Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №2 по курсу

«Операционные системы»

Тема работы

“Изучение взаимодействий между процессами”

Студент: Морозов Артем Борисович
Группа: М8О-208Б-20
Вариант: 17
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021

Содержание

Репозиторий

https://github.com/artemmoroz0v/OS

Постановка задачи

Задача: реализовать программу, в которой родительский процесс создает два дочерних процесса. Родительский процесс принимает строки, которые отправляются в тот или иной дочерний процесс в зависимости от следующего правила: если длина строки больше 10 символов, то строка отправляется во второй дочерний процесс, в противном случае в первый дочерний процесс. Оба процесса удаляют гласные из строк.

Общие сведения о программе

Реализация программы была бы невозможна без специальной библиотеки “unistd.h” для операционной системы Linux, которая позволяет работать с процессами и системными вызовами.

По мере реализации задания используются такие строки(команды), как:

int fd[2] - создание массива из 2 дескрипторов, 0 - чтение (read), 1 - передача (write)

pipe(fd) - конвейер, с помощью которого выход одной команды подается на вход другой (оно же “труба”)

int id = fork () - создание дочернего процесса, в переменной id будет лежать “специальный код” процесса (-1 - ошибка fork, 0 - дочерний процесс, >0 - родительский)

read(...) - команда, предназначенная для чтения данных, посланных из другого процесса, принимающая на вход три параметра: элемент массива дескрипторов с индексом 0, значение **получаемого** объекта (переменной, массива и т.д.), размер **получаемого** объекта (например, в случае переменной int - sizeof(int), в случае массива из 10 переменных типа int - sizeof(int) * 10)

write(...) - команда, принимающая на вход три параметра: элемент массива дескрипторов с индексом 1, значение **посылаемого** объекта (переменной, массива и т.д.), размер **посылаемого** объекта (например, в случае переменной int - sizeof(int), в случае массива из 10 переменных типа int - sizeof(int) * 10)

close(...) - команда, использующаяся, когда нам больше не нужно передавать, либо считывать что-либо из другого процесса.

Общий метод и алгоритм решения

С самого начала программа получает два названия файлов для записи работы дочерних процессов. После этого эти оба файла создаются, и программа запрашивает у пользователя количество строк. Далее выполняется следующий алгоритм: после введения строки в консоль пользователь может увидеть ответ либо от первого дочернего процесса, либо от второго дочернего процесса, так как каждый процесс представляется, прежде чем вывести уже готовую строку пользователю (то есть строку с удаленными гласными). В самой программе удаление гласных представлено посредством вложенных

циклов while и пробегом по строке в поиске гласной при помощи кода ASCII (реализация будет представлена ниже в графе **“Исходный код” -** строки 69-81)

По окончании работы программы пользователь имеет выведенные без гласных строки и в консоли, и в созданном в самом начале файле, как и требовалось в задании.

Лабораторная работа была выполнена в среде Visual Studio code, название файла - Laba2.cpp.

Собирается программа при помощи команды g++ Laba2.cpp -o main, запускается при помощи команды ./main.

