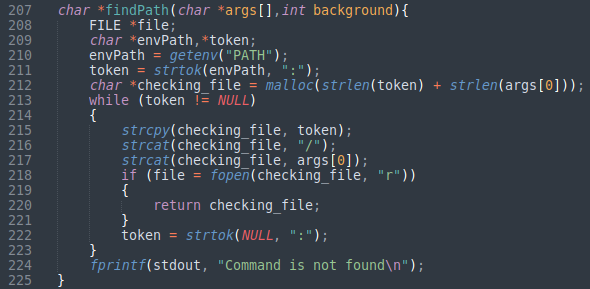
CSE3033 PROJECT 2 REPORT

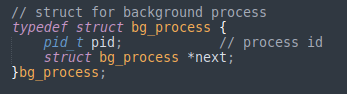
**Hasan Hut**

Part A:

In this part, firstly we have to get PATH variable and then we will split them with “:” because PATH is stored like “/user/bin:/bin:”.

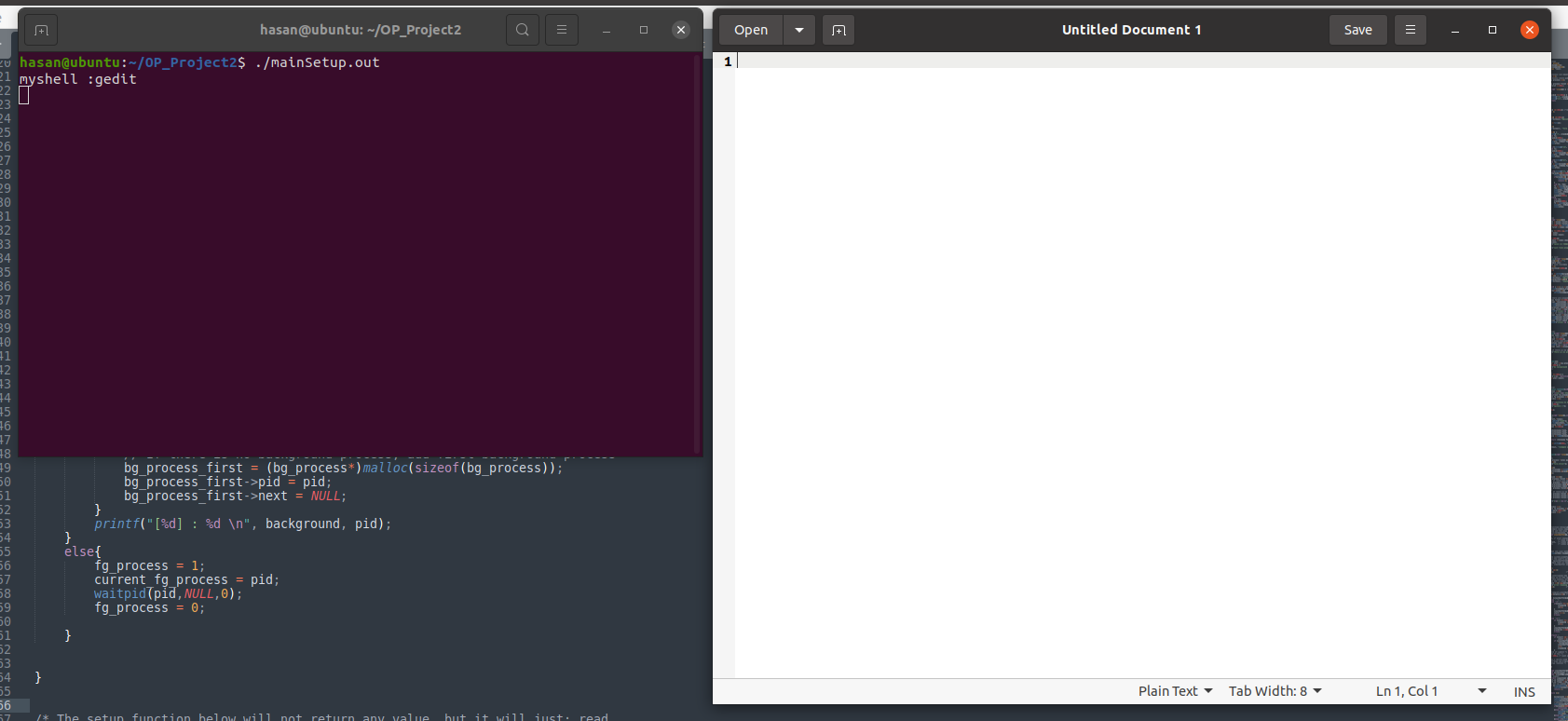


