Assignment 1

CS330: Operating Systems

1 mygrep Implementation - 30 Marks

Implement **mygrep** which takes two arguments, a string and a path (file or directory). **mygrep** should search recursively for the given string in all regular files present in the directory sub-tree. It should output matching lines with full file path. You will submit a single C source file with implementation of **mygrep**.

The implementation of **mygrep**, when executed as **./mygrep Kanpur IITK** should return output as mentioned under **output format** for the directory structure given in Figure 1. Here **Kanpur** and **IITK** are the string and path arguments, respectively.

The directory structure mentioned in Figure 1 is only for representation purpose and the actual test cases will have deeper directory structures.

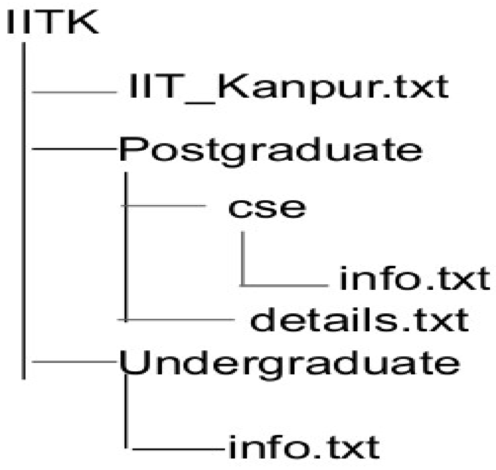


Figure 1: Directory Structure

output format

IITK/Postgraduate/details.txt:Kanpur Postgraduate courses in Engineering offer Master of Technology (M.Tech), MS (R) and Ph.D. degrees ...

IITK/Undergraduate/info.txt:IIT Kanpur offers four-year B.Tech programs in Aerospace Eng ...
IITK/IITK_Kanpur.txt:Faculty Building, IIT Kanpur ...
IITK/IITK_Kanpur.txt:Under the guidance of economist John Kenneth Galbraith, IIT Kanpur was the first institute in India ...

You can use below mentioned APIs to implement this part of the assignment. Refer to man page of these APIs to know about their usage.

- opendir
- readdir
- closedir
- chdir
- open
- read
- close
- stat

2 Implement @ and $ operators - 25 Marks

As part of this task, you need to implement two operators (@ and $) which have following functionalities. You should strictly use **pipe** system call to implement this part of the assignment. Usage of temporary files to store intermediate results will result in zero marks. You will submit single c source file with the implementation of both these operators.

2.1 @ operator

The @ operator takes two arguments, a string and a path. The program should output a number corresponding to count of lines that contain the string in the path provided.

The output of executing **./part2 @ Kanpur IITK** should be equal to **grep -rF Kanpur IITK | wc -l**. This is the first check that you need to do. Here **Kanpur** and **IITK** are the string and path arguments, respectively.

You are free to use **grep** or your implementation of grep from part1 for this task.

2.2 $ operator

The $ operator takes four arguments, a string , a path, an output file and a command. You should get all lines that contain the string in the given path and write the result to output file as well as execute the command on the search result.

The output of executing **./part2 $ Kanpur IITK output.txt sort** should be equal to **grep -rF Kanpur IITK |tee output.txt | sort**.

The output of executing **./part2 $ Kanpur IITK output.txt wc -l** should be equal to **grep -rF Kanpur IITK |tee output.txt | wc -l**. This is the first check that you need to do. Here **Kanpur** and **IITK** are the string and path arguments respectively.

You are free to use **grep** or your implementation of grep from part1 for this task.

You can use below mentioned APIs to implement this part of the assignment. Refer to man page of these APIs to know about their usage.

- pipe
- dup, dup2
- fork
- exec
- open

3 Directory Size - 45 Marks

Write a program to add up sizes of all files in the given directory. Your program will be provided with the path to the root directory (IITK Directory in Figure 2). It should perform the following actions

- Assume that there are N immediate child sub-directories under the root directory. For each immediate child sub-directory under the provided root directory, your program must fork a new process P_i(i will range from 1 to the total number of immediate child sub-directories) that recursively looks in all the sub-directories for the files and compute sum of their sizes.
- Your program should output the size of root directory as well as the size of all immediate child sub-directories.
- Let us see an example of the folder structure given in the Figure 2. We will be providing the path to root directory(IITK) has a command line argument to your program. The root directory is having two immediate child sub-directories (Postgraduate, Undergraduate). So your program has to fork two processes and add up the size of the files in its corresponding directories.

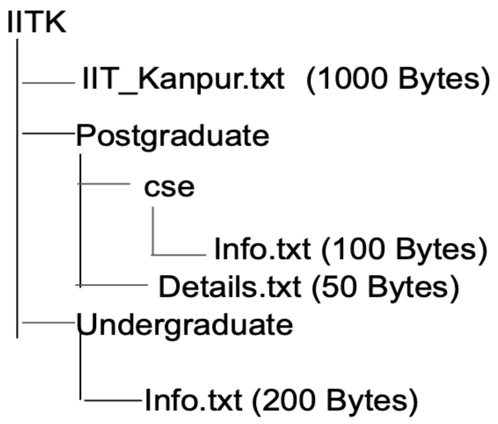
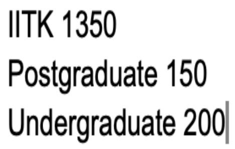


Figure 2: Folder Structure

- Your program should output the names of root, immediate child sub-directories and their corresponding size. The expected output of the Figure 2 directory structure is shown in Figure 3.



```
IITK 1350
Postgraduate 150
Undergraduate 200
```

Figure 3: Expected output

- We have provided another sample explanation in the folder(Part_3\Samples_Explanation). You can also refer that.

3.1 Validation

Your submitted code will be validated based on the following criteria.

- If there are N immediate child sub-directories, your program should fork at least N process.
- Your program should only use pipe (pipe() system call) to communicate between processes. You should neither use sockets nor use any temporary files to communicate. Any violation will result in zero marks.
- There is no restriction on the number of pipes. You can create as many as it requires.
- In top of your file(part3.c) explain your implementation details and you should add comments to your code so that it will be helpful for us to go thorough your code if required.
- Before submitting the file, kindly verify your code with sample test cases which is provided in the directory "Part_3\Test_Cases".

3.2 Useful links

- http://www.cs.loyola.edu/ jglenn/702/S2005/Examples/dup2.html
- http://web.mst.edu/ ercal/284/PipeExamples/Pipe4.c

.

4 Submission guidelines

- You should name submission files as part1.c, part2.c and part3.c
- Source code for each part of the assignment should be placed in respective folders. For example, part1.c should be placed in **/Assignment1/Part_1/src/**. Don't change the structure of Assignment1 folder. You need to submit a zip of the entire folder (Assignment1.zip)
- Source code should be properly commented.
- Some test cases for each part of the assignment are made available in respective folders (Test_Cases). Passing of these test cases will fetch you some mark. We will use extra test cases as part of the evaluation to test the robustness of your implementation.
- Submission will be through canvas. Link for which will be made available through piazza prior to the submission date.