Lab1

Simple Shell

Yasmine Mohamed Medhat Khalil

3855

In this program, an implementation of a UNIX simple shell that conveniently allows you to run other programs.

Description of the code:

In the main function, there’s a while loop that loops continuously until the user inputs “exit”. Each loop the program takes in the user input it, parse it then execute it.

FUNCTIONAITIES:

1]Print the working directory

The function “PrintWorkingDirectory” is called with each loop so it could print the current working directory before the input of the user so they could know where they are working at the moment by simply calling the system command “getcwd”

2]Parsing

The function “Tokenize” takes in an input line from the user and parse it by white spaces or tabs w new lines and returns an array of tokens and the number of tokens, to check whether it’s an empty command so we could loop again, to the man function.

3]Executing

The function “commandHandler” is called each loop unless, it’s an exit command we terminate the program or it’s an empty command we loop again, it takes in the array of tokens.

First, it check whether the input is a background process or not and sets a flag accordingly.

Secondly, it checks if the command is print the current working directory it calls the system function “getcwd” or if it’s change directory then the program call the function “ChangeDirectory” or it’s something else.

If so, it forks the current process, then, it checks whether this process is a child or a parent and goes through if conditions.

If the process is a child, it goes through executing, we system call the function “execvp” to execute the command.

If it’s a parent process, we check first if the background flag is set, if yes, then we loop again regularly and the process is executed in the background, if it’s not a background process, the parent process waits until this process is executed and terminated in the foreground.

4]Change Directory

The function “ChangeDirectory” checks if the user specified a path or just called cd with no arguments.

If there are no arguments, we call the system function getenv(“HOME”) so it could change the directory to the home directory.

If the user specified the path then we simply call chdir with the path.

5]Handling child signals

In the main function, we initialize a sigaction struct and call the function “sigaction” so it would call our function “SIgnalHandler” every time a child process terminates.

The function “SignalHandler” creates a log file if it’s does not exist already, and appends the statement “Child process was terminated” each time a child process terminates.

