

Московский Авиационный Институт  
(Национальный Исследовательский Университет)  
Факультет информационных технологий и прикладной математики  
Кафедра вычислительной математики и программирования

**Лабораторная работа №2 по курсу**  
**«Операционные системы»**

**Тема работы**  
**“Изучение взаимодействий между процессами”**

Студент: Прохоров Данила  
Михайлович                      Группа: М8О-208Б-20  
Вариант: 16  
Преподаватель: Миронов Евгений Сергеевич  
Оценка: \_\_\_\_\_  
Дата: \_\_\_\_\_  
Подпись: \_\_\_\_\_

Москва, 2021  
**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

## Репозиторий

<https://github.com/dmprokhorov/OS>

## Постановка задачи

Задача: реализовать программу, в которой родительский процесс создает один дочерний процесс. Родительский процесс принимает путь к файлу и строки, которые отправляются в тот дочерний процесс, там те из них, которые оканчиваются на символы ';' или '.', записываются в файл, если же строки не удовлетворяют этому правилу, то они возвращаются в родительский процесс. Далее в родительском процессе сначала выводятся строки из файла (если его удалось открыть и там есть хотя бы одна строка), а потом строки, вернувшиеся из родительского процесса в дочерний.

## Общие сведения о программе

В программе используются следующие библиотеки:

- `<iostream>` - для вывода информации на консоль
- `<fstream>` - для записи текста в файл.
- `<unistd.h>` - для системных вызовов `read` и `write` в Ubuntu.
- `<sys/wait.h>` - для функции `waitpid`, когда родительский процесс ждёт дочерний.

В задании используются такие команды и строки, как:

- `int fd[2]` - создание массива из 2 дескрипторов, 0 - чтение (`read`), 1 - передача (`write`). И
- `pipe(fd)` - конвейер, с помощью которого выход одной команды подается на вход другой (также “труба”). Это неименованный канал передачи данных между родственными процессами.

- **pid\_t child = fork ()** - создание дочернего процесса, в переменной child будет храниться “специальный код” процесса (-1 - ошибка fork, 0 - дочерний процесс, >0 - родительский)
- **read(int fd, void\* buf, size\_t count)** (здесь дан общий пример) - команда, предназначенная для чтения данных, посланных из другого процесса, принимающая на вход три параметра: элемент массива дескрипторов с индексом 0 (в моей программе fd1[0], fd2[0]), указатель на память получаемого объекта (переменной, массива и т.д.), размер получаемого объекта (в байтах).
- **write(int fd, void\* buf, size\_t count)** (здесь дан общий пример) - команда, предназначенная для записи данных в другой процесс, принимающая на вход три параметра: элемент массива дескрипторов с индексом 1 (в моей программе fd1[1], fd2[1]), указатель на память посылаемого объекта (переменной, массива и т.д.), размер посылаемого объекта (в байтах).
- **close(int fd)** - команда, закрывающая файловый дескриптор.
- **int wstatus;**

**waitpid(child, &wstatus, 0)** – команда ожидания завершения процесса с id, равным child, если она завершилась без ошибок, то в переменной wstatus будет лежать значение 0, и родительский процесс продолжит своё выполнение, в противном случае там будет лежать значение ненулевое значение, и родительский процесс аварийно завершится.

## Общий метод и алгоритм решения

В начале программа получает на вход путь к файлу, где будут лежать нужные строки, затем пользователю предлагается ввести строки, конец ввода должен сигнализироваться символом Ctrl+D (это некоторое количество строк

считывается как одна (с символами переноса строки и символом конца строки – ‘\0’). Далее программа создаёт дочерний процесс, если этого сделать не удалось, то она аварийно завершается.

В дочернем процессе получается имя файла, а также большая исходная строка. В результате прохода по строке и некоторых операций получают 2 строки `file_string` и `out_string`. Если удалось открыть файл, то `file_string` записывается в файл, `out_string` по неименованному каналу передаётся обратно в родительский процесс, вся память чистится, закрываются файловые дескрипторы и возвращается значение 0. Если же файл открыть не удалось, то выводится информация об ошибке открытия файла, и возвращается значение 1.

Родительский процесс сразу, как только запускается, ждёт дочерний. Если получаемое значение 1, то программа завершается, предварительно очистив всю память и закрыв файловые дескрипторы. Если же возвращаемое значение 0, то тогда программа пытается открыть файл. Если это не получилось сделать, то она аварийно завершается. В противном случае она читает из файла строки и выводит их, предварительно написав, что это строки, удовлетворяющие правилу. Дальше выводятся строки, не удовлетворяющие правилу. Далее чистится вся память и удаляются файловые дескрипторы.

