Operating System

Lab One
Simple Shell

Presented by:

Mostafa Hakam 5525

CONTENTS

Summary

Implementing a simple UNIX shell that supports:

1. Commands with no arguments.
2. Commands with arguments.
3. Commands with (no) arguments executed in the background.
4. Exit command.

Language Used:

- C/C++ Language.

Libraries used:

1. Library “unistd”.
2. Library “stdio”.
3. Library “string”.
4. Library “ srdlib”.
5. Library “signal”.
6. Library “sys/wait”.

Prcedures:

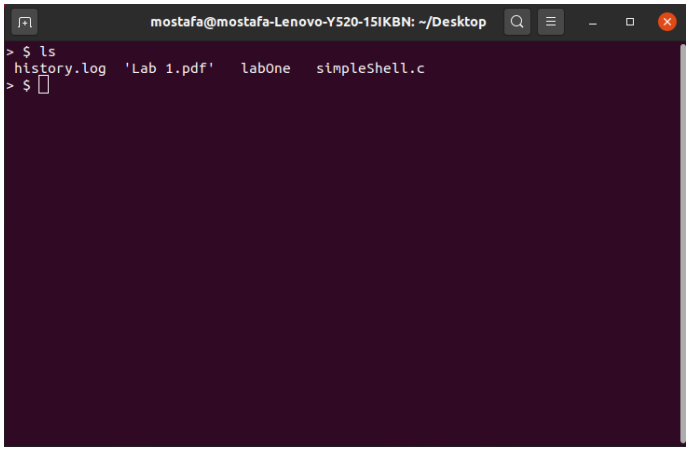
1. Scanning command from user.
2. Tokenising command (and arguments) into strings.
3. Dynamically allocating array of command (and arguments).
4. Using fork(), excevp(), wait(), and exit() functions to create processes.

Code

```c
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/wait.h>

// Handles the signals for commands executed in the
background
void handles() {
    int x, y;
    y = waitpid(-1, &x, WNOHANG);
    if (y > 0) {
        FILE* f = fopen("history.log", "a+");

        if(f == NULL)
            return;

        fprintf(f, "Process %d: terminated successfully in
the background.\n", y);
        fclose(f);
        fgetc(f);
    }
}

int main() {

    // Erase all data in the log
    FILE* f = fopen("history.log", "w");
    fclose(f);

    char** commandArray;

    char userInput[100];
    char* temp, * delimiter = " \n";

    char token[50];

    int commandSize;
    int i, flag, frk, proc;

    // Create signal struct
    struct sigaction sig;
        sig.sa_handler = handles;
        sig.sa_flags = SA_RESTART;

    // Terminal loop until EXIT is called
    while(1) {
        flag = 1;

        printf("> $ ");
        fgets(userInput, 100, stdin);

        // If no command was entered
        if(!strcmp(userInput, "\n"))
            flag = 0;
        else
            strcpy(userInput, strtok(userInput, "\n"));

        // If command was entered
        if(flag) {
            strcpy(token, userInput);
            temp = strtok(token, delimiter);

            i = 0;
            commandSize = 0;

            // Count command size
            while (temp != NULL) {
                commandSize += 1;
                temp = strtok(NULL, delimiter);
            }
commandArray = malloc(sizeof(char *) * commandSize);
            strcpy(token, userInput);
            temp = strtok(token, delimiter);

            // Dynamically allocate command into array of
pointers
            while (temp != NULL) {
                commandArray[i] = (char *)
malloc(sizeof(char) * strlen(temp));
                strcpy(commandArray[i], temp);
                temp = strtok(NULL, delimiter);
                i++;
            }

            i = 0;

            // If EXIT command was called
            if (!strcmp(commandArray[i], "exit")) {
                printf("Proceeding EXIT ...\n");
                exit(0);
            } else if (commandArray[0] == NULL) {
                continue;
            }

            // If command is set to run in background
            if(!strcmp(commandArray[commandSize - 1], "&")) {
                flag = 0;
                commandArray[commandSize - 1] = NULL;
            }

            // Creating processes from command
            frk = fork();
            if (frk >= 0) {
                if (!frk) {
                    proc = execvp(commandArray[0],
commandArray);
                    if (proc == -1)
                        printf("Error: %s: command not
found\n", userInput);
                    exit(0);
                } else if(flag) {
                    wait(NULL);
                } else
                    sigaction(SIGCHLD, &sig, NULL);
            } else
                printf("Forking Error");
            usleep(100000);
        }
    }
}
```

