Project: Basic Shell in C Language

Submitted by – Kashish Aggarwal (102103490)

Submitted to – Dr. Amrita

December 2022

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

1. Introduction

This shell is a basic shell written in c language. With some in built commands. This was a great experience which allowed me to learn many new ways to use and implement header files and helped understanding of the shell and its commands.

A Shell provides you with an interface to the Unix system. It gathers input from you and executes programs based on that input. When a program finishes executing, it displays that program's output.

Shell is an environment in which we can run our commands, programs, and shell scripts. There are different flavors of a shell, just as there are different flavors of operating systems. Each flavor of shell has its own set of recognized commands and functions.

2. Description of Shell Built-in Commands used in Code

help – This command shows list of all the commands built in this shell and gives a brief description about the commands which helps the users to understand the shell.

date – This command shows the current date in dd:mm:yyyy format using header file time.h

time – This command shows the current time in hh:mm:ss format using header file time.h

pwd – This command displays present working directory

clear – This clears the terminal screen

flip – This flip a coin and give you either head or tail and is totally random

user – Shows the name of the current user

ls – List outs all the contents of the current working directory

cd – Changes the directory

info – Gives some basic introduction of this project

bye – Exits a shell with fun

exit – Exits the shell and returns to the terminal window

3. Significance of Header Files Used

stdio.h – this helps in performing input and ouput from user

sys/wait.h – declaration for waiting. Used in k_exe() for WIFEXITED(), WUNTRACED, pid_t, child_pid

unistd.h - defines miscellaneous symbolic constants and types, and declares miscellaneous functions.

stdbool.h – for true and false statements

errno.h – to investigate to what went worng

time.h – to extract time and date and other from the system and display it

stdlib.h – standarad libray header file for C contains many useful function such as perror to print a certain error code during a occurence of the error

4. Screenshots of the Working Prototype

Help

```
Terminal - kashish@kashish-Swift-SFX14-41G: ~/Kashish/Shell Scripting
File Edit View Terminal Tabs Help
kashish@kashish-Swift-SFX14-41G:~/Kashish/Shell Scripting$ ./a.out
 > help
Kashish - KASHISH AGGARWAL 102103490 SHELL
info - display basic info about this project
The followng commands are built in this
cd - change working dir
exit - exit the shell
bye - bye bye from the shell
help - prints this text
date - display current date
time - display current time in 12hr format
pwd - displays pesent working directory
clear - clear the screen/terminal window
flip - flips a coin
user - displays name of the user
ls - list all contents
 >
```

info

```
Terminal - kashish@kashish-Swift-SFX14-41G: ~/Kashish/Shell Scripting
File Edit View Terminal Tabs Help
 > info
This is a basic shell programmed in C Language
 for Course Practical Computing
```

cd

```
 > cd
Kashish: cd: missing argument
```

date

```
 > date
Date: 20-12-2022
```

time

```
 > time
2:29:44 PM
```

pwd

```
 > pwd

 /home/kashish/Kashish/Shell Scripting
```

flip

```
 > flip
Heads
 > flip
Tails
 > flip
Heads
 > flip
Heads
```

user

```
 > user

 kashish
```

ls

```
 > user

 kashish
 > ls
 a.out                else_if.sh    q1        sample.c              test
 array.sh             for_loop.sh   q1.c      Shello.c              until.sh
 basic_operator.sh  'if else.sh'    q1.cpp   'Shell Project.docx'   while.sh
 bye.png              if.sh         Ques1.c   switch_case.sh
 >
```

bye/exit

```
Terminal - kashish@kashish-Swift-SFX14-41G: ~/Kashish/Shell Scripting
File  Edit  View  Terminal  Tabs  Help
2:29:44 PM
 > pwd

 /home/kashish/Kashish/Shell Scripting
 > flip
Heads
 > flip
Tails
 > flip
Heads
 > flip
Heads
 > user

 kashish
 > ls
 a.out                else_if.sh    q1        sample.c              test
 array.sh             for_loop.sh   q1.c      Shello.c              until.sh
 basic_operator.sh  'if else.sh'    q1.cpp   'Shell Project.docx'   while.sh
 bye.png              if.sh         Ques1.c   switch_case.sh
 > bye
kashish@kashish-Swift-SFX14-41G:~/Kashish/Shell Scripting$ ./a.out
 > exit
kashish@kashish-Swift-SFX14-41G:~/Kashish/Shell Scripting$
```

5. Code

File Edit Search View Document Help

```
//Shell - Kashish

#include <stdio.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdbool.h>
#include <errno.h>
#include <time.h>
#include <stdlib.h>
#include <string.h>

//Exit/Bye Function
void k_exit(char **agr)
{
    exit(0);
}

//Bye
void k_bye(char **agr)
{
        system("eog bye.png");
        exit(0);
}

//Date func

void k_date(char **agr)
{
        time_t t;
        t=time(NULL);
        struct tm tm = *localtime(&t);
        printf("Date: %d-%d-%d\n", tm.tm_mday,tm.tm_mon+1,tm.tm_year+1900);
}

//Time Func

void k_time(char **agr)
{
        time_t t;
        t=time(NULL);
        struct tm tm = *localtime(&t);
        if(tm.tm_hour>=12)
    {
                if(tm.tm_hour==12)
                    printf("12");
                else
                    printf("%d", tm.tm_hour-12);
                printf(":%d:%d PM\n", tm.tm_min, tm.tm_sec);
    }
    else
        printf("%d:%d:%d AM\n", tm.tm_hour, tm.tm_min, tm.tm_sec);
}
```

