**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Операционные системы»

**Лабораторная работа № 2**

Тема: Управление процессами в ОС

Студент: Тимофеев А.В.

Группа: 80-207

Преподаватель: Миронов Е. С.

Дата:

Оценка:

**Постановка задачи**

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 21.

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 21) Правило фильтрации: нечетные строки отправляются в pipe1, четные в pipe2. Дочерние процессы инвертируют строки.

**Алгоритм решения**

Родительский процесс считывает две строки, в которых указывается имя первого и в торого файла соответственно. Далее создаются файлы с данными именами, открываются два pipe и создаются 2 дочерних процесса. После создания, дочерние процессы переопределяют свой ввод на

pipe и вывод в файл и запускают саму программу дочернего процесса при

помощи exec.

В это время родительский процесс начинает считывать поток ввода посимвольно, пока не встретиться ‘\n’ все записывается в первый pipe, если встречается ‘\n’ он переключается на другой pipe и пишет в него.

Дочерние процессы считывают символы из каналов в массив char-ов (он ограничен 256 символов).О завершении строки сигнализирует символ переводы строки. После этого строка переворачивается и печатается.

**Листинг программы**

**main.c**

#include <unistd.h>

#include <stdio.h>

#include <string.h>

int main(){

char fn1[256];

char fn2[256];

if (scanf("%s", fn1) <= 0){

perror("scanf error");

return -1;

}

if (scanf("%s", fn2) <= 0){

perror("scanf error");

return -1;

}

FILE\* file1 = fopen(fn1, "wt");

if (file1 == NULL){

perror("fopen error");

return -1;

}

FILE\* file2 = fopen(fn2, "wt");

if (file2 == NULL){

perror("fopen error");

return -1;

}

int fd1[2];

int fd2[2];

if ((pipe(fd1) == -1) || (pipe(fd2) == -1)){

printf("Error creating pipe!");

return 1;

}

int id1 = fork();// первый дочерний процесс

if (id1 == -1) {

perror("fork error");

return -1;

}else if (id1 > 0){// родительский процесс

int id2 = fork();// второй дочерний процесс

if (id2 == -1){

perror("fork error");

return -1;

}else if (id2 > 0){ // родительский процесс

close(fd1[0]);

close(fd2[0]);

fclose(file1);

fclose(file2);

char c;

int n = 1;

while (getchar() != '\n');

while (scanf("%c", &c) > 0){// заполняем pipes

if (c != '\n'){

if (n % 2 != 0){

write(fd1[1], &c, sizeof(char));

}else{

write(fd2[1], &c, sizeof(char));

}

}else{

if (n % 2 != 0){

write(fd1[1], &c, sizeof(char));

}else{

write(fd2[1], &c, sizeof(char));

}

n++;

}

}

close(fd1[1]);

close(fd2[1]);

}else{// второй дочерний процесс

fclose(file1);

close(fd1[0]);

close(fd1[1]);

close(fd2[1]);

if (dup2(fd2[0], fileno(stdin)) == -1){

perror("dup2 error");

return -1;

}

if (dup2(fileno(file2), fileno(stdout)) == -1){

perror("dup2 error");

return -1;

}

fclose(file2);

char \* const \* null = NULL;

if (execv("child.out", null) == -1){

perror("execv error");

return -1;

}

}

}else{// первый дочерний процесс

fclose(file2);

close(fd2[0]);

close(fd2[1]);

close(fd1[1]);

if (dup2(fd1[0], fileno(stdin)) == -1){

perror("dup2 error");

return -1;

}

if (dup2(fileno(file1), fileno(stdout)) == -1){

perror("dup2 error");

return -1;

}

fclose(file1);

char \* const \* null = NULL;

if (execv("child.out", null) == -1){

perror("execv error");

return -1;

}

}

return 0;

}

**child.c**

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main() {

char c;

int n = 0;

int cap = 256;

char str[256];

memset(str,0,256);

char tmp ;

while(read(fileno(stdin), &c, sizeof(char)) > 0){

if((c != '\n') && (n < 254)){

str[n] = c;

++n;

}else{

if(c != '\n'){

str[n] = c;

}

short len = strlen(str);

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

tmp = str[len - i - 1];

str[len - i - 1] = str[i];

str[i] = tmp;

}

printf("%s",str);

printf("\n");

memset(str,0,256);

n = 0;

}

}

}

**Тесты и протокол исполнения**

dude@DESKTOP-545VSUH:/mnt/d/education/education/OCu/My\_OS\_labs/OC/mylaba2$ ./a.out

file

file1

Lorem ipsum dolor sit

amet,

consectetur

adipiscing elit,

sed do eiusmod

tempor

incididunt

ut labore

et dolore

magna aliqua.

dude@DESKTOP-545VSUH:/mnt/d/education/education/OCu/My\_OS\_labs/OC/mylaba2$ ls

a.out child.c child.out example file file1 main.c makefile

dude@DESKTOP-545VSUH:/mnt/d/education/education/OCu/My\_OS\_labs/OC/mylaba2$ cat file

tis rolod muspi meroL

rutetcesnoc

domsuie od des

tnudidicni

erolod te

dude@DESKTOP-545VSUH:/mnt/d/education/education/OCu/My\_OS\_labs/OC/mylaba2$ cat file1

,tema

,tile gnicsipida

ropmet

erobal tu

.auqila angam

**Вывод**:

**Список литературы**

1. Таненбаум Э., Бос Х. Современные операционные системы. – 4-е изд. – СПб.: Издательский дом «Питер», 2018.