Министерство науки и высшего образования 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

по дисциплине: «Разработка программных систем»

Студент	Харитонов Евгений Юрьевич
Группа	РК6-62б
Тип задания	лабораторная работа
Тема лабораторной работы	многопроцессорное программирование

Студент _____ **Харитонов Е.Ю.**
подпись, дата *фамилия, и.о.*

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

Оценка _____

Москва, 2020 г.

Оглавление

Задание на лабораторную работу

Вариант 5 с дополнением:
составить программу, которая заданное число раз (для определенности 7) через переменный временной интервал (начиная с 5 секунд и прибавляем 3 секунды на каждой итерации) повторяет на экране запрос и ожидает стандартный ввод. Программа должна завершаться в случае корректного ответа на запрос или после исчерпывания заданного числа запросов.

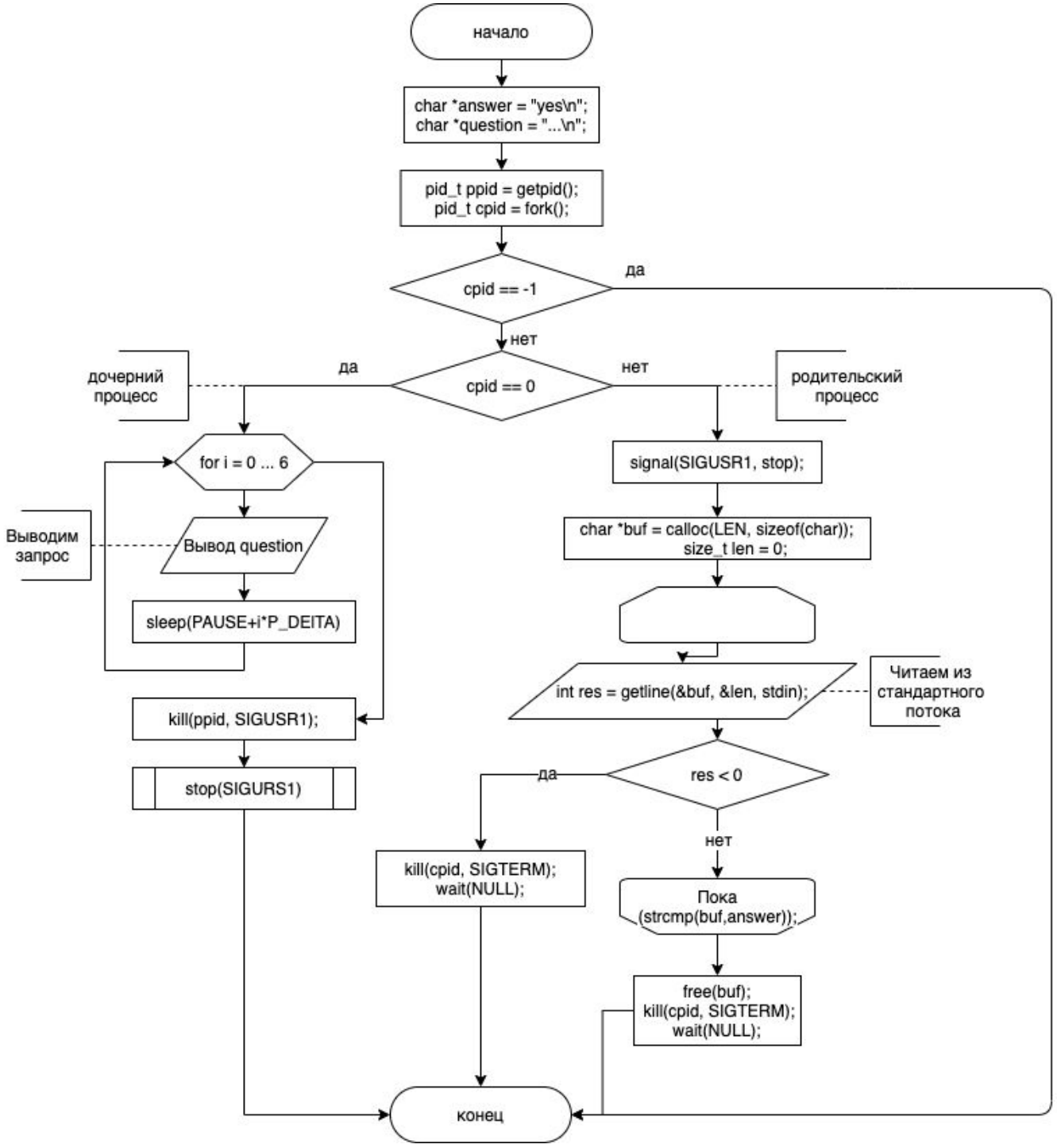
Описание структуры программы и реализованных способов взаимодействия процессов

В начале программы для осуществления коммуникации процессов записываем в переменную ID процесса. С помощью системного вызова fork() создаем копию текущего процесса. В случае успешного выполнения системный вызов возвращает 0 дочернему процессу, родительскому ID дочернего, в иных случаях происходит завершение работы программы. С помощью PID процесса в программе определяется, является текущий процесс родительским или дочерним. Взаимодействие процессов осуществляется отправкой сигналов.

Родительский процесс ожидает ввода от пользователя из стандартного потока. В случае ввода ответа на запрос пользователем, родительский процесс проверяет корректность введенных данных. Если данные неверны, процесс ожидает новый ввод. Если данные верны, процесс отправляет сигнал SIGTERM дочернему процессу, ожидает его завершения и завершается сам.

