Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

**Лабораторная работа №3 по курсу**

**«Операционные системы»**

Группа: М8О-215Б-23

Студент: Тараскаев Д.М.

Преподаватель: Миронов Е.С.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 01.11.24

Москва, 2024

**Постановка задачи**

**Вариант 16.**

**Правило проверки: строка должна оканчиваться на “.” или “;”.**

**Общий метод и алгоритм решения**

В программе использовались следующие системные вызовы:

* shm\_open - открытие объекта разделяемой памяти.
* ftruncate - изменение размера файла.
* mmap - отображение файла в память.
* fork - создание нового процесса.
* execlp - замена текущего процесса новым.
* perror - вывод сообщения об ошибке.
* sleep - приостановка выполнения процесса.
* fopen - открытие файла.
* fputs - запись строки в файл.
* fclose - закрытие файла.
* strlen - вычисление длины строки.
* strncmp - сравнение строк.
* strcpy - копирование строки.

Родительский процесс выполняет следующие действия:

1. Создает объект разделяемой памяти с помощью shm\_open.
2. Устанавливает размер разделяемой памяти с помощью ftruncate.
3. Отображает объект разделяемой памяти в адресное пространство процесса с помощью mmap.
4. Создает новый процесс с помощью fork.
5. Если fork возвращает 0 (дочерний процесс), выполняет программу child с помощью execlp, передавая имя разделяемой памяти.
6. Если fork возвращает положительное значение (родительский процесс), выводит сообщение "Write a doc name".

Дочерний процесс выполняет следующие действия:

1. Открывает объект разделяемой памяти с помощью shm\_open.
2. Отображает объект разделяемой памяти в адресное пространство процесса с помощью mmap.
3. Ожидает, пока в разделяемой памяти не появится имя файла для записи.
4. Открывает файл для записи с помощью fopen.
5. В цикле проверяет содержимое разделяемой памяти:
6. Если сообщение заканчивается на ';' или '.', записывает его в файл и записывает в разделяемую память сообщение "Success".
7. Иначе записывает в разделяемую память сообщение "Not over in ';' or '.'".
8. Завершает работу, если в разделяемой памяти появляется сообщение "exit".
9. Закрывает файл с помощью fclose.

**Код программы**

**parent.c**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/mman.h>

#include <sys/wait.h>

#include <fcntl.h>

#define SHARED\_MEMORY\_SIZE 256

int main() {

const char \*shared\_memory\_name = "/shared\_memory";

int fd = shm\_open(shared\_memory\_name, O\_CREAT | O\_RDWR, 0666);

ftruncate(fd, SHARED\_MEMORY\_SIZE);

char \*shared\_memory = mmap(NULL, SHARED\_MEMORY\_SIZE, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0);

if (shared\_memory == MAP\_FAILED) {

perror("mmap");

return 1;

}

pid\_t pid = fork();

if (pid == -1) {

perror("fork");

return 1;

}

if (pid == 0) {

execlp("./child", "./child", shared\_memory\_name, NULL);

perror("execlp");

return 1;

} else {

printf("Write a doc name\n");

fgets(shared\_memory, SHARED\_MEMORY\_SIZE, stdin);

shared\_memory[strcspn(shared\_memory, "\n")] = 0;

while (1) {

printf("Enter a message (exit for exit)\n");

fgets(shared\_memory, SHARED\_MEMORY\_SIZE, stdin);

if (strncmp(shared\_memory, "exit", 4) == 0) {

strcpy(shared\_memory + 128, "exit");

break;

}

usleep(100000);

printf("Status: %s\n", shared\_memory + 128);

}

waitpid(pid, NULL, 0);

shm\_unlink(shared\_memory\_name);

}

return 0;

}

**child.c**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/mman.h>

#include <fcntl.h>

#define SHARED\_MEMORY\_SIZE 256

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

if (argc < 2) {

fprintf(stderr, "Usage: %s <shared\_memory\_name>\n", argv[0]);

return 1;

}

const char \*shared\_memory\_name = argv[1];

int fd = shm\_open(shared\_memory\_name, O\_RDWR, 0666);

if (fd == -1) {

perror("shm\_open");

return 1;

}

char \*shared\_memory = mmap(NULL, SHARED\_MEMORY\_SIZE, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fd, 0);

if (shared\_memory == MAP\_FAILED) {

perror("mmap");

return 1;

}

char \*read\_msg = shared\_memory;

char \*response\_msg = shared\_memory + 128;

while (strlen(read\_msg) == 0) {

sleep(1);

}

FILE \*fp = fopen(read\_msg, "w");

if (!fp) {

perror("file error");

return -1;

}

while (1) {

sleep(1);

if (strncmp(response\_msg, "exit", 4) == 0) {

break;

}

int len = strlen(read\_msg);

if (len > 0 && (read\_msg[len - 2] == ';' || read\_msg[len - 2] == '.')) {

fputs(read\_msg, fp);

strcpy(response\_msg, "Success");

} else {

strcpy(response\_msg, "Not over in ';' or '.'");

}

}

fclose(fp);

return 0;

}

**Протокол работы программы**

**Тестирование**

$ ./parent

Write a doc name

hellow

Enter a message

hellow;

Status: Success

Enter a message:

qwe

Status Not over in ';' or '.'