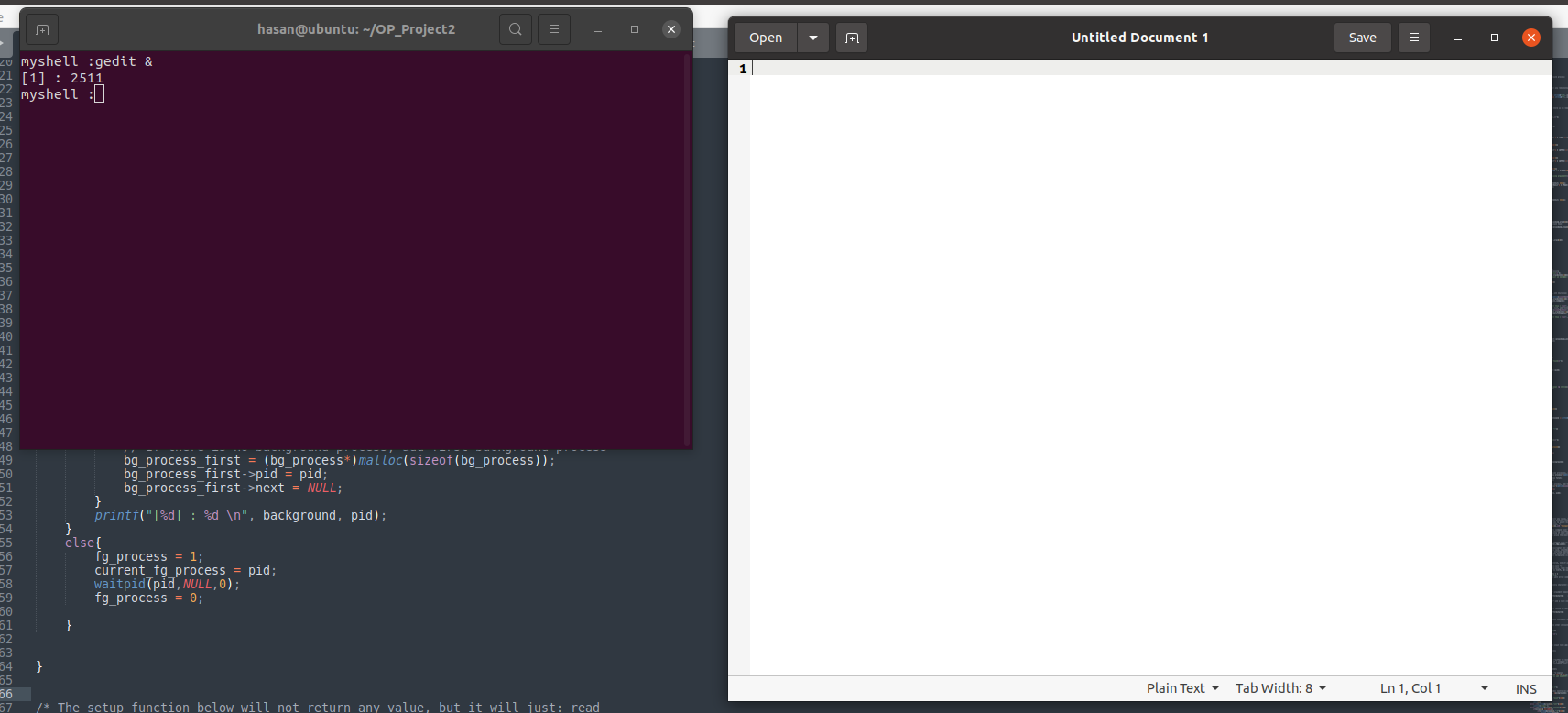
After getting executable path, If the user wants to run a command, we run our function called exec\_command. In this function, we use the fork() operation. We create a child process for the given argument. We put the processes we have created in a linked list structure.





