Exercise 02

For this exercise, you're going to implement input and output redirection. We'll be using the same techniques used by the shell when you use `<` or `>` as part of a command.

Inside our program, we'll fork() a child process. The child will use execl() to run the cat command with no command line arguments. When run like this, cat will just read everything it can from standard input and write everything to standard output. However, before starting cat, the child will replace standard input with a file descriptor that reads from a file, “input.txt”. Likewise, it will replace standard output with a file descriptor that writes to a file, “output.txt”. So, when cat gets started it may think it's reading and writing to the terminal, but it will really be copying from the input.txt file to the output.txt file. This is typical of how fork() and execl() are used. You create a new child process, then, you get to make adjustments in its execution environment before you start the program the child is supposed to run.

I've written a partial implementation of redirect.c. It creates a child process then has the parent wait for the child to finish. You get to add the interesting parts. You'll need to add code to do the following. I've left little comments like “// ...” in the code, where you need to add something.

- We need to open the input file and replace the child's standard input with a file descriptor that reads from “input.txt” instead. This is normally done in three steps. After you create the child, have it use the open() system call to open the file “input.txt” for reading. This will give the child a file descriptor that reads from “input.txt”. Then, use the dup2() system call to close the child's old file descriptor for standard input (there's a peprocessor constant for this, STDIN_FILENO) and replace it with another file descriptor that reads from “input.txt”. The dup2() call does both of these things at once. You will want to check its documentation to see how to use it. Now, close the original file descriptor you got when you opened “input.txt”; with the dup2() call, you just duplicated that file descriptor to replace standard input, so you no longer need the original file descriptor you got back from open().
- Follow similar steps to open “output.txt” for writing and to replace the child's standard output file descriptor with a file descriptor that writes to “output.txt” instead.
- Use execl() to run the cat command in the child, after you’ve changed the files it’s using for standard input and output. Remember, if successful, execl() will not return. Still, it might be good to put an error message or a call to fail() after the call execl() call, to help you diagnose the problems if you have trouble starting up cat. (This shows one reason it’s nice to have a separate output stream for error messages, standard error. After you redirect standard output to a file, you can’t use it to print to the terminal any longer, but you still have standard error, which still goes to the terminal.)

Once your program is working, it shouldn't print any output, but it should successfully copy the file “input.txt” to a

new file, “output.txt”. When you're done, turn in your source code for your completed redirect.c program. Submit the file to the Gradescope assignment named EX02.