

# OPERATING SYSTEM LAB (CSE 3164)

## MINI PROJECT

PROJECT TITLE: UNIX SHELL

TEAM MEMBERS:

1. Isha Sthanpati (200905258)
2. Aaryan Gupta (200905276)
3. Daksh Soni (200905206)
4. Kopal Dixit (200905244)
5. Ritik Nagpal (200905239)

TOPIC: Handling Linux Shell Commands

SUMMARY:

Created a basic Unix Shell in C Language.

FILE	DESCRIPTION
builtin.h	Contains code related to shell builtin commands
handler.h	Contains code related to handling builtin commands and system commands
config.h	Contains code related to general configuration of the shell
parser.h	Contains code related to parsing the commands entered by the user
prompt.h	Contains code related to the shell command prompt
util.h	Contains some useful helper functions
shell.c	Contains main() function

Shell Builtin Commands implemented-

COMMAND	DESCRIPTION
cd	Changes the current working directory
showpid	Prints out the PIDs of the last 10 processes created by the running shell process
exit	Causes normal process termination

The currently active shell process starts a new process for every other command (not mentioned above). `fork()` and `execvp` are mostly utilised in the execution of external commands, and `waitpid()` was employed to prevent zombie processes (dead processes).

### About the commands:

#### Change Directory Command (`cd`):

This command is used to change the current working directory.

Syntax:

```
$ cd [directory]
```

To move inside a subdirectory we use:

```
$ cd [directory-name]
```

Different functionalities of `cd` command:

**`cd /`**: this command is used to change directory to the root directory, The root directory is the first directory in your filesystem hierarchy.

Above, `/` represents the root directory.

**`cd dir_1/dir_2/dir_3`**: This command is used to move inside a directory from a directory

**`cd ~`**: this command is used to change directory to the home directory.

**`cd ..`**: this command is used to move to the parent directory of current directory, or the directory one level up from the current directory.

**`cd "dir name"`**: This command is used to navigate to a directory with white spaces.

#### Showpid command:

Every app, service, or process has an assigned number known as the Process ID.

The Process ID (or PID) is used to identify each running process or suspended process within a system.

Knowing an app's PID helps you identify programs running multiple instances, such as when editing two different files using the same app. Also, the PID helps you when you need to terminate a process manually or if you want to check system resources consumed by a certain process.

Showpid command does this job by getting the current process id and to check whether the process is running or suspended.

#### Exit command:

**`exit`** command in linux is used to exit the shell where it is currently running. It takes one more parameter as `[N]` and exits the shell with a return of status `N`. If `n` is not provided, then it simply returns the status of last command that is executed.

Syntax:

```
$ exit [n]
```

Options for exit:

**`exit`**: Exit Without Parameter

After pressing enter, the terminal will simply close.

**`exit [n]`**: Exit With Parameter

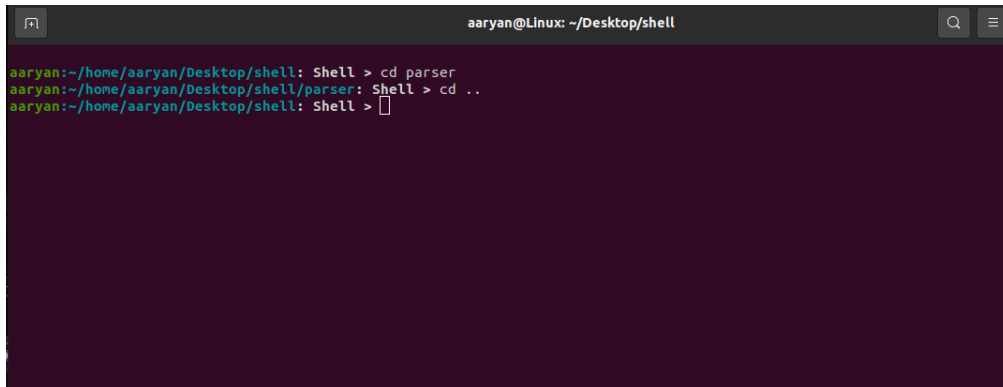
After pressing enter, the terminal window will close and return a status of 110.

"0" means the program has executed successfully.

"1" means the program has minor errors.

