|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 3](#_Toc100669247)

[Описание структуры программы и реализованных способов взаимодействия процессов 3](#_Toc100669248)

[Описание основных используемых структур данных 3](#_Toc100669249)

[Блок-схема программы 4](#_Toc100669250)

[Примеры результатов работы программы 6](#_Toc100669251)

[Текст программы 7](#_Toc100669252)

# 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.

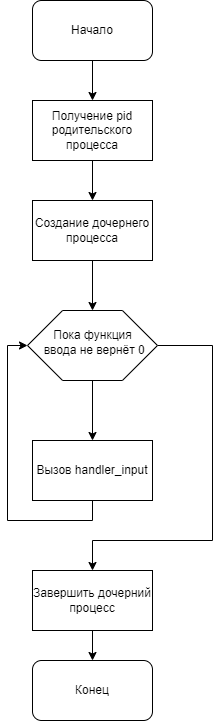


Рисунок 1. Блок-схема родительского процесса.

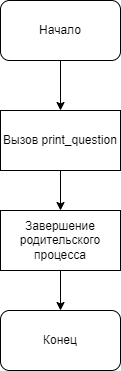


Рисунок 2. Блок-схема дочернего процесса.

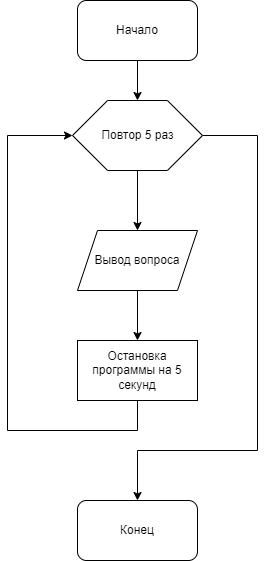


Рисунок 3. Блок-схема print\_question.

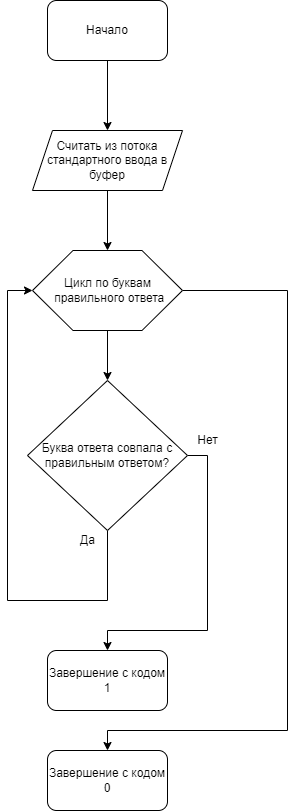


Рисунок 4. Блок-схема функции handler\_input.

# 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

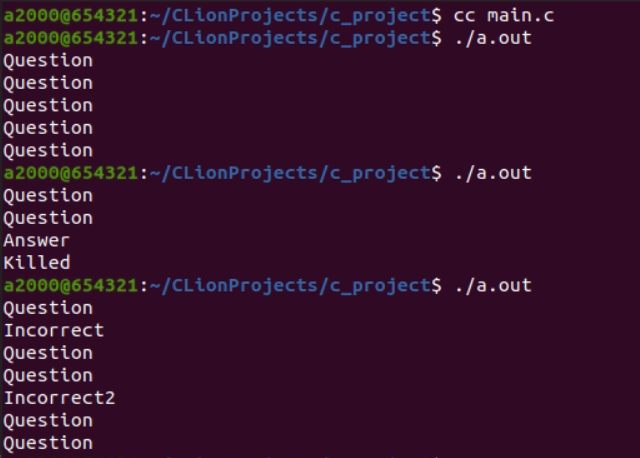


Рисунок . Примеры выполнения работы программы.

# Текст программы

#include <stdio.h>

#include <unistd.h>

#include <string.h>

#include <signal.h>

#define QUESTION "Question\n" // вопрос, котоырй будет выводится

#define PAUSE 10 // длительность задержки между выводом вопроса в секундах

#define COUNT\_QUESTION 5 // количество вывода вопроса

#define ANSWER "Answer" // парвильный ответ

#define BUF\_SIZE 128 // размер буфера для считывания из стандартного потока ввода

void print\_question() { // печать вопроса

for (int i = 0; i < COUNT\_QUESTION; i++) { // Повтор вывода вопроса нужное количество раз

write(1, QUESTION, strlen(QUESTION)); // вывод вопроса

sleep(PAUSE); // останвока программы на PAUSE секунд

}

}

int handler\_input() { // првоерка правильности введённого ответа

char buf[BUF\_SIZE]; // буфер для чтения из потока стандартного ввода

read(0, buf, BUF\_SIZE); // чтение в буфер из потока стандартного ввода

for (int i = 0; i < strlen(ANSWER); i++) { // проходим по каждой букве в правильном ответе

if (buf[i] != ANSWER[i]) { // проверка на совпадения с правильным ответом

return 1;

}

}

return 0;

}

int main() {

int parent\_pid = getpid(); // Получение pid отцовского процесса

int pid = fork(); // создание нового дочернего процесса

if (pid) {

print\_question(); // функция печати вопроса

kill(pid, SIGKILL); // завершение родительского процесса

} else {

while (handler\_input()); // вызов функции считывания овтета, пока он не будет верный

kill(parent\_pid, SIGKILL); // завершение дочернего процесса

}

return 0;

}