Alexandria University

Faculty of Engineering

Electronics and Communication Department

CS 333 – Operating Systems

OPERATING SYSTEMS

Assignment One Report
Simple Shell

Submitted by:

- Aya Abdelaziz Mohamed Abdelaziz	**ID: 7**
- Ahmed Adel Sultan Saad	**ID: 24**
- Omar Samy Megahed Ahmed	**ID: 135**
- Amr Sherif Elsayed Ghazy	**ID: 144**
- Fatma Amr Abdul-Mo'men Amer	**ID: 151**

Contents

1 Overview:

The shell is a program that takes commands from the keyboard and gives them to the operating system to perform. In the old days, it was the only user interface available on a Unix-like system such as Linux. Nowadays, we have graphical user interfaces (GUIs) in addition to command line interfaces (CLIs) such as the shell.

2 Code Description:

The code starts with taking user commands through the prompt until they choose to exit or press CTRL+D. The input line undergoes several processes including: the removal of the leading spaces from each input, and calculation of the line length so that the system knows whether to ask again for an input line or continues to the next stages. The line is then split into an array of strings containing the command, the first command line argument, and any other arguments.

The first element in the array is compared to the command “cd” since we need to use *chdir()* function and not *execvp()* for this particular command.

Each command can be executed in background or foreground. In case of background execution, the parent process, in this case the shell, is not required to wait for the child process to terminate while in case of foreground execution the parent process has to wait for the child process.

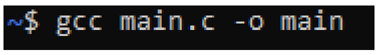
On each child process termination, the main application will be interrupted by a signal where a log file will be updated with a line indicating the termination of a child process. The number of appended lines in the log file is the number of child processes terminations.

If the user input was an undefined command, the user is alarmed by a printed sentence that the command is not found.

3 How to run the program:

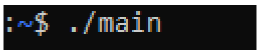
1. Open the terminal in the project folder.

2. Compile the program using the command: “gcc main.c -o main”.



```
~$ gcc main.c -o main
```

3. Run the program using the command “./main”.



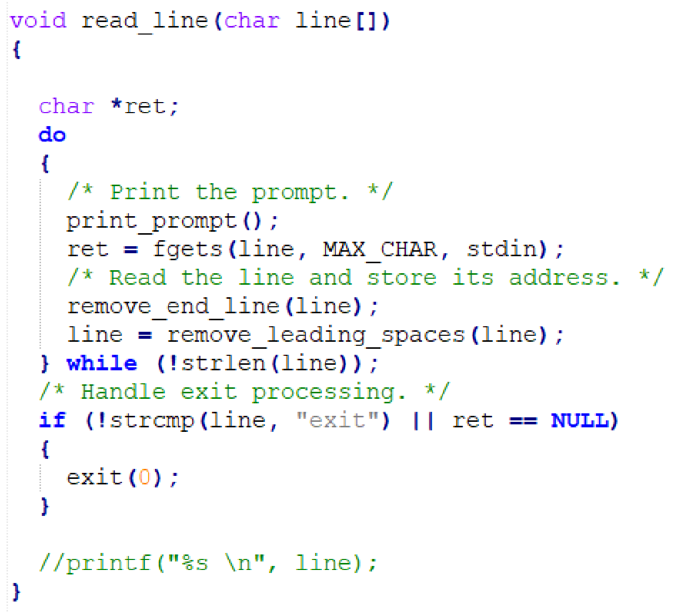
```
:~$ ./main
```

4 Major Functions:

4.1 read_line():

read_line() function is responsible for printing the prompt using the function *print_prompt()* and reading the input line and perform several changes on the line including:

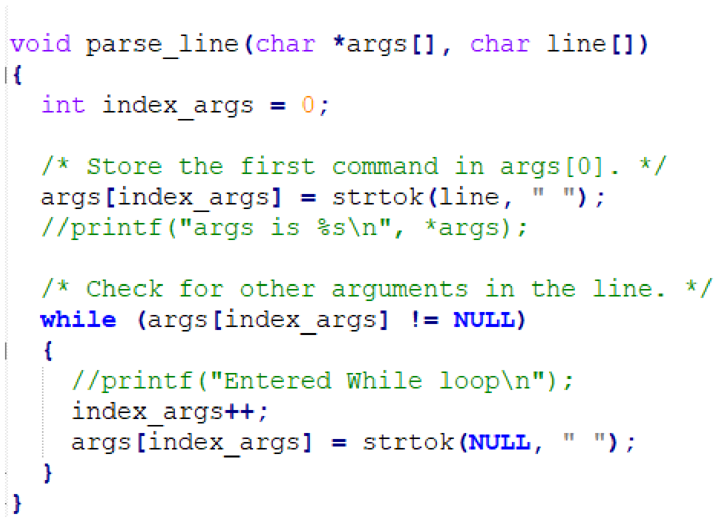
- Removing the end line using the function *remove_end_line()*.
- Removing the leading spaces using the function *remove_leading_spaces()*.



```
void read_line(char line[])
{

  char *ret;
  do
  {
    /* Print the prompt. */
    print_prompt();
    ret = fgets(line, MAX_CHAR, stdin);
    /* Read the line and store its address. */
    remove_end_line(line);
    line = remove_leading_spaces(line);
  } while (!strlen(line));
  /* Handle exit processing. */
  if (!strcmp(line, "exit") || ret == NULL)
  {
    exit(0);
  }

  //printf("%s \n", line);
}
```

