/\*

Assigment-1-Shall-2

Name: Veda Samhitha Manne

Std ID: 1002030416

\*/

//Code for the assignment1- shall-2

#define \_GNU\_SOURCE //define-directive to define macro

//Preprocessor of specified file of directive processor

#include<stdio.h>

#include<sys/types.h>

#include<sys/wait.h>

#include<stdlib.h>

#include<string.h>

#include<signal.h>

#include <unistd.h>

#include<errno.h>

#define WHITESPACE "\t\n" //tab space, separate tokens of the cmd

#define MAX\_CMD\_SIZE 255 //maximum-size-of-the-command-line

#define MAX\_NUM\_ARG 13 //mav-shall supports 10 arg

#define MAX\_PRE\_ARG 14 //mav-shall-storing previous commands

#define MAX\_PRO\_IDS 17 // Process\_ID stored

int main()

{

int a; //decalring-variable-a

// allocating-memory-to-block

char \* command\_string=(char\*)malloc(MAX\_CMD\_SIZE);

//declaring-process\_ID's- index&ct

int pro\_i=0;

int pro\_ct=0;

//Memory allocation to array

char \*pre[MAX\_PRE\_ARG];

int j;

int pre\_i=0;

int pre\_c=0;

int pro\_id=0;

//memory-allocation-to-index

for(a=0; a<MAX\_PRE\_ARG; a++)

{

pre[a]= (char \*)malloc(MAX\_CMD\_SIZE);

}

while(1)

{

//Printing output as msh

printf("msh>");

//while-statement-used to wait-for the user's input, fget-returns-0-per-0-input

while(!fgets(command\_string,MAX\_CMD\_SIZE,stdin));

char \*token[MAX\_NUM\_ARG];

int token\_count=0;

//declaring-pointer-to the charcater

char \*arg\_ptr;

//save-the-copyofthecmd-line,strdup-end-moving-the-poniter

char \*working\_string=strdup(command\_string);

//track-of the original\_value and deallocate\_crt\_amount in end

char \*head\_ptr=working\_string;

j=0;

//First\_element\_of the cmd is"!"

if(working\_string[0]=='!')

{

int pre\_no=atoi(&working\_string[1]);

//verifing if the command is valid

if(pre\_no >=0 && pre\_no < pre\_c)

{

//string copy command

strcpy(working\_string,pre[pre\_no]);

}

else

{

printf("Command not found");

j++;

}

}

//Tokenize the input by using whitespace as a seperator

while(((arg\_ptr=strsep(&working\_string, WHITESPACE))!= NULL) && (token\_count<MAX\_NUM\_ARG))

{

token[token\_count]=strndup( arg\_ptr, MAX\_CMD\_SIZE);

if(strlen( token[token\_count])==0)

{

token[token\_count]=NULL;

}

token\_count++;

}

//Shell FUNCTION

//Null input is checked by null functionality

if(token[0]!=NULL)

{

char \*line=strdup(command\_string);

pre[pre\_i]=line;

//incrementing the element by each time and it is addedto the previous command

pre\_i++;

pre\_c++;

//modifing the pre\_i

if(pre\_i > MAX\_PRE\_ARG -1)

{

pre\_i=0;

}

//Maximum Process-ID is decalred as 15

if(pre\_c > MAX\_PRE\_ARG)

{

pre\_c=MAX\_PRE\_ARG;

}

//ASk if the user wishes to quit the shell

if(strcmp(token[0],"exit")==0 || strcmp(token[0],"quit")==0)

{

exit(0);

}

pid\_t pid\_l[15];

//Start the process for fork() function

pid\_t pid=fork();

//Proccess ID is stored in the Process ID list

pid\_l[pro\_i]=pro\_id;

//performing increment operation for the index and count of the process ID

pro\_i++;

pro\_ct++;

//modify the index value of the process ID

if(pro\_i >MAX\_PRO\_IDS -1)

{

pro\_i=0;

}

if(pro\_ct > MAX\_PRO\_IDS)

{

pro\_ct=MAX\_PRO\_IDS;

}

//Performing parent process

if(pro\_id !=0)

{

//string compare is used to verify the "CD" command is entered or not

if(strcmp(token[0],"cd")==0)

{

chdir(token[1]);

}

}

//chlid function

if(pro\_id==0)

{

//exex() command to call new program

//executionof the command

int bha = execvp(token[0], &token[0]);

if( bha == -1)

{

//command cannot be executed

if(strcmp(token[0],"listpro\_ids")!=0 && strcmp(token[0],"history")!=0 && strcmp(token[0],"cd")!=0 && j!=1)

{

printf("%S: Cmd not found.\n",token[0]);

}

break;

}

}

else

{

int new;

wait(&new);

}

//User will get list of process IDS, if it was typed as listpro\_id

if(strcmp(token[0],"listpro\_ids")==0)

{

printf("\n");

printf("Listing the Pro\_ID:\n");

//display the total no.of Pro\_ID

for(a=0;a < pro\_ct;a++)

{

printf("%d. %d\n", a+1,pid\_l[a]);

}

printf("\n");

}

//if the user types"pre" then the user will get previous history

if(strcmp(token[0],"pre")==0)

{

printf("\n");

}

printf("HISTORY:\n");

//total no.of cmds in the record

for(a=0; a<pre\_c; a++)

{

printf("%d. %s",a,pre[a]);

}

printf("\n");

}

//deallocating the memory blocks

free(head\_ptr);

}

for(a=0; a<MAX\_PRE\_ARG; a++)

{

free(pre[a]);

}

return 0;

}