If the background process linked list structure is empty, we first create a new background process. But, if this structure is full, we add a new element to the list.

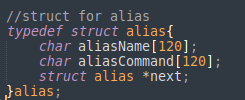




Adding & to the end of the command will make it run on background.

Part B:

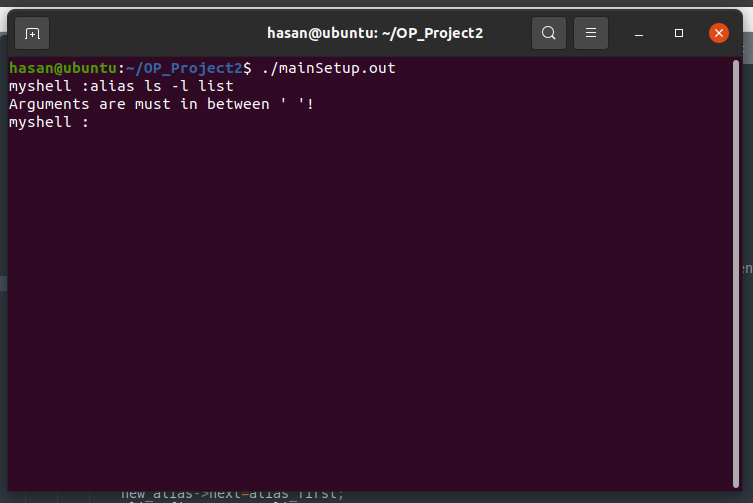
As in background processes, we keep aliases in structs. Our struct structure is like this.



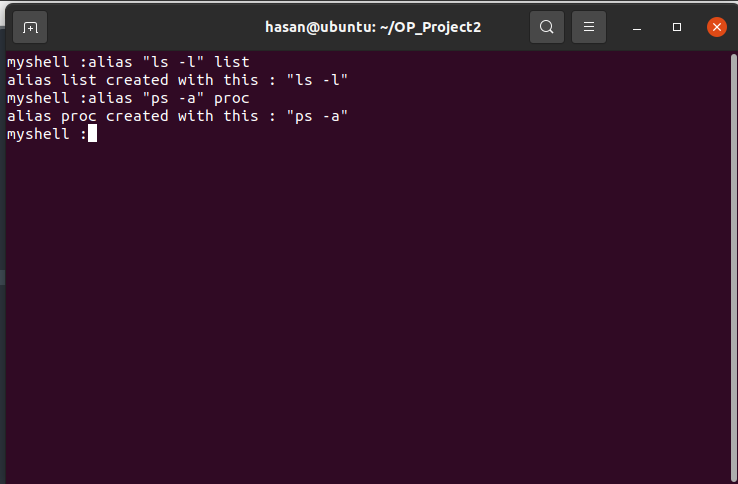
aliasName stores the name of the alias. aliasCommand stores the command we gave for alias. When the user wants to add alias, the function first calculates how many arguments the program has entered.

Then, if one of the arguments is empty or if the arguments are not in the apostrophe, the program throws an error.

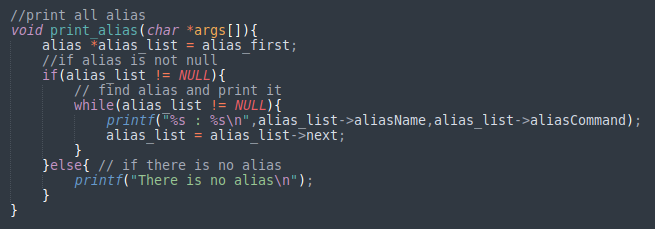
Finally, if the alias linked list structure is empty, it creates a new alias and adds it to the list, but if the linked list head is not null, it adds the alias to the next part of the list.

