

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

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

Процессы операционных систем

Студент: Никулин Кристиан Ильич

Группа: М8О–208Б–21

Вариант: 3

Преподаватель: Соколов Андрей Алексеевич

Оценка: _____

Дата: _____

Подпись: _____

Москва, 2022.

Постановка задачи

Цель работы

Приобретение практических навыков в:

- Управление процессами в ОС
- Обеспечение обмена данными 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.

Вариант 3

Пользователь вводит команды вида: «число число число». Далее эти числа передаются от родительского процесса в дочерний. Дочерний процесс производит деление первого числа, на последующие, а результат выводит в файл. Если происходит деление на 0, то тогда дочерний и родительский процесс завершают свою работу. Проверка деления на 0 должна осуществляться на стороне дочернего процесса. Числа имеют тип int. Количество чисел может быть произвольным.

Общие сведения о программе

Отдельно компилируются две программы из файлов parent.c и child.c

Общий метод и алгоритм решения.

С помощью вызова fork создаются родительский и дочерний процессы, родительский процесс считывает название будущего файла и строку целых чисел, которые передаются в дочерний процесс. Дочерний процесс создаёт файл и записывает в него деление первого числа из этой строки на все последующие. Данные передаются между процессами с помощью pipe и потоков ввода-вывода. Ключей для запуска программа не имеет.

Основные файлы программы

parent.c:

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

```
#include <stdio.h>
```

```

#include <string.h>
#include <stdlib.h>
#include <fcntl.h>
#include <wait.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/uio.h>
#define BUF_SIZE 128
char * read_string() {
    char *x = calloc(sizeof(char), BUF_SIZE);
    if (read(0, x, sizeof(char) * BUF_SIZE) == -1)
    {
        perror("Error during reading string\n");
        exit(1);
    }
    char *y = malloc(sizeof(char) * strlen(x));
    strncpy(y, x, strlen(x));
    free(x);
    return y;
}
int main()
{
    int fd[2];
    if (pipe(fd) == -1)
    {
        perror("Error during creating pipe\n");
        exit(3);
    }
    // FILE NAME INPUT
    char *print_file = "Print output file name:\n";
    write(1, print_file, sizeof(char) * strlen(print_file));
    char * file_name = read_string();
    // INT NUMBERS INPUT
    char *print_numbers = "Enter the numbers:\n";

```

```

write(1, print_numbers, sizeof(char) * strlen(print_numbers));
char * a = read_string();
int a_len = strlen(a), a_len1;
// WRITE ARRAY TO PIPE
if ((a_len1 = write(fd[1], a, a_len)) != a_len)
{
    perror("write arr error");
    exit(2);
}
sleep(1);
// CHILD
int child = fork();
if (child == -1)
{
    perror("Error during creating fork\n");
    exit(4);
}
else if (child == 0)
{
    char *args[] = {file_name, NULL};
    if (close(fd[1]) == -1)
    {
        perror("close fd error");
        exit(5);
    }
    if (dup2(fd[0], STDIN_FILENO) == -1)
    {
        perror("Descriptor fd error output\n");
        exit(7);
    }
    fflush(stdout);
    // EXEC
    execv("./child", args);
    perror("Exec error\n");
}