4.2 parse_line():

parse_line() function is responsible for parsing the input line into an array of strings so each command line argument will be stored in the array.



```
void parse_line(char *args[], char line[])
{
  int index_args = 0;

  /* Store the first command in args[0]. */
  args[index_args] = strtok(line, " ");
  //printf("args is %s\n", *args);

  /* Check for other arguments in the line. */
  while (args[index_args] != NULL)
  {
    //printf("Entered While loop\n");
    index_args++;
    args[index_args] = strtok(NULL, " ");
  }
}
```

4.3 execute():

execute() function is responsible for the birth of a new child that will execute the command entered by the user using *fork()* function.

```c
void execute(char *command, char *args[])
{
  int background = is_background(args);
  // Make a new process.
  int child_pid = fork();
  if (child_pid == 0)
  {
    // In the child process execute the entered command.
    if (execvp(args[0], args) == -1)
    {
      printf(ANSI_COLOR_RED "Command not found.\n");
    }
  }
  else
  {
    // In the parent process wait till the child process is finished.
    if (!background)
    {
      waitpid(child_pid, NULL, 0);
    }
  }
}
```

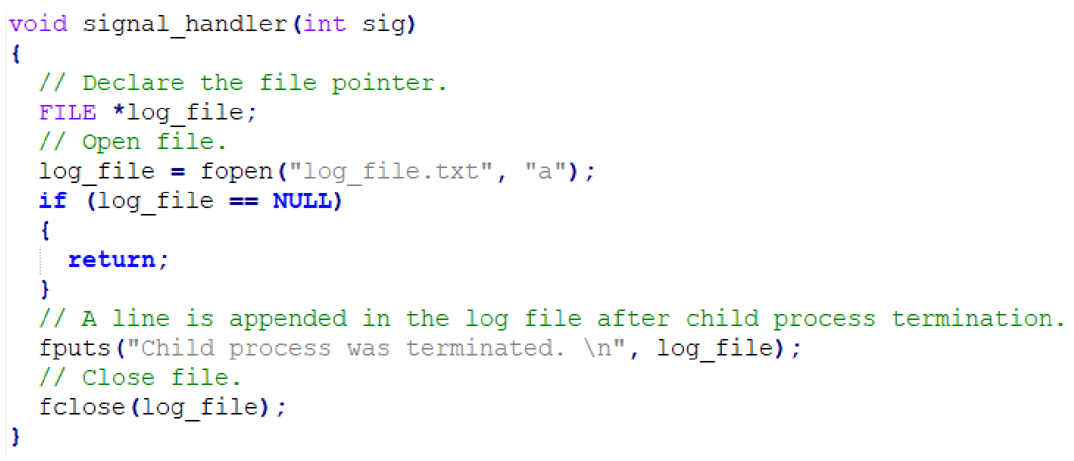
4.4 is_background():

is_background() function is responsible for determining whether the execution is in background or foreground.

```c
int is_background(char *args[])
{
  int background = 0;
  int argsLength = 0;
  while (args[argsLength] != NULL)
  {
    argsLength++;
  }
  if (strcmp(args[argsLength - 1], "&") == 0)
  {
    background = 1;
  }
  return background;
}
```

4.5 signal_handler():

signal_handler() function is responsible for the interrupt signal handling where a log file is updated each time a child process is terminated.

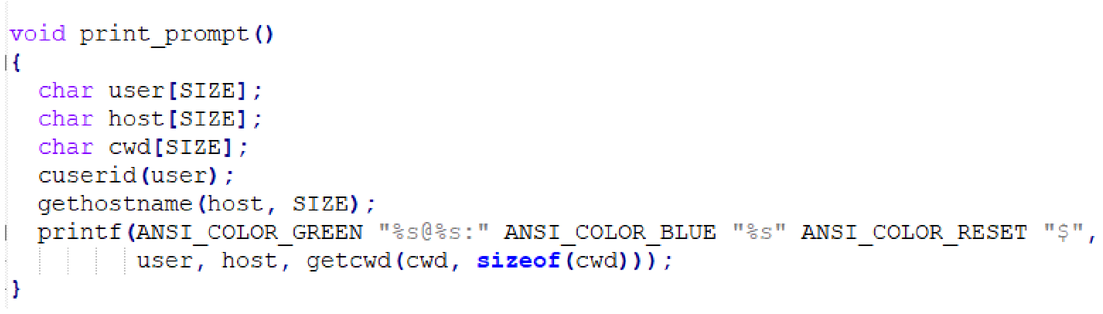


```
void signal_handler(int sig)
{
  // Declare the file pointer.
  FILE *log_file;
  // Open file.
  log_file = fopen("log_file.txt", "a");
  if (log_file == NULL)
  {
    return;
  }
  // A line is appended in the log file after child process termination.
  fputs("Child process was terminated. \n", log_file);
  // Close file.
  fclose(log_file);
}
```

