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

Лабораторная работ №2 по курсу

«Операционные системы»

Управление процессами в ОС

Студент: Шорохов Алексей Павлович
Группа: М8О-206Б-18
Вариант: 22
Преподаватель: Соколов Андрей Алексеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2019

Постановка задачи.

Родительский процесс представляет собой сервер по работе с массивами и принимает команды со стороны дочернего процесса.

Общие сведения о программе

Программа компилируется из файла os.c. В программе используются следующие системные вызовы:

1. **read** – для чтения данных из pipe.
2. **write** – для записи данных в pipe.
3. **pipe** – для создания однонаправленного канала, через который из дочернего процесса данные передаются родительскому. Возвращает два дескриптора файлов: для чтения и для записи.
4. **fork** – для создания дочернего процесса.
5. **close** – для закрытия pipe после окончания считывания результата выполнения команды.

Общий метод и алгоритм решения.

С помощью fork() создаётся дочерний процесс. Для обмена данными между дочерним и родительским процессами создаём pipe. Если pid равен нулю, то идёт выполнение процессов клиента. В нём мы принимаем данные массива из файла input.txt, которые мы перенаправляем в наш сервер, где и происходит работа с массивом, в моей реализации это умножение элементов на 10. Далее сервер записывает полученные данные в файл output.txt. Для объективности проверки работы программы используется собственный генератор тестов на Python.

Файлы программы.

os.c

```
#include <unistd.h>
#include <sys/types.h>
#include <errno.h>
#include <stdio.h>
#include <sys/wait.h>
#include <stdlib.h>



void server(int fd1[2], int fd2[2]){
    close(fd1[0]); // close reading end of first pipe
    close(fd2[1]); // close writing end of second pipe

    int writefd = fd1[1], readfd = fd2[0];

    FILE * write = fopen("output.txt", "w");
```

```
    int size;

    read(readfd, &size, sizeof(int));

    int array[size];
    for (int i = 0; i < size; i++) {
        read(readfd, &array[i], sizeof(int));
        array[i] *= 10;
    }

    fprintf(write, "%d\n", size);

    for (int i = 0; i < size; i++) {
        fprintf(write, "%d ", array[i]);
    }

    fclose(write);
    close(writefd);
    close(readfd);
}

void client(int fd1[2], int fd2[2]){
    close(fd1[1]);
    close(fd2[0]);

    int writefd = fd2[1], readfd = fd1[0];

    FILE * read = fopen("input.txt", "r");

    int size;
```

```
    fscanf(read, "%d", &size);

    int array[size];

    for (int i = 0; i < size; i++) {
        fscanf(read, "%d", &array[i]);
    }

    write(writefd, &size, sizeof(int));

    for (int i = 0; i < size; i++) {
        write(writefd, &array[i], sizeof(int));
    }

    fclose(read);

    close(writefd);
    close(readfd);
    exit(0);
}

int main(){
    // fd - read, write
    // fd1 - for parent to write
    // fd2 - for parent to read
    int fd1[2], fd2[2];
    if(pipe(fd1) == -1){
        fprintf(stderr, "Pipe failed");
        return 1;
```

```
    }
    if(pipe(fd2) == -1){
        fprintf(stderr, "Pipe failed");
        return 1;
    }



    pid_t childPID = fork();
    if(childPID == 0)client(fd1, fd2);
    else if (childPID > 0) server(fd1, fd2);
    else {
        fprintf(stderr, "fork() was not finished successfully");
        return 1;
    }
    return 0;
}
```

gen.py

```
import random
f = open('input.txt', 'w')
f.write('50\n')
for i in range(50):
        a = str(random.randint(-10000, 10000))
        f.write(a)
        f.write(' ')
f.close()
```

Демонстрация работы программы.

dobriyalien@LAPTOP-2MM4OK81:/mnt/c/Users/shoro/Desktop/MAI/2course/os/final $ python gen.py

dobriyalien@LAPTOP-2MM4OK81:/mnt/c/Users/shoro/Desktop/MAI/2course/os/final $ cat input.txt

50

4412 -6068 -5337 -5401 -285 -6386 2297 8935 4186 9585 719 319 -6937 9376 4026 -3569 9762 9740 -418 -96 598 -2170 -3170 7531 976 -6874 5587 -5162 5869 5569 3295 -1913 -3438 1699 225 -2310 903 6466 -5896 -9666 -3132 8962 7787 -2998 -1612 -3740 903 8726 4694 -730 dobriyalien@LAPTOP-2MM4OK81:/mnt/c/Users/shoro/Desktop/MAI/2course/os/final $ gcc os.c -o run

dobriyalien@LAPTOP-2MM4OK81:/mnt/c/Users/shoro/Desktop/MAI/2course/os/final $ ./run

dobriyalien@LAPTOP-2MM4OK81:/mnt/c/Users/shoro/Desktop/MAI/2course/os/final $ cat output.txt

50

44120 -60680 -53370 -54010 -2850 -63860 22970 89350 41860 95850 7190 3190 -69370 93760 40260 -35690 97620 97400 -4180 -960 5980 -21700 -31700 75310 9760 -68740 55870 -51620 58690 55690 32950 -19130 -34380 16990 2250 -23100 9030 64660 -58960 -96660 -31320 89620 77870 -29980 -16120 -37400 9030 87260 46940 -7300 dobriyalien@LAPTOP-2MM4OK81:/mnt/c/Users/shoro/Desktop/MAI/2course/os/final $

Вывод

Я научился создавать процессы, используя системный вызов fork(). Также я обрел навыки взаимодействия между процессами с помощью pipe(), получил знания о файловых дескрипторах и познакомился с моделью “сервер – клиент”.