Исходный код

```
#include <iostream>
#include <string>
#include <fstream>
#include "unistd.h"
int main () {
std:: string path_child1, path_child2;
std:: cout << "Congrats, you are in parent process. Please enter name of files for first and second child!" << std:: endl;
std:: cout << "For first child: " << std:: endl;
std:: cin >> path_child1;
std:: cout << "For second child: " << std:: endl;
std:: cin >> path_child2;
std:: fstream fs;
int fd1[2];
int fd2[2];
if (pipe(fd1) == -1) {
    std:: cout << "An error occured with opening the pipe" << std:: endl;
    return 1;
}
if (pipe(fd2) == -1) {
    std:: cout << "An error occured with opening the pipe" << std:: endl;
    return 2;
}
int first_identificator = fork();
```

```
if (first_identificator == -1) {
    std:: cout << "Fork error!" << std:: endl;
    return -1;
}
else if (first_identificator == 0) {
    fs.open(path_child1, std:: fstream:: in | std:: fstream:: out | std:: fstream:: app);
    int a;
    read(fd1[0], &a, sizeof(int));
    std:: cout << "Congrats, you are in child #1 process" << std:: endl;
    while (a > 0) {
        int size_of_string;
        read(fd1[0], &size_of_string, sizeof(int));
        char string_array[size_of_string];
        read(fd1[0], string_array, sizeof(char) * size_of_string);
```

```
        std::string string;
        for (int i = 0; i < size_of_string; i++){
            string.push_back(string_array[i]);
        }
        int x = 0;
        while (x < string.size()) {
            while ((string[x] == char(65)) || (string[x] == char(69)) || (string[x] == char(73)) || (string[x] == char(79)) ||
                (string[x] == char(85)) || (string[x] == char(89)) || (string[x] == char(97)) || (string[x] == char(101)) ||
                    (string[x] == char(105)) || (string[x] == char(111)) || (string[x] == char(117)) || (string[x] == char(121)))
            {
                string.erase(string.begin() + x);
            }
                x++;
            }
        fs << string << std:: endl;
        std::cout << "From child #1, your string is: " << string << std:: endl;
            --a;
    }
        close(fd1[0]);
        close(fd1[1]);
    }
else {
    int second_identificator = fork();
    if (second_identificator == -1) {
        std:: cout << "Fork error!" << std:: endl;
        return -1;
    }
    else if (second_identificator == 0) {
    fs.open(path_child2, std:: fstream:: in | std:: fstream:: out | std:: fstream:: app);
    int a;
    read(fd2[0], &a, sizeof(int));
    std:: cout << "Congrats, you are in child #2 process!" << std:: endl;
    while (a > 0) {
    int size_of_string;
    read(fd2[0], &size_of_string, sizeof(int));
```

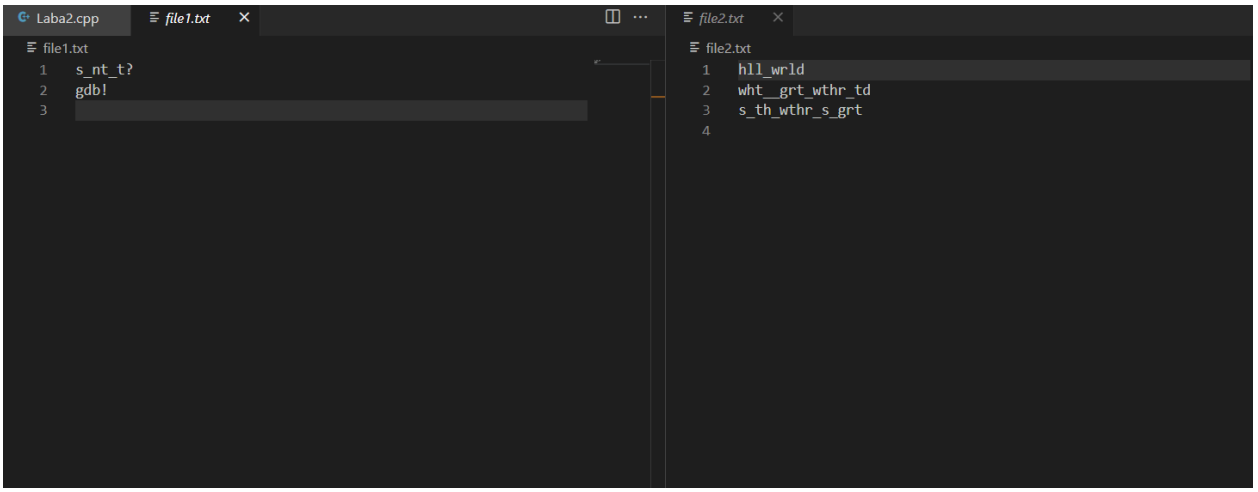
```
    char string_array[size_of_string];
    read(fd2[0], string_array, sizeof(char) * size_of_string);
    std::string string;
    for (int i = 0; i < size_of_string; i++){
    string.push_back(string_array[i]);
    }
    int x = 0;
    while (x < string.size()) {
        while ((string[x] == char(65)) || (string[x] == char(69)) || (string[x] == char(73)) || (string[x] == char(79)) ||
            (string[x] == char(85)) || (string[x] == char(89)) || (string[x] == char(97)) || (string[x] == char(101)) ||
                (string[x] == char(105)) || (string[x] == char(111)) || (string[x] == char(117)) || (string[x] == char(121)))
        {
            string.erase(string.begin() + x);
        }
            x++;
        }
    fs << string << std:: endl;
    std::cout << "From child #2, your string is: " << string << std:: endl;
    a = a - 1;
    }
    close(fd2[0]);
    close(fd2[1]);
    }
else {
    int a;
    std:: cout << "From parent: enter amount of strings" << std::endl;
    std:: cin >> a;
    write(fd1[1], &a, sizeof(int));
    write(fd2[1], &a, sizeof(int));
    std:: cout <<"Good, let's delete vowels in your strings " << a << " times: " << std::endl;
    for (int i = 0; i < a; i++){
    std:: string string;
    std:: cin >> string ;
    int n = string.size();
```

```
        char string_array[n];
        for (int i = 0; i < n; i++){
        string_array[i] = string[i];
        }
        if (string.size() <= 10) {
        write(fd1[1], &n, sizeof(int));
        write(fd1[1], string_array, sizeof(char) * n);
        }
        else {
        write(fd2[1], &n, sizeof(int));
        write(fd2[1], string_array, sizeof(char) * n);
        }
        }
        close(fd1[0]);
        close(fd1[1]);
        close(fd2[0]);
        close(fd2[1]);
        }
        }
        return 0;
}
```