5 Associated Functions:

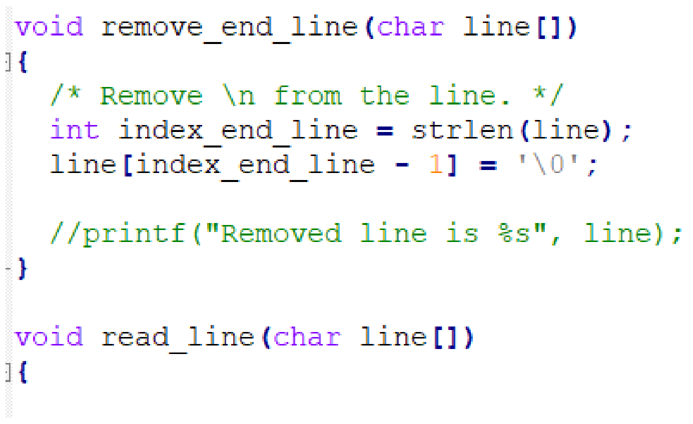
Those functions are used to meet some needs inside the major functions.

5.1 print_prompt():



```
void print_prompt()
{
  char user[SIZE];
  char host[SIZE];
  char cwd[SIZE];
  cuserid(user);
  gethostname(host, SIZE);
  printf(ANSI_COLOR_GREEN "%s@%s:" ANSI_COLOR_BLUE "%s" ANSI_COLOR_RESET "$",
         user, host, getcwd(cwd, sizeof(cwd)));
}
```

5.2 remove_end_line():



```
void remove_end_line(char line[])
{
  /* Remove \n from the line. */
  int index_end_line = strlen(line);
  line[index_end_line - 1] = '\0';

  //printf("Removed line is %s", line);
}

void read_line(char line[])
{
```

5.3 remove_leading_spaces():

```
char *remove_leading_spaces(char *str)
{
  static char new_str[MAX_CHAR];
  int count = 0, j, k;

  // Iterate String until last
  // leading space character
  while (str[count] == ' ')
  {
    count++;
  }

  // Putting string into another
  // string variable after
  // removing leading white spaces
  for (j = count, k = 0;
       str[j] != '\0'; j++, k++)
  {
    new_str[k] = str[j];
  }
  new_str[k] = '\0';
  return new_str;
}
```

5.4 read_parse_line():

```
int read_parse_line(char *args[], char line[])
{
  /* Read the line. */
  read_line(line);

  /* Parse the line. */
  parse_line(args, line);

  return 1;
}
```

6 Main Function:

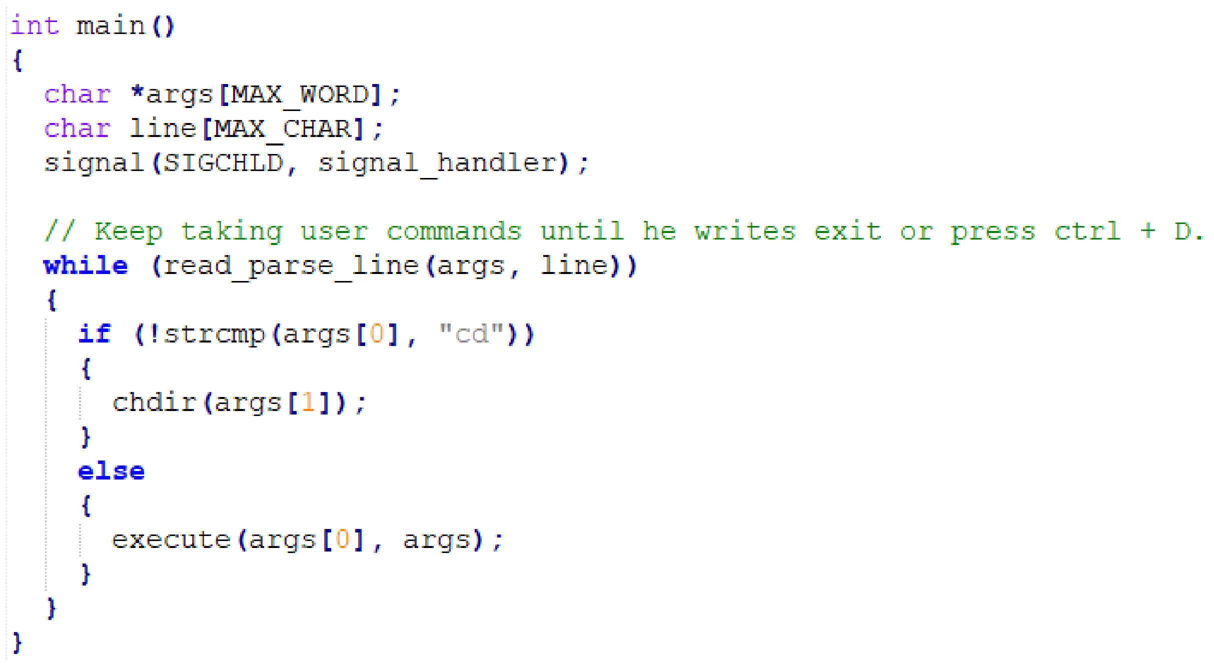
- In the main function, the program is executed in an infinite loop, it is halted only if the user pressed CTRL + D or entered “exit”.