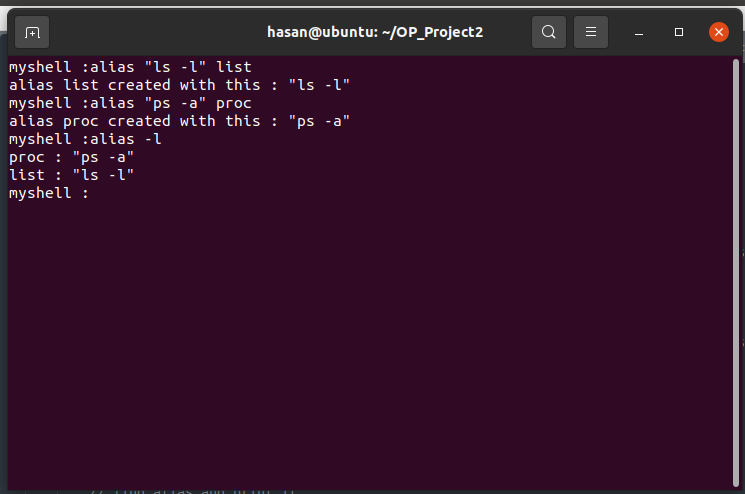


It gives an error because we wrote the structure wrong here.

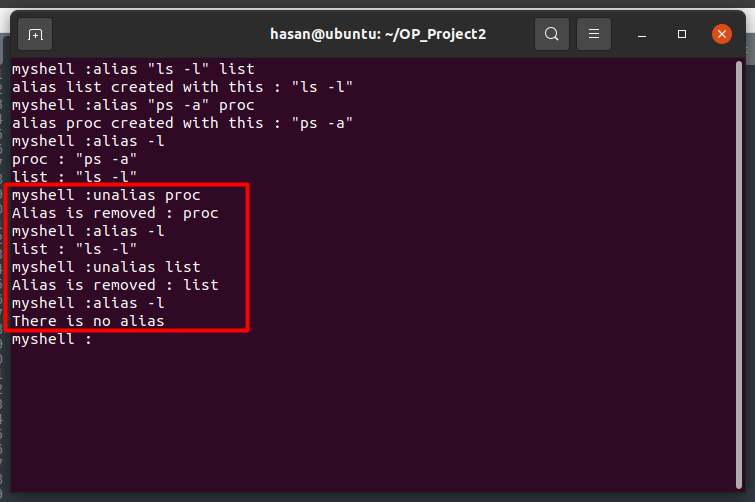


Alias ​​created successfully. If you want to see the list of alias, you can use alias -l command. We can list all aliases by looping through the alias list.

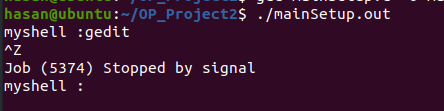




If you want to delete alias you can use unalias [aliasName] command. In this function, it matches the argument entered on the command line with the contents of the linked list structure. If the match is correct, it deletes this structure from the linked list.



* Ctrl^z = We control ctrl^Z with signal handler. If there is a foreground process, it terminates that process.

