Operating Systems
Assignment 1
National University of Computer and Emerging Sciences
Department of Computer Science

- ***You must submit .sh & .c files only through Google Classroom.***
- ***Plagiarism will result in ZERO marks in all the future assignments too***
- **Due: Friday, October 7th, 2022.**

Part1 [5+10+10 = 25 Marks]

Task1:

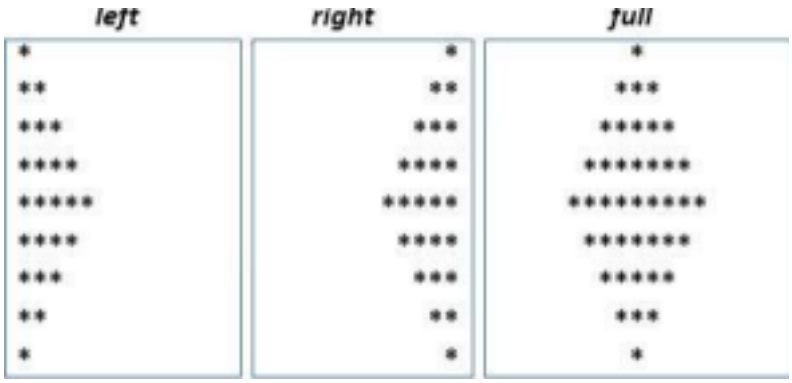
Write a script that takes two parametric variables: **[10]**

./script pattern_option number

Where,

pattern_option = {left, right, full}, number = could be any positive integer

You have to output the pattern depending on the input parameters as shown in Fig1. Generate error messages for invalid input parameter.



Task2

Write a script that displays a main menu and perform tasks based on the input value. Valid *input values = {1, 2, 3, 4, exit}.*
The different options *1,2,3,4* will display the output as follow:

1. Input a filename from user and display permissions of that particular file. Then invert the permissions e.g. If permissions were *r-x* change them to *-w-*. Then again display the updated permissions of that file.

2. Input a *filename* and a *string* and search it in the file. Output the lines of file where that *string* is found. But if the *string* contains a *dot(.)* it means any character can fill the place. For example: *string = c.t = {cat, cot, c t,}*

3. Create a file *dummy.txt* and add the content of all the files in the current directory to dummy. But copy the content in such a way that if files in current directory = {f1, f2, f3, f4, ..., fn}. Then copy first *N* lines of files at even location *{f2, f4,.}* and last *N* lines of files at odd location *{f1,f3,. }*. Input value of *N* from user.

4. Input a filename from user and check modified date of that file. If modified date is greater than 24 hours from the current time change the modified date to current date. Along with displaying the output on terminal, maintain a log file that contains the information of the script. Format of the log file is given below:

Format of the log File

Option 01 selected at *date and time*
File name: *filename.txt*
Permissions of *filename.txt*: *Show permissions*
Permissions changed
Updated Permissions of *filename.txt*: *Show permissions*

Option 02 selected at *date and time*
Filename: *filename.txt*
String: *string*
Output all the lines in *filename* where *string* is found.

Option 03 selected at *date and time*
Files at odd location: *f1, f3, f5*
Files at even location: *f2, f4*
Dummy.txt is created and *N* lines of each file copied in it.

Option 04 selected at *date and time*
Filename: *filename.txt*
Current modified date: *date*
Modified time updated or not

Option exit
Script terminated at *date and time.*

Task3

Fig 2 shows the basic hierarchy of the files. Input (read) the name of the *Main Directory*. In the *Main Directory* there are multiple directories *Dir_1, Dir_2, ..., Dir_N*. In each subdirectory there are multiple files with different extensions. Now insert a loop that takes a string value as input and terminate when user enters *any string starting with e*. The hierarchy of the files will change when user enters either *forward or backward.*

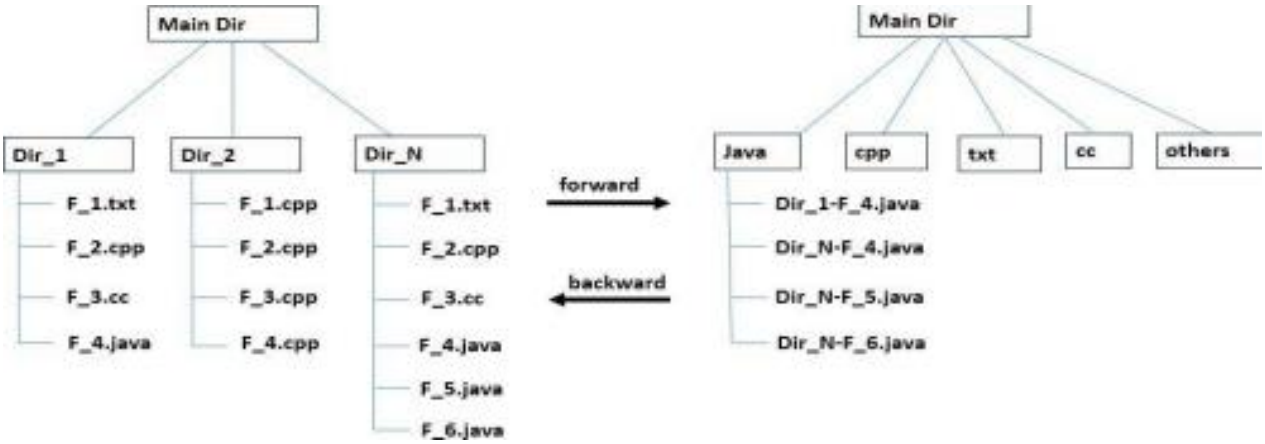


Figure 2: Hierarchy of the main directory

1. **Forward:** Create 5 different directories namely *java, txt, cpp, cc and other*. Copy each file in the directories depending on their file extension. Before copying the file follow the protocol to name the file as *DirectoryName-Filename.extension*. Delete all the extra folders from *main Dir* namely *Dir_1, Dir_2,. , Dir_n*.

2. **Backward** Now in the backward cycle, extract name of directories from the file name. Create the directories again and copy the files to their original directory restoring their default names.

Part2 [25 Marks]

You are to use any distribution of the Linux operating system to create two C programs. Program *consumerProducer.c* and program *producerConsumer.c*. These two programs communicate with each other through ***ordinary pipes.***

The producerConsumer program will

1) open a text file with the name *editSource.txt* and will write all sentences of the created file on one pipe to be read by the consumerProducer program.

The consumerProducer program will

1) read from the pipe, and it will
2) count the number of characters,
3) count the number of words, and
4) count the number of lines in the received sentences. It will then
5) create a file named *theCount.txt*,
6) write the produced output to *theCount.txt,* and will
7) write the contents of the file *theCount.txt* on a second pipe to be read the producerConsumer process.

The producerConsumer process will display the received file on standard output.

The contents of the *editSource.txt* file is as follows:

Source code
represents the part of
process that
contains the programming
language itself. You may
use a text editor to
write your source code file.

The format of *theCount.txt* file is:

Number of characters: number of characters found in *editSource.txt*
Number of words: number of words found in *editSource.txt*
Number of lines: number of lines found in *editSource.txt*

Note:
Source code of both your producerConsumer.c and consumerProducer.c files should display the output produced by your solution.
Your *solution must use Ordinary Pipes.*

Good Luck 😊