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

Лабораторная работа № 2 по курсу

«Операционные системы»

Студент: Пирогов М.Д.
Группа: М8О-207Б-21
Вариант: 12
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/pirogovmark/OS-Labs/

Постановка задачи

Цель работы

Приобретение практических навыков в управлении процессов в C++ и обеспечении обмена данных между процессами посредством каналов.

Задание

Составить и отладить программу на языке Си, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программа (основной процесс) должен создать для решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или каналы (pipe). Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

12 вариант) Child1 переводит строки в верхний регистр. Child2 убирает все задвоенные пробелы.

Общие сведения о программе

Программа компилируется из файла main.c. Также используется заголовочные файлы: unistd.h, ... В программе используются следующие системные вызовы:

1. pipe() - создает канал для чтения и записи
2. fork() - создает новый процесс, копируя вызывающий процесс
3. read() & write() - соответственно чтение и запись из файлового дескриптора в буфер
4. close() - закрывает файловый дескриптор

Общий метод и алгоритм решения

С помощью системного вызова fork создается Child 1. В Parent снова используем вызов fork(), создавая тем самым Child 2. Pipe вызывается 3 раза для связи Parent - Child 1, Child 1 - Child 2 и Child 2 - Parent. Через каждый вызов pipe будем передавать данные в соответствии со схемой задания варианта.

Исходный код

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

```
#include <string>


std::string spaces(std::string s) {
    std::string out = "";
    char prev = 'a';
    for (int i = 0; i < s.size(); ++i) {
        if (!(prev == ' ' && s[i] == ' ')) {
            out += s[i];
            prev = s[i];
        }
    }

    return out;
}


std::string to_upper(std::string s) {
    for (int i = 0; i < s.size(); ++i) {
        if (s[i] >= 'a' && s[i] <= 'z') {
            s[i] = toupper(s[i]);
        }
    }

    return s;
}


int main() {
    int pipefd_p1[2]; // Pipe for communicating Parent and Child1
    pipe(pipefd_p1);

    int pipefd_12[2]; // Pipe for communicating Child1 and Child2
    pipe(pipefd_12);

    int pipefd_2p[2]; // Pipe for communicating Child2 and Parent
```

```
    pipe(pipefd_2p);

    int id_1 = fork();

    if (id_1 == -1) {
        return -1;
    } else if (id_1 == 0) {        // Child 1
        std::string s_1;
        read(pipefd_p1[0], &s_1, sizeof(std::string));
        std::cout << "Child 1 in: " << s_1 << "\n";
        close(pipefd_p1[0]);
        close(pipefd_p1[1]);

        s_1 = to_upper(s_1);
        std::cout << "Child 1 out: " << s_1 << "\n";
        write(pipefd_12[1], &s_1, sizeof(std::string));
        close(pipefd_12[0]);
        close(pipefd_12[1]);

        std::cout << '\n';
    } else {
        int id_2 = fork();

        if (id_2 == -1) {
            return -1;
        } else if (id_2 == 0) {    // Child 2
            std::string s_2;
            read(pipefd_12[0], &s_2, sizeof(std::string));
            std::cout << "Child 2 in: " << s_2 << "\n";
            close(pipefd_12[0]);
            close(pipefd_12[1]);

            s_2 = spaces(s_2);
            std::cout << "Child 2 out: " << s_2 << "\n";
            write(pipefd_2p[1], &s_2, sizeof(std::string));
            close(pipefd_2p[0]);
```

```
            close(pipefd_2p[1]);

            std::cout << '\n';
        } else {                // Parent
            std::cin >> s;
            std::cout << "Parent in: " << s << "\n\n";
            write(pipefd_p1[1], &s, sizeof(std::string));
            close(pipefd_p1[0]);
            close(pipefd_p1[1]);

            read(pipefd_2p[0], &s, sizeof(std::string));
            std::cout << "Parent out: " << s << "\n";

            close(pipefd_2p[0]);
            close(pipefd_2p[1]);
        }

    }

    return 0;
}
```

Демонстрация работы программы

```
markp@MacBook-Air-Mark Lab_2 % g++ OSLab_2.cpp
markp@MacBook-Air-Mark Lab_2 % ./a.out
Parent in: Too    many   spaces?

Child 1 in: Too    many   spaces?
Child 1 out: TOO    MANY   SPACES?

Child 2 in: TOO    MANY   SPACES?
Child 2 out: TOO MANY SPACES?

Parent out: TOO MANY SPACES?
```

Выводы

Я приобрел навыки управления процессами в C++ и обеспечении обмена данных между процессами через каналы.