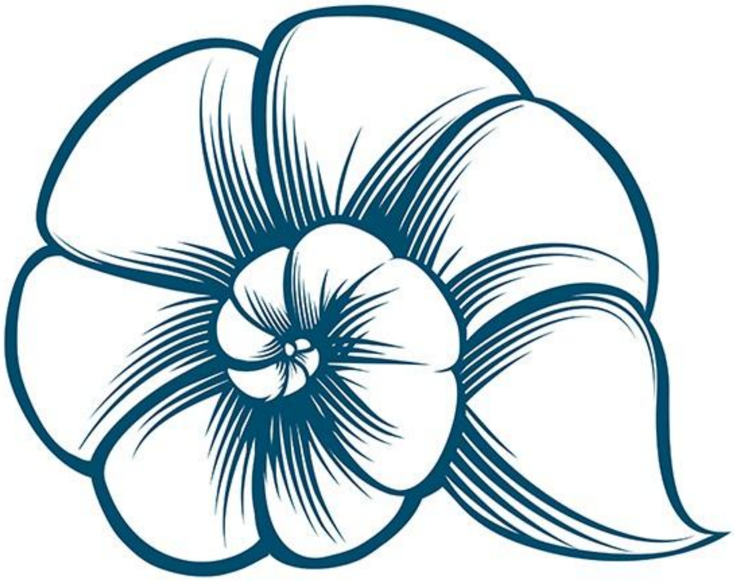
# [Simple Shell]

[Ahmad Mahmoud Elreffaiy]    [4534]



## [About the program]

Let’s look at a shell from the top down. A shell does three main things in its lifetime.

- **Initialize**: In this step, a typical shell would read and execute its configuration files. These change aspects of the shell’s behavior.

- **Interpret**: Next, the shell reads commands from stdin (which could be interactive, or a file) and executes them.

- **Terminate**: After its commands are executed, the shell executes any shutdown commands, frees up any memory, and terminates.

-

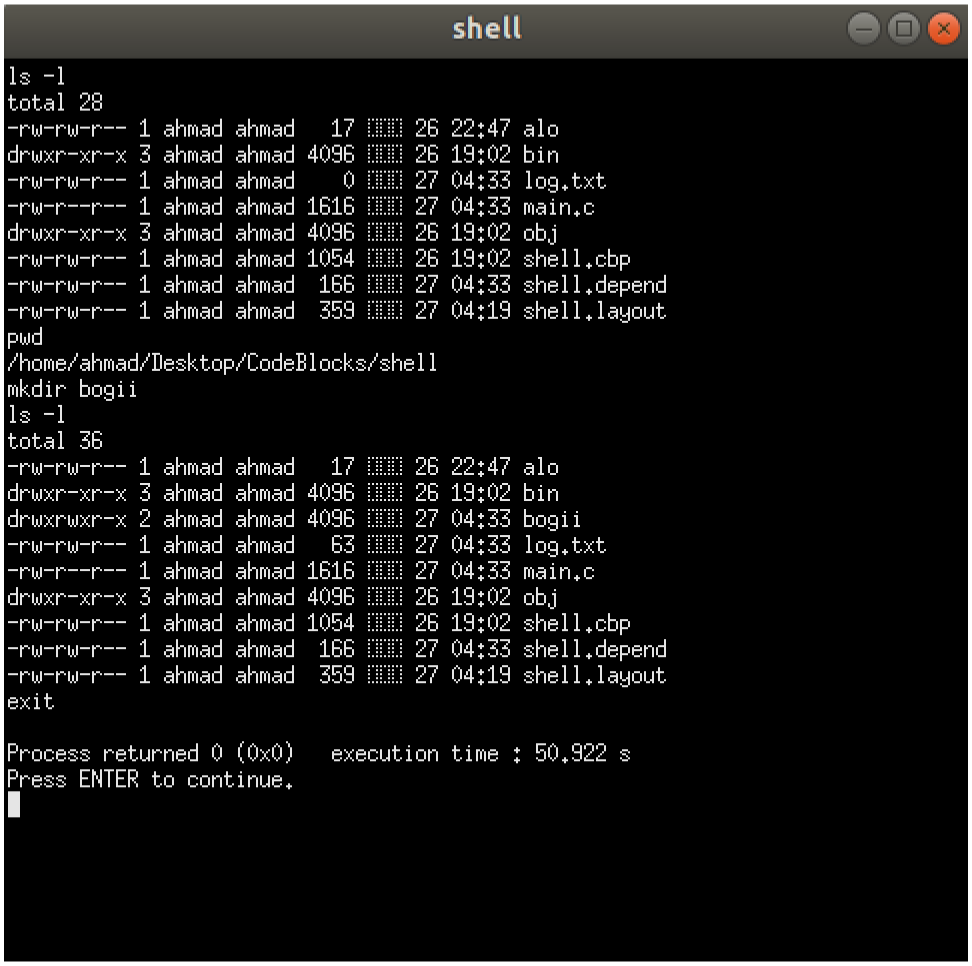
## [Basic loop of a shell]

So we’ve taken care of how the program should start up. Now, for the basic program logic: what does the shell do during its loop? Well, a simple way to handle commands is with three steps:

- **Read**: Read the command from standard input.
- **Parse**: Separate the command string into a program and arguments.
- **Execute**: Run the parsed command.

