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: Слесарчук Василий Анатольевич
Группа: М8О–210Б–22
Вариант: 15
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. Если да, результат записывается в файл.

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

Программа компилируется при помощи Makefile. Также используется заголовочные файлы: iostream, string, stdio.h, unistd.h, cstdlib, sys/wait.h, stream, fcntl.h, sys/stat.h. 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

4.

parent.cpp:

```
#include "WriteRead.h"

#define READ 0
#define WRITE 1

int main() {
    int pipe_fd1[2];
    int pipe_fd2[2];
    WriteString(STDOUT_FILENO, "Type the name of file: \n");
    int fd_file;
    mode_t mode = S_IRWXU;
        int flags = O_WRONLY | O_CREAT | O_APPEND;
    if ((fd_file = open(ScanString(STDIN_FILENO).c_str(), flags, mode)) < 0) {
        WriteString(STDOUT_FILENO, "File isn't opened\n");
        perror("file");
    }
    if (pipe(pipe_fd1) == -1) {
        perror("pipe");
        return -1;
    }
    if (pipe(pipe_fd2) == -1) {
        perror("pipe");
        return -1;
    }
    pid_t pid = fork();
```

```
    if (pid == 0) {
        close(pipe_fd2[READ]);
        close(pipe_fd1[WRITE]);
        dup2(pipe_fd2[WRITE], STDERR_FILENO);
        dup2(pipe_fd1[READ], STDIN_FILENO);
        dup2(fd_file, STDOUT_FILENO);
        execl("son.out", "son.out", NULL);
    } else if (pid > 0) {
        std::string input;
        close(pipe_fd1[READ]);
        close(pipe_fd2[WRITE]);
        WriteString(STDOUT_FILENO, "Type string to check: \n");
        WriteString(pipe_fd1[WRITE], ScanString(STDIN_FILENO));
        WriteString(STDOUT_FILENO, ScanString(pipe_fd2[READ]));
        wait(NULL);
        close(pipe_fd1[WRITE]);
        close(pipe_fd2[READ]);
    } else {
        perror("fork");
        return -1;
    }
}

#include "WriteRead.h"

int main() {
    std::string line;
    line = ScanString(STDIN_FILENO);
    if (isupper(line[0])) {
        WriteString(STDOUT_FILENO, line);
```

```
        WriteString(STDERR_FILENO, "String is valid, check the file\n");
    } else {
        WriteString(STDERR_FILENO, "String is not valid\n");
    }
}
```

WriteRead.cpp

```
#include "WriteRead.h"

std::string ScanString(int stream) {
    char c;
    std::string line;
    while (1) {
        if (c == '\n') break;
        read(stream, &c, sizeof(char));
        line.push_back(c);
    }
    return line;
}

void WriteString(int stream, std::string line) {
    write(stream, line.c_str(), line.size());
}
```

WriteRead.h

```
#pragma once
#include <iostream>
#include <string>
#include "stdio.h"
#include "unistd.h"
#include <cstdlib>
#include "sys/wait.h"
#include <fstream>
#include <fcntl.h>
#include <sys/stat.h>
```

```
void WriteString(int stream, std::string line);
std::string ScanString(int stream);
```

Пример работы

1. test1:

Input: res1.txt

Abcd ABcd asdmfl

Output: String is valid, check the file

Abcd ABcd asdmfl

2. test2:

Input: res2.txt

gogogogogo gogogogo everton

Output: String is not valid

3. test3:

Input: res3.txt

Output: String is not valid

4. test3:

Input: res4.txt

123 Abcd Abcd djdjdjjd

Output: String is not valid

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.