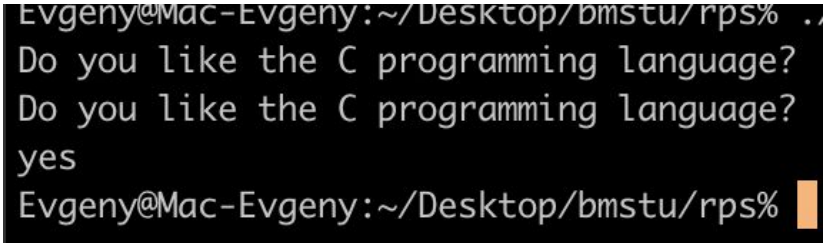
Дочерний процесс через определенные интервалы времени выводит запрос на стандартный поток вывода. Если запрос был выведен заданное число раз, родительскому процессу отправляется сигнал USRSIG1. Его получение приводит к приостановке работы родительского процесса, ожиданию завершения работы дочернего процесса и завершению работы родительского процесса.

Блок схема программы

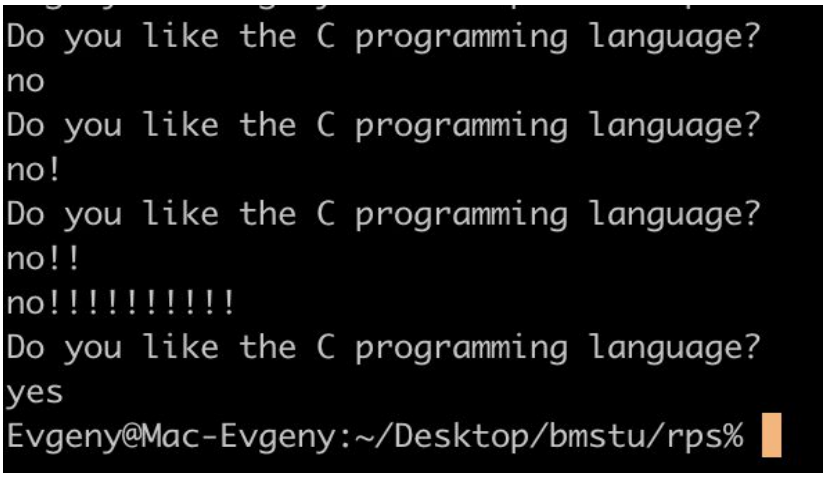


Примеры результатов работы программы

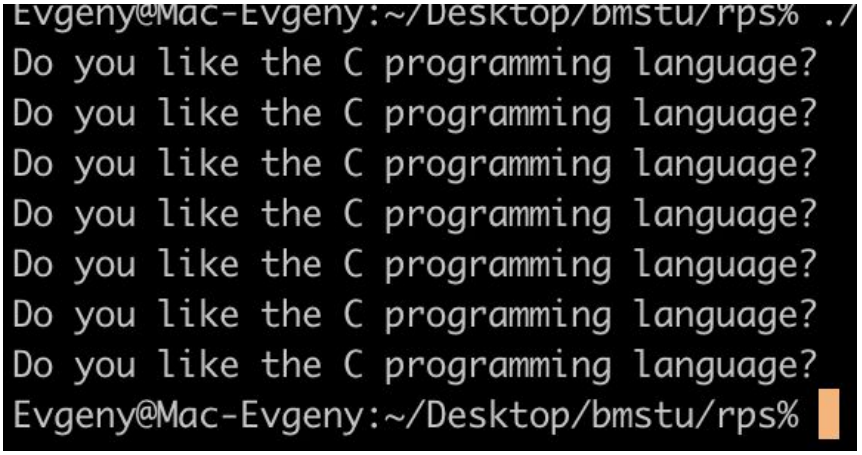
1. Программа через определенное время выводит запрос. При вводе правильного ответа на запрос программа завершает работу.



2. Программа через определенное время выводит запрос. При вводе неправильного ответа на запрос программа ожидает новый ответ. При вводе правильного ответа завершает работу.



3. Программа через определенное время выводит запрос. При отсутствии ответов, по достижении 7 запросов, программа завершает работу.



Текст программы

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

#define N 7
#define PAUSE 5
#define DELTA_PAUSE 3
#define BUFLEN 64


//функция обработчик сигнала
void stop(int sig) {
   wait(NULL);
   exit(0);
}

int main() {

   char *answer = "yes\n";
   char *question = "Do you like the C programming language?\n";

   // получаем pid процесса (в будущем это pid родителя)
   pid_t ppid = getpid();

   //создаем новый процесс
   pid_t cpid = fork();

   //проверка создания нового процесса
   if (cpid == -1) {
      perror("fork");
      exit(EXIT_FAILURE);
   }

   if (cpid == 0) { //процесс ребенок
      for (size_t i = 0; i < N; i++) { //пишет N раз в stdout
         write(STDOUT_FILENO, question, strlen(question));
         sleep(PAUSE + i * DELTA_PAUSE);
```

```
        }

        //если написали N раз, отправляем сигнал SIGUSR1 родителю
        kill(ppid, SIGUSR1);

    } else {
        //устанавливаем обработчик сигнала
        signal(SIGUSR1, stop);

        //создаем буфер для ввода
        char *buf = calloc(BUFLEN, sizeof(char));
        size_t len = 0;

        //цикл считываения
        do {
            int res = getline(&buf, &len, stdin);

            //при неудачном считывании завершаем работу
            if (res < 0) {
                kill(cpid, SIGTERM);
                wait(NULL);
                exit(EXIT_FAILURE);
            }
        }
        while (strcmp(buf, answer)); //цикл выполняется, пока считанная строка отличается от ожидаемой

        free(buf);

        //убиваем сына
        kill(cpid, SIGTERM);
        wait(NULL);
    }

    return 0;
}
```