Date: December 1, 2015

Project Report

Project Title: UNIX Shell and History Feature

<u>**Description**</u> :

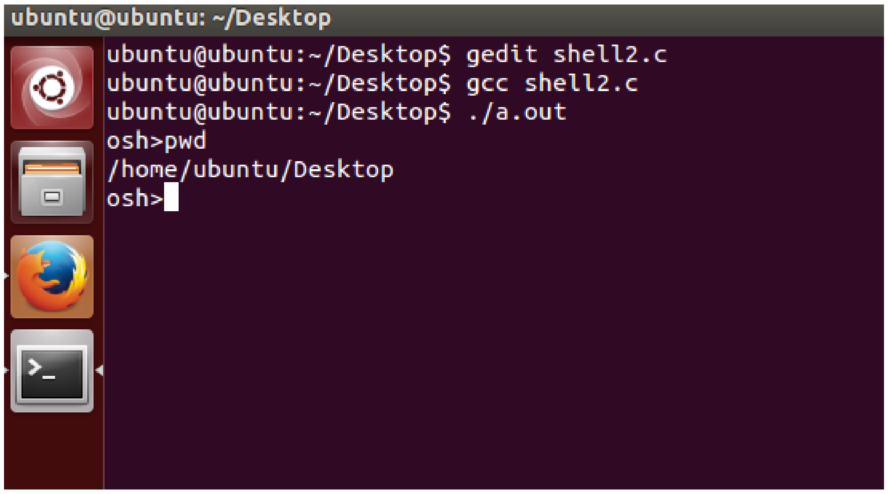
Part -1 Shell Interface:

Algorithm for interface and command execution:

- Enter command in shell.
- The command string is read into inputBuffer[] array(using read()) from the command line.
- Command string is formatted and tokenized into arguments using formatCommand() function and is passed to args[] array.
- The '\t' or ' '(blank space) in the iput indicates the end of the word.Following that word will be a new argument word. Thus a single command line is separated in different argument words. When a '\n' is encountered, that is treated as end of command line and NULL character is added in the 'args[]' array to denote the same
- The arguments are checked for correctness of the command and is stored in history[10][BUFFER_SIZE] array.
- In the main function the child process is created using 'fork()'. 'execvp()' loads the process with the given command if successful otherwise gives respective errors.
- Parent process enters in to wait (calls wait()) if & is not entered this is done through using the 'flag' variable
- To exit the virtual shell prompt enter 'ctrl+c'.

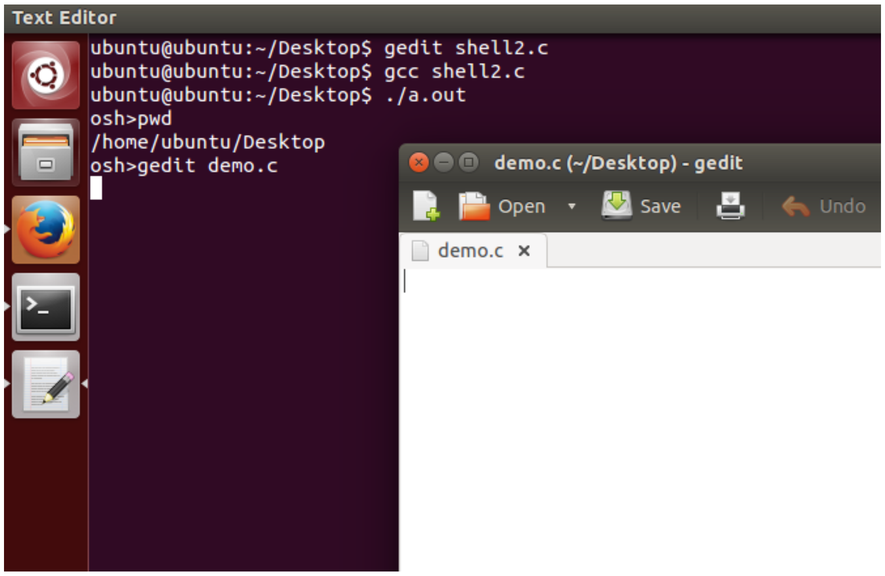
Output for different Cases:

1.Executing a command:



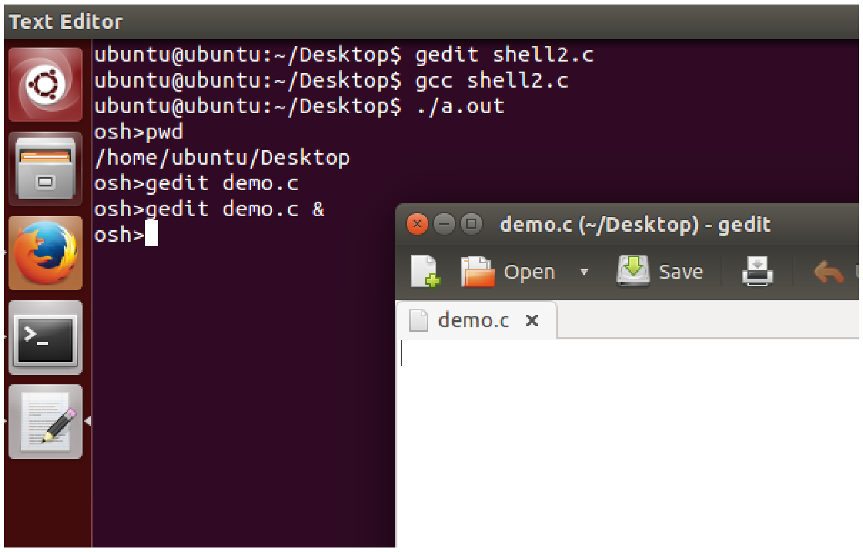
```
ubuntu@ubuntu:~/Desktop$ gedit shell2.c
ubuntu@ubuntu:~/Desktop$ gcc shell2.c
ubuntu@ubuntu:~/Desktop$ ./a.out
osh>pwd
/home/ubuntu/Desktop
osh>
```

2.Executing a command without '&': As shown in the below image parent process will wait for the child process (gedit) to complete.



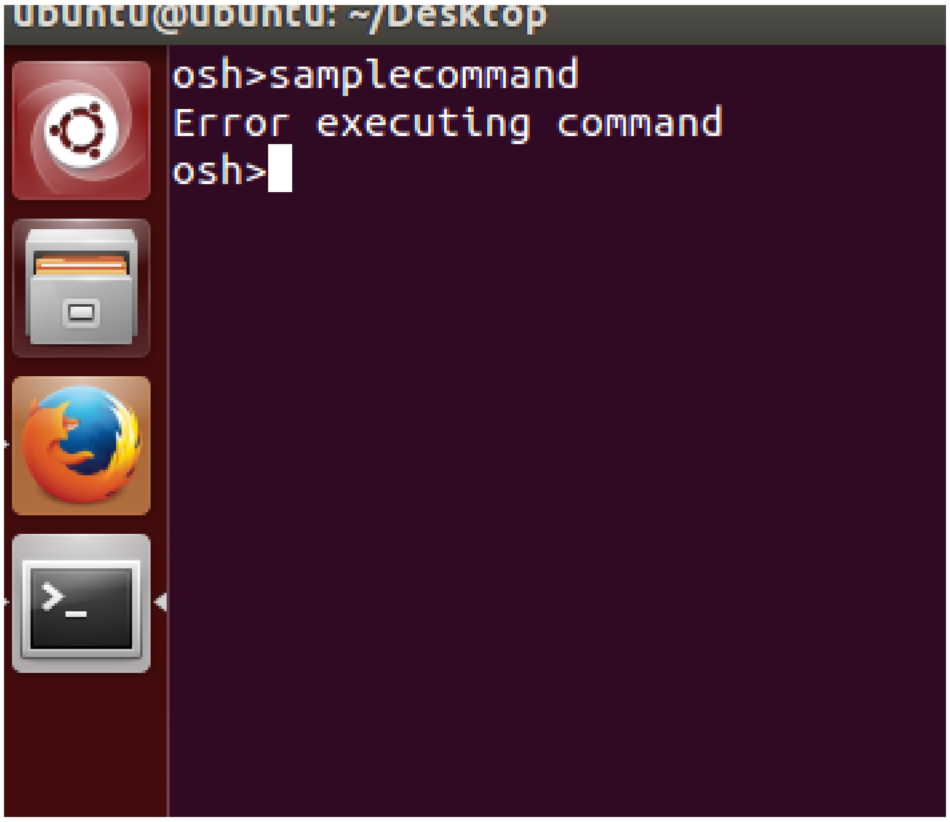
```
ubuntu@ubuntu:~/Desktop$ gedit shell2.c
ubuntu@ubuntu:~/Desktop$ gcc shell2.c
ubuntu@ubuntu:~/Desktop$ ./a.out
osh>pwd
/home/ubuntu/Desktop
osh>gedit demo.c
```

