Assignment 1

Task 2

Errors handled

To catch the errors and display what they indicate and why they occurred ,proper error handling has been done.The header file for the same #include <errno.h>, has been included which defines errno,which stores the value of any error that occurred the most recently in our running process.Perror() a supporting function interprets this error number and displays the corresponding descriptive detail for the error.
List of errors handled:

	Error
1	/* Operation not permitted */
2	/* No such file or directory */
3	/* No such process */
4	/* Interrupted system call */
5	/* I/O error */
6	/* No such device or address */
7	/* Argument list too long */
8	/* Exec format error */
9	/* Bad file number */
10	/* No child processes */
11	/* Try again */
12	/* Out of memory */
13	/* Permission denied */
14	/* Bad address*/
15	/* Command not recognised*/
16	/* Invalid __ command*/ (can be ls,cat,echo etc)
17	/*File exits */
18	/*Directory exists*/
19	/*No data available*/
20	/*Invalid arguments*/

Apart from these if the user implements a command that is not interpreted by the shell,an error message is displayed. Also if our shell interprets the command but not the particular flag then so is also displayed.

Thus for each of the commands mentioned below all these errors and their handlings are taken in consideration.

Options the shell commands

Spaces have to be taken care of while entering commands,less or more spaces can result in unrecognisable command

Internal commands:

1. **cd**

this command is used to change directory to the home directory.

1.1. cd ..

this command is used to move to the parent directory of current directory

1.2. cd dirname

This command is used to navigate to a directory specified as dirname

1.3. cd -P dirname

Navigate to dirname though its physical or absolute address.

1.4. cd ~

this command is used to change directory to the home directory.

2. echo args

Display the args, removing \ since not interpreted by it and followed by a
newline, on the standard output.

2.1. echo -E args

explicitly suppress interpretation of backslash escapes

2.2. echo -n args

do not append a newline

3. history

Display the history list with line numbers.Since no permanent file created,is available and updates in the program itself.

3.1. history -c

clear the history list by deleting all of the entries

3.2. history -n

read all history lines not already read from the history

4. pwd

Print the name of the current working directory though symbolic interpretation of path

4.1. pwd -L

Print the name of the current working directory though symbolic interpretation of path

4.2. pwd -P

Print the name of the current working directory though absolute interpretation of path

5. Exit

Exit the shell.

<u>External commands:</u>

6. ls

List information about the FILEs (in the current directory by default).

6.1. ls -a

do not ignore entries starting with .

6.2. ls -i

print the index number of each file

7. Cat file

read standard input file

7.1. cat -n

number all output lines

7.2. cat -E

display $ at end of each line

8. date

Display the current time of the system date.

8.1. date -u

print or set Coordinated Universal Time (UTC)

8.2. date -R

output date and time in RFC 5322 format.

9. Rm file

Remove the FILE

9.1. rm -i

prompt before every removal

9.2. rm -f

ignore nonexistent files and arguments, never prompt

10. mkdir

Create the DIRECTORY, if they do not already exist.

10.1. mkdir -v

print a message for each created directory

10.2. mkdir -p

no error if existing, make directories as needed

Assumptions in the shell commands

In history, since history list and file are not both being implemented it is assumed that at every point in time the list is the same as the file.
The echo command does not include any quotes,if so then expected to be printed.
The user is assumed to follow the shell commands list to obtain the desired functionality.

Working

Components: file.c ls.c mkdir.c rm.c date.c cat.c
Working:
The file.c is the main shell that is responsible for implementing all the functionalities.For the external commands it implements a fork to the respective c files of the commands implementation.
Details:
cd and its flags : Uses chdir() to get the absolute path to the required directory.
pwd -L & pwd -P:Uses getenv() to get the symbolic path and getcwd() to get an absolute path to implement them respectively.
history: Uses List implementation to store the commands entered.
echo:Interpret the entered string character by character.
ls:Read the directory using readdir() from dir structure.
cat:Used file open,close and read commands to do the needful.
date:Used the functions in time.h header
mkdir:Used the mkdir() functionality
Rm:used the remove() command.

