Operating Systems
Constructor University
Dr. Jürgen Schönwälder

Module: CO-562
Date: 2024-11-01
Due: 2024-11-08

Problem Sheet #9

Problem 9.1: *computer science multiple-choice quiz* (3+2+3+2 = 10 points)

Write a program that writes multiple choice questions to the standard output and then reads answers from the standard input until the correct answer has been given or the number of attempts to provide an answer has been exceeded. The game repeats until the user ends the standard input (by pressing CTRL-D on Linux/MacOS or CTRL-Z on Windows). The game then shows the final score.

```
$ ./quiz
Answer multiple choice questions about computer science.
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.

This mobile OS held the largest market share in 2012.

[a] Symbian
[b] Android
[c] BlackBerry
[d] iOS

(8pt) > d
Congratulation, answer [d] is correct. Your current score is 8/8 points.

The programming language "Swift" was created to replace what other programming language?

[a] Ruby
[b] C#
[c] Objective-C
[d] C++

(8pt) > c
Congratulation, answer [c] is correct. Your current score is 12/16 points.

What is the name given to layer 4 of the Open Systems Interconnection (ISO) model?

[a] Session
[b] Transport
[c] Data link
[d] Network

(8pt) > d
Answer [d] is wrong, try again.
(4pt) > a
Answer [a] is wrong, try again.
(2pt) > b
Congratulation, answer [b] is correct. Your current score is 14/24 points.

What does GHz stand for?

[a] Gigahatz
[b] Gigahotz
[c] Gigahetz
[d] Gigahertz
```

```
(8pt) > ^D
Your current score is 14/24 points.

Thanks for playing today.
Your final score is 14/24 points.
```

Your program obtains 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 produced by curl. An example invocation of `curl` in your shell may look like this:

```
$ curl -s 'https://opentdb.com/api.php?amount=1&category=18&type=multiple'
{
  "response_code": 0,
  "results": [
    {
      "category": "Science: Computers",
      "type": "multiple",
      "difficulty": "medium",
      "question": "What was the name given to Android 4.3?",
      "correct_answer": "Jelly Bean",
      "incorrect_answers": [
        "Lollipop",
        "Nutella",
        "Froyo"
      ]
    }
  ]
}
```

The URL requests a single multiple-choice question from the computer science category 18. If the request is successful, the curl program returns a JSON document, which includes the question, the correct answer, as well as several wrong answers.

Further information how to structure your implementation is provided in the following header file:

```c
/*
 * quiz/quiz.h --
 */

#ifndef QUIZ_H
#define QUIZ_H

typedef struct {
    unsigned n;          /* current question number (starting at 1) */
    unsigned score;      /* current total score */
    unsigned max;        /* possible max score */
    char *question;      /* next question (dynamically allocated) */
    char *answer;        /* next answer (dynamically allocated) */
    char *choices[4];         /* */
} quiz_t;

/*
 * Fetch the content of the given url by running 'curl' as a child
 * process and reading the response from a pipe. The response is
 * returned as a malloc'ed string, or NULL if there was an error.
 *
 * Implement the fetch() function in a fetch.c module.
 */

extern char* fetch(char *url);
```

```c
/*
 * Parse a JSON encoded msg and fill the next question into the
 * quiz_t. Use a JSON parsing library (e.g., jansson or json-c). The
 * function returns 0 or -1 if there was a parsing error.
 *
 * Implement the parse() function in a parse.c module.
 */

extern int parse(quiz_t *quiz, char *msg);

/*
 * Play one round of the quiz game by first fetching and parsing a
 * quiz question and then interacting with the user. The function
 * returns 0 or -1 if there was an error.
 *
 * Implement the play() function in a play.c module.
 */

extern int play(quiz_t *quiz);

#endif
```

a) Implement a function `char* fetch(char *url)` in the file `fetch.c` that opens a pipe to `curl` running as a child process. The function returns the response obtained by `curl`.

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

b) Implement a function `int parse(quiz_t *quiz, char *msg)` in the file `parse.c` that parses the JSON input and adds the next question and the expected answer to the quiz data structure. Consider to use a JSON parsing library such as `jansson` or `json-c`.

c) Implement a function `int play(quiz_t *quiz)` in the file `play.c` that plays one iteration of the game.

d) Implement a main program in the file `quiz.c` that plays the quiz in a loop until either the user did end the standard input or the user pressed CTRL-C to interrupt the program.

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.