~/Desktop/K

File Edit Search View Document Help

```
void k_time(char **agr)
{
        time_t t;
        t=time(NULL);
        struct tm tm = *localtime(&t);
        if(tm.tm_hour>=12)
    {
                if(tm.tm_hour==12)
                    printf("12");
                else
                    printf("%d", tm.tm_hour-12);
                printf(":%d:%d PM\n", tm.tm_min, tm.tm_sec);
    }
    else
        printf("%d:%d:%d AM\n", tm.tm_hour, tm.tm_min, tm.tm_sec);
}

//info

void k_info(char **agr)
{
        printf("This is a basic shell programmed in C Language\n for Course Practical Computing\n");
}

//pwd

void k_pwd(char **agr)
{
        char *buf;
        buf=(char *)malloc(100*sizeof(char));
        getcwd(buf,100);
        printf("\n %s \n",buf);
}

//current username

void k_user(char *agr)
{
        char *buf;
        buf=(char *)malloc(10*sizeof(char));
        buf=getlogin();
        printf("\n %s \n",buf);
}

//flip

void k_flip(char **agr)
{
        int f=rand()%10;
        if(f%2==0)
        {printf("Tails\n");
}
        else if(f%2!=0)
                {printf("Heads\n");
}
}
```

File Edit Search View Document Help

```
}


//Cd function

void k_cd(char **agr)
{
    if(agr[1]==NULL)
    {
        fprintf(stderr,"Kashish: cd: missing argument\n");
    }
    else
    {
        if(chdir(agr[1])!=0)
        {
            perror("kashish: cd");
        }
    }
}

//Clear func

void k_clear(char **agr)
{
        system("clear");
}

//Help function

void k_help(char **agr)
{
    char *txt =
    "Kashish - KASHISH AGGARWAL 102103490 SHELL\n"
"info - display basic info about this project\n"
    "The followng commands are built in this\n"
    "cd - change working dir\n"
    "exit - exit the shell\n"
    "bye - bye bye from the shell\n"
    "help - prints this text\n"
"date - display current date\n"
"time - display current time in 12hr format\n"
"pwd - displays pesent working directory\n"
"clear - clear the screen/terminal window\n"
"flip - flips a coin\n"
"user - displays name of the user\n"
"ls - list all contents\n"
    ;
    printf("%s",txt);
}

//It helps in assigning the cmd name to the handler func

struct bults
{
    char *name;
    void (*func)(char **agr);
};
```

```
};

//Other Functions

int k_num_bult()
{
    return sizeof(bult)/sizeof(struct bults);
}

//Executer Bit

void k_exe(char **agr)
{
    for(int i=0;i<k_num_bult();i++)
    {
        if(strcmp(agr[0],bult[i].name)==0)
        {
            bult[i].func(agr);
            return;
        }
    }

    pid_t child_pid=fork();
    if(child_pid==0)
    {
        execvp(agr[0], agr);
        perror("Kashish");
        exit(1);
    }
    else if(child_pid>0)
    {
        int stat;
        do
        {
            waitpid(child_pid, &stat, WUNTRACED);
        }
        while(!WIFEXITED(stat)&&!WIFEXITED(stat));
    }
    else{perror("Kashish");};
}
```

```c
//Line Splitter and Praizer

char **k_split(char *line)
{
    int len=0;
    int cap=16;
    char **tokens = malloc(cap*sizeof(char*));
    if(!tokens)
    {
        perror("Kashish");
        exit(1);
    }
    char *delmiters = "\t\r\n";
    char *token=strtok(line, delmiters);

    while(token!=NULL)
    {
        tokens[len]=token;
        len++;
        if(len>=cap)
        {
            cap=(int)(cap*1.6);
            tokens = realloc(tokens, cap*sizeof(char*));
            if(!tokens)
            {
                perror("Kashish");
                exit(1);
            }
        }
        token=strtok(NULL, delmiters);
    }
    tokens[len=NULL];
    return tokens;
}
```

```c
//Reading the Line using getline

char *k_read_line()
{
    char *line =NULL;
    size_t buflen = 0;
    errno = 0;
    ssize_t stlen = getline(&line, &buflen, stdin);
    if(stlen<0)
    {
        if(errno)
        {
            perror("Kashish");
        }
        exit(1);
    }
    return line;
}

int main() // Main Function handling the loop to read and break the line to understand the commands
{
    while(true)
    {
        printf(" > ");
        char *line=k_read_line();
        char **tokens=k_split(line);
        if(tokens[0]!=NULL)
        {
            k_exe(tokens);
        }
        free(tokens);
        free(line);
    }
}
```