Comp 304 Project I Report

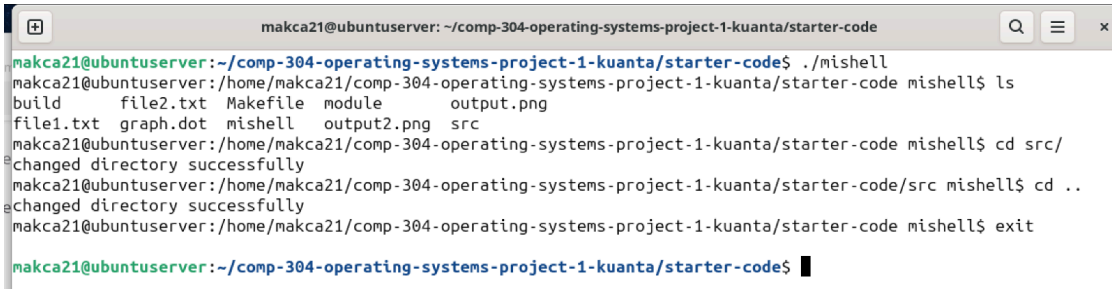
Mislina Akça - 80006 & Duru Tandoğan - 79479

PART 1

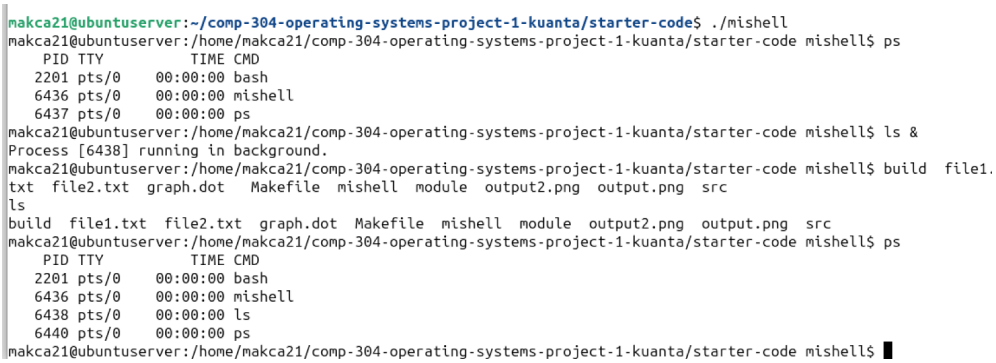
We used the command make to compile our code after the changes done in the c file shell-skeleton.
Then we switched to other shell with the command ./mishell
We implemented the built-in cd command with a minor change in the starter-code file. In the cd part of the process_command method, we changed the arg[0] to arg[1] to match the desired outcome.

For the built-in command exit, we did not added any code since it was already working in the starter code provided to us.

Both commands demonstrated:

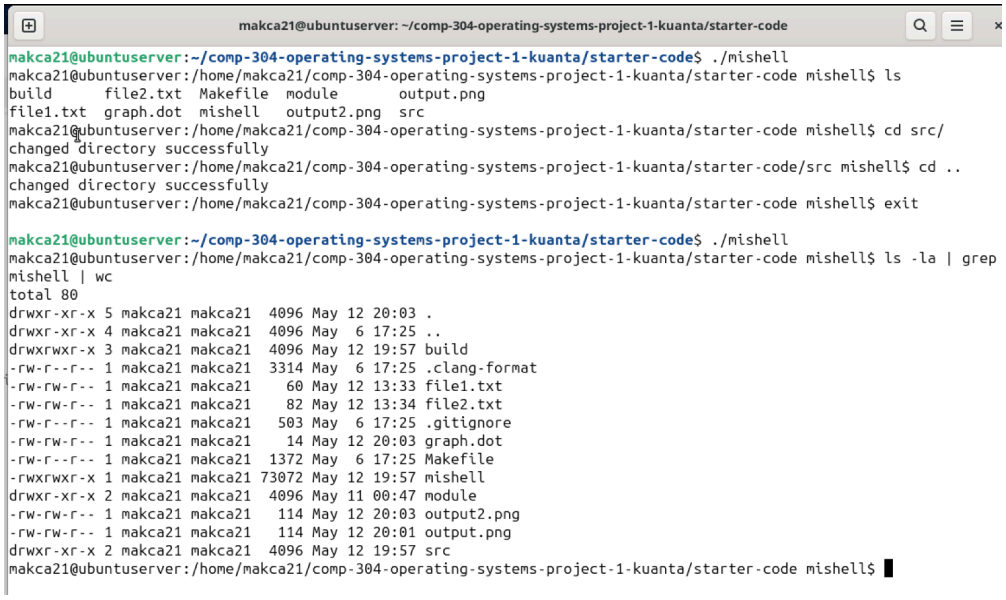


As for the background, we checked for the background flag which is a boolean. If the flag is false, the parent process waits until the parent process finishes. If it is true, parent continues executing while the child is also running.