**Run samples:**

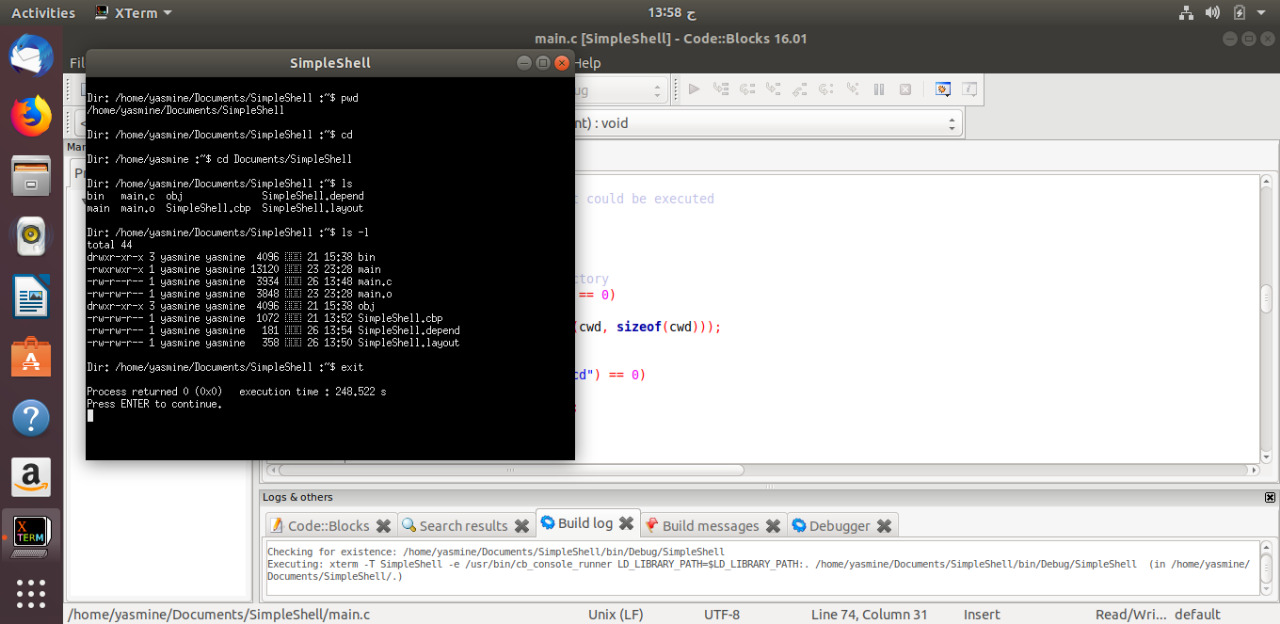


Figure 1: Commands pwd, cd with no path, cd with path, ls, and ls -l.

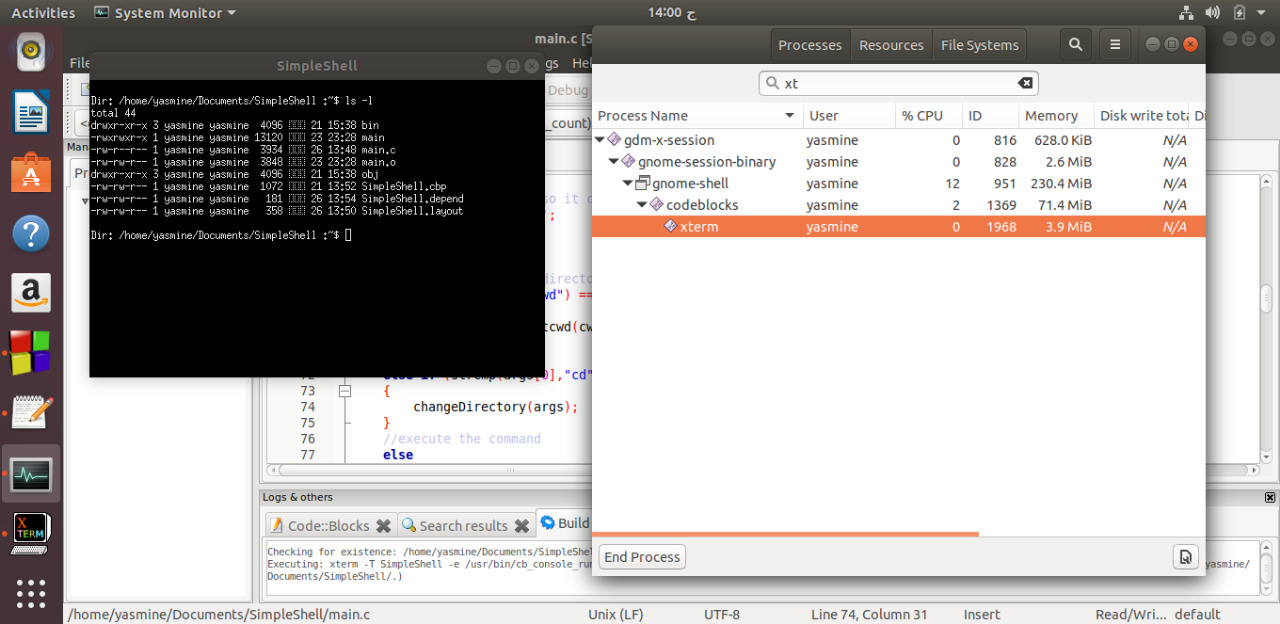


Figure 2: system monitor with xterminal process running.