Собирается программа при помощи команды `g++ task.cpp -o task`, запускается при помощи команды `./task`.

## Исходный код

```
#include <iostream>
#include <fstream>
#include <unistd.h>
#include <sys/wait.h>
int main()
{
    int fd1[2], fd2[2];
    if ((pipe(fd1) == -1) || (pipe(fd2) == -1))
    {
        std::cout << "Failed to open pipes between parent and child processes" << std::endl;
        exit(1);
    }
    pid_t child;
    if ((child = fork()) == -1)
    {
        std::cout << "Failed to create child process" << std::endl;
```

```

        exit(1);
    }
    pid_t child;
    if ((child = fork()) == -1)
    {
        std::cout << "Failed to create child process" << std::endl;
        exit(1);
    }
    else if (child > 0)
    {
        std::cout << "You're in the parent process with id [" << getpid() << "]" << std::endl;
        std::string file_path, string;
        std::cout << "Input path to the file" << std::endl;
        getline(std::cin, file_path);
        int length = file_path.length() + 1;
        write(fd1[1], &length, sizeof(int));
        write(fd1[1], file_path.c_str(), length * sizeof(char));
        char symbol, *in = (char*) malloc (2 * sizeof(char));
        int counter = 0, size_of_in = 2;
        std::cout << "Now input some strings. If you want to end input, press Ctrl+D" << std::endl;
        while ((symbol = getchar()) != EOF)
        {
            in[counter++] = symbol;
            if (counter == size_of_in)
            {
                size_of_in *= 2;
                in = (char*) realloc (in, size_of_in * sizeof(char));
            }
        }
        in = (char*) realloc (in, (counter + 1) * sizeof(char));
        in[counter] = '\0';
        write(fd1[1], &counter, sizeof(int));
        write(fd1[1], in, counter * sizeof(char));
        close(fd1[0]);
        close(fd1[1]);
        int out_length, wstatus;
        waitpid(child, &wstatus, 0);
        if (wstatus)
        {
            close(fd2[0]);
            close(fd2[1]);
            free(in);
            exit(1);
        }
        std::cout << "You're back in parent process with id [" << getpid() << "]" << std::endl;
        read(fd2[0], &out_length, sizeof(int));
        char* out = (char*) malloc (out_length);
        read(fd2[0], out, out_length * sizeof(char));
        std::ifstream fin(file_path.c_str());
        if (!fin.is_open())
        {
            close(fd2[0]);
            close(fd2[1]);
            free(in);
            std::cout << "Failed to open file to read strings" << std::endl;
            exit(1);
        }
        std::cout << "-----" << std::endl << "These strings end in character '.' or ';' : " << std::endl;
        if (fin.peek() != EOF)
        {
            while (!fin.eof())
            {
                getline(fin, string);
                std::cout << string << std::endl;
            }
        }
        fin.close();
        std::cout << "-----" << std::endl << "These strings don't end in character '.' or ';' : " << std::endl;
        for (int i = 0; i < out_length - 1; i++)
        {
            std::cout << out[i];
        }
        close(fd2[0]);
        close(fd2[1]);
        free(in);
        free(out);
    }
    else
    {
        int length;
        read(fd1[0], &length, sizeof(int));
        char* c_file_path = (char*) malloc (length * sizeof(char));
        read(fd1[0], c_file_path, length * sizeof(char));
        int counter;
        read(fd1[0], &counter, sizeof(int));
        char* inc = (char*) malloc (counter * sizeof(char));
        read(fd1[0], inc, counter * sizeof(char));
        close(fd1[0]);
        close(fd1[1]);
        std::cout << "Now you are in child process with id [" << getpid() << "]" << std::endl;
        std::string string = std::string(), out_string = std::string(), file_string = std::string();
        for (int i = 0; i < counter; i++)
        {
            if (inc[i] != '\0')