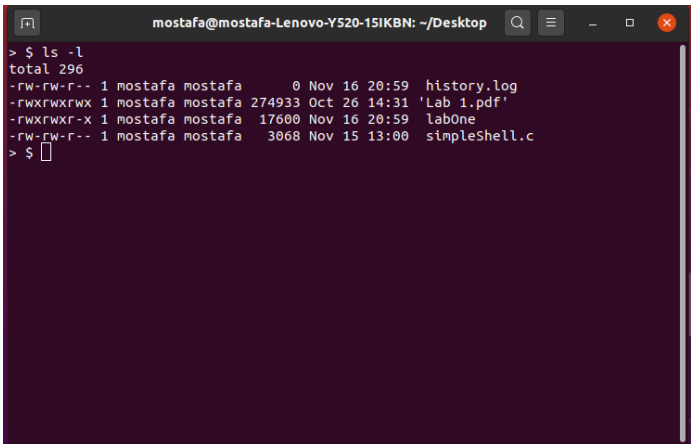
Sample Runs

Commands with no aguments:



Using command “ls” with no arguments to list directory in UNIX shell.

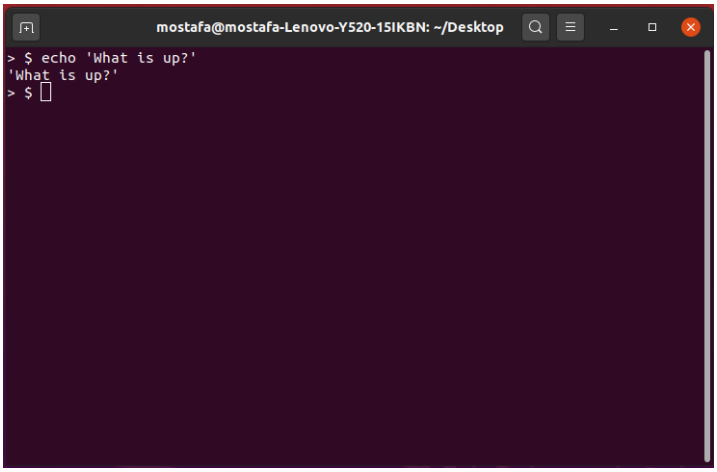
Commands with aguments:



Using command “ls” with argument “-l” to list directory in UNIX shell with long format.

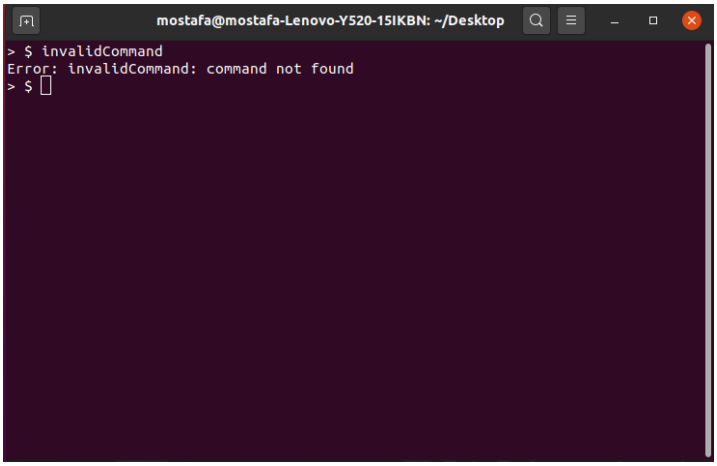
Sample Runs

Commands with variable arguments:



Using “echo” command is used to display line of text “What is up?” that are passed as an argument.

