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

Лабораторная работа 1 по курсу
«Операционные системы»
III Семестр
Управление процессами. Каналы.

Студент: Шляхтуров А.В
Группа: М8О-201Б-22
Преподаватель: Миронов Е. С.
Вариант 12
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2023

Цель работы

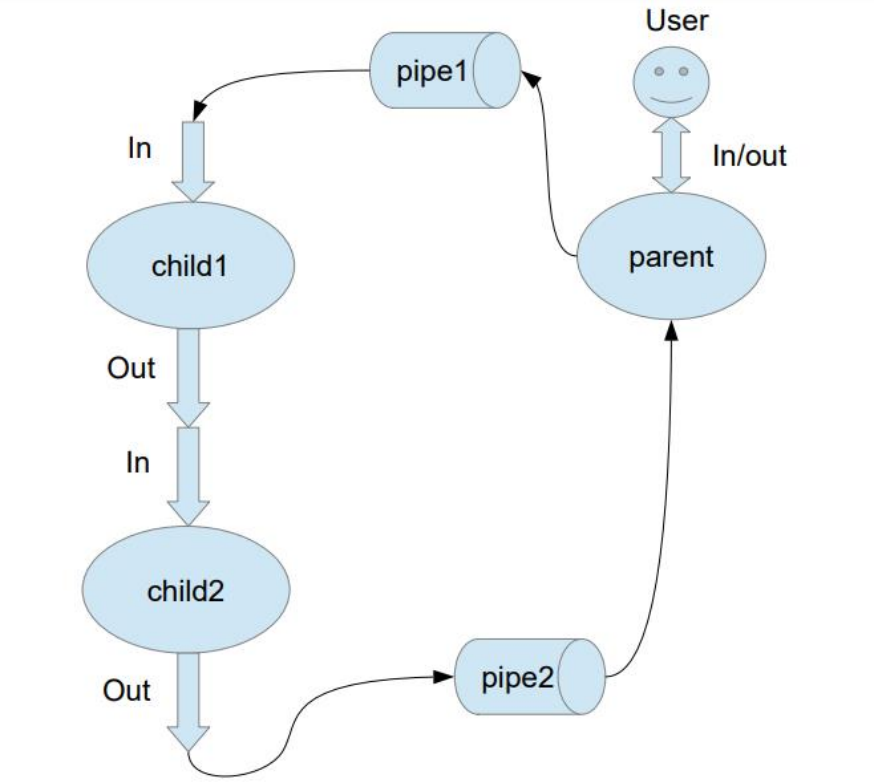
Приобретение практических навыков в управлении процессами в ОС и обеспечении обмена данных между процессами посредством каналов

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.

Описание идеи задачи

Родительский процесс создает два дочерних процесса. Перенаправление стандартных потоков ввода-вывода показано на картинке выше. Child1 и Child2 можно «соединить» между собой дополнительным каналом. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1. Процесс child1 и child2 производят работу над строками. Child2 пересылает результат своей работы родительскому процессу. Родительский процесс полученный результат выводит в стандартный поток вывода. Child1 переводит строки в верхний регистр. Child2 убирает все задвоенные пробелы.



Общий метод и алгоритм решения

Родительский процесс создает первый дочерний процесс. Первый дочерний процесс создает второй дочерний процесс. Далее, посредством pipe родительский процесс отправляет считанные данные в дочерний процесс. Дочерний процесс в свою очередь считывает их, переводит в верхний регистр и отправляет второму дочернему процессу. Второй дочерний процесс убирает все задвоенные пробелы и возвращает посредством pipe данные в родителя. Родитель выводит в консоль.