- If condition is used to handle “cd” command, as if the user entered cd, chdir() function is called instead of execute().

- The line entered is stored in an array of characters “string” which has a maximum number of characters defined in the beginning of the program.

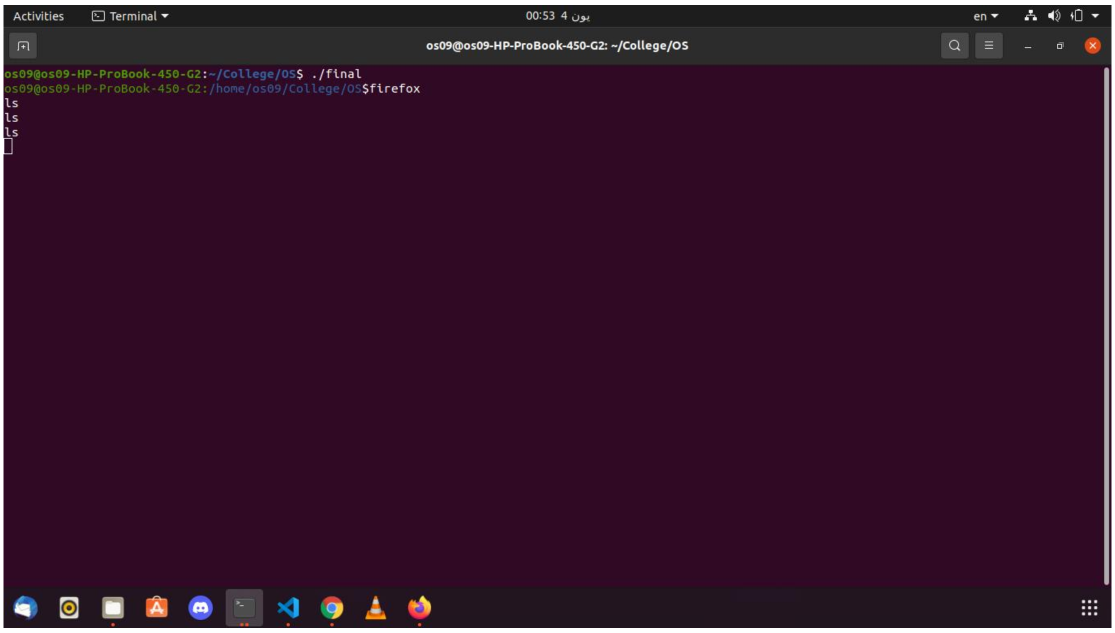
- The arguments which are inside the entered line are stored in an array of strings which has a maximum number of arguments defined in the beginning of the program.

- For the arguments, we should use a pointer to a string to be able to implement a string of arrays.



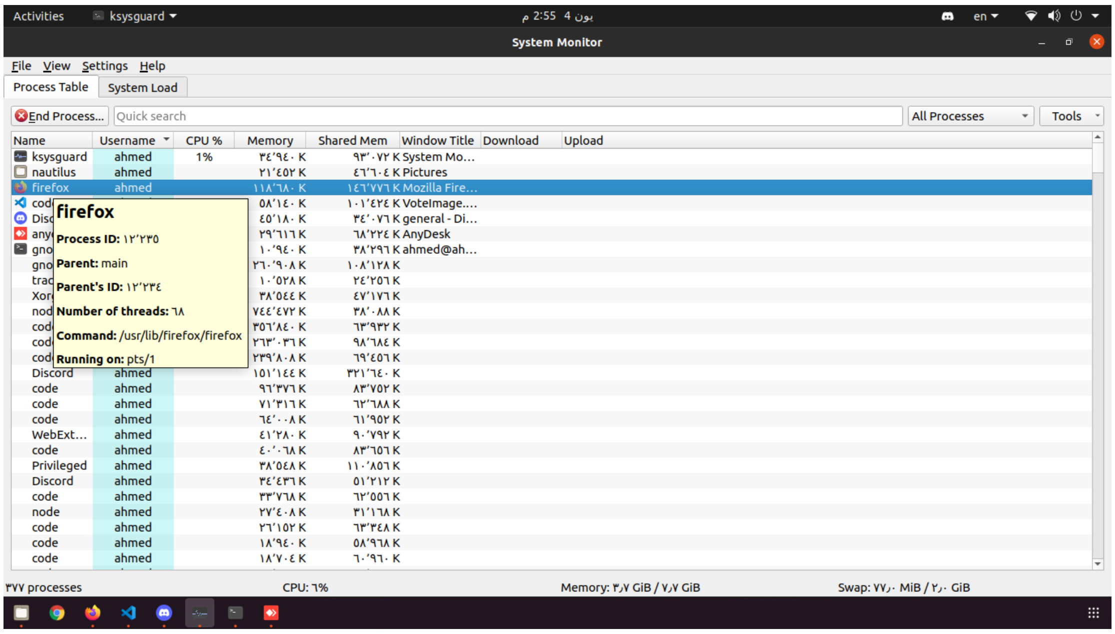
```
int main()
{
  char *args[MAX_WORD];
  char line[MAX_CHAR];
  signal(SIGCHLD, signal_handler);

  // Keep taking user commands until he writes exit or press ctrl + D.
  while (read_parse_line(args, line))
  {
    if (!strcmp(args[0], "cd"))
    {
      chdir(args[1]);
    }
    else
    {
      execute(args[0], args);
    }
  }
}
```