3.Executing a command with '&': As shown in the below image parent process will not wait for the child process (gedit) to complete.



```
ubuntu@ubuntu:~/Desktop$ gedit shell2.c
ubuntu@ubuntu:~/Desktop$ gcc shell2.c
ubuntu@ubuntu:~/Desktop$ ./a.out
osh>pwd
/home/ubuntu/Desktop
osh>gedit demo.c
osh>gedit demo.c &
osh>
```

4.Executing a invalid command



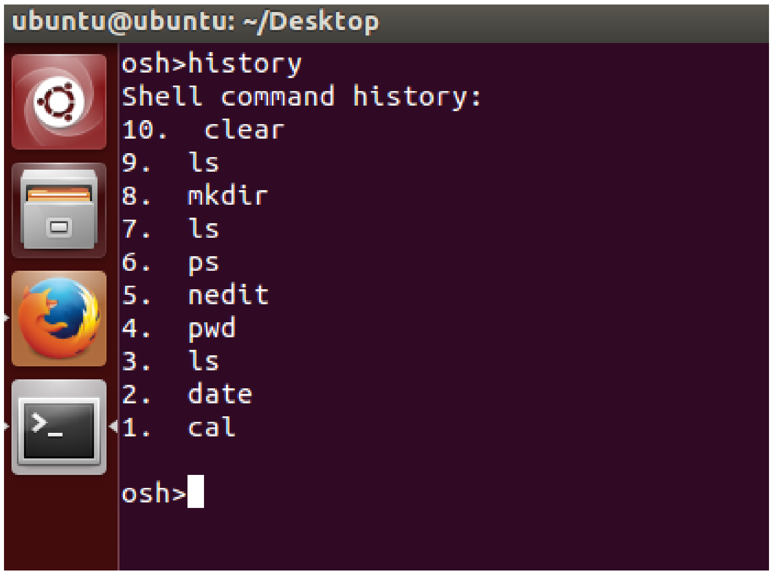
```
osh>samplecommand
Error executing command
osh>
```

Algorithm for History Feature

- Create a two dimensional character array history[10][BUFFER_SIZE] to store commands entered in terminal
- Define a function displayHistory() to display the previous commands on terminal when history command is executed in terminal
- Max size given is 10 i.e., it can store upto 10 recent commands entered.
- Each row stores a command. Each column of a row stores the separate argument words in the command entered with NULL character as the last character.
- Created a variable named histcount for counting the commands entered. When a command is entered in terminal the histcount value will be increased by one.
- The oldest entered value will be overwritten for all newly entered commands. Thus the history array can store 10 commands at a time. So, the array basically holds the 10 recent commands entered in the terminal at a particular time.
- Type history in the terminal to display the commands entered upto 10.
- For 'history_count' less than 10, the commands entered are displayed keeping the other positions in history vacant and not accessible.
- Type '!!' to reexecute the most recent command executed.
- Type '!n' where n refers to the index of the rows in history array to reexecute that command i.e., to execute nth command.