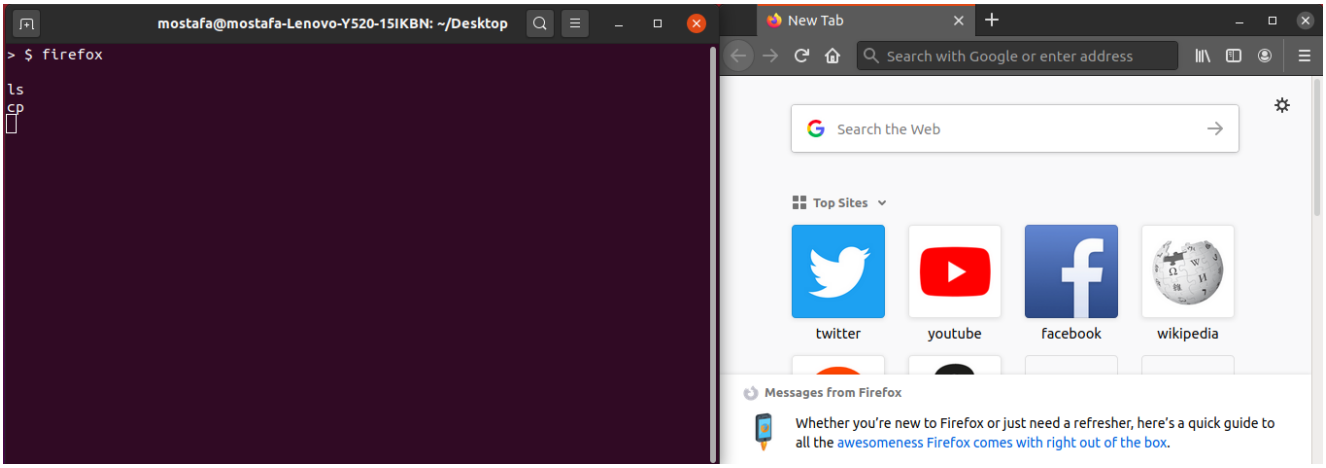
Invalid commands:



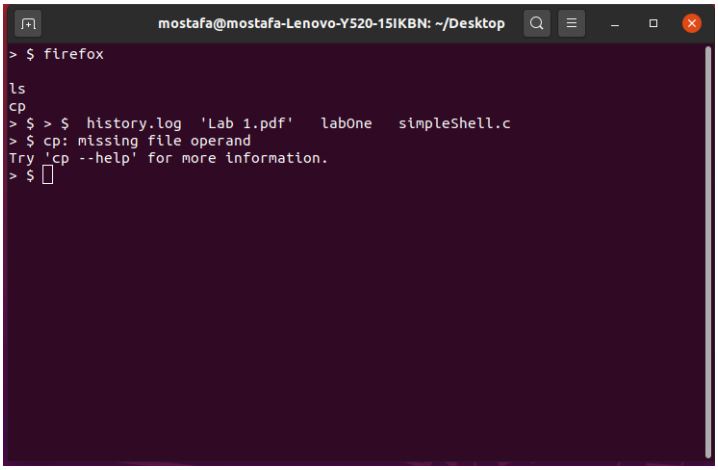
Entering invalid command to receive Error “Command not found”.