7 Screenshots of Sample Runs:



```
os09@os09-HP-ProBook-450-G2:~/College/OS$ ./final
os09@os09-HP-ProBook-450-G2:/home/os09/College/OS$firefox
ls
ls
ls
```

Firefox running in foreground so parent process must wait for child process to terminate to execute other commands.



Name	Username	CPU %	Memory	Shared Mem	Window Title	Download	Upload
ksysguard	ahmed	1%	٣٤٬٩٤٠ K	٩٣٬٠٧٢ K	System Mo…		
nautilus	ahmed		٢١٬٤٥٢ K	٤٦٬٦٠٤ K	Pictures		
firefox	ahmed		١١٨٬٦٨٠ K	١٤٦٬٧٧٦ K	Mozilla Fire…		
cod			٥٨٬١٤٠ K	١٠١٬٤٢٤ K	VoteImage.…		
Disc			٤٥٬١٨٠ K	٣٤٬٠٧٦ K	general - Di…		
any			٢٩٬٦١٦ K	٦٨٬٢٢٤ K	AnyDesk		
gno			١٠٬٩٤٠ K	٣٨٬٢٩٦ K	ahmed@ah…		
gno			٢٦٠٬٩٠٨ K	١٠٨٬١٢٨ K			
trac			١٠٬٥٢٨ K	٢٤٬٢٥٦ K			
Xor			٣٨٬٥٤٤ K	٤٧٬١٧٦ K			
nod			٧٤٤٬٤٧٢ K	٣٨٬٠٨٨ K			
cod			٣٥٦٬٨٤٠ K	٦٣٬٩٣٢ K			
cod			٢٦٣٬٠٣٦ K	٩٨٬٦٨٤ K			
cod			٢٣٩٬٨٠٨ K	٦٩٬٤٥٦ K			
Discord	ahmed		١٥١٬١٤٤ K	٣٢١٬٦٤٠ K			
code	ahmed		٩٦٬٣٧٦ K	٨٣٬٧٥٢ K			
code	ahmed		٧١٬٣١٦ K	٦٢٬٦٨٨ K			
code	ahmed		٦٤٬٠٠٨ K	٦١٬٩٥٢ K			
WebExt…	ahmed		٤١٬٢٨٠ K	٩٠٬٧٩٢ K			
code	ahmed		٤٠٬٠٦٨ K	٨٣٬٦٥٦ K			
Privileged	ahmed		٣٨٬٥٤٨ K	١١٠٬٨٥٦ K			
Discord	ahmed		٣٤٬٤٣٦ K	٥١٬٢١٢ K			
code	ahmed		٣٣٬٧٦٨ K	٦٢٬٥٥٦ K			
node	ahmed		٢٧٬٤٠٨ K	٣١٬١٦٨ K			
code	ahmed		٢٦٬١٥٢ K	٦٣٬٣٤٨ K			
code	ahmed		١٨٬٩٤٠ K	٥٨٬٩٦٨ K			
code	ahmed		١٨٬٧٠٤ K	٦٠٬٩٦٠ K			

