Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа**

**по курсу «Операционные системы»**

**III Семестр**

**Задание 2**

**Вариант 11**

|  |  |
| --- | --- |
| Студент: | Анисимов В.А. |
| Группа: | М80-208Б-18 |
| Преподаватель: | Миронов Е.С |
|  |  |
| Оценка: |  |
| Дата: |  |

Москва 2019

**1. Описание задания**

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 11:

Написание собственного простого целочисленного калькулятора с операциями "+","-". В дочернем процессе должны происходить вычисления выражений. В родительском процессе ввод/вывод.

**2. Код программы**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

int main()

{

int fd1[2], fd2[2];

pid\_t pid;

int arr[4];

char op;

if (pipe(fd1) < 0 || pipe (fd2) < 0){

printf("Cant create pipes\n");

return -1;

}

pid = fork();

if(pid == -1){

printf("Can\'t fork\n");

return -1;

}else if(pid == 0){

close(fd1[1]);

close(fd2[0]);

while(read(fd1[0], arr, 4 \* sizeof(int))){

if (arr[1] == 1)

arr[3] = arr[0] + arr[2];

if (arr[1] == -1)

arr[3] = arr[0] - arr[2];

write(fd2[1], arr, 4 \* sizeof(int));

}

close(fd1[0]);

close(fd2[1]);

}else{

close(fd1[0]);

close(fd2[1]);

while(scanf("%d %c %d", &arr[0], &op, &arr[2]) == 3) {

if (op == '+')

arr[1] = 1;

if (op == '-')

arr[1] = -1;

write(fd1[1], arr, 4 \* sizeof(int));

read(fd2[0], arr, 4 \* sizeof(int));

printf("%d %c %d = %d\n", arr[0], op, arr[2], arr[3]);

}

close(fd1[1]);

close(fd2[0]);

}

return 0;

}

**3. Протокол работы программы**

walien@PC-name:~/2kurs/OS/lab2$ ./OS\_lab2

5+5

5 + 5 = 10

7-7

7 - 7 = 0

999-888

999 - 888 = 111

400-200

400 - 200 = 200

**5. Объяснение работы программы**

Создаётся пайп и дочерний процесс при помощи команды fork(). Родительский процесс считывает из стандартного потока ввода два числа и операнд и записывает их в виде целочисленного массива arr в pipe.

Далее родитель ожидает, пока потомок считает из пайпа массив с данными, произведёт вычисления и запишет результат обратно в пайп.

После чего родительский процесс считывает из пайпа результат работы дочернего процесса и выводит его в стандартный поток вывода.

**6. Strace**

walien@PC-name:~/2kurs/OS/lab2$ strace ./OS\_lab2

execve("./OS\_lab2", ["./OS\_lab2"], 0x7ffd445c8400 /\* 63 vars \*/) = 0

brk(NULL) = 0x55fdbf0ed000

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

fstat(3, {st\_mode=S\_IFREG|0644, st\_size=83072, ...}) = 0

mmap(NULL, 83072, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f5485cb7000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libpthread.so.0", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0000b\0\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=144976, ...}) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f5485cb5000

mmap(NULL, 2221184, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f5485886000

mprotect(0x7f54858a0000, 2093056, PROT\_NONE) = 0

mmap(0x7f5485a9f000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19000) = 0x7f5485a9f000

mmap(0x7f5485aa1000, 13440, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f5485aa1000

close(3) = 0

access("/etc/ld.so.nohwcap", F\_OK) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\260\34\2\0\0\0\0\0"..., 832) = 832

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2030544, ...}) = 0

mmap(NULL, 4131552, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f5485495000

mprotect(0x7f548567c000, 2097152, PROT\_NONE) = 0

mmap(0x7f548587c000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7f548587c000

mmap(0x7f5485882000, 15072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f5485882000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f5485cb2000

arch\_prctl(ARCH\_SET\_FS, 0x7f5485cb2740) = 0

mprotect(0x7f548587c000, 16384, PROT\_READ) = 0

mprotect(0x7f5485a9f000, 4096, PROT\_READ) = 0

mprotect(0x55fdbd39b000, 4096, PROT\_READ) = 0

mprotect(0x7f5485ccc000, 4096, PROT\_READ) = 0

munmap(0x7f5485cb7000, 83072) = 0

set\_tid\_address(0x7f5485cb2a10) = 3459

set\_robust\_list(0x7f5485cb2a20, 24) = 0

rt\_sigaction(SIGRTMIN, {sa\_handler=0x7f548588bcb0, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_SIGINFO, sa\_restorer=0x7f5485898890}, NULL, 8) = 0

rt\_sigaction(SIGRT\_1, {sa\_handler=0x7f548588bd50, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7f5485898890}, NULL, 8) = 0

rt\_sigprocmask(SIG\_UNBLOCK, [RTMIN RT\_1], NULL, 8) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

pipe([3, 4]) = 0

pipe([5, 6]) = 0

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f5485cb2a10) = 3460

close(3) = 0

close(6) = 0

fstat(0, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(136, 0), ...}) = 0

brk(NULL) = 0x55fdbf0ed000

brk(0x55fdbf10e000) = 0x55fdbf10e000

read(0, 5+5

"5+5\n", 1024) = 4

write(4, "\5\0\0\0\1\0\0\0\5\0\0\0\375U\0\0", 16) = 16

read(5, "\5\0\0\0\1\0\0\0\5\0\0\0\n\0\0\0", 16) = 16

fstat(1, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(136, 0), ...}) = 0

write(1, "5 + 5 = 10\n", 115 + 5 = 10

) = 11

read(0, 7-7

"7-7\n", 1024) = 4

write(4, "\7\0\0\0\377\377\377\377\7\0\0\0\n\0\0\0", 16) = 16

read(5, "\7\0\0\0\377\377\377\377\7\0\0\0\0\0\0\0", 16) = 16

write(1, "7 - 7 = 0\n", 107 - 7 = 0

) = 10

read(0, "", 1024) = 0

close(4) = 0

close(5) = 0

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

Выполнив лабораторную работу, я получил первоначальные знания о работе процессов в ОС Linux, научился пользоваться командами fork, pipe, wait, создавать с их помощью подпроцессы, передавать между ними данные, управлять состоянием процессов.