CS3411 Project 5 - Filter I/O and Run a Program

Due : December 10, 2021 midnight

In this project, you will be developing a *filter* program that forks a child and then executes a program passed as an argument and intercepts all output and input from that program. The syntax for the execution of the *filter* program is given by:

`filter <program name> <arguments>`.

Any input (on stdin) to the filter program which starts with a slash character (/) is a control command to the filter program. All other input is passed on to the executed program subject to the current mode of operation. The program has three modes of operation:

1. Input only mode : In this mode, any input data from the standard input is sent to the child process. The output from the child program is not displayed. This mode is entered with the command `/i`.
2. Command mode : In this mode the program should only accept commands starting with the slash character. Any other input should result in an appropriate error message. This mode is entered either by user request through the command `/c` or when the child process completes its execution.
3. Input/Output mode : In this mode the program should send all input data (except commands) to the child process's input and return output data from the child upto a maximum of specified lines of data onto the standard output of the filter command. The mode is entered through the command `/o`.

Example:

```
filter cat
```

should cause the program filter to fork a child, create three pipes which become the input, the output and the error file descriptors of the forked child and then exec the program `cat`. Once the program starts executing, filter program will be in command mode, awaiting input. Then the command :

```
/i
```

should put the filter program into the input only mode such that all data that is read from the standard input is sent to the program `cat`'s input unless it starts with the control character. If the user enters `/o` than the program should display data from the `cat` program's output upto a maximum lines of text (max-text, default 20). If the program has more data than the max-text, or when the available data is displayed, the filter program should go into the command mode. Note that any data on standard error of the child should immediately be sent to the standard error of the parent. Similarly, a message about the termination of the child should be immediately sent to the standard error of the parent together with the return value (i.e., *The child < pid > has terminated with code < nn >*).

The following commands must be implemented:

1. `//` : Pass the character `/` as input to the program.
2. `/i` : Go to input only mode.
3. `/o` : Go to input/output mode.

4. `/c` : Go to command mode.
5. `/m <integer>` : Set the max-text, maximum number of lines to be displayed.
6. `/k <integer>` : Send the child process indicated signal.

After receiving each command, the program should send back a prompt indicating the current mode and if there is more data to be displayed. The prompt syntax is :

```
<pid> m <more> #
```

where `<pid>` is the process id of the child process, `m` is the current mode (i (input only), c (command), o(input/output)), optional `<more>` is the text `more` if there is data available to be displayed, and lastly the pound character.

Your program should remain responsive to the input of commands **at all times**. This means you cannot simply read and write, but you have to make sure that reading and writing will not block you. I strongly recommend the use of *select* system call and I recommend against using non-bloking i/o.

Pragmatics

Source Template
You are provided a source template that includes:

- Makefile with the following commands:
  - all - compile project into filter.
  - filter - compile and generate filter executable
  - clean - removes executable and object files.
  - submission - generates prog5.tgz with all source files needed for submission. Upload this file to Canvas.
- filter.c - C source for implementation of filter

Formatting

- Commands and input for the exec'd program will be entered separately. Thus a line contains either a command or data for the program. Data may include escape characters ("//" which should be sent as '/').
- When entering input output mode (/o) the parent should print up to max-text number of lines (a line is any text up to and including a new line character). Preceding the output from the child should be the prompt as given on the spec.
- After receiving each command, the program should send back a prompt indicating the current mode and if there is more data to be displayed. The prompt syntax is :

  ```
  <pid> m <more> #
  ```

- A line is any text up to and including a new line character
- Say there's 29 lines of output from the child. When we enter /o the first 20 lines should be printed. If we enter /o again, the remaining 9 should be printed.

Format Example

```
An example is shown below.
Note that any lines starting with % are showing what is entered on stdin of the parent.
Lines without the % are what the parent writes to stdout.

% ./filter <pgm> <args>
% /i
<pid> i #
%<this text should be sent to pgm as input>
%<this too is input>
%/o
<up to max-text lines of output (which may have from prior input but not yet printed).
 Since the prompt printed by the parent says more, there is more to print>
<pid> o more #
%/o
<up to max-text lines of output (continuing from where the last print left off).
Since the prompt here does not say more, there is no more buffered output from the child>
<pid> o #
%<this text sent to pgm>
%/m 25
<pid>o #
%/i
<pid> i #
%<text to send to pgm>
%/c
<pid> c #
%/k 9
The child <pid> has terminated with code <nn>
```

Submission Requirements

Your submission must be written in C.

Use Canvas to submit a tar file named `prog5.tar` that contains generated by running make submission.

Ground Rules and Restrictions This assignment may be provided with updates as the project goes on. Any updates or revisions will be posted to the assignment page on Canvas. Make sure that you attend the classes and watch the class mailing list.
You may not borrow or reuse any code from other resources, and all the code must be your own. In addition, the following rules apply:

- You may discuss the program with others.
- You may not bring any printed/written material into the discussion with you. (You may not show anyone your code; you may not view the code of anyone else. This includes others both enrolled in the course and others not enrolled in the course.)
- You may generate written material during the discussion, but it must be destroyed immediately afterward. Don't carry anything away from the discussion.

- If you are in doubt about the acceptability of any collaboration activity, I expect you to check with me before engaging in the activity.