```
#include <sys/wait.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include<sys/file.h>
#include<sys/wait.h>
int fib_num;

// char default_commands[][10] = {"quit","cd","help","fibonacci","hello","list"};

int fibonacci(int number)
{
    int fib_series[number + 1];
    fib_series[0] = 0;
    fib_series[1] = 1;
    int i;
    for (i = 2; i < number; i++)
    {
        fib_series[i] = fib_series[i - 1] + fib_series[i - 2];
    }
    printf("The first %d values: ", number);
    for (i = 0; i < number-1; i++)
    {
        printf("%d, ", fib_series[i]);
    }
    printf("%d",fib_series[number-1]);
    printf("\n");
    return 0;
}

int main(int argc, char **argv)
{
    char *line = NULL;
    // char *arguments = NULL;
    size_t n;
    size_t maxlen = 0;

    // char *data[100];
    // int i = 0;
    // char array[1000];
    // char path[1000];
    // getcwd(array,1000);
    // strcat(path,array);
    // strcat(path,"/uab_sh.log");
    // printf("%s",array);
    int fd,fin,fout;
    pid_t pid;
    int status;
    // char *temp;


    fd= open("uab_sh.log",O_CREAT | O_APPEND | O_WRONLY, 0644);
    if(fd==-1){
        printf("Error in creatin the file..");
    }
    pid = fork();
    if(pid==0){
        while (1)
    {

            printf("uab_sh >");
            if ((n = getline(&line, &maxlen, stdin)) > 0)
            {
                size_t length = strlen(line);
                fin=open("uab_sh.log",O_WRONLY|O_APPEND);
                // char *string;
                // if(strlen(line)>1){
                //     strcpy(string,line);
                //     string[length+1] = '\0';
```

```
                // }

                write(fin,line,length );
                close(fin);

                if (strstr(line, "hello"))
                {
                    printf("Hello World. \n");
                }
                if(strstr(line, "fibonacci"))
                {
                    if (strstr(line, " "))
                    {
                        strtok(line, " ");
                        char *word = strtok(NULL, " ");
                        fib_num = atoi(word);
                        fibonacci(fib_num);
                    }
                    else
                    {
                        char *fib = NULL;
                        printf("How many elements you want to display:");
                        getline(&fib,&maxlen,stdin);
                        fibonacci(atoi(fib));
                    }
                }
                if(strstr(line,"list")){
                    pid_t cpid,cstatus;
                    cpid = fork();
                    if(cpid ==0){
                        char *argv[]={"/bin/ls", 0};
                        char *envp[]={0};
                        execve(argv[0],argv,envp);
                    }
                    else{
                        wait(&cstatus);
                        WIFEXITED(cstatus);
                    }

                }
                if(strstr(line,"cd")){
                    if(strlen(line)==3){
                        printf("Please Enter the arguments...\n");
                    }
                    else{
                        int i;
                        char array[1000];
                        strtok(line," ");
                        char *word = strtok(NULL," ");
                        char *newword = (char*) malloc(strlen(word)-1);
                        for(i=0;i<strlen(word)-1;i++){
                            newword[i] = word[i];
                        }
                        getcwd(array,1000);
                        if (newword == NULL)
                        {
                            fprintf(stderr, "shell: expected argument to \"cd\"\n");
                        }
                        else{
                            printf("The before working directory %s\n",array);
                            chdir(newword);
                            getcwd(array,1000);
                            printf("The after working directory %s\n",array);
                        }
                    }
                }
                if(strstr(line,"help")){
                    printf("Hello      \t -> For greeting Hello\n");
                    printf("Fibonacci  \t -> For promping for number to print the fibonac
ci numbers until the given number\n");
```

```
                    printf("Fibonacci n\t -> For printing the fibonacci numbers untill \n
");
                    printf("cd argument\t -> For changing directory according to the argu
ment\n");
                    printf("help\t\t -> For the details of command\n");
                    printf("list       \t -> For printing the files in the current workin
g directory\n");
                    printf("history    \t -> For printing the history of commands\n");
                }
                if(strstr(line,"history")){
                    char history[1000];
                    int x;
                    fout = open("uab_sh.log", O_RDONLY);
                    while((x = read(fout,history,1000))>0){
                        printf("%s",history);
                    }

                }
                if(strstr(line,"quit")){
                    if(remove("uab_sh.log")==0){
                        printf("History Deleted succesfully Exiting the program\n");
                        exit(-1);
                    }
                    else{
                        printf("History not deleted\n");
                    }
                }

            }
    }

    }
    else{
        wait(&status);
        WIFEXITED(status);
    }
    return 0;

}
```