Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу
«Операционные системы»

Взаимодействие между процессами

Студент: Калиниченко Артём Андреевич
Группа: М8О–210Б–22
Вариант: 9
Преподаватель: Соколов Андрей Алексеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2023.

Постановка задачи

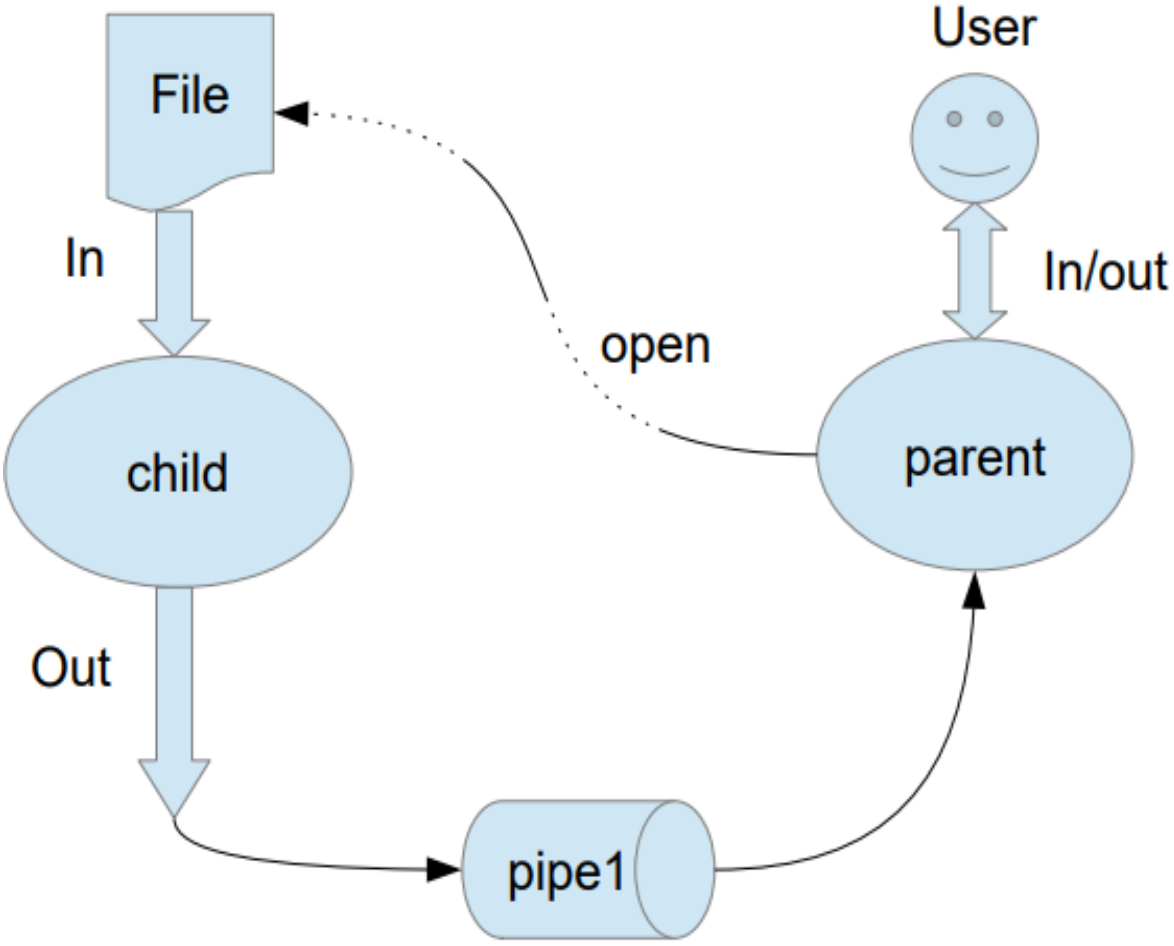
Цель работы

Целью является приобретение практических навыков в:

- Работе с несколькими процессами
- Создание программ, которые используют межпроцессное взаимодействие

Задание

Требуется написать программу, которая создает дочерний процесс взаимодействует с ним и проверяет, начинается ли входящая строка с большой буквы. Межпроцессное взаимодействие согласно схеме:



В конечном итоге, программа должна состоять из следующих частей:

- Программа родительского процесса
- Программа дочернего процесса
- Библиотечный файл с функциями записи и чтения в/из потока

Провести анализ работы программы

Общие сведения о программе

В программе используются следующие системные вызовы:

1. **read** – читает из потока f в переменную s n байт.
2. **write** – записывает в поток f значение переменной s размером в n байт.
3. **execl** – подменяет образ текущего процесса процессом proc, и отправляет на вход proc входные данные arg1, agr2,…
4. **fork –** создает новый дочерний процесс. Возвращает pid.
5. **pipe –** создает новый pipe. Возвращает файловые дескрипторы fd1 и fd2 на чтение и запись соответственно.

Общий метод и алгоритм решения.

Для реализации поставленной задачи необходимо:

1. Изучить принципы работы write, read, exec*.
2. Написать программу дочернего и родительского процессов.
3. Организовать простейший командный интерфейс в файлах parent.cpp и son.cpp

Основные файлы программы

main.cpp

```
#include <unistd.h>
#include <iostream>
#include <string>
#include <fcntl.h>

using namespace std;

int main() {
    string file_name;
    char el;
    while (read(STDIN_FILENO, &el, sizeof(char))!=0) {
```

```
    if (el != '\n') file_name.push_back(el);
    else break;
}

int file = open(file_name.c_str(), O_RDONLY);
if (file == -1) {
    for (char el : "error pipe\n") {
        write(STDERR_FILENO, &el, sizeof(char));
    }
    return 1;
}

int pipefd[2];
if (pipe(pipefd) == -1){
    for (char el : "error pipe\n") {
        write(STDERR_FILENO, &el, sizeof(char));
    }
    return 1;
}

pid_t pid = fork();
if (pid == -1) {

    for (char el :"error fork\n") {
        write(STDERR_FILENO, &el, sizeof(char));
    }
    return 1;

} else if (pid == 0) {

    close(pipefd[0]);
```

```
        dup2(file, STDIN_FILENO);
        dup2(pipefd[1], STDOUT_FILENO);

        execl("./child", "child", NULL);

        for (char el :"Ошибка запуска дочернего процесса!") {
            write(STDERR_FILENO, &el, sizeof(char));
        }

    } else {

        close(pipefd[1]);

        char p;
        while ((read(pipefd[0], &p, sizeof(p))) != 0) {
            write(STDOUT_FILENO, &p, sizeof(p));
        }

        for (char el : "You want to '/' with zero.\n") {
            write(STDERR_FILENO, &el, sizeof(el));
        }

    close(pipefd[0]);
    close(file);
    }
return 0;
}
```

child.cpp

```
#include <iostream>
#include <unistd.h>

using namespace std;

int main(int argc, char *argv[]) {

    char c, enter = '\n';
    float num = 0, res;
    int counter = 0, isfirstnum = 1;
    while (read(STDIN_FILENO, &c, sizeof(c)) != 0) {

        if (c == ' ') {

            if (isfirstnum == 1) {
                res = num;
                isfirstnum = 0;

            } else {
                if (num != 0) res /= num;
                else return 1;
            }
            num = 0;
            counter = 0;

        } else if (c == '.') counter = 10;

        else if (c == '\n') {
            if (num != 0) {
                res /= num;
                string buf = to_string(res);

                for (char el : buf) {
                    write(STDOUT_FILENO, &el, sizeof(el));
                }
                write(STDOUT_FILENO, &enter, sizeof(char));

                counter = 0;
                isfirstnum = 1;
                num = 0;

            } else return 1;
```

```
        } else {
            if (counter == 0) {
                num *= 10;
                num += c - '0';
            } else {
                num += (float)(c - '0') / counter;
                counter *= 10;
            }
        }
    }
return 0;
}
```

Пример работы

1. test1:

Input: res1.txt

```
1 2
2 3 1
1 0
```

Output:

```
0.5
0.6666667
You’re gonna to “/” with zero
```

Вывод

При выполнении данной лабораторной работы я научился работать с несколькими процессами. Обучился межпроцессному взаимодействию при помощи именнованных каналов pipe. Понял, что под scanf, printf, cin, cout лежат системные вызовы write и read. Научился отлаживать программы при помощи strace. Написал собственную программу использующую несколько процессов и каналы pipe.