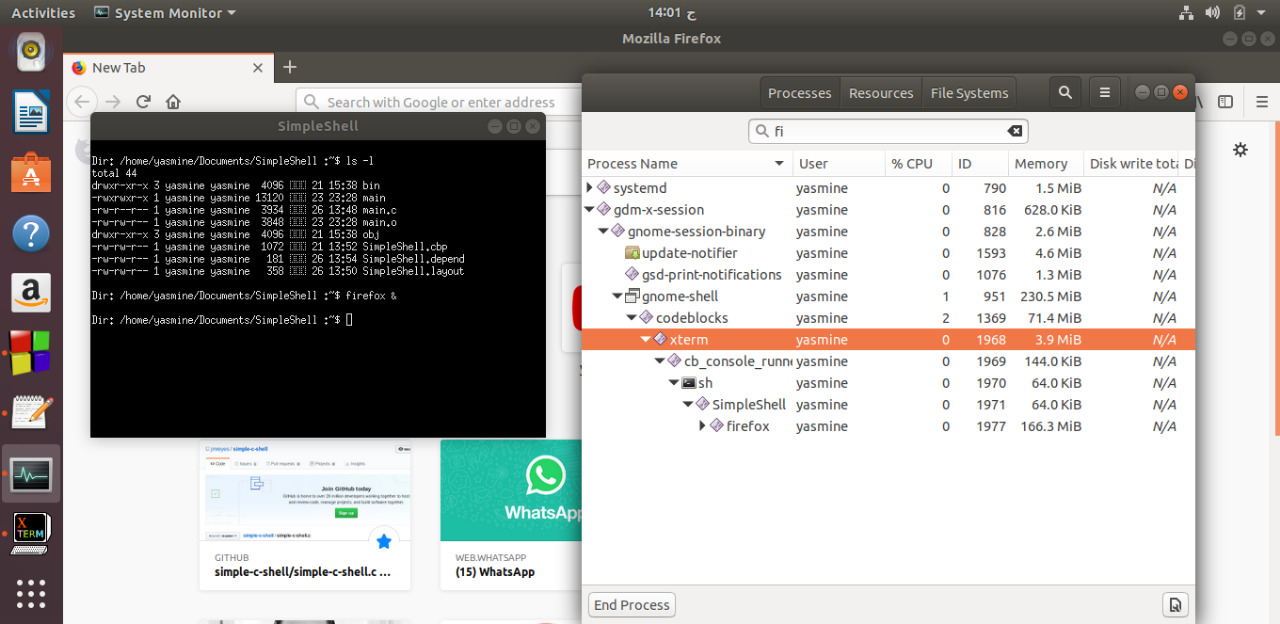


Figure 3: firefox running in the background.

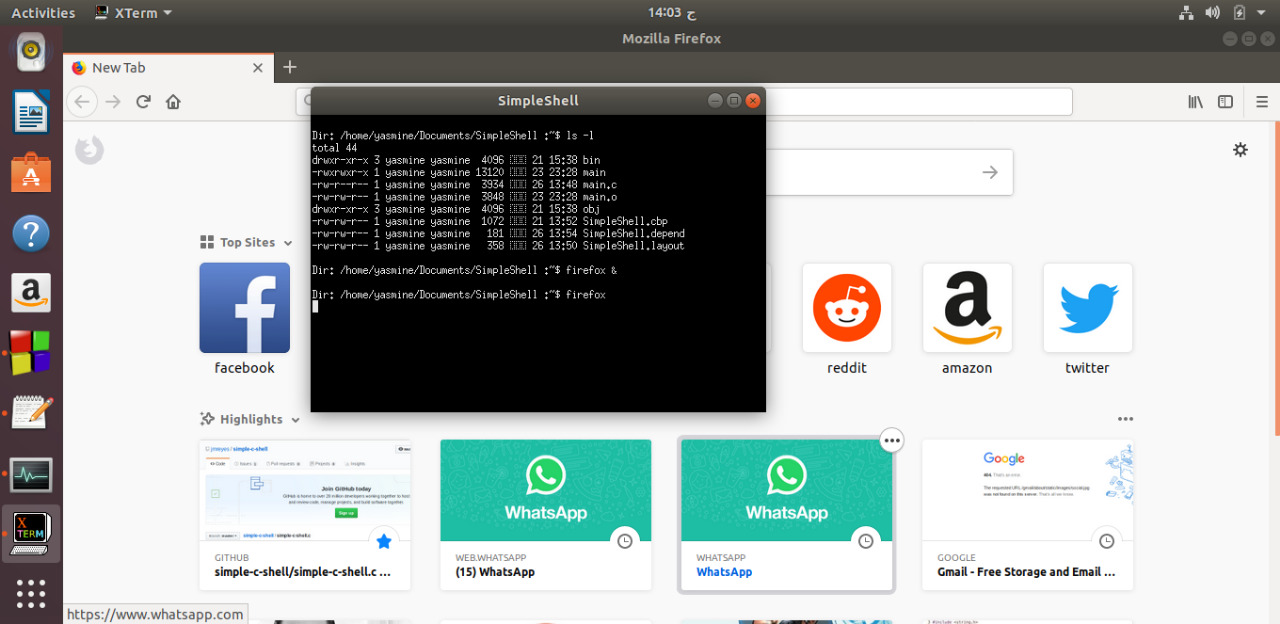


Figure 4: Firefox running in the foreground and parent process waiting.

