Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Робототехники и комплексной автоматизации

КАФЕДРА Системы автоматизированного проектирования (РК-6)

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ №1

Студент	Журавлев Николай Вадимович
Группа	РК6-62б
Тип задания	Лабораторная работа
Тема лабораторной работы	Многопроцессорное программирование

Студент _____ **Н.В. Журавлев**
подпись, дата *фамилия, и.о.*

Преподаватель _____ **В.Г. Федорук**
подпись, дата *фамилия, и.о.*

Оценка _____

Москва, 2022 г.

Оглавление

Текст задания

Составить программу, которая заданное число раз (для определенности 5) через определенный временной интервал (5 сек.) повторяет на экране запрос и ожидает стандартный ввод. Программа должна завершаться в случае корректного ответа на запрос или после исчерпывания заданного числа запросов. Рекомендуется использовать два процесса: один (например, отцовский) обрабатывает стандартный ввод, а другой - периодически выводит запрос.

Описание структуры программы и реализованных способов взаимодействия процессов

В функции main с помощью функции getpid получаем pid отцовского процесса. Затем вызываем функцию fork, после чего создаётся 2 процесса.

Далее, если мы находимся в отцовском процессе, то вызывается функция handler_input, в которой создаётся массив char размером BUF_SIZE (Константа определяющая размер буфер). Затем, используя функцию read, этот массив заполняется. После, проходя по каждому символу правильного ответа в цикле, сравниваем каждый символ в буфере и в правильном ответе. Если символ не совпал, то функция завершается с кодом 1 и вызов функции handler_input из main повторяется. Если введен верный ответ, то функция завершается с кодом 0, и в main завершается дочерний процесс.

Если мы находились в дочернем процессе, то вызывается функция print_question, в которой выводится через цикл for 5 раз вопрос, затем ставится пауза на 5 секунд. После завершения функции в main завершается родительский процесс.

Описание основных используемых структур данных

int parent_pid – хранит pid родительского процесса.

int pid - pid дочернего процесса для отцовского процесса и 0 для дочернего.

char buf[BUF_SIZE] – массив размером BUF_SIZE для хранения введённой информации для дальнейшей проверки на правильность ответа.

Блок-схема программы

В результате выполнения программы создаётся 2 процесса рис.1 и рис.2. Они вызывают функции print_question и handler_input для печати вопроса и проверки ответа соответственно. Функция print_question представлена на рис.3, а handler_input на рис.4.

Рисунок 1. Блок-схема родительского процесса.

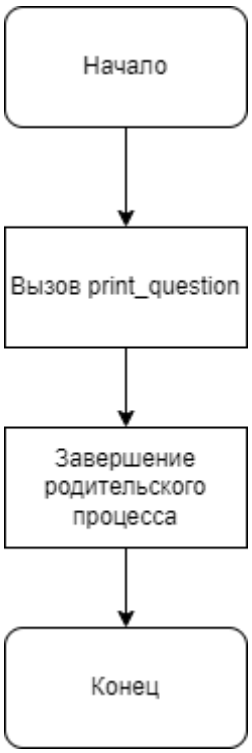

Рисунок 2. Блок-схема дочернего процесса.

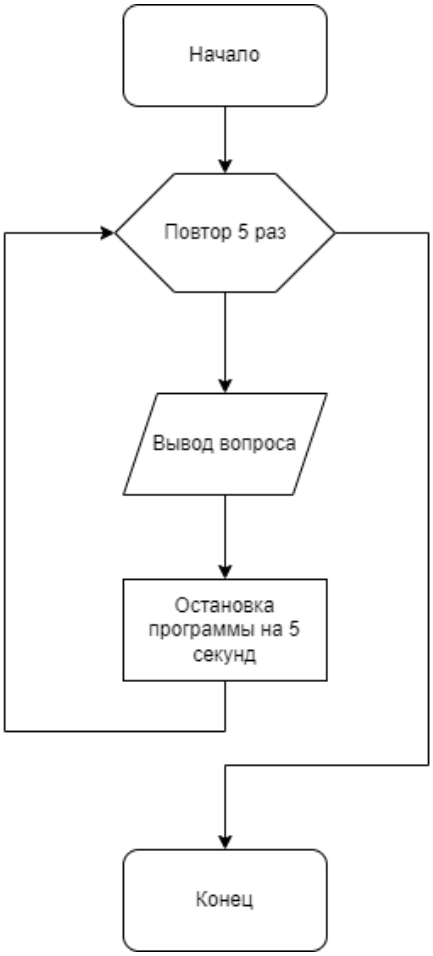

Рисунок 3. Блок-схема print_question.

Рисунок 4. Блок-схема функции handler_input.

Рисунок 5. Блок-схема функции end_program

Примеры результатов работы программы

При первом запуске программа отработала без ввода и завершилась после 5 повторения вопроса. Во втором случае, программа завершилась после ввода правильного ответа и завершилась. В третьем случае при попытке ввести неправильного слова это не повлияло на работу программы и после 5 вопросов она завершилась. Результаты представлены на рис.5

Рисунок 6. Примеры выполнения работы программы.

Текст программы

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <signal.h>
```

```
#define QUESTION "Question\n" // вопрос, который будет выводится
#define PAUSE 5 // длительность задержки между выводом вопроса в секундах
#define COUNT_QUESTION 5 // количество вывода вопроса
#define ANSWER "Answer" // правильный ответ
#define BUF_SIZE 128 // размер буфера для считывания из стандартного потока ввода

void end_program(int _) { // функция для завершения процесса
    exit(0);
}
void print_question() {
    for (int i = 0; i < COUNT_QUESTION; i++) { // Повтор вывода вопроса нужное количество раз
        write(1, QUESTION, strlen(QUESTION));
        sleep(PAUSE    }
}

int handler_input() {
    char buf[BUF_SIZE];
    read(0, buf, BUF_SIZE);
    for (int i = 0; i < strlen(ANSWER); i++) { // проходим по каждой букве в правильном ответе
        if (buf[i] != ANSWER[i]) {
            return 1;
        }
    }
    return 0;
}

int main() {
```

```
    signal(SIGUSR1, end_program);
    int pid = fork(); // создание нового дочернего процесса
    if (pid) {
        print_question(); // функция печати вопроса
        kill(pid, SIGUSR1); // завершение родительского процесса
    } else {
        while (handler_input()); // вызов функции считывания овтета, пока он не будет верный
        kill(getppid(), SIGUSR1); // завершение дочернего процесса
    }
    return 0;
}
```