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 4**

**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 решения задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется с помощью отображения файла на память.

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 <signal.h>

#include <stdio.h>

#include <string.h>

#include <pthread.h>

#include <unistd.h>

#include <sys/mman.h>

#include <fcntl.h>

#include <semaphore.h>

#include <stdlib.h>

int main()

{

FILE\* tmp\_file = fopen("file", "w+");

if (!tmp\_file) {

printf("can`t open file\n");

return -1;

}

int fd = fileno(tmp\_file);

write(fd, "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 20);

int\* file\_ptr = mmap(NULL, 5, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0);

if (file\_ptr == MAP\_FAILED) {

printf("failed to map\n");

return -1;

}

sem\_t \*sem\_calc = sem\_open("/calc", O\_CREAT, 777, 0);

sem\_t \*sem\_out = sem\_open("/out", O\_CREAT, 777, 0);

if (sem\_calc == SEM\_FAILED || sem\_out == SEM\_FAILED) {

printf("Cant create semaphore(s)\n");

return -1;

}

sem\_unlink("/calc");

sem\_unlink("/out");

pid\_t pid;

int arr[4];

char op;

pid = fork();

if(pid == -1){

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

return -1;

}else if(pid == 0){

while(1){

sem\_wait(sem\_calc);

if(file\_ptr[4] == 1){

break;

}

for(int i = 0; i < 3; i++){

arr[i] = file\_ptr[i];

}

if (arr[1] == 1)

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

if (arr[1] == -1)

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

file\_ptr[3] = arr[3];

sem\_post(sem\_out);

}

sem\_close(sem\_calc);

sem\_close(sem\_out);

munmap(file\_ptr, 20);

close(fd);

}else{

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

if (op == '+')

arr[1] = 1;

if (op == '-')

arr[1] = -1;

for(int i = 0; i < 3; i++){

file\_ptr[i] = arr[i];

}

sem\_post(sem\_calc);

sem\_wait(sem\_out);

arr[3] = file\_ptr[3];

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

}

file\_ptr[4] = 1;

sem\_post(sem\_calc);

sem\_close(sem\_calc);

sem\_close(sem\_out);

close(fd);

}

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**

Межпроцессорное взаимодействие осуществляется через файл, отражённый в память.

Процесс-родитель считывает из стандартного ввода математическое выражение и записывает его в отражённый на память файл, после чего оповещает дочерний процесс, что данные готовы для вычисления посредством семафора sem\_calc. Дочерний процесс считывает данные, производит вычисления, записывает ответ и оповещает родительский процесс, что данные готовы к выводу посредством семафора sem\_out. Родительский процесс выводит решение в стандартный вывод.

Для межпроцессорного взаимодействия используются именованные семафоры.

**6. Strace**

walien@PC-name:~/2kurs/OS/lab4$ strace ./os\_lab4

execve("./os\_lab4", ["./os\_lab4"], 0x7ffc1770d900 /\* 63 vars \*/) = 0

brk(NULL) = 0x559612c74000

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=83168, ...}) = 0

mmap(NULL, 83168, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f949fe08000

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) = 0x7f949fe06000

mmap(NULL, 2221184, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f949f9d7000

mprotect(0x7f949f9f1000, 2093056, PROT\_NONE) = 0

mmap(0x7f949fbf0000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x19000) = 0x7f949fbf0000

mmap(0x7f949fbf2000, 13440, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f949fbf2000

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) = 0x7f949f5e6000

mprotect(0x7f949f7cd000, 2097152, PROT\_NONE) = 0

mmap(0x7f949f9cd000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e7000) = 0x7f949f9cd000

mmap(0x7f949f9d3000, 15072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f949f9d3000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f949fe03000

arch\_prctl(ARCH\_SET\_FS, 0x7f949fe03740) = 0

mprotect(0x7f949f9cd000, 16384, PROT\_READ) = 0

mprotect(0x7f949fbf0000, 4096, PROT\_READ) = 0

mprotect(0x559610f46000, 4096, PROT\_READ) = 0

mprotect(0x7f949fe1d000, 4096, PROT\_READ) = 0

munmap(0x7f949fe08000, 83168) = 0

set\_tid\_address(0x7f949fe03a10) = 6457

set\_robust\_list(0x7f949fe03a20, 24) = 0

rt\_sigaction(SIGRTMIN, {sa\_handler=0x7f949f9dccb0, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_SIGINFO, sa\_restorer=0x7f949f9e9890}, NULL, 8) = 0