firefox

Process ID: ١٢٬٢٣٥

Parent: main

Parent's ID: ١٢٬٢٣٤

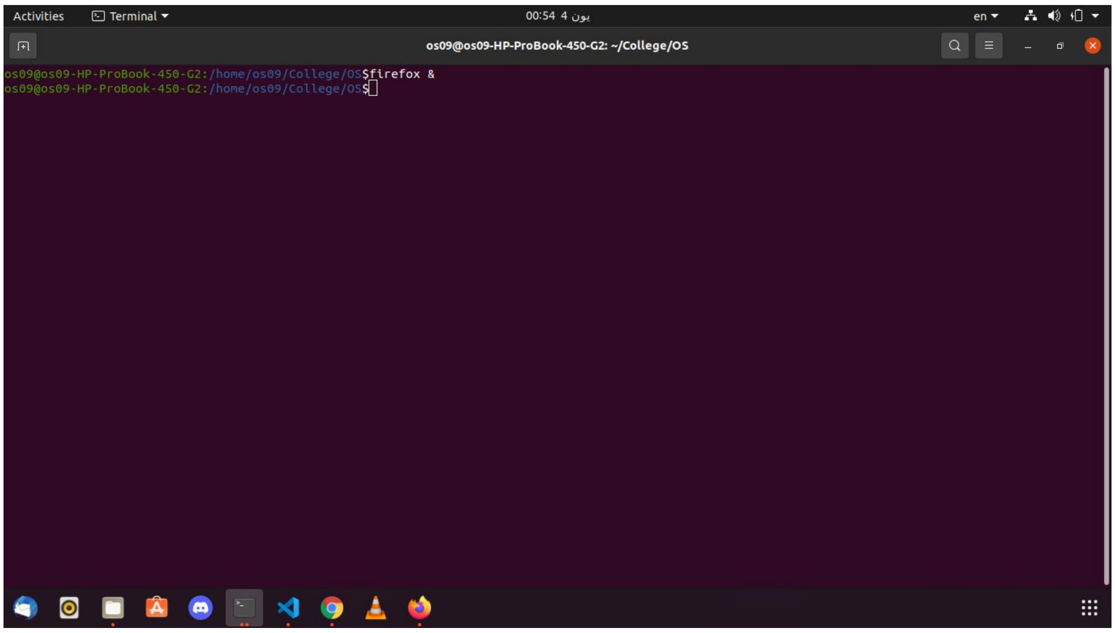
Number of threads: ٦٨

Command: /usr/lib/firefox/firefox

Running on: pts/1

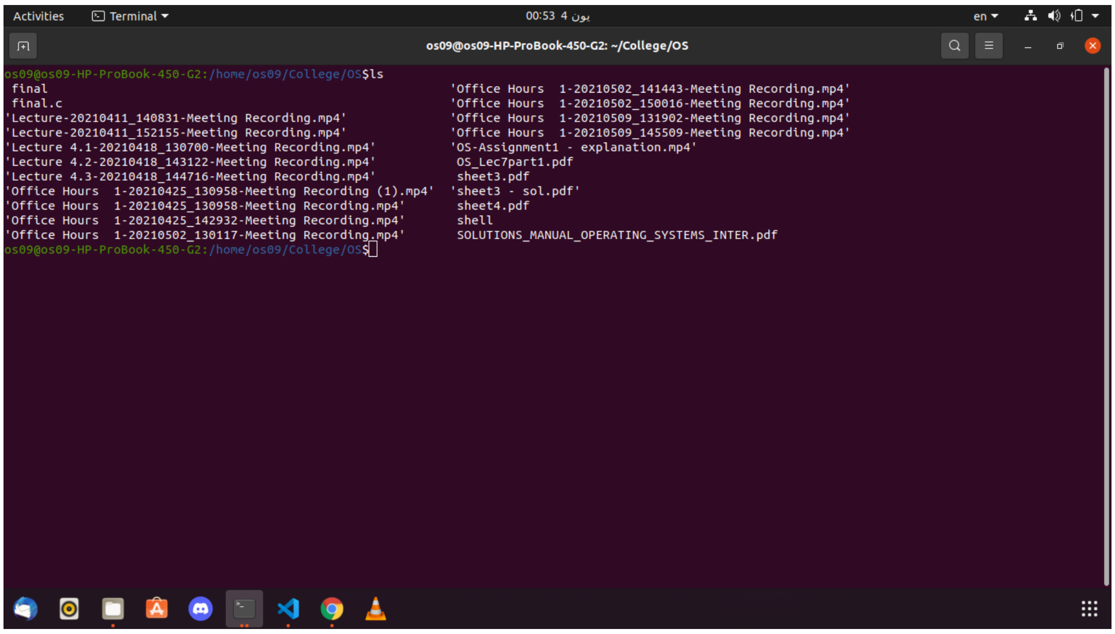
٣٧٧ processes   CPU: ٦%   Memory: ٣٫٧ GiB / ٧٫٧ GiB   Swap: ٧٧٫٠ MiB / ٢٫٠ GiB

The parent of this process is 'main' which is our shell.



