****Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работ №2 по курсу**

**«Операционные системы»**

Группа: М80 – 207Б-18

Студент: Син Денис Дмитриевич

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2019.

**Содержание**

1. Постановка задачи
2. Общие сведения о программе
3. Общий метод и алгоритм решения
4. Основные файлы программы
5. Демонстрация работы программы
6. Вывод

**Постановка задачи**.

Написать целочисленный калькулятор с операциями +, -. В дочернем процессе должны происходить вычисления выражений. В родительском процессе должны происходить ввод и вывод.

**Общие сведения о программе**

Программа компилируется из одного файла main.c. В данном файле используются заголовочные файлы stdio.h, unistd.h, string.h, stdlib.h. В программе используются следующие системные вызовы:

1. **read** – для чтения данных из файла
2. **write** – для записи данных в файл
3. **pipe** – для создания однонаправленного канала, через который могут общаться два процесса. Системный вызов возвращает два дескриптора файлов. Один для чтения из канала, другой для записи в канал.
4. **fork** – для создания дочернего процесса.
5. **close** – для закрытия файла.

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Используя системный вызов pipe создать 2 канала, по которым будут обмениваться данными два процесса.
2. Используя системный вызов fork создать дочерний процесс.
3. В родительском процессе считывать данные cо стандартного потока в цикле в формате: **<left arg> <operation> <right arg>** пока данные вводятся.
4. Как в родительском процессе данные считались, необходимо записать их в канал с помощью системного вызова write. Затем родительский процесс считывает результат из второго канала. Но пока дочерний процесс не запишет данные во второй канал, родительский процесс будет заблокирован.
5. Пока родительский процесс на записал данные в канал. Дочерний процесс блокируется. И как только родительский процесс записал данные в первый канал дочерний процесс считывает их, производит вычисления и записывает результат во второй канал.
6. Как только дочерний процесс запишет результат вычислений во второй канал родительский процесс получит результат из второго канала и выведет их в стандартный поток. И затем снова будет ждать ввода со стандартного потока.

**Основные файлы программы**.

**Файл main.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#define PIPE\_FD\_SIZE 2

void DataWritePipe(int \**lhs*, int \**rhs*, char \**op*, int *fd*) {

    write(fd, lhs, sizeof(int));

    write(fd, op, sizeof(char));

    write(fd, rhs, sizeof(int));

}

void DataReadPipe(int \**res*, int *fd*) {

    read(fd, res, sizeof(int));

}

int Sum(int *lhs*, int *rhs*) {

    return lhs + rhs;

}

int Sub(int *lhs*, int *rhs*) {

    return lhs - rhs;

}

int main(void) {

    int fd1[PIPE\_FD\_SIZE];

    int fd2[PIPE\_FD\_SIZE];

    int pr;

    if (pipe(fd1) == -1 || pipe(fd2) == -1) {

        printf("Can\'t create pipe\n");

        exit(0);

    }

    pr = fork();

    if (pr == -1) {

        printf("Can\'t fork child\n");

        exit(0);

    } else if (pr > 0) {

        close(fd1[0]);

        close(fd2[1]);

        int res,

            lhs,

            rhs;

        char op;

        while (scanf("%d %c %d", &lhs, &op, &rhs) == 3) {

            DataWritePipe(&lhs, &rhs, &op, fd1[1]);

            DataReadPipe(&res, fd2[0]);

            printf("%d\n", res);

        }

        close(fd1[1]);

        close(fd2[0]);

    } else {

        close(fd1[1]);

        close(fd2[0]);

        int res,

            lhs,

            rhs;

        char op;

        while (read(fd1[0], &lhs, sizeof(int))) {

            read(fd1[0], &op, sizeof(char));

            read(fd1[0], &rhs, sizeof(int));

            res = (op == '+') ? Sum(lhs, rhs) : Sub(lhs, rhs);

            write(fd2[1], &res, sizeof(int));

        }

        close(fd1[0]);

        close(fd2[1]);

    }

    return 0;

}

**Демонстрация работы программы.**

[20:21:38] sindenis:lab02 $ gcc main.c -o calculator

[20:21:46] sindenis:lab02 $ ./calculator

1 + 0

1

1 - 0

1

0 - 0

0

45 - 5

40

1000 + 2000

3000

65 - 90

-25

-89832 + 8000

-81832

-89832 + 90000

168

123456789 - 987654321

-864197532

76 + 76

152

[20:23:00] sindenis:lab02 $

**Вывод**

Межпроцессорное взаимодействие можно осуществлять с помощью канала. В СИ канал создается с помощью системного вызова pipe. На мой взгляд, такой способ общения процессов очень удобен, так как при данном подходе не приходится сталкиваться с гонками, так как при использовании блокирующих системных вызовов read и write процессы блокируются, если им нечего считывать или буфер для записи полный. Так же одним из плюсов такого подхода к межпроцессорному взаимодействию является то, что каналом могут пользоваться только родственные процессы, так как канал находится в пределах ядра.