rt\_sigaction(SIGRT\_1, {sa\_handler=0x7f949f9dcd50, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7f949f9e9890}, 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

brk(NULL) = 0x559612c74000

brk(0x559612c95000) = 0x559612c95000

openat(AT\_FDCWD, "file", O\_RDWR|O\_CREAT|O\_TRUNC, 0666) = 3

write(3, "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 20) = 20

mmap(NULL, 5, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 3, 0) = 0x7f949fe1c000

statfs("/dev/shm/", {f\_type=TMPFS\_MAGIC, f\_bsize=4096, f\_blocks=255097, f\_bfree=255071, f\_bavail=255071, f\_files=255097, f\_ffree=255070, f\_fsid={val=[0, 0]}, f\_namelen=255, f\_frsize=4096, f\_flags=ST\_VALID|ST\_NOSUID|ST\_NODEV}) = 0

futex(0x7f949fbf5370, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

openat(AT\_FDCWD, "/dev/shm/sem.calc", O\_RDWR|O\_NOFOLLOW) = -1 ENOENT (No such file or directory)

getpid() = 6457

lstat("/dev/shm/Pav60c", 0x7ffd9b943fe0) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/dev/shm/Pav60c", O\_RDWR|O\_CREAT|O\_EXCL, 01411) = 4

write(4, "\0\0\0\0\0\0\0\0\200\0\0\0\224\177\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 32) = 32

mmap(NULL, 32, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 4, 0) = 0x7f949fe1b000

link("/dev/shm/Pav60c", "/dev/shm/sem.calc") = 0

fstat(4, {st\_mode=S\_IFREG|S\_ISVTX|0411, st\_size=32, ...}) = 0

unlink("/dev/shm/Pav60c") = 0

close(4) = 0

openat(AT\_FDCWD, "/dev/shm/sem.out", O\_RDWR|O\_NOFOLLOW) = -1 ENOENT (No such file or directory)

getpid() = 6457

lstat("/dev/shm/0a8acj", 0x7ffd9b943ff0) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/dev/shm/0a8acj", O\_RDWR|O\_CREAT|O\_EXCL, 01411) = 4

write(4, "\0\0\0\0\0\0\0\0\200\0\0\0\224\177\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0", 32) = 32

mmap(NULL, 32, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 4, 0) = 0x7f949fe1a000

link("/dev/shm/0a8acj", "/dev/shm/sem.out") = 0

fstat(4, {st\_mode=S\_IFREG|S\_ISVTX|0411, st\_size=32, ...}) = 0

unlink("/dev/shm/0a8acj") = 0

close(4) = 0

unlink("/dev/shm/sem.calc") = 0

unlink("/dev/shm/sem.out") = 0

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f949fe03a10) = 6458

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

read(0, 100+50

"100+50\n", 1024) = 7

futex(0x7f949fe1b000, FUTEX\_WAKE, 1) = 1

futex(0x7f949fe1a000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, 0xffffffff) = 0

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

write(1, "100 + 50 = 150\n", 15100 + 50 = 150

) = 15

read(0, 200-50

"200-50\n", 1024) = 7

futex(0x7f949fe1b000, FUTEX\_WAKE, 1) = 1

futex(0x7f949fe1a000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, 0xffffffff) = -1 EAGAIN (Resource temporarily unavailable)

write(1, "200 - 50 = 150\n", 15200 - 50 = 150

) = 15

read(0, 3-2

"3-2\n", 1024) = 4

futex(0x7f949fe1b000, FUTEX\_WAKE, 1) = 1

futex(0x7f949fe1a000, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 0, NULL, 0xffffffff) = 0

write(1, "3 - 2 = 1\n", 103 - 2 = 1

) = 10

read(0, "", 1024) = 0

futex(0x7f949fe1b000, FUTEX\_WAKE, 1) = 1

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=6458, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

munmap(0x7f949fe1b000, 32) = 0

munmap(0x7f949fe1a000, 32) = 0

close(3) = 0

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

Memory-mapped files – механизм, позволяющий отображать файлы на участок памяти. Их использование может дать существенный прирост в производительности, по сравнению с обычной буферизированной работе с файлами. Так же они могут использоваться для межпроцессорного взаимодействия, т.к. при отображении файла на участок памяти все процессы разделяют эту память и имеют доступ к данным.