## [Sample Runs & Screenshots]

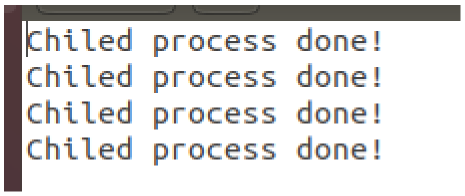
Trying some simple commands like (ls -l , pwd , exit)



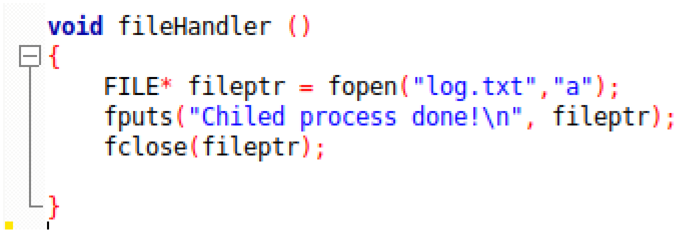
```
ls -l
total 28
-rw-rw-r-- 1 ahmad ahmad   17 ▯▯ 26 22:47 alo
drwxr-xr-x 3 ahmad ahmad 4096 ▯▯ 26 19:02 bin
-rw-rw-r-- 1 ahmad ahmad    0 ▯▯ 27 04:33 log.txt
-rw-r--r-- 1 ahmad ahmad 1616 ▯▯ 27 04:33 main.c
drwxr-xr-x 3 ahmad ahmad 4096 ▯▯ 26 19:02 obj
-rw-rw-r-- 1 ahmad ahmad 1054 ▯▯ 26 19:02 shell.cbp
-rw-rw-r-- 1 ahmad ahmad  166 ▯▯ 27 04:33 shell.depend
-rw-rw-r-- 1 ahmad ahmad  359 ▯▯ 27 04:19 shell.layout
pwd
/home/ahmad/Desktop/CodeBlocks/shell
mkdir bogii
ls -l
total 36
-rw-rw-r-- 1 ahmad ahmad   17 ▯▯ 26 22:47 alo
drwxr-xr-x 3 ahmad ahmad 4096 ▯▯ 26 19:02 bin
drwxrwxr-x 2 ahmad ahmad 4096 ▯▯ 27 04:33 bogii
-rw-rw-r-- 1 ahmad ahmad   63 ▯▯ 27 04:33 log.txt
-rw-r--r-- 1 ahmad ahmad 1616 ▯▯ 27 04:33 main.c
drwxr-xr-x 3 ahmad ahmad 4096 ▯▯ 26 19:02 obj
-rw-rw-r-- 1 ahmad ahmad 1054 ▯▯ 26 19:02 shell.cbp
-rw-rw-r-- 1 ahmad ahmad  166 ▯▯ 27 04:33 shell.depend
-rw-rw-r-- 1 ahmad ahmad  359 ▯▯ 27 04:19 shell.layout
exit

Process returned 0 (0x0)   execution time : 50.922 s
Press ENTER to continue.
```

the log file function and what prints in it during the run



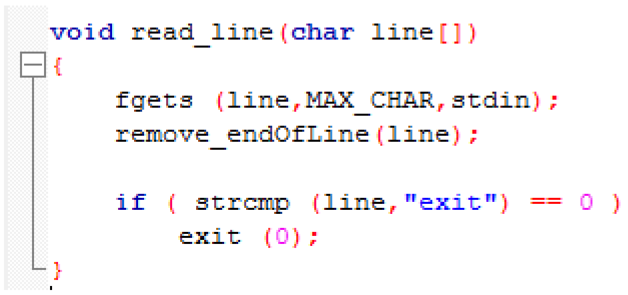
```
Chiled process done!
Chiled process done!
Chiled process done!
Chiled process done!
```



```c
void fileHandler ()
{
    FILE* fileptr = fopen("log.txt","a");
    fputs("Chiled process done!\n", fileptr);
    fclose(fileptr);

}
```

Reading line function and inside the exit command imp.



```
void read_line(char line[])
{
    fgets (line,MAX_CHAR,stdin);
    remove_endOfLine(line);

    if ( strcmp (line,"exit") == 0 )
        exit (0);
}
```

## [Main functions used]

- parser ( )
- exit ( )
- read_line ( )
- remove_endOfLine ( )
- process_line ( )
- fileHandler ( )