Programming Assignment (PA) -1

(Shell Command Execution Simulation in C)

CS307 – Operating Systems

FALL 2021/2022

Submission Report

by

Furkan Kerim Yıldırım

28138

Problem Description

In this project assignment, we were expected to write a C program that simulates the piped execution of man and grep commands as if they were written in a shell. For this purpose, it was necessary to use the fork, wait and exec system calls that we saw in the lessons.

Command Formation & Argument Selection

Before the simulation was applied, values for command and option variables had to be picked. The command selected in this assignment is as follows:

```
man –P cat ls | grep –e –A –A 1 –m 1 > output.txt
```

"ls" command, displays the name as well as the requested, associated information for each operand that names a file of a type other than directory. "–A" flag lists all entries except for "." and "...". The reason for choosing the above-mentioned commands in this project is because a computer engineer prefers the "ls" command when checking the contents of the directory in BASH and prefers the "–A" option to see the contents of the directory more regularly. "–P cat", "–e", "–A 1" and "–m 1" commands are used for the program and command to get more regular output. "–P cat", specifies that "cat" pager will be used, and this option overrides the "MANPAGER" environment variable. "–e" option specifies a pattern used during the input search. An input row is selected if it matches any of the specified patterns. This option is most useful when using multiple "–e" options to specify multiple patterns, or when a pattern begins with a hyphen ('-'). "–A 1" option prints 1 line of trailing context after each match and "–m 1" option stops reading the file after 1 match.

Finally, "> output.txt" ensures that the output is printed to output.txt after the man and grep processes are finished.

C Program Implementation

First of all, in the pipeSim.c file, the main process in the project was determined as the "SHELL" process and the process number of the "SHELL" process is printed to the console. For the "SHELL" process, 2 child processes were created. These are the "MAN" and "GREP" processes. To connect the "MAN" and "GREP" processes, a file descriptor called fd[2] was defined, and this file descriptor was piped together with the pipe() function. Then, a variable named pid1 is initialized with the fork() function so that the "MAN" child process can be started. In this process, after making sure that the process has been created successfully, the process number of the child process is printed to the console. Then the "MAN" process is executed with the execlp() function and the dup2() function writes the result of the man operation to the write end of the fd. After the "MAN" process is successfully performed, a variable named pid2 is initialized with the fork() function so that the "GREP" child process can be started. In this process, after making sure that the process has been created successfully, as in the "MAN" process, the process number of the subprocess is printed to the console. Then the output.txt file is created, the output of the "MAN" process is taken from the read-end of the fd with dup2() function, the grep process is executed with the execlp() function and with the dup2() function, the result of the "GREP" process is printed to output.txt. At the end of the program, the completion of the "MAN" and "GREP" child processes is expected with the waitpid() functions in the "SHELL" main process, and the program is terminated by printing the process number of the main process to the console.