Демонстрация работы программы

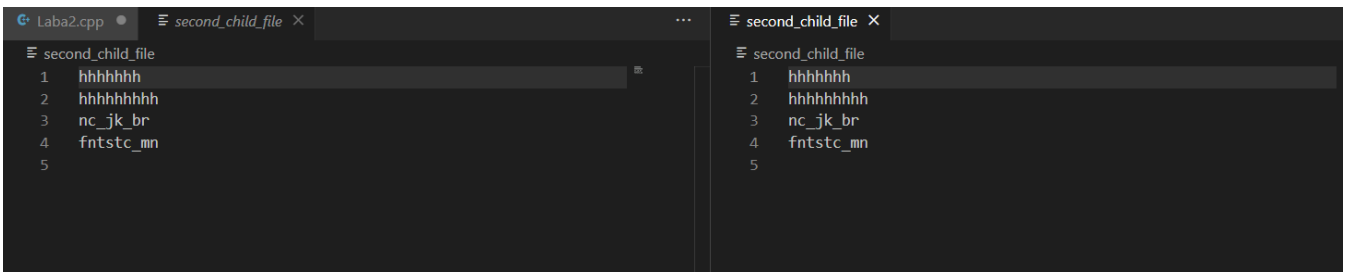
Тест 1.

```
moroz0v@LAPTOP-T5JMDNV1:~$ g++ Laba2.cpp -o main
moroz0v@LAPTOP-T5JMDNV1:~$ ./main
Congrats, you are in parent process!
Please enter name of files for first and second child!
For first child:
file1.txt
For second child:
file2.txt
From parent: enter amount of strings
5
Good, let's delete vowels in your strings 5 times:
Congrats, you are in child #1 process
Congrats, you are in child #2 process!
hello_world
From child #2, your string is: hll_wrld
what_a_great_weather_today
From child #2, your string is: wht__grt_wthr_td
is_not_it?
From child #1, your string is: s_nt_t?
yes_the_weather_is_great
From child #2, your string is: s_th_wthr_s_grt
goodbye!
From child #1, your string is: gdb!
moroz0v@LAPTOP-T5JMDNV1:~$
```



Тест 2.

```
moroz0v@LAPTOP-T5JMDNV1:~$ ./main
Congrats, you are in parent process!
Please enter name of files for first and second child!
For first child:
first_child_file
For second child:
second_child_file
From parent: enter amount of strings
10
Good, let's delete vowels in your strings 10 times:
Congrats, you are in child #2 process!
Congrats, you are in child #1 process
ahaha
From child #1, your string is: hh
ahahaha
From child #1, your string is: hhh
ahahahahahahaha
From child #2, your string is: hhhhhhh
ahahahahahahahah
From child #2, your string is: hhhhhhhhh
nice_joke_bro
From child #2, your string is: nc_jk_br
ahahah
From child #1, your string is: hhh
great
From child #1, your string is: grt
beatiful
From child #1, your string is: btfl
wonderful
From child #1, your string is: wndrfl
fantastic_man
From child #2, your string is: fntstc_mn
moroz0v@LAPTOP-T5JMDNV1:~$
```



Выводы

Данная лабораторная работа помогла мне ознакомиться с тем, как устроены процессы в Linux. Я осознал принцип работы вышеперечисленных команд и системных вызовов, а также узнал некоторые тонкости работы процессоров.