PART 2

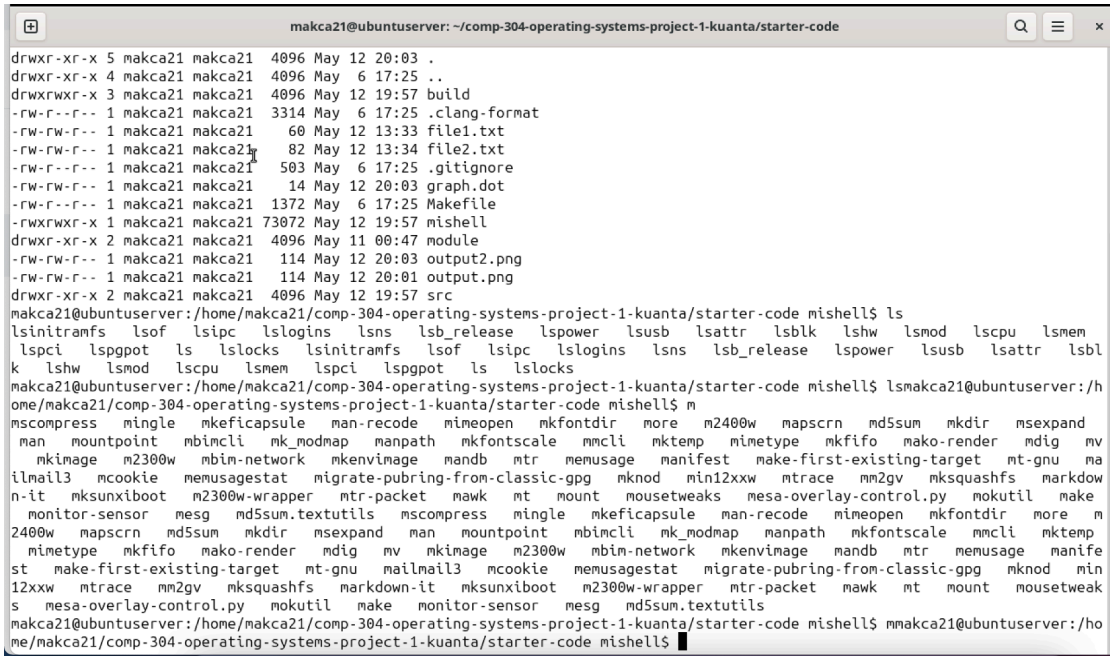
For this part, we created a pipe in the method process_command before the child process, and after executing the command directed the output of the first execution to the write end of the pipe with the built-in method dup2(...) After that, we closed the write end and recursively called the next command after the pipe with the process_command function to do the same thing until there are no next commands left.



PART 3

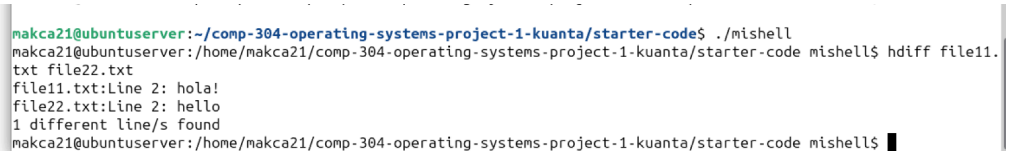
1. **AUTO-COMPLETE**

   In this part, it is first checked if the tab key is pressed with the command->background. After, we format the input since it contained an ‘?’ whenever it is pressed to the tab when there are no whitespaces. After extracting the inputted command, we check each directory by getting the path and parsing the tokens of the path, and finally initializing them for later use if they are not null. Later, we check inside of the directories initialized at the previous step. Inside, we compare each entry with our formatted input and if they do match, we print the entry to the console. By doing this for every directory, we show the user each possible entry they may want to associate with.



2. Hdiff

For the first part, we handled file opening actions initially. We created 2 counters for each file. Choosing first inputted file initially, we created a while loop to read this file line by line and inside this loop lines from the second file were also taken. We compared both lines from these files and printed them to the console if they are different. To handle file pairs which have uneven lines, we first handled the file 2 being short case inside the main while loop. For the case of file 1 being short, we created another while loop for file 2 to count that lines and adding them to the different lines counter.