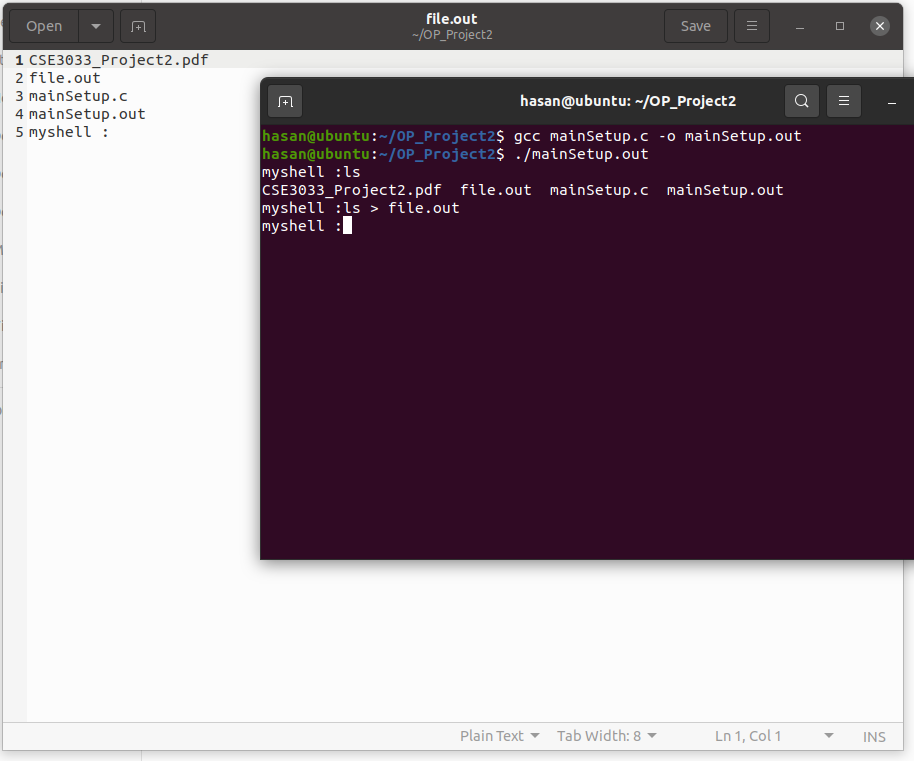


* Exit = When “exit” command is entered, background processes are checked first. If there is a background process, the shell does not close but if there is no background process, the program terminates.

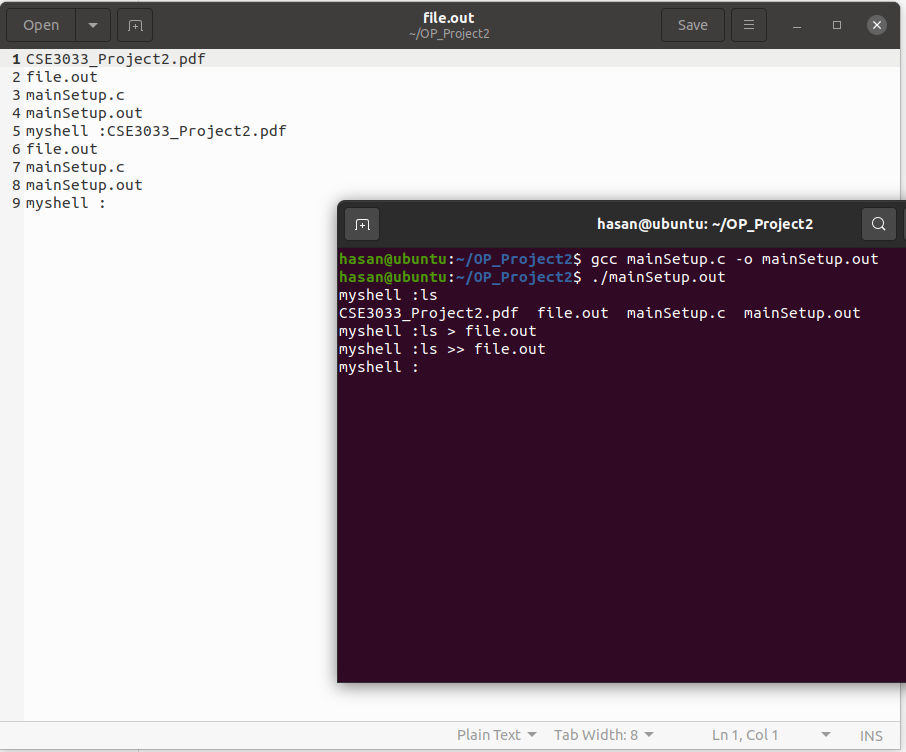
Part C:

For this part, we first checked if the args contains any redirection operators. If it does, we fork out the child.

* prog[args] > file.out



* prog [args] >> file.out



References:

* stackoverflow.com
* geeksforgeeks.com
* http://akademik.marmara.edu.tr/zuhal.altuntas