Output :

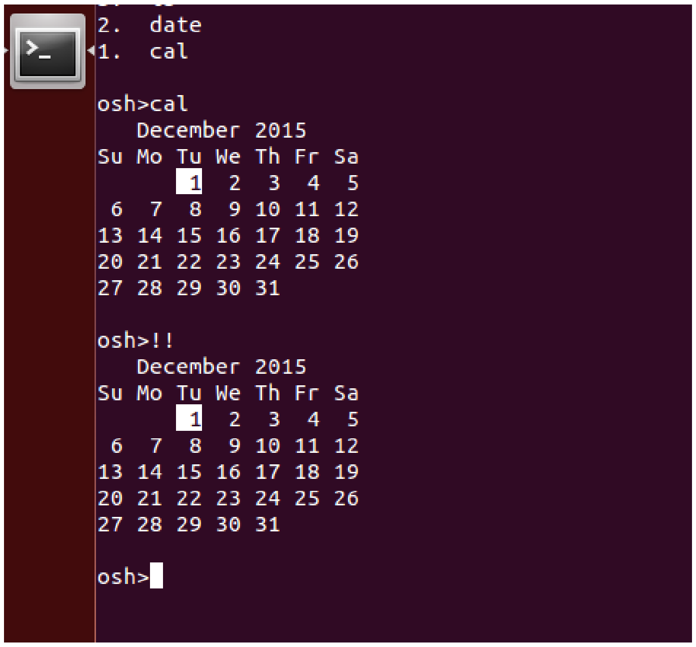
1.Executing history command to display previous 10 commands



```
ubuntu@ubuntu: ~/Desktop
osh>history
Shell command history:
10.  clear
9.   ls
8.   mkdir
7.   ls
6.   ps
5.   nedit
4.   pwd
3.   ls
2.   date
1.   cal

osh>
```

2. Executing the recent command by using '!!' in the terminal



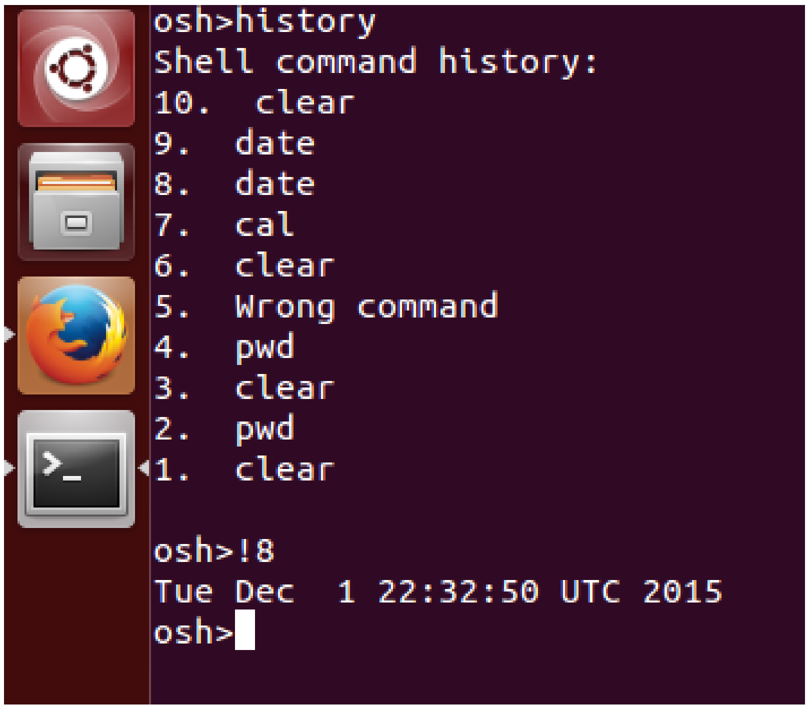
```
2.   date
1.   cal

osh>cal
   December 2015
Su Mo Tu We Th Fr Sa
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31

osh>!!
   December 2015
Su Mo Tu We Th Fr Sa
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31

osh>
```