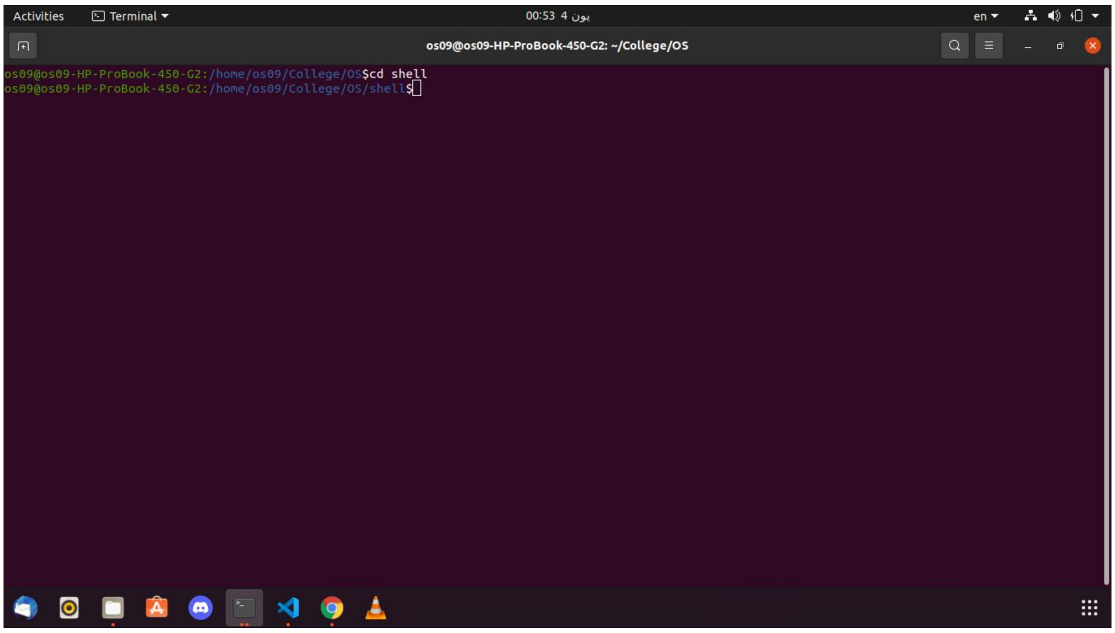
```
os09@os09-HP-ProBook-450-G2:/home/os09/College/OS$firefox &
os09@os09-HP-ProBook-450-G2:/home/os09/College/OS$
```

Running Firefox in background.



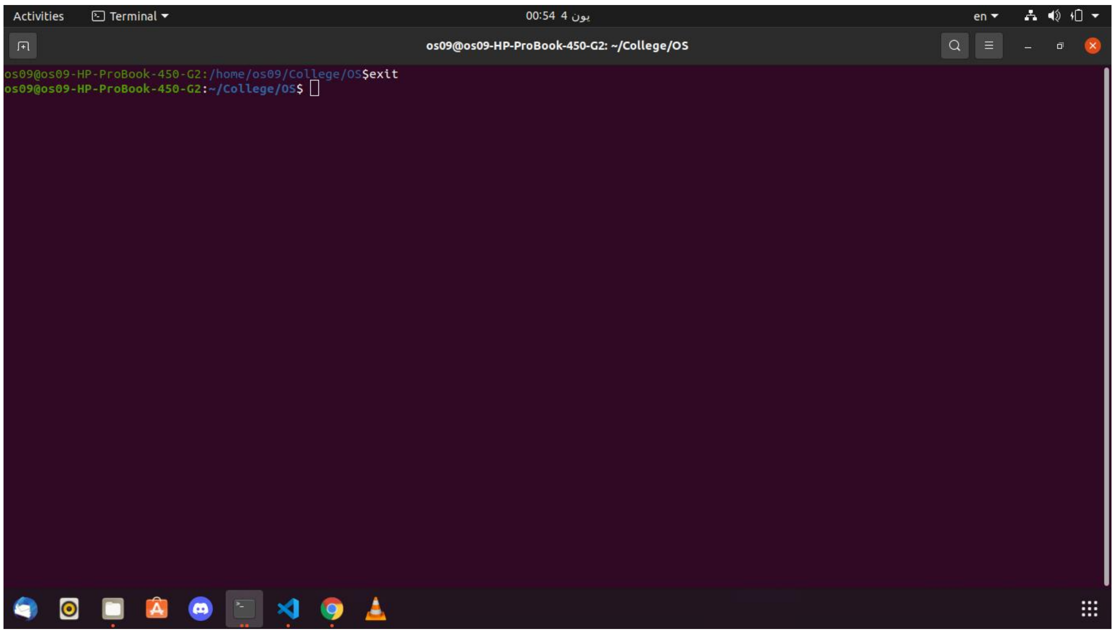
```
os09@os09-HP-ProBook-450-G2:/home/os09/College/OS$ls
 final                                                       'Office Hours  1-20210502_141443-Meeting Recording.mp4'
 final.c                                                     'Office Hours  1-20210502_150016-Meeting Recording.mp4'
'Lecture-20210411_140831-Meeting Recording.mp4'              'Office Hours  1-20210509_131902-Meeting Recording.mp4'
'Lecture-20210411_152155-Meeting Recording.mp4'              'Office Hours  1-20210509_145509-Meeting Recording.mp4'
'Lecture 4.1-20210418_130700-Meeting Recording.mp4'          'OS-Assignment1 - explanation.mp4'
'Lecture 4.2-20210418_143122-Meeting Recording.mp4'           OS_Lec7part1.pdf
'Lecture 4.3-20210418_144716-Meeting Recording.mp4'           sheet3.pdf
'Office Hours  1-20210425_130958-Meeting Recording (1).mp4'  'sheet3 - sol.pdf'
'Office Hours  1-20210425_130958-Meeting Recording.mp4'       sheet4.pdf
'Office Hours  1-20210425_142932-Meeting Recording.mp4'       shell
'Office Hours  1-20210502_130117-Meeting Recording.mp4'       SOLUTIONS_MANUAL_OPERATING_SYSTEMS_INTER.pdf
os09@os09-HP-ProBook-450-G2:/home/os09/College/OS$
```

