| CSE 3100 Systems Programming | Spring 2024 |
|---|---|
| Lab #6 | |

Welcome to Lab 6! Read each question carefully before beginning work. Submit your .c file to gradescope. Note a Makefile is provided.

**Part 1. (100 points) Guess My Number**. In the **Guess My Number** game, two people, C and P, agree upon a range of integer values. One person, C, picks a random integer within the range. The other person, P, tries to guess the number. For each guess P makes, C tells P if the guess is correct, too large, or too small. P then makes another guess based on the feedback. This process repeats until P has guessed the correct number. P tries to minimize the number of guesses.

In this problem, we will use two processes, a parent process and a child process, to simulate the game. The parent process plays person P, and the child process plays person C. The child process generates a random number between 1 and `MAX_VALUE` (defined as 1000), inclusively, for the parent process to guess. The parent process and the child process use two pipes for two-way communications.

The process works as follows.

1. The child process tells the parent process the largest possible value.
2. The parent process makes a guess (picking an integer in the range) and sends it to the child process.
3. The child process checks if the guess is correct and sends the result to the parent process. The result is 0 if the guess is correct, 1 if the guess is smaller, or -1 if the guess is larger. If the guess is correct, the child process also sends a final message and then exits. The final message includes the number of attempts the parent has made to guess the correct number.
4. If the guess is not correct, the parent process adjusts the range, based on the feedback from the child process, and goes to Step 2.
5. If the guess is correct, the parent receives the final message from the child process and prints it to `stdout`.

The parent process uses a binary search to minimize the number of the guesses (over many games). `guess_my_number()` in the starter code demonstrates the process. It prints the guesses the parent process makes and the final message from the child process. The function does not involve multiple processes or pipes.

The task in this problem is to complete the starter code so the parent process and child process can play the game by communicating through pipes. Search `TODO` in the starter code to find the places where work needs to be done. The detailed steps are described in comments. Most of the steps are operations on pipes, e.g., reading from or writing to a pipe. Below are some requirements and suggestions on the implementation.

- Integers are stored in variables of type `int`. An integer is written to the pipe as 4 (`sizeof(int)`) bytes in a binary format.
- The final message is transmitted as a sequence of ASCII characters. It ends with a new line character. The parent does not assume the message has a specific length. Therefore, it has to read characters until it finds a new line character. The child does not send a NULL character to the pipe to indicate the end of the sequence of characters (that is, the sequence is not a string).
- The child process calls functions to operate on `gmn`. It does not access fields in the structure directly.
- Put commonly used operations, e.g., writing an integer to a pipe, in functions.
- If `read()` or `write()` fails, call `die()` to exit.

The program `guess-my-number` takes optional arguments from the command line. An argument can be one of the following.

- A positive integer. It will be used as the seed for generating pseudorandom numbers. If no seed is specified on the command line, a default value is used.
- `demo`. Call the demo function and exit. It helps us to test our implementation using two processes.

The following are two sessions of running the program. The first session calls the demo function. The second session is from our implementation using two processes.

```
./guess-my-number demo 3000
My guess: 500
My guess: 250
My guess: 375
My guess: 312
My guess: 281
My guess: 265
My guess: 273
My guess: 277
My guess: 279
My guess: 280
It took you 10 attempt(s) to guess the number 280.
$ ./guess-my-number 3000
My guess: 500
My guess: 250
My guess: 375
My guess: 312
My guess: 281
My guess: 265
My guess: 273
My guess: 277
My guess: 279
My guess: 280
It took you 10 attempt(s) to guess the number 280.
```