CS307 - FALL 23-24 - PROGRAMMING ASSIGNMENT 1

Shell Command Execution Simulation in C

AHMET ALPEREN GÜNGÖR

28847

- In programming assignment 1, the main goal is to simulate shell command execution using fork(), wait() and exec() commands. The task requires combining “man” and “grep” commands using UNIX pipes.

Choice of command and options:

- For this task, I chose “ls” command since it is one of the commands that I use the most, especially for directories of large projects that contain many items. “-R” option lists subdirectories recursively, which is especially helpful since it allows us to see all contents with a single command, while understanding their hierarchy.
- **Command.txt: man ls | grep -a -w -m 1 -A 1 -- -R > output.txt**
- man command gives information about other commands, and in that case it gives information about ls command. By using proper pipe commands, output of man is directed into grep command as input. In order to correctly display information about ls -R, some additional options for grep are used:
    - -a is used to make sure input is processed as a txt file
    - -w is used to make sure only lines that form whole words are selected
    - -m 1 is used to stop grep from searching after the first match for optimization
    - – A 1 is used to capture 1 line after matched line

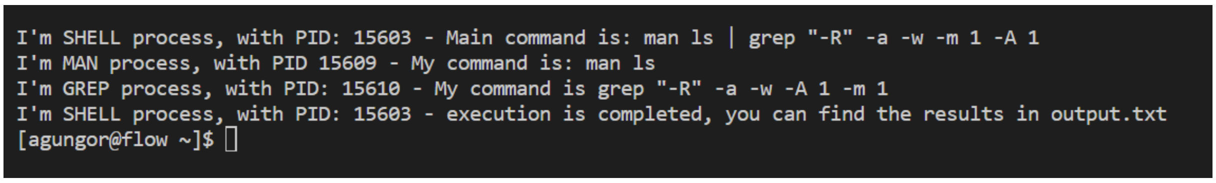
Program flow:

- Pipe is created by calling fd[] into pipe(). fd[0] will be used for reading from pipe while fd[1] will be used for writing from pipe.
- Fork is called to call the first child, which will be the man process. There is a check in case fork fails (<0). This process will generate manual page for ls and write it to pipe for the next process to read. First, reading pipe will be closed – since it is not required. Stdout will be changed with pipe, by dup2. Execvp will execute the arguments for man – making the child process fully functional man process.
- Main process calls fork again, this time for creating the grep process. This process reads the data from the pipe as input – by changing Stdin with pipe, closes standard output and opens a file called output.txt where output of grep will be written. Execvp will execute the arguments for grep – making this child a fully functional grep process. ,
- Therefore, in total, there will be 3 process active at the same time: main, man and grep.
- Main waits for children to finish with wait(NULL), prints information to the console (not to txt file since main process’s stdout is still console)

The program has 2a properties defined in the pdf:

Both man and grep are created from the main process – which makes them siblings.

Man writes to pipe while concurrently grep takes the information from pipe – running concurrently.



```
I'm SHELL process, with PID: 15603 - Main command is: man ls | grep "-R" -a -w -m 1 -A 1
I'm MAN process, with PID 15609 - My command is: man ls
I'm GREP process, with PID: 15610 - My command is grep "-R" -a -w -A 1 -m 1
I'm SHELL process, with PID: 15603 - execution is completed, you can find the results in output.txt
[agungor@flow ~]$ 
```