Programming Assignment 1
CS962: Operating System Principles

Due Date: 8th August 2025, 11:59 PM

1 Be a Seeker

In this part you will learn about searching a string in a file.

1.1 Init [20 Marks]

Write a C program `Q1.1/init.c` which takes two arguments, a string and a file path. It should then search for the given string in that file and print **FOUND** if string is found, otherwise print **NOT FOUND**.

Syntax

```
$ ./init <search_term> <file_name>
# Here the <search_term> will be a single word
# consists of only alphanumeric letters.
```

Example

Consider a file `courses.txt` with the following contents:

```
esc101 is a great course to start with.
phy201 is a course number for physics department.
eco201
ee201
cs330 is a course for intro to operating system.
cs768
cs234
ec212
chm112
mt216cs193
```

```
$ ./init cs330 courses.txt
FOUND
```

Output

Print the output as mentioned in the problem. *Stick to the exact format.*

Error handling

In case of any error, print **"Error"** as output.

System calls and library functions allowed

You **must only use** the below mentioned APIs **to perform file handling operations** in this question.

```
    - open                  - read
    - close                 - write
    - lseek                 - malloc
    - strlen                - free
    - strcpy                - strcat
    - strcmp                - strto* family
    - ato* family           - printf family
    - exit
```

Testing

Run `Q1.1/run_tests.sh` script to check whether your implementation passes the test cases or not. A correctly implemented program would generate following output:

```
Test case 1 passed
Test case 2 passed
...
...
Test case 8 passed
```

1.2 Level Up [30 Marks]

Write a C program `Q1.2/level_up.c` which takes 5 arguments, a string, a file path and three positive integers where first integer denotes the start offset, second integer denotes the end offset in the given file and third integer denotes the order to search. If the value of `order` input is 0 then search in forward direction, i.e. from start offset to end offset else search in reverse order, i.e. from end offset to start offset. It should then search for the string according to the given order in that file in the provided offset range and print **FOUND** if string is found in the given offset range, otherwise print **NOT FOUND**.

Syntax

```
$ ./level_up <search_term> <file_name> <start_offset> <end_offset> <order>

# Here the <search_term> would be a string of alpanumric
# letters only. The value of <order> would be either 0
# or 1.
```

Example

Consider a file `courses.txt` with the following contents:

```
cs330 is a course for intro to operating system.
cs768
cs234
ec212
chm112
mt216cs193


$ ./level_up cs330 courses.txt 2 50 0
NOT FOUND

$ ./level_up 03 courses.txt 0 50 1
FOUND
```

```
# Here please note that in backward search from
# offset 50 to offset 0 the string "03" is present
# but it is not present for forward search.
```

Output

Print the output as mentioned in the problem. *Stick to the exact format.*

Note

- The file start with offset value as 0.

Error handling

If any offset (start or end) is not correct according to the given file then print **"Invalid Offset"**. In case of any other error, print **"Error"** as output.

System calls and library functions allowed

You **must only use** the below mentioned APIs **to perform file handling operations** in this question.

```
    - malloc/ free          - strlen
    - strcpy/ strcat        - strstr
    - ato* family           - printf family
    - exit                  - All file related syscalls
```

Testing

Run `Q1.2/run_tests.sh` script to check whether your implementation passes the test cases or not. A correctly implemented program would generate following output:

```
Test case 1 passed
Test case 2 passed
...
...
Test case 12 passed
```

2 Pipe Dream: The Great Redirection Adventure [50 Marks]

In this part you will implement Command Pipelining feature of Linux command-line shell. Write a C program `Q2.1/pipes.c` which takes a string as an input. The string will contain shell commands augmented with pipe symbols. Your task is to parse this string and execute all the commands in a pipelined way, i.e. the output from the first command will be the input for the second command and so on.

Examples with command pipeline

```
$ cat input.txt | grep cs330
# Here the first command "cat input.txt" will show
# entire content of the file which will become
# input to the second command then second command
# "grep cs330" will output strings containing cs330.
```

Syntax

```
$ ./pipes <command_string>
```

Input

A string containing shell commands augmented with pipe symbols. The string will not contain any single inverted (') or double inverted (") commas except at the start and end of the string (see the examples below).

```
$ ./pipes "cat input.txt | grep cs330"
$ ./pipes "echo hello"

# Here the command string contains double inverted commas
# only at the start and end of the command string.
```

Output

The output generated after executing all the given shell commands in a pipelined way.

Note

- The input can have at most 16 pipe symbols.

Error handling

In case of any error, print **"Error"** as output.

System calls and library functions allowed

You **must only use** the below mentioned APIs to implement this question.

```
    - fork/ exit            - malloc/ free
    - exec* family          - strcpy/ strcat/ strcmp
    - strtok                - strto* family
    - ato* family           - wait/ waitpid
    - printf, sprintf       - strlen
    - pipe                  - All File related syscalls
```

Testing

Run `Q2.1/run_tests.sh` script to check whether your implementation passes the test cases or not. A correctly implemented program would generate following output:

Test case 1 passed
Test case 2 passed
...
Test case 12 passed

3 Submission

- Make sure that your implementation doesn't print unnecessary data. Your output should match exactly with the expected output specified in each question.
- You have to submit zip file named your_roll_number.zip Eg: 12345.zip containing **only** the following files in specified folder format:

```
12345.zip
|
|------ 12345
              |-------- Q1.1
              |            |----- init.c
              |
              |-------- Q1.2
              |            |----- level_up.c
              |
              |-------- Q2.1
                           |----- pipes.c
```

- Your submission will be evaluated on a recent Linux distribution (Ubuntu 22.04) with GCC compiler.
- Include a README report describing instructions to compile and execute the program.
- Use of Boost library is not allowed. Boost is not part of the standard C/C++ distribution.
- You will get 0 if we identify any plagiarism in your submission.

All the best. Looking forward to the submissions !!..... PS: Start Early.