Sample Runs

Commands stopping other processes:



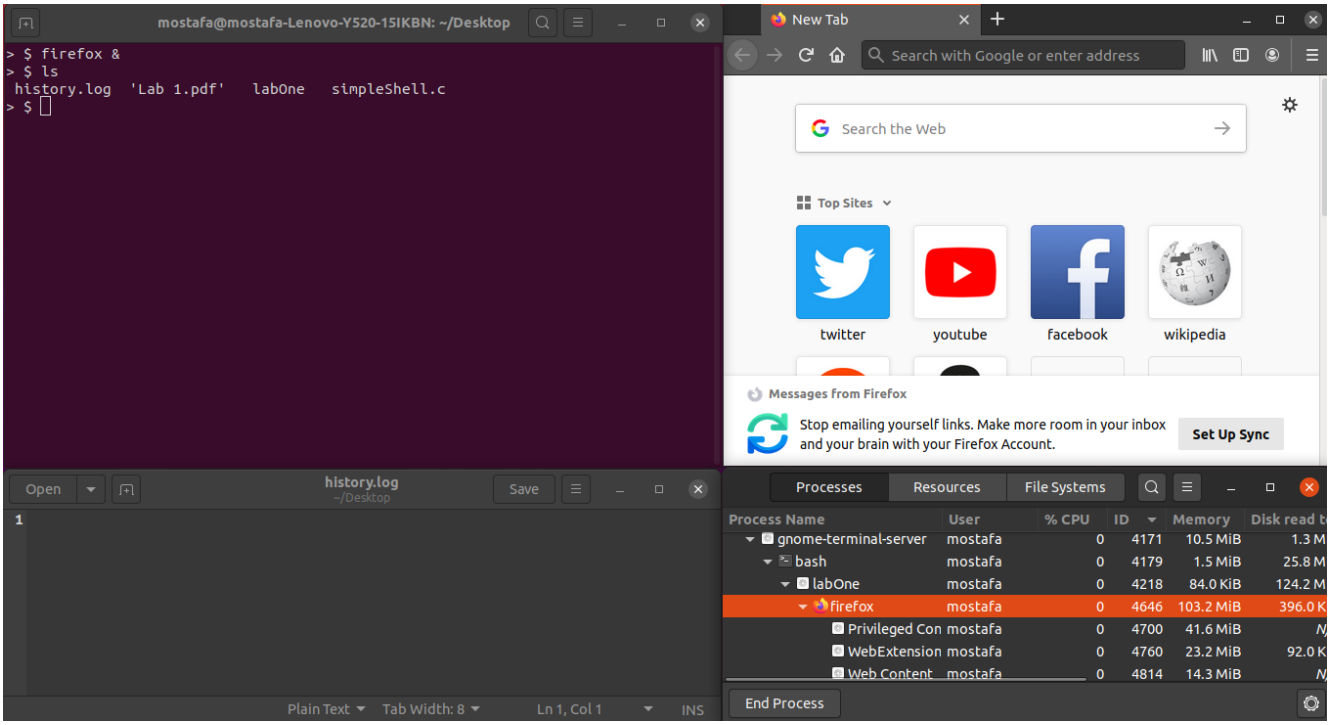
Firefox was process was born after command “Firefox” was called. It’s run in the forgeound. Thus, all next commands are put in the queue waiting for the first process to terminate.
Even though “ls” and “cp” are called, no action occurred because firefox is still runinng.



As soon as firefox process was terminated, all pending operations in queue, “ls” and “cp”, take action.

Sample Runs

Commands running in background:



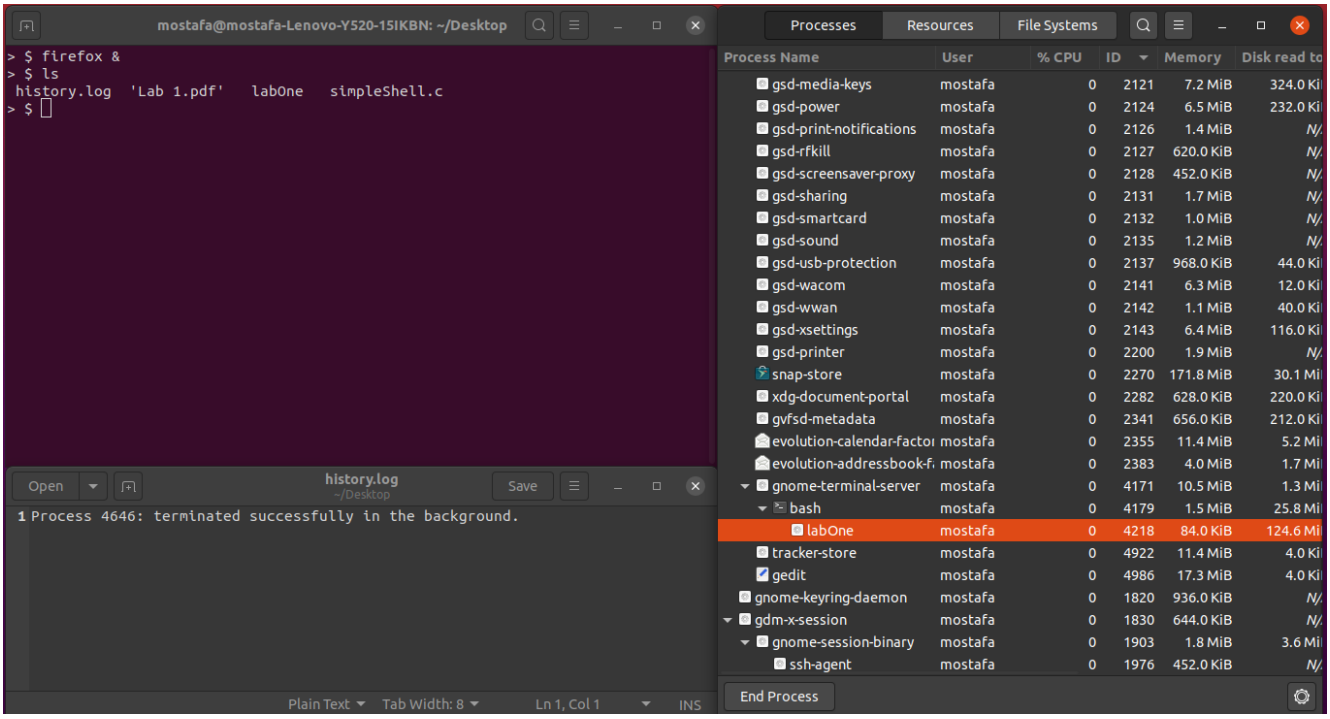
Adding an “&” to the end of the “firefox” command forces it to run in the background allowing all next processes to take action even though the firefox process isn’t terminated yet.

Firefox process (ID = 4646) is running in the background as a child of the UNIX shell.

The history log is empty because no processes in the background were terminated.

Sample Runs

Terminating process in background:

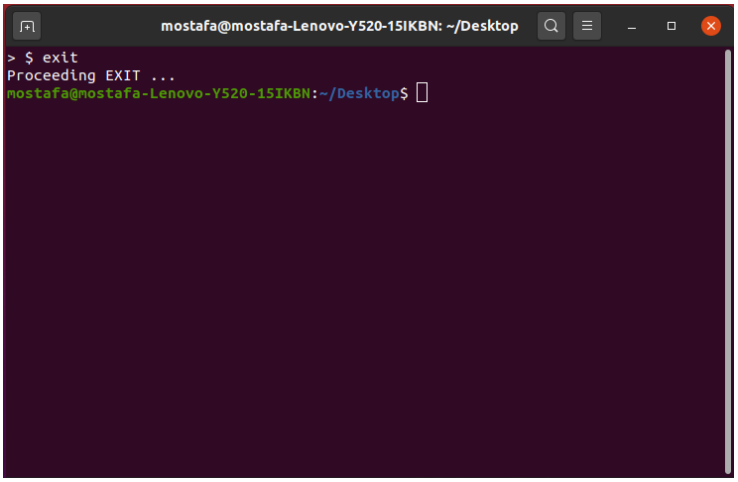


Firefox process was terminated in the background and its parent process has no childs anymore.

The UNIX shell won't mark any changes, but the terminated process sends a signal to add note in the history log that it was terminated.

Sample Runs

Exiting shell:



EXIT command terminates the shell and return to original terminal.