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: Речинская Ангелина Юрьевна

Группа: М80-206Б-20

Вариант: 1

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 решения 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 дочерний процесс. Первой строчкой пользователь в консоль родительского процесса пишет имя файла, которое будет передано при создании дочернего процесса. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс передает команды пользователя через pipe1, который связан с стандартным входным потоком дочернего процесса. Дочерний процесс при необходимости передает данные в родительский процесс через pipe2. Результаты своей работы дочерний процесс пишет в созданный им файл. Допускается просто открыть файл и писать туда, не перенаправляя стандартный поток вывода.

1 вариант) Пользователь вводит команды вида: «число число число». Далее эти числа передаются от родительского процесса в дочерний. Дочерний процесс считает их сумму и выводит её в файл. Числа имеют тип int. Количество чисел может быть произвольным.

**Листинг программы**

**child.c**

#include <stdio.h>

#include <unistd.h>

#include <ctype.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

int main(int argc, char\* argv[]){

if (argc != 1){

perror("incorrect number of arguments\n");

exit(EXIT\_FAILURE);

}

int out=open(argv[0],O\_CREAT | O\_WRONLY | O\_TRUNC);

if (out==-1){

close(out);

perror("I can not open file\n");

exit(EXIT\_FAILURE);

}

int k;

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

while(k!=-1){

int M[k];

read(0, &M, sizeof(int)\*k);

int sum=0;

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

sum = sum + M[i];

}

//printf("%d\n",sum);

char t[11];

int u;

int j=0;

int sum1=sum;

if (sum1<0){

t[j]='-';

j++;

sum1\*=-1;

}

if (sum==0){

t[j]='0';

j++;

}

while(sum1!=0){

u=sum1%10;

sum1=sum1/10;

t[j]=u+'0';

j++;

}

char g[j];

if (t[0]=='-'){

g[0]='-';

for(int i=j-1;i>0;i--){

g[j-i]=t[i];

}

}

else{

for(int i=j-1;i>-1;i--){

g[j-i-1]=t[i];

}

}

char o='\n';

write(out,&g,sizeof(char)\*j);

write(out,&o,sizeof(char));

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

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

}

close(out);

return 0;

}

**main.c**

#include <stdio.h>

#include <unistd.h>

#include <ctype.h>

#include <stdlib.h>

int main(){

char string[100];

char p;

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

int z=0;

while(p!='\n'){

string[z]=p;

z++;

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

}

char file\_name[z];

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

file\_name[i]=string[i];

}

int fd1[2], fd2[2];

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

perror("pipe1");

exit(EXIT\_FAILURE);

}

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

perror("pipe2");

exit(EXIT\_FAILURE);

}

int id1 = fork();

if (id1 == -1) {

perror("fork error");

exit(EXIT\_FAILURE);

} else if (id1 == 0){

close(fd1[1]);

close(fd2[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("child.out", file\_name, NULL);

} else{

close(fd1[0]);

close(fd2[1]);

int M[64];

int k=0;

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

while(p!='\n'){

int n=0;

int z=1;

while(p==' ' || p=='\t'){

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

}

if (p=='-'){

z=-1;

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

}

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

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

}

exit(EXIT\_FAILURE);

}

while(p!=' ' && p!='\n' && p!='\t'){

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

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

}

exit(EXIT\_FAILURE);

}

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

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

}

M[k]=z\*n;

k++;

}

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

write(fd1[1], &M, sizeof(int)\*k);

int sum=0;

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

char t[11];

int u;

int j=0;

if (sum<0){

p='-';

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

sum\*=-1;

}

if (sum==0){

p='0';

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

}

while(sum!=0){

u=sum%10;

sum=sum/10;

t[j]=u+'0';

j++;

}

for(int i=j-1;i>-1;i--){

write(1, &t[i], sizeof(char));

}

p='\n';

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

k=0;

}

k=-1;

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

close(fd1[1]);

close(fd2[0]);

}

return 0;

}

**Примеры работы**

salutik@salutik:~/un/OS/lab2$ ./z

qwerty

1 2 3 4 5

15

1 -1

0

0 0

0

1

1

-3 -5 -7

-15

salutik@salutik:~/un/OS/lab2$ cat qwerty

15

0

0

1

-15

**Вывод**

В процессе работы над лабораторной я научилась основам работы с конвейерами и процессами в Си. Процессы занимают важную роль в разработке ПО, так как программы зачастую состоят из нескольких, относительно обособленных, подпрограмм, то есть процессов. Конвейеры как один из способов обмена данными между процессами также играют немаловажную роль.