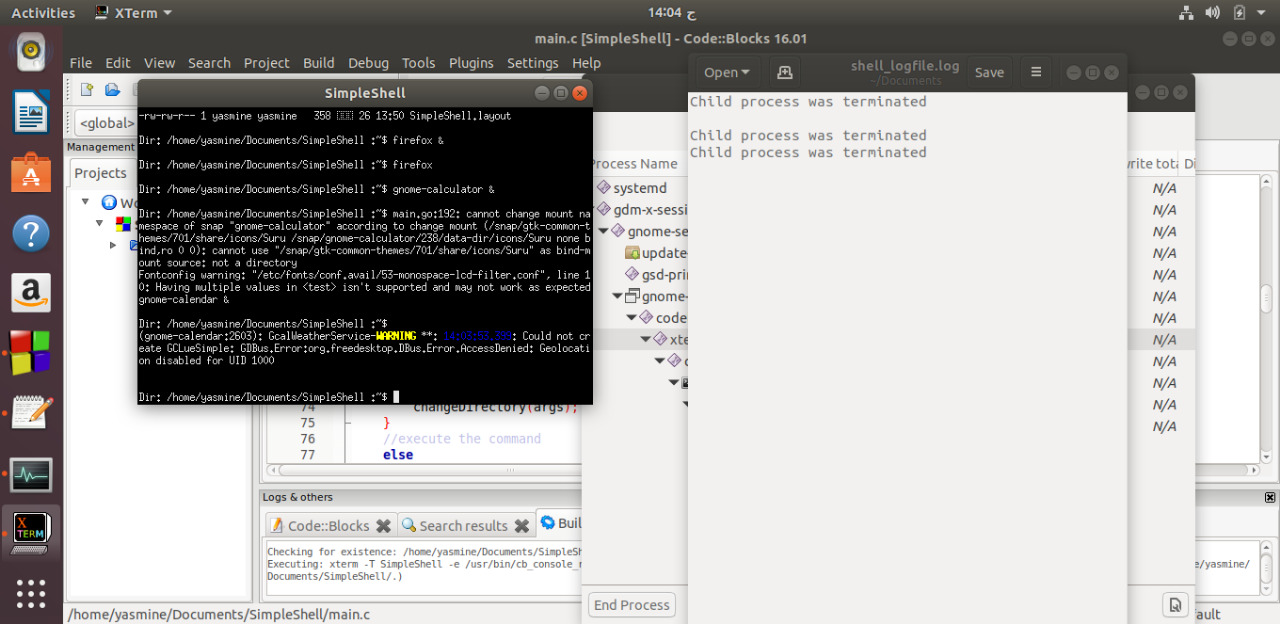


Figure 5

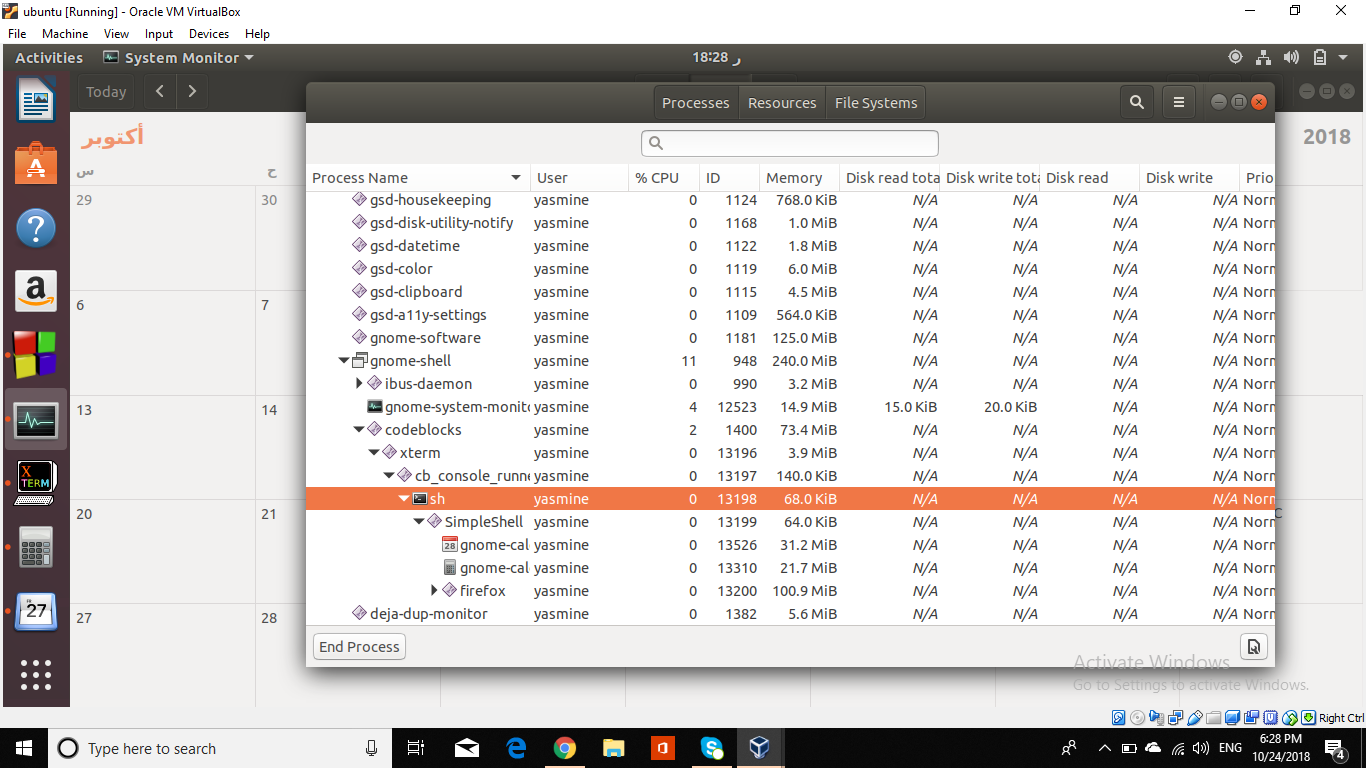


Figure 6: firefox, calculator and calendar running in the background.

