Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

**Управление процессами в ОС. Обеспечение обмена данных между процессами посредством каналов.**

Студент: Почечура Артемий Андреевич

Группа: М80-206Б-20

Вариант: 14

Преподаватель: Соколов Андрей Алексеевич

Дата: 9.10.2021

Оценка: 5

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**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.

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.

14 вариант) Child1 переводит строки в нижний регистр. Child2 убирает все задвоенные пробелы.

**Листинг программы**

**child1.c**

#include <stdio.h>

#include <unistd.h>

#include <ctype.h>

int main(){

int n;

read(0,&n,sizeof(int));

char S[n];

read(0, &S, sizeof(char)\*n);

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

S[i]=tolower(S[i]);

}

write(1, &n, sizeof(int));

write(1, &S, sizeof(char)\*n);

return 0;

}

**child2.c**

#include <stdio.h>

#include <unistd.h>

#include <ctype.h>

int main(){

int n;

read(0, &n, sizeof(int));

char S[n];

char K[n];

int r=1;

read(0, &S, sizeof(char)\*n);

K[0]=S[0];

for(int i=1;i<n;i++){

if(S[i]==' ' && S[i-1]==' '){

continue;

}

K[r]=S[i];

r++;

}

write(1, &r, sizeof(int));

write(1, &K, sizeof(char)\*r);

return 0;

}

**main.c**

#include <stdio.h>

#include <unistd.h>

#include <ctype.h>

int main(){

int fd1[2], fd2[2], fd3[2];

pipe(fd1);

pipe(fd2);

pipe(fd3);

int id1 = fork();

if (id1 == -1) {

perror("fork error");

return -1;

} else if (id1 == 0){

if (dup2(fd1[0], STDIN\_FILENO) < 0) {

perror("Can't redirect stdin for child process");

}

if (dup2(fd2[1], STDOUT\_FILENO) < 0) {

perror("Can't redirect stdout for child process");

}

close(fd1[0]);

close(fd2[1]);

execl("child1.out", "", NULL);

} else{

int id2 = fork();

if (id2 == -1) {

perror("fork error");

return -1;

} else if(id2==0){

if (dup2(fd2[0], STDIN\_FILENO) < 0) {

perror("Can't redirect stdin for child process");

}

if (dup2(fd3[1], STDOUT\_FILENO) < 0) {

perror("Can't redirect stdout for child process");

}

close(fd2[0]);

close(fd3[1]);

execl("child2.out", "", NULL);

} else {

int n=0;

char p;

read(0,&p,sizeof(char));

while(p!='\n'){

if(p<'0' || p>'9'){

printf("Wrong values\n");

while(read(0,&p,sizeof(char))){

}

return -1;

}

n=n\*10+p-'0';

read(0,&p,sizeof(char));

}

char S[n];

read(0,&S,sizeof(char)\*n);

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

write(fd1[1], &S, sizeof(char)\*n);

read(fd3[0], &n, sizeof(int));

read(fd3[0], &S, sizeof(char)\*n);

write(1, &S, sizeof(char)\*n);

p='\n';

write(1, &p, sizeof(char));

close(fd1[1]);

close(fd3[0]);

}

}

return 0;

}

**Примеры работы**

root@DESKTOP-5HM2HTK:~#./a.out

10

AAAA AAA

aaaa aaa

root@DESKTOP-5HM2HTK:~#./a.out

5

KKKK

kkkk

root@DESKTOP-5HM2HTK:~#./a.out

8

R y

r y

**Вывод**

Данная работа была для меня сложной из-за того, что я в первый раз работал с процессами. Но в то же время было крайне интересно передавать строку между тремя процессами, контролируя последовательность выполнения программы. Эта работа научила меня правильно создавать pipes на Си, создавать новые процессы, переназначать ввод и вывод программы в pipe с помощью dup2. Думаю, что эти знания мне пригодятся в будущем.