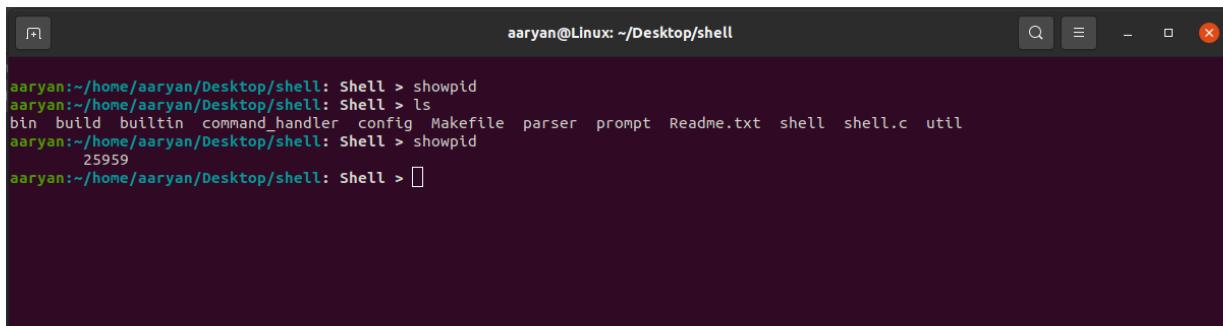
## OUTPUT:

Changing directories-

A terminal window titled 'aaryan@Linux: ~/Desktop/shell' with a dark purple background. It shows a sequence of commands and their outputs: 'cd parser' changes the directory to 'parser', 'cd ..' returns to the parent directory, and the prompt returns to 'Shell >'.

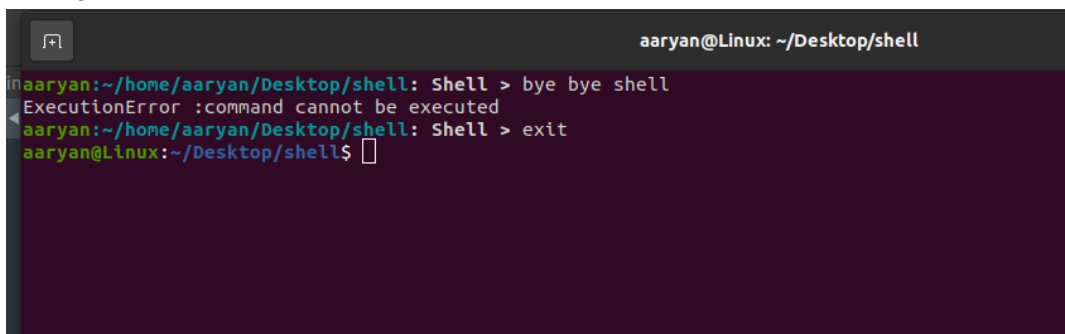
```
aaryan@Linux: ~/Desktop/shell
aaryan:~/home/aaryan/Desktop/shell: Shell > cd parser
aaryan:~/home/aaryan/Desktop/shell/parser: Shell > cd ..
aaryan:~/home/aaryan/Desktop/shell: Shell > 
```

Showing process ids:

A terminal window titled 'aaryan@Linux: ~/Desktop/shell' with a dark purple background. It shows the 'showpid' command being used to display the process IDs of various files in the current directory. The output lists files like 'bin', 'build', 'builtin', etc., along with their corresponding PIDs.

```
aaryan@Linux: ~/Desktop/shell
aaryan:~/home/aaryan/Desktop/shell: Shell > showpid
aaryan:~/home/aaryan/Desktop/shell: Shell > ls
bin build builtin command_handler config Makefile parser prompt Readme.txt shell shell.c util
aaryan:~/home/aaryan/Desktop/shell: Shell > showpid
25959
aaryan:~/home/aaryan/Desktop/shell: Shell > 
```

Exiting shell-

A terminal window titled 'aaryan@Linux: ~/Desktop/shell' with a dark purple background. It shows the user typing 'bye bye shell' and 'exit'. The first command results in an 'ExecutionError :command cannot be executed' message, while the second command successfully exits the shell, returning to the 'aaryan@Linux' prompt.

```
aaryan@Linux: ~/Desktop/shell
aaryan:~/home/aaryan/Desktop/shell: Shell > bye bye shell
ExecutionError :command cannot be executed
aaryan:~/home/aaryan/Desktop/shell: Shell > exit
aaryan@Linux:~/Desktop/shell$ 
```

## OUTCOME:

We learnt how system calls work in Linux, and the internal working of these commands and the shell using C. Along with this, we learnt how to use GitHub while we communicated with each other. We understood in depth the working of functions like `fork()`, `exec()`, etc.

## CONCLUSION:

Before exploring and comprehending the inner workings of it all, features like terminals and shells appear to the untrained eye—and even to computer science students—to be

extremely difficult. These are the fundamental components of a terminal and shell, including system calls and the comprehension and categorization of statements using string matching. The following conclusions can be made once this project was successfully planned and carried out:

1. For versatility and efficiency, shells must be very modular.
2. Creating basic components from complicated characteristics is relatively straightforward in C.