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>

#include <sys/types.h>

#include <sys/wait.h>

int main()

{

int fd[2], rv;

pid\_t pid;

int arr[4];

char op;

if (pipe(fd) < 0){

printf("Cant create pipe\n");

return -1;

}

pid = fork();

if(pid == -1){

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

return -1;

}else if(pid == 0){

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

close(fd[0]);

if (arr[1] == 1)

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

if (arr[1] == -1)

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

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

close(fd[1]);

exit(rv);

}else{

scanf("%d%c%d", &arr[0], &op, &arr[2]);

if (op == '+')

arr[1] = 1;

if (op == '-')

arr[1] = -1;

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

close(fd[1]);

wait(&rv);

if(WIFEXITED(rv) == 0){

printf("Child error\n");

return -2;

}

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

close(fd[0]);

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

}

return 0;

}

**3. Протокол работы программы**

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

15+15

15 + 15 = 30

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

99-66

99 - 66 = 33

**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"], 0x7ffccc8d71b0 /\* 63 vars \*/) = 0

brk(NULL) = 0x563877c06000

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) = 0x7f4b6689c000

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, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f4b6689a000

mmap(NULL, 4131552, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f4b66299000

mprotect(0x7f4b66480000, 2097152, PROT\_NONE) = 0

mmap(0x7f4b66680000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7f4b66680000

mmap(0x7f4b66686000, 15072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f4b66686000

close(3) = 0

arch\_prctl(ARCH\_SET\_FS, 0x7f4b6689b500) = 0

mprotect(0x7f4b66680000, 16384, PROT\_READ) = 0

mprotect(0x563876307000, 4096, PROT\_READ) = 0

mprotect(0x7f4b668b1000, 4096, PROT\_READ) = 0

munmap(0x7f4b6689c000, 83072) = 0

pipe([3, 4]) = 0

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f4b6689b7d0) = 5677

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

brk(NULL) = 0x563877c06000

brk(0x563877c27000) = 0x563877c27000

read(0, 5+5

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

write(4, "\5\0\0\0\1\0\0\0\5\0\0\0008V\0\0", 16) = 16

close(4) = 0

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=5677, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

wait4(-1, [{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 5677

read(3, "\5\0\0\0\1\0\0\0\5\0\0\0\n\0\0\0", 16) = 16

close(3) = 0

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

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

) = 11

lseek(0, -1, SEEK\_CUR) = -1 ESPIPE (Illegal seek)

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.