Executing 'ls' command.



```
os09@os09-HP-ProBook-450-G2:/home/os09/College/OS$cd shell
os09@os09-HP-ProBook-450-G2:/home/os09/College/OS/shell$
```

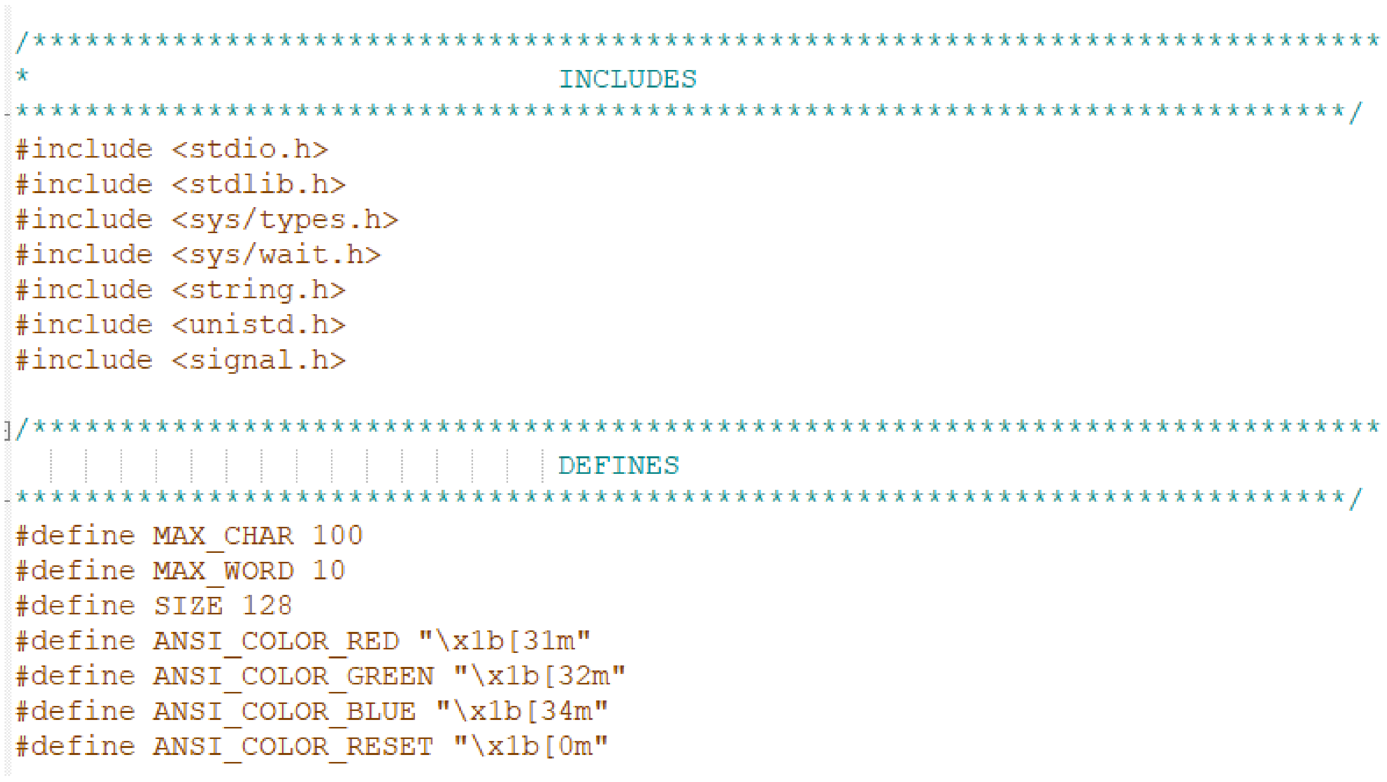
Executing 'cd' command.



```
os09@os09-HP-ProBook-450-G2:/home/os09/College/OS$exit
os09@os09-HP-ProBook-450-G2:~/College/OS$ 
```

Executing exit command.

8 Defines and Includes:



```
/*****************************************************************************
*                              INCLUDES
*****************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <string.h>
#include <unistd.h>
#include <signal.h>

/*****************************************************************************
                               DEFINES
*****************************************************************************/
#define MAX_CHAR 100
#define MAX_WORD 10
#define SIZE 128
#define ANSI_COLOR_RED "\x1b[31m"
#define ANSI_COLOR_GREEN "\x1b[32m"
#define ANSI_COLOR_BLUE "\x1b[34m"
#define ANSI_COLOR_RESET "\x1b[0m"
```