Operating Systems
Jacobs University Bremen
Dr. Jürgen Schönwälder

Module: CO-562
Date: 2020-10-22
Due: 2020-10-29

OS 2020 Problem Sheet #7

Problem 7.1: *math quiz using an online oracle* (2+3+2+2+1 = 10 points)

Write a program that writes random mathematical questions to the standard output and then reads an answerf from the standard input until the correct answer has been given and the number of attempts to provide an answer has been exceeded. Answers are restricted to the range $[1..100]$.

```
$ ./quiz
Answer questions with numbers in the range [1..100].
You score points for each correctly answered question.
If you need multiple attempts to answer a question, the
points you score for a correct answer go down.

Q1: What is the number of inches in a yard?
8 pt> 36
Congratulation, your answer 36 is correct.
Your total score is 8/8 points.

Q2: What is the smallest number that can be written as the sum of of 2 squares in 2 ways?
8 pt> 42
Too small, try again.
4 pt> 70
Too large, try again.
2 pt> 55
Too large, try again.
1 pt> 50
Congratulation, your answer 50 is correct.
Your total score is 9/16 points.

Q3: What is the largest cube in the Fibonacci sequence?
8 pt> 1
Too small, try again.
4 pt> 2
Too small, try again.
2 pt> 3
Too small, try again.
1 pt> 5
Too small, the correct answer was 8.
Your total score is 9/24 points.

Q4: What is a composite number, its proper divisors being 1, 2, 3, 6 and 9?
8 pt> ^D
Your total score is 9/24 points.
```

You obtain the questions from a web service. Since you do not want to implement a network protocol yourself (yet), you simply run a tool like `curl` to fetch questions and you parse the output generated by curl. An example invocation of `curl` in your shell may look like this:

```
$ curl -s 'http://numbersapi.com/random/math?min=1&max=100&fragment&json'
{
 "text": "a composite number, its proper divisors being 1, 2, 3, 6 and 9",
 "number": 18,
 "found": true,
 "type": "math"
}%
```

The URL requests a math question with the answer being in the range 1 to 100. If the request is successful, the program returns a JSON document, which includes the question text, the correct answer, the indication that an answer was found, and the category of the lookup. (In addition to math, you can ask for trivia questions.)

a) Implement a function `char* readall(int fd)` that reads data from the file descriptor `fd` until the end of file has been reached and returns all data in a dynamically allocated string.

b) Implement a function `char* fetch()` that creates a pipe and subsequently runs a child process to executes `curl` in order to fetch a question as a JSON document. The result produced by the child process executing `curl` is directed into the write end of the pipe. The parent process reads the data from the read end of the pipe (using `readall()`) and returns a pointer to malloc'ed memory holding the result returned by `curl`.

   Do not use `popen()` or `system()`. Do not use the `libcurl` C API.

c) Write a function or two to extract the question and the right answer from the JSON response. You do not need to implement a full JSON parser, it is sufficient to look for the `text` and number fields.

d) Write a function `unsigned play(unsigned n, unsigned score, char *text, long answer)` that receives the number of the current question, the current score, the question text and the answer. The function posts the question and iterates through the attempts to answer the question. The function returns the updated score.

e) The main program implements a loop fetching a question, parsing it, and then playing one round of the game. This loop continues as long as the standard input has not reached the end of file. Your program should not create any zombie processes.

Make sure that your program handles error situations and is robust regarding malicious inputs. It is recommended that you test your program using `valgrind`, a tool that can identify a number of programming errors.