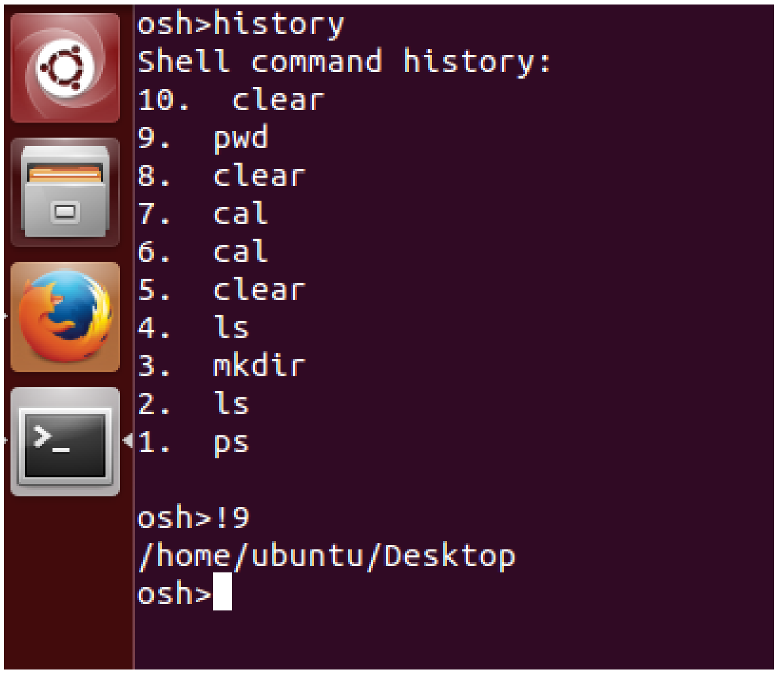
3. **Executing 8th recent command by using '!8'**



```
osh>history
Shell command history:
10.  clear
9.   date
8.   date
7.   cal
6.   clear
5.   Wrong command
4.   pwd
3.   clear
2.   pwd
1.   clear

osh>!8
Tue Dec  1 22:32:50 UTC 2015
osh>
```

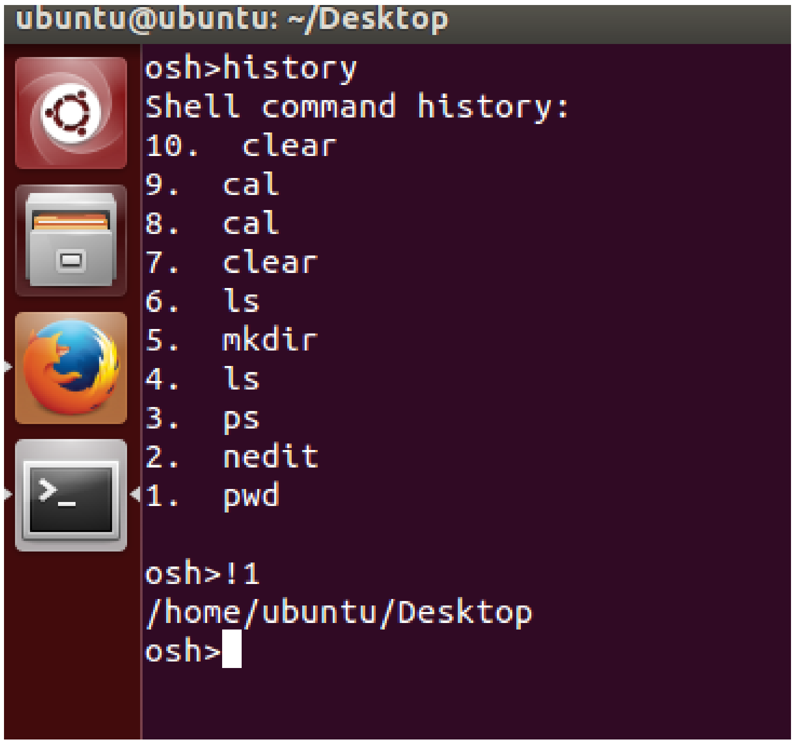
4. **Executing 9th recent command by using '!9'**



```
osh>history
Shell command history:
10.  clear
9.   pwd
8.   clear
7.   cal
6.   cal
5.   clear
4.   ls
3.   mkdir
2.   ls
1.   ps

osh>!9
/home/ubuntu/Desktop
osh>
```

