ASSIGNMENT1 PART2
Basic Linux/Unix shell

Ayush Mahant
Section A
2019353
CSSS

Basic indications-

The whole path is considered using chdir and getcwd.
The exec files for all the c files should be in the same folder.
The text file created for the history should be in that same folder.
The conventions of naming the exec files must be the same during checking.

Fork and process handling-

For each external command fork is used where in the child process, the execution takes place and in the parent process, it waits for the child to execute .

External commands

1. Ls command(explaining the ls1.c file)

Arguments passed from the main shell file are, in argc and argv .The argc stores the number of arguments sent and argv contains the arguments. First, we check if the option along with ls,
If it is null then we need to display the directories and files in the current directory which we get using getcwd. Then using opendir we open the directory with the given path and if the path is not found we print the error using perror. If error is not found we store the file names to an array which is further sorted using bubble sort and printed on the shell screen and after hiding the hidden files which in fact starts with '.'.
Then we check if along with 'ls' the option is "-1", and then if along with it filenames are given,
Then we run a while loop for displaying file names in all the paths given or directories given.

If no command is given along with “-1” the current directory’s file names are given in row wise.
Then we check if along with ‘ls’ the option is “-r”, and then if along with it filenames are given, Then we run a while loop for displaying file names in all the paths given or directories given in reverse order . If no command is given along with “-1” the current directory’s file names are given in reverse order.
Then we check if along with ‘ls’ the option is “-a”, and then if along with it filenames are given, Then we run a while loop for displaying file names in all the paths given or directories primarily along the hidden files. If no command is given along with “-1” the current directory’s file names are given along with the hidden files..
Now if multiple files are given along with the ls command then we do the same task for all the files(thus handling multiple files).
If command is wrong or the file name is not there we print “command not found or no directory found error”.

Errors handled-

1.If the child process fails to be created, an error -fork failed , is displayed.
2.If wrong option, i.e, a flag is given to be interpreted as a file name is shown as an error.
3.Wrong command error is shown if no flags or options or filenames are present in the command line.
4. For all file names while reading through “opendir” if the fine is not present, then error-directory is not found error prompts.(for all the options)
5.Also, if the shell can’t interpret certain options, then command cannot interpret error prompts.

Corner cases-

1.Multiple files have been handled in all the options.
2.Execv is used because of the multiple file handling.
3.Execl would be limited to the arguments.
4.I have sorted all the filenames as is done on bash shell.
5. Also, I have interpreted the concept of hidden files.

2**.rm command(explaining the rm1.c file)**

Arguments passed from the main shell file are, in argc and argv .The argc stores the number of arguments sent and argv contains the arguments. First, we check if the option along with rm. If it is null an error shows up.
If it is not null then we need to remove the following files provided along the command.

If the ‘rm’ command ‘-ii’ is given as the option, then the shell prompts the user if the user wants to remove the specific file.
If the ‘rm’ command ‘-v’ is given as the option, the shell prompts a message displaying that the file has been removed or not.

Error Handling-
1.If the child process fails to be created, an error -fork failed , is displayed.
2.If wrong option, i.e, a flag is given to be interpreted as a file name is shown as an error.
3.Wrong command error is shown if no flags or options or filenames are present in the command line.
4. IF no argument is provided along with ‘rm’ , error prompts saying not enough arguments.
5.Also, if the shell can’t interpret certain options, then command cannot interpret error prompts.
6. In the ‘-i’ option if the user enters no to removing files, I have displayed the same in the shell.

Corner cases-

1.Multiple files have been handled in all the options.
2.Execv is used because of the multiple file handling.
3.Execl would be limited to the arguments.
4. The user can’t have the file name starting with “-” file .This have also been handled.

3. **Mkdir command (explaining the mkdir.c)**

Arguments passed from the main shell file are, in argc and argv (multiple files handled).
If along with “mkdir “ “-v” is given as the option, the message is displayed, if the directory is created or not.We have used the mkdir function for the creation of the directory.
If “mkdir “ “-p” is given as the option, we need to create parent directories.
I have done this using recursion where fullname and subnames repeatedly come which in fact is created using the mkdir function.
If along mkdir multiple file names are given using while loop they have been created successfully.

Error handling-

1.If the child process fails to be created, an error -fork failed , is displayed.
2.If wrong option, i.e, a flag is given to be interpreted as a file name is shown as an error.
3.Wrong command error is shown if no flags or options or filenames are present in the command line.
4. IF no argument is provided along with ‘mkdir ’ , error prompts saying not enough arguments.

5.Also, if the shell can’t interpret certain options, then command cannot interpret error prompts.
6. If the directory is already there, it is not created again and the error message is prompted.

Corner cases-

1.Multiple files have been handled in all the options.
2.Execv is used because of the multiple file handling.
3.Execl would be limited to the arguments.
4. The user can’t have the file name starting with “-” file .This have also been handled.
5. Parent directories have been created.

4.
date command (explaining the date.c)

If only “date” is the command, using time_t, localtime and strftime which are provided in the time.h file, the system’s date and localtime are displayed.
If along with “date” is the command,”-u” is given using gmtime and strftime which are provided in the time.h file, the system’s date and localtime are displayed in gmtime.
If along with “date” is the command,”-r” is given and the file name is given we need to display the latest modification date of the file .This has been done using sys/stat.h and time.h files which has ctime function which gives the time and stat gives the possible file properties.

Error handling-

1.If the child process fails to be created, an error -fork failed , is displayed.
2.If wrong option, i.e, a flag is given to be interpreted as a file name is shown as an error.
3.Wrong command error is shown if no flags or options or filenames are present in the command line except for the “date” command.