For the second part, we followed a similar pattern with the first one. Differently, we opened the files with the “rb” option to open every file with any extensions. We created 3 conditions with:

- while their sizes are compatible, first while loop will run and calculate the difference between each unit step by step
- if one of them reaches the end of the file, meaning one is shorter than the other, according to the file which did not reach to the end yet; second or third loop will run and will add each part to the byte difference directly

We will get the byte difference in each case with the method fgetc() and print the result of the final difference in byte to the user.

In this question, just like instructed in the pdf; if -a or -b is not provided -a version will launch automatically.

```
makca21@ubuntuserver:/home/makca21/comp-304-operating-systems-project-1-
kuanta/starter-code mishell$ hdiff -b file1.txt file2.txt
The two files are different in 1895 bytes
makca21@ubuntuserver:/home/makca21/comp-304-operating-systems-project-1-
kuanta/starter-code mishell$ exit
```

3. **Custom Command**
    a. **Mislina's Command: "dog_encrypt -h"**
    When the user writes this command to the console, a greeting message will be printed out to the console asking for input from the user. User gives the input with -n in front of their input, the program will encrypt this word using dog ASCII numbers.

    b. **Duru's Command : "inspire"**
    Here, we have created a new command called "inspire". When the user writes this command on the console, it will randomly select an inspiration quote from the array defined and print it to the console.

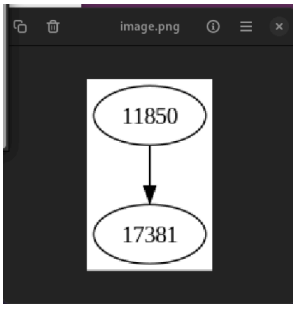
```
duru@fedora:~/Documents/comp-304-operating-systems-project-1-kuanta/starter-code$ ./mishell
duru@fedora:/home/duru/Documents/comp-304-operating-systems-project-1-kuanta/starter-code mishell$ inspire
Your self-worth is determined by you. You don't have to depend on someone telling you who you are. — Beyonce
duru@fedora:/home/duru/Documents/comp-304-operating-systems-project-1-kuanta/starter-code mishell$ inspire
All our dreams can come true, if we have the courage to pursue them. — Walt Disney
duru@fedora:/home/duru/Documents/comp-304-operating-systems-project-1-kuanta/starter-code mishell$
```

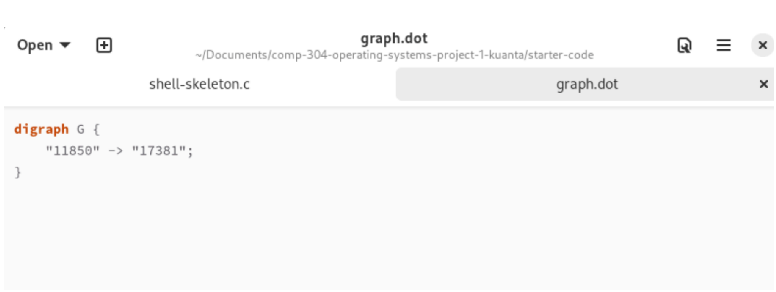
PART 4

We have added another if condition to the process_command function to check the command "psvis".

When the user writes "psvis pid output_file.png" to console a file will be created in the current path which is the graph. The graph comes from the draw_graph function call which creates the image file that we will use as the output file. For each entry, it checks whether the child pid and the parent of the child pid matches, if it does, it uses dot unix language to draw the graph.

```
duru@fedora:/home/duru/Documents/comp-304-operating-systems-project-1-kuanta/starter-code mishell$ ps
    PID TTY          TIME CMD
  11850 pts/0    00:00:00 bash
  17217 pts/0    00:00:00 mishell
  17376 pts/0    00:00:00 ps
duru@fedora:~/Documents/comp-304-operating-systems-project-1-kuanta/starter-code$ ./mishell
duru@fedora:/home/duru/Documents/comp-304-operating-systems-project-1-kuanta/starter-code mishell$ psvis 11850 image.png
duru@fedora:/home/duru/Documents/comp-304-operating-systems-project-1-kuanta/starter-code mishell$
```





Here is the resource of the dot language that we used: https://graphviz.org/doc/info/lang.html

IMPORTANT NOTE: Each time, a new output file should be given as an argument. If not it will only show the first one.