```

```

{
    if (inc[i] != '\0')
    {
        string += inc[i];
    }
    if ((inc[i] == '\n') || (inc[i] == '\0'))
    {
        if ((i > 0) && ((inc[i - 1] == '.') || (inc[i - 1] == ',')))
        {
            file_string += string;
        }
        else
        {
            out_string += string;
        }
        string = std::string();
    }
}
if ((file_string.length() && (file_string[file_string.length() - 1] == '\n'))
{
    file_string.pop_back();
}
std::ofstream fout(c_file_path);
if (!fout.is_open())
{
    std::cout << "Failed to create or open file to write strings" << std::endl;
    free(inc);
    free(c_file_path);
    close(fd2[0]);
    close(fd2[1]);
    return 1;
}
int out_length = out_string.length() + 1;
write(fd2[1], &out_length, sizeof(int));
write(fd2[1], out_string.c_str(), out_length * sizeof(char));
close(fd2[0]);
close(fd2[1]);
if (!file_string.empty())
{
    fout << file_string;
}
fout.close();
free(inc);
free(c_file_path);
}
return 0;

```

## Демонстрация работы программы

Тест 1.

Вводится корректное имя файла, 2 строки оканчиваются на точку или точку с запятой, 2 нет.

```

danila@danila-VirtualBox:~/operation_systems/OS2$ ./task
You're in the parent process with id [4351]
Input path to the file
input.txt
Now input some strings. If you want to end input, press Ctrl+D
dlkvneow;
ewewgview
ewvwewv.
wegwegwew
Now you are in child process with id [4352]
You're back in parent process with id [4351]

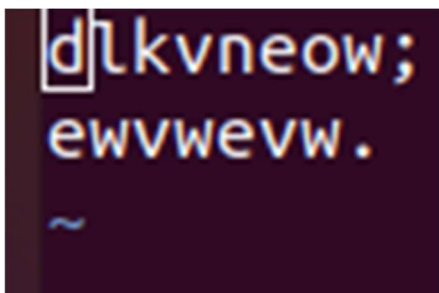
```

```

-----
These strings end in character '.' or ';' :
dlkvneow;
ewvwevw.
-----
These strings don't end in character '.' or ';' :
ewewgview
wegwegwew
danila@danila-VirtualBox:~/operation_systems/OS2$

```

Содержание файла input.txt



```

dlkvneow;
ewvwevw.
~

```

Тест 2.

Вводится корректное имя файла, но все строки удовлетворяют правилу.

```

danila@danila-VirtualBox:~/operation_systems/OS2$ ./task
You're in the parent process with id [4499]
Input path to the file
input.txt
Now input some strings. If you want to end input, press Ctrl+D
d;svmsd;
dsvsdvds.
.
ddevewfefew;
.
;;;;
Now you are in child process with id [4500]
You're back in parent process with id [4499]
-----
These strings end in character '.' or ';' :
d;svmsd;
dsvsdvds.
.
ddevewfefew;
.
;;;;
-----
These strings don't end in character '.' or ';' :
danila@danila-VirtualBox:~/operation_systems/OS2$ 

```



Содержание файла input.txt

```
d;svmsd;  
dsvsdvds.  
  
.  
ddevwfew;  
  
.  
;;;;
```

### Тест 3

Вводится корректное имя файла, но все строки не удовлетворяют правилу (в строке “regre.” После точки следуют ещё 3 пробела, так что формально она имеет вид “regre. “. Пустые строки также формально не оканчиваются на точку или точку с запятой.

```
danila@danila-VirtualBox:~/operation_systems/OS2$ ./task  
You're in the parent process with id [4626]  
Input path to the file  
input.txt  
Now input some strings. If you want to end input, press Ctrl+D  
ljvnd  
vjkrbjkbre  
revrebgre.t  
rgjrlkrewew;  
rgre.  
  
etetette  
Now you are in child process with id [4627]  
You're back in parent process with id [4626]  
-----  
These strings end in character '.' or ';' :  
-----  
These strings don't end in character '.' or ';' :  
ljvnd  
vjkrbjkbre  
revrebgre.t  
rgjrlkrewew;  
rgre.  
  
etetette  
danila@danila-VirtualBox:~/operation_systems/OS2$
```

Содержание файла input.txt



#### Тест 4

Вводится некорректное имя файла (просто символ переноса строки)

```
danila@danila-VirtualBox:~/operation_systems/OS2$ ./task
You're in the parent process with id [4766]
Input path to the file

Now input some strings. If you want to end input, press Ctrl+D
dsvdsv
dsvsdvkm sdvse;
egewgew;
ef.
Now you are in child process with id [4767]
Failed to create or open file to write strings
danila@danila-VirtualBox:~/operation_systems/OS2$
```

#### Выводы

Это была моя первая лабораторная работа по курсу “Операционные системы”. Было интересно узнать много нового и про системные вызовы, и про межпроцессное взаимодействие.