Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

Студент: Николаева Елизавета Сергеевна

Группа: М8О-201Б-20

Вариант: 20

Преподаватель: Миронов Евгений Сергеевич

Москва, 2021

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

https://github.com/esnikolaeva/OS\_2

**Постановка задачи**

Родительский процесс создает два дочерних процесса. Первой строкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия File с таким именем на запись для child1. Аналогично для второй строки и процесса child2. Родительский и дочерний процесс должны быть представлены разными программами.

Родительский процесс принимает от пользователя строки произвольной длины и пересылает их в pipe1 или в pipe2 в зависимости от правила фильтрации. Процесс child1 и child2 производят работу над строками. Процессы пишут результаты своей работы в стандартный вывод.

Вариант 20) Правило фильтрации: строки длины больше 10 символов отправляются в pipe2, иначе в pipe1. Дочерние процессы инвертируют строки.

**Общие сведения о программе**

Вся программа содержится в одном файле laba2.cpp

**Общий метод и алгоритм решения**

Запуск осуществляется при помощи ввода в командную строку Unix:

./laba2

**Исходный код**

#include <iostream>

#include <string>

#include <fstream>

#include "unistd.h"

int main(){

std::string th\_child1;

std::string th\_child2;

std::cout << "This is parent process" << std::endl;

std::cout << "Enter names for first and second childs" << std::endl;

std::cin >> th\_child1;

std::cin >> th\_child2;

std::fstream fs;

int fd1[2];

pipe(fd1);

int fd2[2];

pipe(fd2);

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

std::cout << "Error" << std::endl;

return 1;

}

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

std::cout << "Error" << std::endl;

return 1;

}

int first\_id = fork();

if (first\_id == -1){

std::cout << "Error" << std::endl;

return -1;

}

else if (first\_id == 0){ //work with 1 child (when <= 10 words)

fs.open(th\_child1, std:: fstream:: in | std:: fstream:: out | std:: fstream:: app);

int a = 0;

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

std::cout << "Your in child 1 process" << std::endl;

while(a > 0){

int size;

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

char array[size];

read(fd1[0], array, sizeof(char)\*size);

std::string string;

for (int i = 0; i < size; i++){

string.push\_back(array[i]);

}

for(int i = 0; i < size/2; i++){ //invert

char tmp = string[i];

string[i] = string[size-i-1];

string[size-i-1] = tmp;

}

fs << string << std::endl;

std::cout<<"After work in child 1 your string look as: "<< string <<std::endl;

a = a - 1;

}

close(fd1[0]);

close(fd1[1]);

}

else {

int second\_id = fork();

if (second\_id == -1){

std::cout<<"Error"<<std::endl;

return -1;

}

else if (second\_id == 0) { //work with 2 child (when > 10 words)

fs.open(th\_child1, std:: fstream:: in | std:: fstream:: out | std:: fstream:: app);

int a;

read(fd2[0], &a, sizeof(int));

std::cout << "Your in child 2 process" << std::endl;

while(a > 0) {

int size;

read(fd2[0], &size, sizeof(int));

char array[size];

read(fd2[0], array, sizeof(char)\*size);

std::string string;

for (int i = 0; i<size; i++){

string.push\_back(array[i]);

}

for (int i = 0; i < size/2; i++){ //invert

char tmp = string[i];

string[i] = string[size-i-1];

string[size-i-1] = tmp;

}

fs << string << std::endl;

std::cout << "After work in child 2 your string look as: "<< string << std::endl;

a = a - 1;

close(fd2[0]);

close(fd2[1]);

}

}

else {

int a;

std::cout << "PARENT: Please, enter number of string" << std::endl;

std::cin >> a;

write(fd1[1], &a, sizeof(int));

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

std::cout << "PARENT: Please, enter your strings" << a << "time" << std::endl;

for(int i = 0; i < a; i++){

std::string string;

std::cin >> string;

int num = string.size();

char array[num];

for (int i = 0; i < num; i++){

array[i] = string[i];

}

if (string.size() <= 10){

write(fd1[1], &num, sizeof(int));

write(fd1[1], array, sizeof(char)\* num);

}

else{

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

write(fd2[1], array, sizeof(char)\* num);

}

}

close(fd1[0]);

close(fd1[1]);

close(fd2[0]);

close(fd2[1]);

}

return 0;

}

}

**Выводы**

Я приобрела навыки в управлении процессами в ОС Unix и обеспечении обмена данных между процессами при помощи каналов.