Код

Main.c

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <sys/types.h>
#include <sys/wait.h>

#include "../include/include.h"

int main()
{
    const int max_size_buff = 1024;

    int pipe1[2];
    int pipe2[2];

    if (pipe(pipe1) == -1 || pipe(pipe2) == -1)
    {
        perror("pipe");
        exit(EXIT_FAILURE);
    }

    pid_t child1 = create_processe();

    if (child1 == 0)
    {

        closeFD(pipe1[1]);
        closeFD(pipe2[0]);

        dup2FD(pipe1[0], 0);
        dup2FD(pipe2[1], 1);

        execl("../build/child1", " ", NULL);

        perror("child1");
        exit(EXIT_FAILURE);
    }

    int pipe3[2];
    if (pipe(pipe3) == -1)
    {
        perror("pipe");
        exit(EXIT_FAILURE);
    }

    pid_t child2 = create_processe();

    if (child2 == 0)
    {
```

```
        closeFD(pipe1[1]);
        closeFD(pipe1[0]);

        closeFD(pipe2[1]);
        closeFD(pipe3[0]);

        dup2FD(pipe2[0], 0);
        dup2FD(pipe3[1], 1);
        execl("../build/child2", " ", NULL);

        perror("child2");
        exit(EXIT_FAILURE);
    }

    closeFD(pipe1[0]);
    closeFD(pipe2[0]);
    closeFD(pipe2[1]);
    closeFD(pipe3[1]);

    char input_buffer[max_size_buff];

    char enter[20] = "Enter a string: \n";

    int x = write(1, enter, 30);

    // printf("записали врайтом симоволов: %d\n", x);

    // for(int i = 15; i <= 29; i++){
    //     printf("HERE %d\n", enter[i]);
    // }


    ssize_t size;
    while ((size = read(0, input_buffer, max_size_buff)) > 0)
    {
        write(pipe1[1], input_buffer, size);
    }

    if (size < 0)
    {
        perror("read 1");
        exit(EXIT_FAILURE);
    }

    closeFD(pipe1[1]);

    char result_buffer[max_size_buff];
    ssize_t n;

    char result[20] = "Result: \n";
```

```
    write(1, result, 20);


    while ((n = read(pipe3[0], result_buffer, sizeof(result_buffer))) > 0)
    {
        write(1, result_buffer, n);
    }

    if (n < 0)
    {
        perror("read 2");
        exit(EXIT_FAILURE);
    }

    closeFD(pipe3[0]);

    return 0;
}
```

Child1.c

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

#include "../include/include.h"

int main()
{

    char buffer[1];
    ssize_t n;
    while ((n = read(0, buffer, sizeof(buffer))) > 0)
    {
        char upper_array[n];
        for (int i = 0; i < n; i++)
        {
            upper_array[i] = toupper((unsigned char)buffer[i]);
        }
        write(1, upper_array, n);
    }
    if (n < 0)
    {
        perror("read 1");
        exit(EXIT_FAILURE);
    }
```

```
    closeFD(0);
    closeFD(1);
}
```

Child2.c

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

#include "../include/include.h"

int main()
{
    char buffer[1024];
    ssize_t n;
    while ((n = read(0, buffer, sizeof(buffer))) > 0)
    {
        char result[1024];
        int i = 0;
        int j = 0;
        while (j < n)
        {
            if (buffer[j] == ' ' && buffer[j + 1] == ' ')
            {
                if (buffer[j + 2] != ' ')
                {
                    result[i] = buffer[j + 2];
                    i += 1;
                    j += 3;
                }
                else
                {
                    j += 2;
                }
            }
            else
            {
                result[i] = buffer[j];
                j += 1;
                i += 1;
            }
        }
        write(1, result, i + 1);
    }
```

```
    if (n < 0) {
        perror("read 1");
        exit(EXIT_FAILURE);
    }

    closeFD(0);
    closeFD(1);
}
```

Пример работы

Input: aaa bb   cc dd

Output: AAABBCC DD

Выводы

В этой лабораторной работе мною были реализованы три программы, которые могли общаться между собой.

Я научился создавать процессы и связывать их с помощью канала pipe, чтобы передавать между ними информацию.

Эта работа была для меня интересной, потому что до этого я никогда не имел дела с созданием процессов в своем коде. Знания, полученные мною после выполнения этой лабораторной работы, могут помочь мне в будущем для написания параллельно исполняемого кода, где каждая его часть выполняется изолируемо от другой в своем адресном пространстве и без доступа к памяти другой части.