```

```
    }  
    return 0;  
}
```

child.c

```
#include <unistd.h>  
#include <stdio.h>  
#include <string.h>  
#include <stdlib.h>  
#include <fcntl.h>  
#include <wait.h>  
#include <sys/types.h>  
#include <sys/stat.h>  
#include <sys/uio.h>  
#define BUF_SIZE 128  
int main(int argv, char *args[])  
{  
    char buf[BUF_SIZE];  
    char * file_name = args[0];  
    int f = open(file_name, O_WRONLY | O_CREAT | O_TRUNC, 0777);  
    if (f == -1)  
    {  
        perror("Error during creating file\n");  
        exit(1);  
    }  
    if (read(STDIN_FILENO, buf, BUF_SIZE) < 0)  
    {  
        perror("read error");  
        exit(2);  
    }  
    int neg = 0;  
    // FIRST NUMBER  
    char * result = malloc(sizeof(char) * BUF_SIZE);  
    int len = 0;  
    char c = buf[len];
```

```

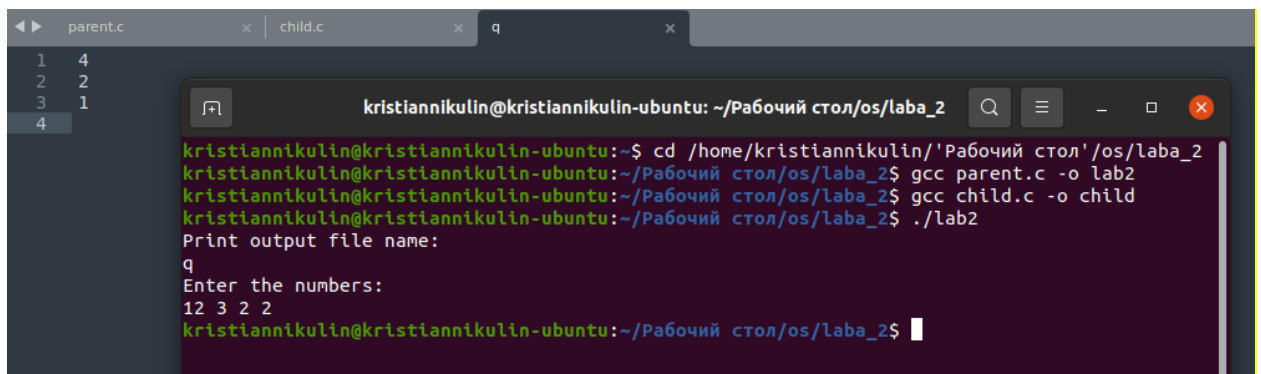
if (c == '-')
{
    neg = 1;
}
while(c != ' ') {
    result[len] = c;
    len++;
    c = buf[len];
}
len++;
result[len] = '\0';
c = buf[len];
int res = atoi(result);
if (neg)
{
    res *= -1;
}
// NUMBERS
int index = 0;
while((c >= 48 && c <= 57) || c == ' ' || c == '-') {
    char * number = malloc(sizeof(char) * BUF_SIZE);
    index = 0;
    while(c != ' ') {
        number[index] = c;
        index++;
        len++;
        c = buf[len];
    }
    len++;
    c = buf[len];
    index++;
    number[index] = '\0';
    int num = atoi(number);
    if (num < 0)

```

```
{
    num *= -1;
    if (neg == 1)
    {
        neg = 0;
    }
    else
    {
        neg = 1;
    }
}
if (num == 0)
{
    char *zero = "Zero found\n";
    write(1, zero, sizeof(char) * strlen(zero));

    break;
}
res /= num;
char buffer[30];
int len = sprintf(buffer, "%d", res);
if (neg)
{
    write(f, "-", 1);
}
write(f, buffer, len);
write(f, "\n", 1);
free(number);
}
return 0;
}
```

Пример работы



```
parent.c x child.c x q x
1 4
2 2
3 1
4

kristiannikulin@kristiannikulin-ubuntu: ~/Рабочий стол/os/laba_2
kristiannikulin@kristiannikulin-ubuntu:~/Рабочий стол/os/laba_2$ cd /home/kristiannikulin/'Рабочий стол'/os/laba_2
kristiannikulin@kristiannikulin-ubuntu:~/Рабочий стол/os/laba_2$ gcc parent.c -o lab2
kristiannikulin@kristiannikulin-ubuntu:~/Рабочий стол/os/laba_2$ gcc child.c -o child
kristiannikulin@kristiannikulin-ubuntu:~/Рабочий стол/os/laba_2$ ./lab2
Print output file name:
q
Enter the numbers:
12 3 2 2
kristiannikulin@kristiannikulin-ubuntu:~/Рабочий стол/os/laba_2$
```

Вывод

Проделав лабораторную работу, я приобрёл практические навыки в управлении процессами в ОС Unix и обеспечении обмена данных между процессами с помощью каналов.