PROGRAMMING ASSIGNMENT (PA) 1

SHELL COMMAND EXECUTION STIMULATION IN C REPORT

CS307 OPERATING SYSTEMS

SPRING 2022-2023

DEADLINE: 3 APRIL 2023

REPORT BY

ZEYNEP KURTULUS

29045

Problem Description

In this programing assignment we are asked to implement a C program which stimulates a shell command execution admitting a specified pattern. According to the pattern our program will be stimulating the piped execution of man and grep commands as if they are being written in a shell. To be able to perform the given task we have used fork, execvp and wait systems calls.

Command Formation & Argument Selection

The pattern that we were asked in order to stimulate the execution of the shell command was the following:

man $< \textit{command} >$ **| grep** $< \textit{option} >$ **> output.txt**

We may see that we have variables that has to be determined, before the stimulation takes place. Here *command* and *option* variables are the ones that we are expected to determine, they will be arguments to the man and grep commands. For this assignment as specified in the command.txt file the command that is selected is the following:

```
man grep | grep -A7 -e-w -m1 > output.txt
```

To begin with, **`man grep`** command will show the manual page of the grep command and specifically the grep command itself is used for searching the specific text patters within a file. Moving on, **|** symbol is known as the pipe, pipe is used for combining to or more commands together. When two or more commands are piped together the output of the first command is the input to the second command in the pipeline. When we have to perform more complex tasks that require combining multiple operations pipe is being used. In the command that we use in the assignment, the pipe will take the output of the man grep command which is the manual page of the grep command, then sends it to the left-hand side of the symbol. The left-side of the pipe symbol has multiple jobs. You may see the extra options that we provided inside the argument. -A7 option tells the grep command to display seven lines of context after a match is found regarding the specific pattern that is being searched by the grep command. -e option is for providing the search pattern to the grep command so accordingly, -e-w command basically tells the grep command to search for the exact letter which is “w”. Finally -m1 option means that, the grep command will stop after it has found the first match, so in other words -m1 command makes sure that the grep command will stop searching for the letter “w” after it has found the first match in the manual page of the grep command. Overall if we combine all these extra options, the command that is being used in the assignment does the following: it searches the first occurrence of option -w inside the manual page of the grep command, prints that line and 7 lines after that match. The reason that we have asked 7 lines to be printed is, we have seen that -w option has 7 lines of description in the manual page once we used the man grep command in the terminal. Lastly, **`> output.txt`** performs the print operation of the output to the output.txt file once man and grep processes are finished. The reason that I have chosen the grep command is that in Computer Science searching is one of the fundamental problems and an important task. Accordingly, the grep command simply makes it really easy and useful to find the specific pieces of information that we are looking for inside lines of definitions, lines of code or large files I wanted to choose the grep command in this assignment.

C Program Implementation

As we all know we will initially start with having a main process inside our pipeSim.c program and in our case the main process (i.e. parent process) is the SHELL process. First we see that the Shell process and its process id is printed to the console. At the beginning of the program we are creating an integer array of size 2 called fd, this array will be provided as an argument to the pipe function and sets the file descriptors for the read and write ends of the pipe. N order words when we write pipe(fd) in our code our program creates a pipe, then at fd[0] the file descriptor for the read end of the pipe will be store and at fd[1] the file descriptor for the write end of the pipe will be stored. After checking the pipe was successful, variable rc1_id will be initialized with the fork() call, after this point the parent process (Shell process) has one child. This child will be used to start the MAN process. Then the program checks whether the fork was a success or not, if it is a success then we check if rc1_id (pid of the MAN process) is equal to 0 because we know that if the pid of a process is 0 it indicates that we are inside the child process which is MAN in our case. Once we are inside the MAN process we print a line indicating we are inside the MAN process and also the pid of it. Moving on, we close the read of the pipe process (fd[0]) because the job of the MAN process is only to write the manual page of the grep command so it will only use the write end of the pipe (fd[1]). Then we use dup2 function call, this function redirects the result of the MAN process to the write end of the pipe so that the other processes can also read the data. Once this process is done we close the write end of the pipe. After that we created an array which will be used to pass arguments to the execvp system call. At this point the execvp system call is used to execute the man command with the help of the grep command. After this task is done as specified in the assignment document we use waitpid command to make the SHELL process wait until the execution of the MAN process is completed. Then, variable rc2_id will be initialized with the fork() call, after this point the parent process (Shell process) mow has two children. The second child will be used to start the GREP process. Then the program checks whether the fork was a success or not, if it is a success then we check if rc2_id (pid of the GREP process) is equal to 0 (indicates that we are inside the GREP process). Once we are inside the GREP process we print a line indicating we are inside the GREP process and also the pid of it. Adding up to this, , we close the write end of the pipe process (fd[1]) because the job of the GREP process is only to read the manual page of the grep command so it will only use the read end of the pipe (fd[0]). Then we again use dup2 function call, this means that any read from the standard input fd[0] instead. We perform this operation so that when one process writes data to the pipe the other process can read this data. Once this process is done we close the read end of the pipe. After that first we generated the output.txt file if it does not already exist then we again created an array which will be used to pass arguments to the execvp system call. Moving on, by using the dup2 system call we ensure that instead of displaying each output to the console they will be directed to the file specified which is output.txt then we again call the execvp system call, it basically executes the grep command with the arguments that are stored inside the arr2 lastly we close the write end of the pipe. Finally we again use waitpid system call to make the SHELL process wait until the execution of the GREP process is done. After both 2 children processes are finished (i.e MAN and GREP) the SHELL process (i.e.) parent process will finish its execution and on the console we will the information regarding the SHELL process.