Testcase

This test case checks all the functionalities.

To run this test case create 2 files in your working directory,dele1 and dele2

```
1 cd ..
2 ls
3 cd ~
4 ls
5 cd Desktop
6 ls
7 cd -P OS1.1
8 ls
9 cd Q1
10 ls
11 cd ..
12 ls
13 cd Q2
```

// these above commands will differ depending on your storage of this file.Mine is as follows:

HOME->Desktop->OS1.1->Q2

```
14 ls -i
15 ls -a
16 cat makefile
17 cat -n makefile
18 cat -E makefile
19 date
20 date -u
21 date -R
22 pwd
23 pwd -L
24 pwd -P
25 echo Ananya\nhello
26 echo -E Ananya\nhello
27 echo -n Asnsnx
28 mkdir hello
29 ls
30 mkdir -i hello
31 mkdir -v hello
32 mkdir -v new
33 mkdir -f new
34 mkdir -p new
35 ls
36 rm dele1
37 ls
38 rm -i dele2
39 ls
40 rm -f dele2
41 ls
42 history
43 echo new
44 echo line
45 history -n
46 history -c
47 exit
```

OUTPUT on IMPLEMENTATION

```
ananya@ubuntu:~$ cd Desktop/OS1.1/Q2
ananya@ubuntu:~/Desktop/OS1.1/Q2$ make
gcc ls.c -o ls
gcc cat.c -o cat
gcc date.c -o date
gcc rm.c -o rm
gcc mkdir.c -o mkdir
gcc -c file.c
./a.out
Please Provide Valid Commands
>>>cd ..
>>>ls
Q1 Q2
>>>cd ~
>>>ls
Documents Desktop Templates add.asm
prog_add.c Downloads Public Music snap Videos
examples.desktop Pictures
>>>cd Desktop
>>>ls
OS.01 Assignment0_2_2019408 OS1.1.zip
OS1.1 OS.02 Quiz1 Assignment0_1_2019408
>>>cd -P OS1.1
>>>ls
Q1 Q2
>>>cd Q1
>>>ls
```

```
parent child parent.c child.c makefile csv-os.csv
file.c a.out WriteUp1.pdf file.o
>>>cd ..
>>>ls
Q1 Q2
>>>cd Q2
>>>ls -i
1->cat.c 2->dele1 3->makefile 4->ls 5->ls.c
6->file.c 7->WriteUp2.pdf 8->cat 9->a.out
10->rm.c 11->mkdir.c 12->mkdir 13->dele2
14->rm 15->date.c 16->date 17->file.o
>>>ls -a
 cat.c  dele1  ..  makefile  ls  ls.c  file.c
WriteUp2.pdf  cat  a.out  rm.c  mkdir.c  mkdir
dele2  .  rm  date.c  date  file.o
>>>cat makefile
all:
        gcc ls.c -o ls
        gcc cat.c -o cat
        gcc date.c -o date
        gcc rm.c -o rm
        gcc mkdir.c -o mkdir
        gcc -c file.c
        gcc file.o
        ./a.out
obj_files:
        gcc ls.c -o ls
        gcc cat.c -o cat
        gcc date.c -o date
        gcc rm.c -o rm
        gcc mkdir.c -o mkdir
assembling:
        gcc -c file.c
linking:
        gcc file.o
run:
        ./a.out
>>>cat -n makefile
  1  all:
  2       gcc ls.c -o ls
  3       gcc cat.c -o cat
  4       gcc date.c -o date
  5       gcc rm.c -o rm
  6       gcc mkdir.c -o mkdir
  7       gcc -c file.c
  8       gcc file.o
  9       ./a.out
  10  obj_files:
  11      gcc ls.c -o ls
  12      gcc cat.c -o cat