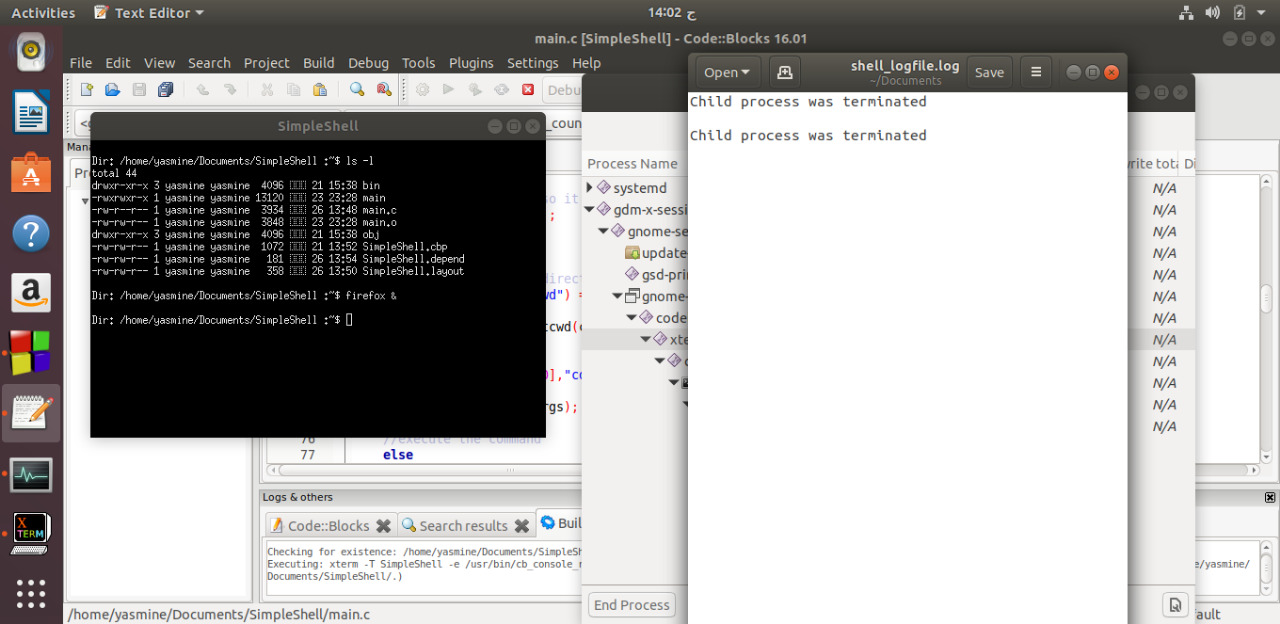


Figure 7: the .log file after the 2 children processes, ls and firefox, terminated.

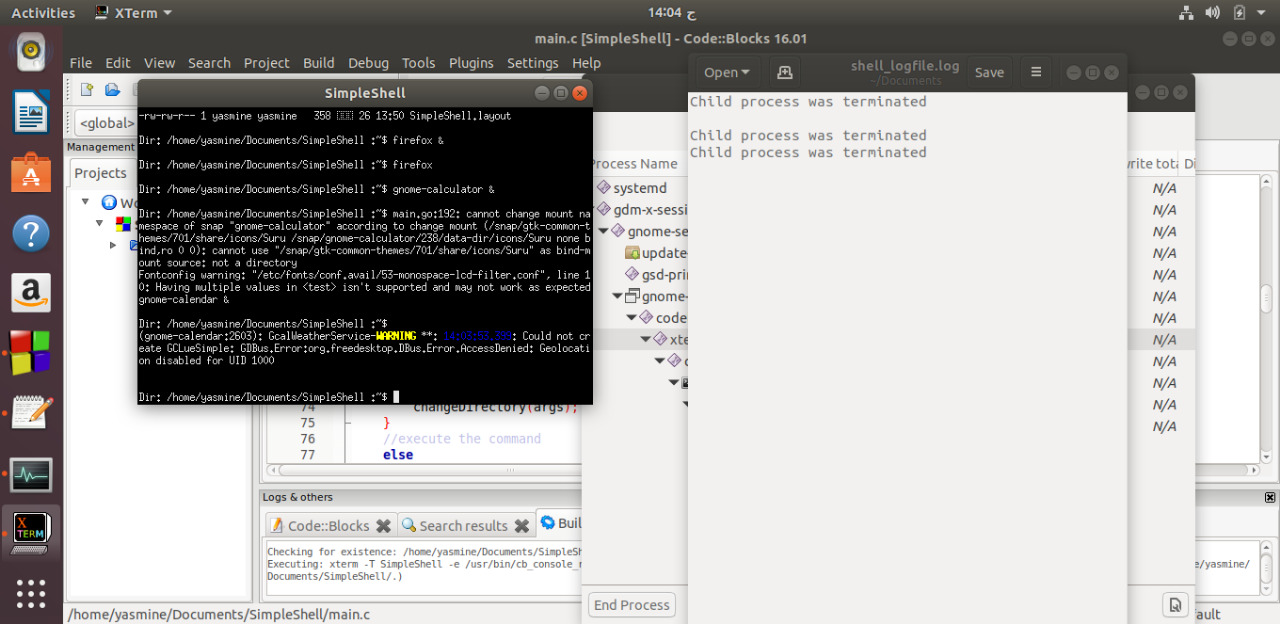


Figure 8: the .log file after the 3 processes, firefox Calendar and calculator, terminated.