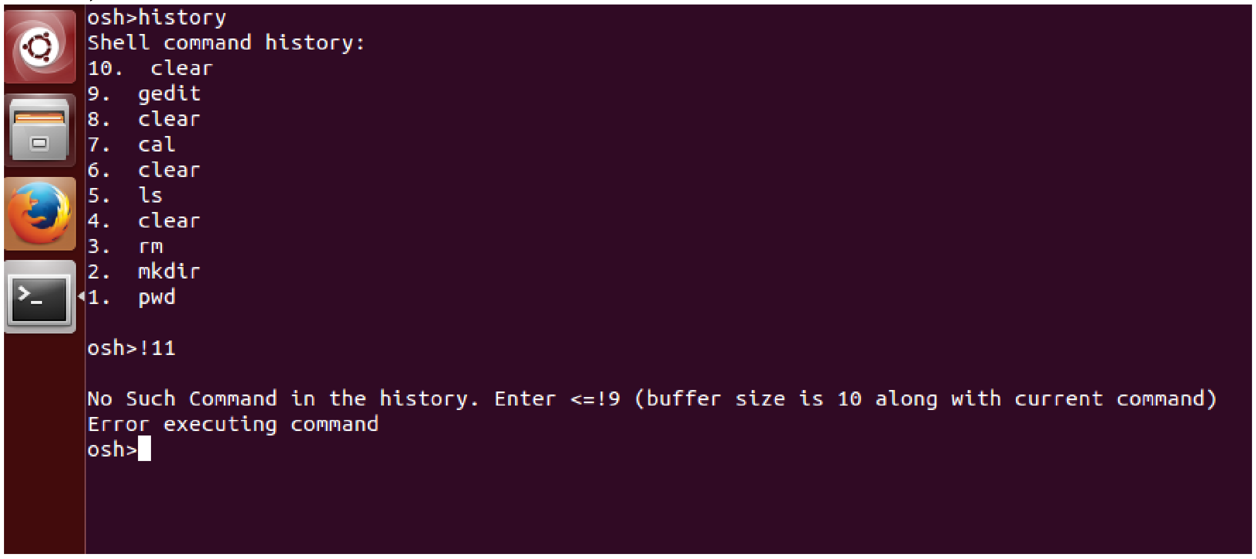
5. Executing 1st recent command by using '!1' in the terminal



```
ubuntu@ubuntu: ~/Desktop
osh>history
Shell command history:
10.  clear
9.   cal
8.   cal
7.   clear
6.   ls
5.   mkdir
4.   ls
3.   ps
2.   nedit
1.   pwd

osh>!1
/home/ubuntu/Desktop
osh>
```

6.Showing the error when 11th command is accessed (history will have only 10 commands stored)

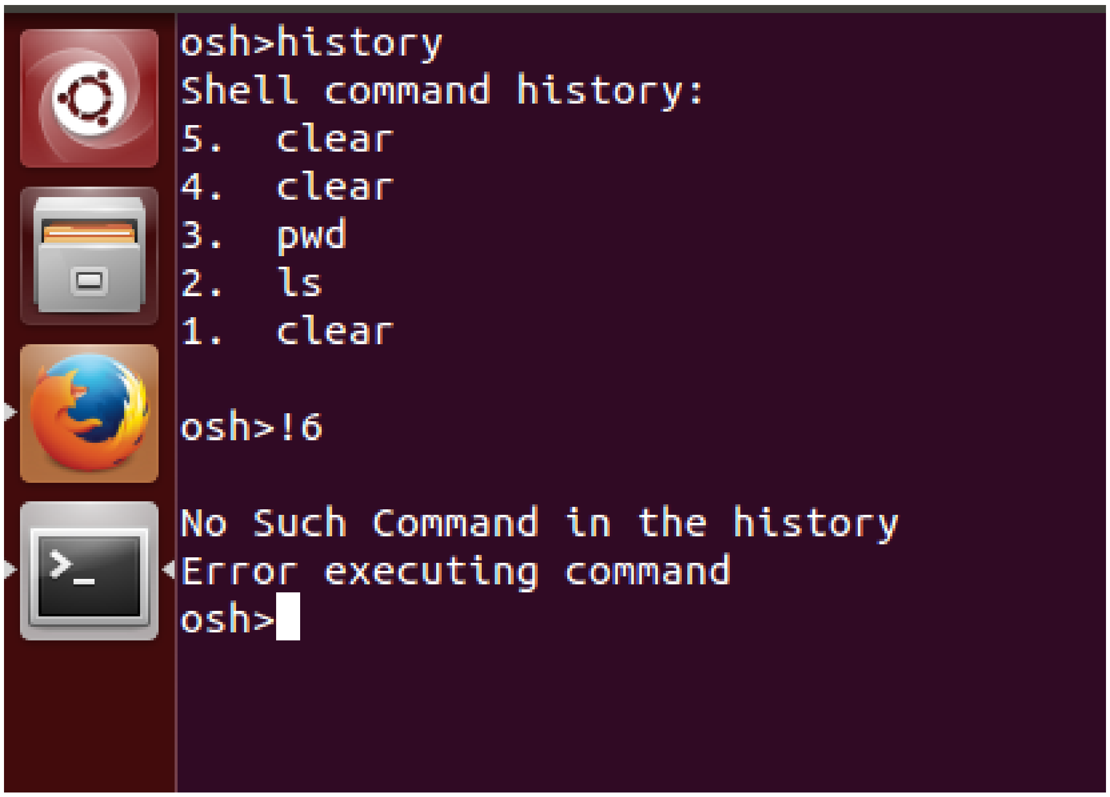


```
osh>history
Shell command history:
10.  clear
9.   gedit
8.   clear
7.   cal
6.   clear
5.   ls
4.   clear
3.   rm
2.   mkdir
1.   pwd

osh>!11

No Such Command in the history. Enter <=!9 (buffer size is 10 along with current command)
Error executing command
osh>
```