  13      gcc date.c -o date
  14      gcc rm.c -o rm
  15      gcc mkdir.c -o mkdir
  16  assembling:
  17      gcc -c file.c
  18  linking:
  19      gcc file.o
  20  run:
  21      ./a.out
>>>cat -E makefile
all:$
        gcc ls.c -o ls$
        gcc cat.c -o cat$
        gcc date.c -o date$
        gcc rm.c -o rm$
        gcc mkdir.c -o mkdir$
        gcc -c file.c$
        gcc file.o$
        ./a.out$
obj_files:$
        gcc ls.c -o ls$
        gcc cat.c -o cat$
        gcc date.c -o date$
        gcc rm.c -o rm$
        gcc mkdir.c -o mkdir$
assembling:$
        gcc -c file.c$
linking:$
        gcc file.o$
run:$
        ./a.out
>>>cate
Error:: No such file or directory
>>>date
Wed Sep 30 06:41:09 PDT 2020
>>>date -u
Wed Sep 30 13:41:19 UTC 2020
>>>date -R
Wed , 30 Sep 2020 06:41:25 -0700
>>>pwd
/home/ananya/Desktop/OS1.1/Q2
>>>pwd -L
/home/ananya/Desktop/OS1.1/Q2
>>>pwd -P
/home/ananya/Desktop/OS1.1/Q2
>>>echo Ananya\nhello
Ananyanhello
>>>echo -E Ananya\nhello
 Ananya\nhello
>>>echo -n ananya\nhello
ananyanhello>>>mkdir hello
>>>ls
```

```
cat.c dele1 hello makefile ls ls.c file.c
WriteUp2.pdf cat a.out rm.c mkdir.c mkdir dele2
rm date.c date file.o
>>>mkdir -i hello
Invalid mkdir Command
>>>mkdir -v hello
Error:: File exists
>>>mkdir -v new
 New Directory created
>>>mkdir -p new
>>>ls
cat.c dele1 hello makefile ls ls.c file.c
WriteUp2.pdf cat a.out rm.c mkdir.c mkdir dele2
rm date.c new date file.o
>>>rm dele1
>>>ls
cat.c hello makefile ls ls.c file.c WriteUp2.pdf cat
a.out rm.c mkdir.c mkdir dele2 rm date.c new date
file.o
>>>rm -i dele2
Are you sure you wish to delete dele2 (Y/N)Y
 dele2 Removed
>>>ls
cat.c hello makefile ls ls.c file.c WriteUp2.pdf cat
a.out rm.c mkdir.c mkdir rm date.c new date file.o
>>>rm -f dele2
>>>ls
cat.c hello makefile ls ls.c file.c WriteUp2.pdf cat
a.out rm.c mkdir.c mkdir rm date.c new date file.o
>>>history

1 cd ..
2 ls
3 cd ~
4 ls
5 cd Desktop
6 ls
7 cd -P OS1.1
8 ls
9 cd Q1
10 ls
11 cd ..
12 ls
13 cd Q2
14 ls -i
15 ls -a
16 cat makefile
17 cat -n makefile
18 cat -E makefile
19 cate
20 date
21 date -u
22 date -R
23 pwd
24 pwd -L
25 pwd -P
26 echo Ananya\nhello
27 echo -E Ananya\nhello
28 echo -n ananya\nhello
29 mkdir hello
30 ls
31 mkdir -i hello
32 mkdir -v hello
33 mkdir -v new
34 mkdir -p new
35 ls
36 rm dele1
37 ls
38 rm -i dele2
39 ls
40 rm -f dele2
41 ls
42 history
>>>echo new
new
>>>echo line
line
>>>history -n

43 echo new
44 echo line
45 history -n
>>>history -c

>>>history

1 history
>>>exit
ananya@ubuntu:~/Desktop/OS1.1/Q2$
```