4.Also, if the shell can’t interpret certain options, then command cannot interpret error prompts.
5. If for the command date -r filename , filename is wrong or not present ,error prompts in the shell.

Corner cases-

1. As the number of arguments were limited ,execl was used for simplicity.
2. Multiple file handling was not used in it.
3. File name needs to be correct.
4. Timezone is not being displayed.

5. Cat command(explaining the cat.c file)(multiple file handled)

If along with cat command “-n” is given and the files name we need to display the files information along with line numbers. This had been done using fgets and fopen.

If along with cat command “-e” is given and the file name we need to display the file information along with the ‘$’ sign.
If only cat and file names are given as command , then we display file information of all the files.

Error handling-

1.If the child process fails to be created, an error -fork failed , is displayed.
2.If wrong option, i.e, a flag is given to be interpreted as a file name is shown as an error.
3.Wrong command error is shown if no flags or options or filenames are present in the command line.
4. IF no argument is provided along with ‘cat ’ , error prompts saying not enough arguments.
5.Also, if the shell can’t interpret certain options, then command cannot interpret error prompts.
6. If file is not found ,then file not found error prompts.

Corner cases-
1.Multiple files have been handled in all the options(assuming all the file names are correct).
2.Execv is used because of the multiple file handling.
3.Execl would be limited to the arguments.
4. The user can’t have the file name starting with “-” file .This have also been handled.
5. If one file name is not correct, then the termination also stops for the other files in ‘-e’ command option.

Internal commands

1. **Cd command**

   If cd .. is the command we need to go to the previous directory using chdir(“..”) .
   If cd or cd ~ is the command we need to go to the home directory using chdir(getenv(“HOME”).
   If cd -L filename is the command then, the default change directory is validated.
   If cd --help is given we give the help of cd command.
   If the cd filename is given then, the directory changes to the given one.

   Error handled-
   File not found error

   Corner cases-

   cd ..
   cd ~
   cd
   cd .
   All these arguments have been implemented.

2. ### Pwd command

   If pwd --help is the command, we display the help provided for the pwd in our shell.
   If pwd is the command we display the current directory using getcwd.
   If pwd -P is the command we display the current directory   using getcwd(default).

   Error handled-
   Wrong commands arguments or options given along with pwd.
   Corner cases-

   The directory changes are reflected by chdir.
   The whole path is displayed.

3. ### Exit command

   If   exit --help is given, we provide the help for exit.

If exit is the command we exit using exit system call.

4. Echo command

If echo --help, is the command , we display the help provided with the echo in the shell.
If echo and some display is given we display the provided argument.
If echo -n is a command we display the given argument in the same line in the shell.
If echo -E is the command we display the given argument in the next line in the shell.

Errors handled-

File name is wrong or the options or commands have not been interpreted.
Corner cases-

The string with quotes have been displayed after removing the quotes in all the options.
The next line have also been considered in the option ‘-E’.

5. History command

We are using “a+” mode in the fopen for appending the commands to the text file.
We are globally storing the history history1.txt file whenever a new command is provided.
When we give history command ,the history stored in the history1.txt is all displayed and History of commands in the previous shell execution is also displayed.
If history -a is the command we are storing the history of the current shell execution in the history1.txt file.
If history -c is the command we are deleting everything in the text file using “w” mode in the fopen.

Error handled-
If command is not interpreted.
Corner cases-
The previous shell execution’s commands are also displayed .
The “-c “ option is also implemented for deleting the text file contents.

ASSUMPTIONS-

1. Getcwd works for the path receiver works fine.

2. History1.txt file for history is in the same path as all other exe files.
3. All exe files and the .c files are in the same folder along with makefile.
4. The size of filenames and directory names or the options are limited to upto 400- 500 bytes.
5. Readline could be used.
6. The files names are separated by space even when present in a string format that is double quotes.
7. Exit command have only one option --help
8. Could use all the general functions like mkdir and remove
9. Efficiency not considered (used bubble sort)
10. The filename iis not starting with a ‘/’.
11. The flags work on all linux based systems.
12. Realpath is not used and the exe files for all ls, mkdir.. are having the specified names I have provided.
13. The final executable file name is “./gg”

Test case1-

```
cd
pwd
cd somebody
ls -a
ls
cd ..
cd ~
cwd
cd Downloads
cd somebody
cd ..
cd ..
cd .
pwd
cd Desktop
cd ayushmahant
ls
ls -a fork newassign
ls -1 fork newassign
ls -r fork
pwd
```

Test case2-

```
history
ls
mkdir p2/p3/p4
mkdir -p p2/p3/p4
pwd
mkdir p2
mkdir -v p2
mkdir -v p5 p6 p7
ls
rm p5
rm p5 p6 p7
mkdir bv bvb bvb
cd ..
ls
date
history
date -u
date -rre
date -r overall.c
history --help
cat
cat ls1.c
cat ls1.c rm1.c
cat -i ls1.c rm1.c
cat -n ls1.c rm1.c
cat -e ls1.c rm1.c
cat gjglsgl
rm p2
rm bvb bvbn
mkdir bvv ggr
ls
rm -v bvv bvbn
mkdir bvv ggr
rm -i
rm -i bvbv
rm -i bvbn
rm -i bvbtrr
echo “fjf jfjdj kf”
echo
echo --help
```

```
cd --help
history --help
pwd --help
exit --help

echo -E “dff fdf  ere”
echo -n “ee  ete et “
pwd -P
mkdir frfdgsfdhg
history
history -c
```

____-