7. Showing the error when user access a command not in history.



```
osh>history
Shell command history:
5.  clear
4.  clear
3.  pwd
2.  ls
1.  clear

osh>!6

No Such Command in the history
Error executing command
osh>
```

<u>Problems Faced and How We Solved Them</u>

1. We faced problem when trying to compare strings. The string compare can be easily done by using strcmp() function but we have used ascii values for comparison. If strcmp() is used we have compare the input with every case of !n like strcmp(str,”!1”), strcmp(str,”!2”), strcmp(str,”!3”) etc. Instead we used ASCII value as described in below code for compact code and easy access of command.

```
else if (args[0][0]-'!' ==0)
        {       int x = args[0][1]- '0';
                int z = args[0][2]- '0';

                if(x>count) //second letter check
                {
                printf("\nNo Such Command in the history\n");
                strcpy(inputBuffer,"Wrong command");
                }
                else if (z!=-48) //third letter check
                {
```

```
                printf("\nNo Such Command in the history. Enter <=!9 (buffer size is 10 along with current command)\n");
                strcpy(inputBuffer,"Wrong command");
                }
                else
                {

                        if(x==-15)//Checking for '!!',ascii value of '!' is 33.
                        {       strcpy(inputBuffer,history[0]);   // this will be your 10 th(last) command
                        }
                        else if(x==0) //Checking for '!0'
                        {       printf("Enter proper command");
                                strcpy(inputBuffer,"Wrong command");
                        }

                        else if(x>=1) //Checking for '!n', n >=1
                        {
                                strcpy(inputBuffer,history[count-x]);

                        }

                }
        }
```

Features of the project:

1.Exit command → program exits on entering 'exit' commands

2. Running in background → child process runs in background on adding '&'

3. History feature → shows recently entered 10 commands

case 2: history_count > 10

4. Recent command entered → executes recent command on '!!'

5. Executing Nth command in history → on entering '!N', where 'N' is the history index

6. Error on entering invalid command in command line

7. Error on entering '!!' as the first command (when there is no command in history)

8. Error on entering '!N' when Nth index is vacant

8. Error on entering '!N' when Nth index is greater than 10

//Source Code

```
//Venkata Sai deepak Aitha, Chandan dev, shanmukha priya, Madhu
anchuri, saransh singh
//Enter command  'history' for history feature and CTRL - c to exit
the 'osh>' shell
/*Header files */
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>

#define MAX_LINE 80 /* The maximum length of a command */
#define BUFFER_SIZE 50
#define buffer "\n\Shell Command History:\n"

//declarations
char history[10][BUFFER_SIZE]; //history array to store history
commands
int count = 0;


//function to display the history of commands
void displayHistory()
{

    printf("Shell command history:\n");

    int i;
    int j = 0;
    int histCount = count;

    //loop for iterating through commands
    for (i = 0; i<10;i++)
    {
        //command index
        printf("%d.  ", histCount);
        while (history[i][j] != '\n' && history[i][j] != '\0')
        {
           //printing command
            printf("%c", history[i][j]);
            j++;
        }
        printf("\n");
        j = 0;
        histCount--;
        if (histCount ==  0)
            break;
    }
    printf("\n");
}
```

```
//Fuction to get the command from shell, tokenize it and set the args
parameter

int formatCommand(char inputBuffer[], char *args[],int *flag)
{
     int length; // # of chars in command line
     int i;     // loop index for inputBuffer
     int start;  // index of beginning of next command
     int ct = 0; // index of where to place the next parameter into
args[]
     int hist;
     //read user input on command line and checking whether the
command is !! or !n

     length = read(STDIN_FILENO, inputBuffer, MAX_LINE);


    start = -1;
    if (length == 0)
        exit(0);   //end of command
    if (length < 0)
    {
        printf("Command not read\n");
        exit(-1);  //terminate
    }

   //examine each character
    for (i=0;i<length;i++)
    {
        switch (inputBuffer[i])
        {
            case ' ':
            case '\t' :               // to seperate arguments
                if(start != -1)
                {
                    args[ct] = &inputBuffer[start];
                    ct++;
                }
                inputBuffer[i] = '\0'; // add a null char at the end
                start = -1;
                break;

            case '\n':                 //final char
                if (start != -1)
                {
                    args[ct] = &inputBuffer[start];
                    ct++;
                }
                inputBuffer[i] = '\0';
                args[ct] = NULL; // no more args
                break;

            default :
                if (start == -1)
                    start = i;
                if (inputBuffer[i] == '&')
                {
                    *flag  = 1; //this flag is the differentiate
whether the child process is invoked in background
                    inputBuffer[i] = '\0';
                }
        }
    }

    args[ct] = NULL; //if the input line was > 80

if(strcmp(args[0],"history")==0)
        {
               if(count>0)
           {

                displayHistory();
           }
           else
           {
           printf("\nNo Commands in the history\n");
           }
           return -1;
        }

      else if (args[0][0]-'!' ==0)
      {    int x = args[0][1]- '0';
           int z = args[0][2]- '0';

           if(x>count) //second letter check
           {
           printf("\nNo Such Command in the history\n");
           strcpy(inputBuffer,"Wrong command");
           }
           else if (z!=-48) //third letter check
           {
           printf("\nNo Such Command in the history. Enter <=!9
(buffer size is 10 along with current command)\n");
           strcpy(inputBuffer,"Wrong command");
           }
           else
           {

                if(x==-15)//Checking for '!!',ascii value of '!' is
33.
```

```
                {     strcpy(inputBuffer,history[0]);  // this will be
your 10 th(last) command
                }
                else if(x==0) //Checking for '!0'
                {     printf("Enter proper command");
                     strcpy(inputBuffer,"Wrong command");
                }

                else if(x>=1) //Checking for '!n', n >=1
                {
                     strcpy(inputBuffer,history[count-x]);

                }

           }
      }
 for (i = 9;i>0; i--) //Moving the history elements one step higher
           strcpy(history[i], history[i-1]);

    strcpy(history[0],inputBuffer); //Updating the history array with
input buffer
    count++;
     if(count>10)
     { count=10;
     }
}

int main(void)
{
    char inputBuffer[MAX_LINE]; /* buffer to hold the input command */
    int flag; // equals 1 if a command is followed by "&"
    char *args[MAX_LINE/2 + 1];/* max arguments */
    int should_run =1;

    pid_t pid,tpid;
    int i;


    while (should_run) //infinite loop for shell prompt
    {
        flag = 0; //flag =0 by default
        printf("osh>");
        fflush(stdout);
        if(-1!=formatCommand(inputBuffer,args,&flag)) // get next
command
     {
           pid = fork();

           if (pid < 0)//if pid is less than 0, forking fails
           {

                     printf("Fork failed.\n");
```

```
                        exit (1);
                }

                        else if (pid == 0)//if pid ==0
                {

                        //command not executed
                        if (execvp(args[0], args) == -1)
                        {

                        printf("Error executing command\n");
                        }
                        }

                        // if flag == 0, the parent will wait,
                // otherwise returns to the formatCommand() function.
                else
                {
                        i++;
                        if (flag == 0)
                         {
                        i++;
                        wait(NULL);
                         }
                }
            }
          }
  }
```