command: “man ls | grep -e -B -A 2 -m 1”

Principal purpose :

man: Used to display the user manual of any command.

ls: Used for listing files.

grep: It is a filter that is used for searching specific lines.

Used grep flags:

- -e : specifies the pattern for search.
- -A 2: Gives 2 lines after match
- -m 1 : Stops after 1 match.

Motivation Behind the Command

First, I picked this command because I was wondering how I could make the file searches quicker and easier. Moreover, with this question in mind, I wanted to start with the ls command which is used for listing the files. We know that the “grep” command is used as a filter to search specific lines. By adding 32new flags, which are -e and -m 1, to the grep command I extended the usage of this command. Furthermore, with “-e,” I added search pattern to my command line which gave me the flexibility to search for my desired flag. After that, I used the “-A 2” flag to get the 2 lines after matching occurrence. Lastly, I used the “-m 1” flag to stop the search after finding the first match.

Program Structure

My man and grep have a sibling relationship and they can run concurrently (2a). They are sibling because they both have a parent-child relationship with the SHELL(Parent) process. They can run concurrently because once the main process started. The parent process (SHELL) continues through execution and creates the grep process using the fork() function again. To be sure, I also added a message through the pipe form man process which is read from the grep process to print the outputs (process outputs in the console) in a correct order.

Program Implementation

First, as required in the assignment documentation, the main process was named SHELL. With the given syntax (I’m SHELL process, with PID: 38810 - Main command is: <insert piped command here>) the main command and PID were printed to the console. After that, I created the first process, which is for the man process, using the "fork( )" function and named it "process_man". From now on, we need a file descriptor to connect the first created process which will be created in the program the grep process. So, "pipe_fd[2]" was created as a file director and passed to the pipe function. Now, we need to check the "process_man". To check, we need if, else if, and else structure. Their parameters will be checking what is returned from the fork() function which was equalized to "process_man" previously. If it is equalized to zero, we can be sure that the man process is initialized successfully and we will see the process name and its PID from the console in the given syntax(I'm MAN process, with PID: 38811 - My command is: <insert man command here>). Moreover, there is a need to close the reading end of the pipe because there is no need for reading operation in this process. After that, I wanted to create a message and write it through the pipe's write end. The main reason behind this is to be sure of the order of the console. By using the dup2() function we are copying a file descriptor and writing the STDOUT_FILENO to the write end of the pipe. Moreover, I created an "args" array to give my commands to the execvp() function before I closed the write end of the pipe. The process is performed and now it will go to the inside of the else statement and create a new process for the grep. The syntax will be the same as the man process but this time after successfully using fork() I closed the write end of the pipe because there is no need to write in this process. After that, I read the message that was sent from the man process which makes us sure about the program concurrency and print order of the console. In the if statement blocks it will check the returned number from the "process_grep" If it is equal to zero the process will be initialized successfully. The result from the man process will be taken from the read end of the pipe_fd using the dup2() function. Also, another copy using the dup2 function will be formed to write through the output file. After that, I also closed the read end of the pipe, and following this execvp will be executing the commands. Lastly, with waitpid () our process will wait with their order of pid (grep will be after then man) and after these functions, we will see the finishing information of the program in the console.

.