MINI PROJECT
[COM-312]

PRESENTED BY

Saif Ullah(2021A1R093)

Amit Bali(2021A1R107)

Anuj Raina(2022A1L006)

Isha (2022A1L018)

SUBMITTED TO

Miss. Pragati Jamwal

TABLE OF CONTENTS

Build your own Linux shell and execute a simple (ls-l command)

INTRODUCTION

In this project we have to develop a fully functional Linux Shell that can parse and execute a fairly complex set of commands , loops ,and expressions . Linux shell manages to parse and execute commands by actually writing the code .

LINUX SHELL

The shell can be defined as a command interpreter within an operating system like Linux/GNU or Unix . It is a program that runs other programs. The shell facilitates every user of the computer as an interface to the Unix/GNU Linux system

FOR DEVELOPING SHELL

The shell is a complex piece of software that contains many different parts. The core part of any Linux shell is the Command Line Interpreter, or CLI. This part serves two purposes: it reads and parses user commands, then it executes the parsed commands. You can think of the CLI itself as having two parts: a parser (or front-end), and an executor (or back-end).

The parser scans input and breaks it down to tokens. A token consists of one or more characters (letters, digits, symbols), and represents a single unit of input. For example, a token can be a variable name, a keyword, a number, or an arithmetic operator.

The parser takes these tokens, groups them together, and creates a special structure we call the Abstract Syntax Tree, or AST. You can think of the AST as a high-level representation of the command line you gave to the shell. The parser takes the AST and passes it to the executor, which reads the AST and executes the parsed command

SIMPLE COMMAND

A simple command consists of a list of words, separated by whitespace characters (space, tab, newline). The first word is the command name, and is mandatory (otherwise, the shell won't have a command to parse and execute!). The second and subsequent words are optional. If present, they form the arguments we want the shell to pass to the executed command.

CODE TO BUILD A LINUX SHELL

```
#include <stdlib.h>
#include <string.h>
#include "shell.h"

int main(int argc, char **argv)
{
    char *cmd;

    do
    {
        print_prompt1();

        cmd = read_cmd();

        if(!cmd)
        {
            exit(EXIT_SUCCESS);
        }
```

CONTINUE

```
if(cmd[0] == '\0' ||
        {
            free(cmd);
            continue;
        }

        if(strcmp(cmd, "exit\n") == 0)
        {
            free(cmd);
            break;
        }

        printf("%s\n", cmd);

        free(cmd);

    } while(1);

    exit(EXIT_SUCCESS);
  }
```

Now create the source file prompt.c

```
#include <stdio.h>
#include "shell.h"

void print_prompt1(void)
{
    fprint(stderr, "$ ");
}

void print_prompt2(void)
{
    fprintf(stderr, "> ");
}
```

```
rint_prompt1(void); void print_prompt2(void); char *read_cmd(void); #endif
```

CONTINUE

```
echo "This is a very long line of input,
\
      one that needs to span two, three,
\
      or perhaps even more lines of
input, \
      so that we can feed it to the
shell"
```

CONTINUE

```
define SHELL_H

void print_prompt1(void);
void print_prompt2(void);

char *read_cmd(void);

#endif
```

OUTPUTS

```
mima@DaBossLaptop:~$ cd projects/sh
mima@DaBossLaptop:~/projects/sh$ ls
main.c  prompt.c  shell.h
mima@DaBossLaptop:~/projects/sh$ gcc -o shell main.c prompt.c
mima@DaBossLaptop:~/projects/sh$ ls
main.c  prompt.c  shell  shell.h
mima@DaBossLaptop:~/projects/sh$
```

```
mima@DaBossLaptop:~$ cd projects/sh
mima@DaBossLaptop:~/projects/sh$ ls
main.c  prompt.c  shell.h
mima@DaBossLaptop:~/projects/sh$ gcc -o shell main.c prompt.c
mima@DaBossLaptop:~/projects/sh$ ls
main.c  prompt.c  shell  shell.h
mima@DaBossLaptop:~/projects/sh$ ./shell
$ echo Hello World
echo Hello World

$ echo "This is a very long line of input, one that needs to span two, three, or
 perhaps even more lines of input, so that we can feed it to the shell"
echo "This is a very long line of input, one that needs to span two, three, or p
erhaps even more lines of input, so that we can feed it to the shell"

$ echo "This is a very long line of input, \
> one that needs to span two, three, \
> or perhaps even more lines of input, \
> so that we can feed it to the shell"
echo "This is a very long line of input, one that needs to span two, three, or p
erhaps even more lines of input, so that we can feed it to the shell"

$
```

OUTPUT

```
mima@DaBossLaptop:~$ cd projects/sh
mima@DaBossLaptop:~/projects/sh$ ls
main.c  prompt.c  shell.h
mima@DaBossLaptop:~/projects/sh$ gcc -o shell main.c prompt.c
mima@DaBossLaptop:~/projects/sh$ ls
main.c  prompt.c  shell  shell.h
mima@DaBossLaptop:~/projects/sh$ ./shell
$ echo Hello World
echo Hello World

$ echo "This is a very long line of input, one that needs to span two, three, or
 perhaps even more lines of input, so that we can feed it to the shell"
echo "This is a very long line of input, one that needs to span two, three, or p
erhaps even more lines of input, so that we can feed it to the shell"

$ echo "This is a very long line of input, \
> one that needs to span two, three, \
> or perhaps even more lines of input, \
> so that we can feed it to the shell"
echo "This is a very long line of input, one that needs to span two, three, or p
erhaps even more lines of input, so that we can feed it to the shell"

$ exit
mima@DaBossLaptop:~/projects/sh$
```

- ***THANK YOU***