Enter a message

asfdd;

Status: Success

Enter a message

xddddd;

Status: Success

Enter a message

exit

**Strace:**

$ strace -f ./parent

execve("./parent", ["./parent"], 0x7fff222b3548 /\* 50 vars \*/) = 0

brk(NULL) = 0x58cbfc040000

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=121891, ...}) = 0

mmap(NULL, 121891, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7e9538879000

close(3) = 0

openat(AT\_FDCWD, "/usr/lib/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\340\_\2\0\0\0\0\0"..., 832) = 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2014520, ...}) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7e9538877000

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

mmap(NULL, 2034616, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7e9538686000

mmap(0x7e95386aa000, 1511424, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x24000) = 0x7e95386aa000

mmap(0x7e953881b000, 319488, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x195000) = 0x7e953881b000

mmap(0x7e9538869000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e3000) = 0x7e9538869000

mmap(0x7e953886f000, 31672, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7e953886f000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7e9538683000

arch\_prctl(ARCH\_SET\_FS, 0x7e9538683740) = 0

set\_tid\_address(0x7e9538683a10) = 30517

set\_robust\_list(0x7e9538683a20, 24) = 0

rseq(0x7e9538684060, 0x20, 0, 0x53053053) = 0

mprotect(0x7e9538869000, 16384, PROT\_READ) = 0

mprotect(0x58cbf187c000, 4096, PROT\_READ) = 0

mprotect(0x7e95388d1000, 8192, PROT\_READ) = 0

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

munmap(0x7e9538879000, 121891) = 0

openat(AT\_FDCWD, "/dev/shm/shared\_memory", O\_RDWR|O\_CREAT|O\_NOFOLLOW|O\_CLOEXEC, 0666) = 3

ftruncate(3, 256) = 0

mmap(NULL, 256, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 3, 0) = 0x7e9538896000

clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLDstrace: Process 30518 attached

, child\_tidptr=0x7e9538683a10) = 30518

[pid 30518] set\_robust\_list(0x7e9538683a20, 24 <unfinished ...>

[pid 30517] fstat(1, <unfinished ...>

[pid 30518] <... set\_robust\_list resumed>) = 0

[pid 30517] <... fstat resumed>{st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}) = 0

[pid 30517] getrandom("\x0b\x43\xf8\xda\x33\xbc\x71\x00", 8, GRND\_NONBLOCK) = 8

[pid 30518] execve("./child", ["./child", "/shared\_memory"], 0x7ffe560e3b68 /\* 50 vars \*/ <unfinished ...>

[pid 30517] brk(NULL) = 0x58cbfc040000

[pid 30517] brk(0x58cbfc061000) = 0x58cbfc061000

[pid 30517] write(1, "Write a doc name\n", 17Write a doc name

) = 17

[pid 30517] fstat(0, {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}) = 0

[pid 30517] read(0, <unfinished ...>

[pid 30518] <... execve resumed>) = 0

[pid 30518] brk(NULL) = 0x5b3b8209b000

[pid 30518] access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

[pid 30518] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

[pid 30518] fstat(3, {st\_mode=S\_IFREG|0644, st\_size=121891, ...}) = 0

[pid 30518] mmap(NULL, 121891, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7ef86a452000

[pid 30518] close(3) = 0

[pid 30518] openat(AT\_FDCWD, "/usr/lib/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

[pid 30518] read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\340\_\2\0\0\0\0\0"..., 832) = 832

[pid 30518] pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

[pid 30518] fstat(3, {st\_mode=S\_IFREG|0755, st\_size=2014520, ...}) = 0

[pid 30518] mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ef86a450000

[pid 30518] pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

[pid 30518] mmap(NULL, 2034616, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7ef86a25f000

[pid 30518] mmap(0x7ef86a283000, 1511424, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x24000) = 0x7ef86a283000

[pid 30518] mmap(0x7ef86a3f4000, 319488, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x195000) = 0x7ef86a3f4000

[pid 30518] mmap(0x7ef86a442000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1e3000) = 0x7ef86a442000

[pid 30518] mmap(0x7ef86a448000, 31672, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7ef86a448000

[pid 30518] close(3) = 0

[pid 30518] mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ef86a25c000

[pid 30518] arch\_prctl(ARCH\_SET\_FS, 0x7ef86a25c740) = 0

[pid 30518] set\_tid\_address(0x7ef86a25ca10) = 30518

[pid 30518] set\_robust\_list(0x7ef86a25ca20, 24) = 0

[pid 30518] rseq(0x7ef86a25d060, 0x20, 0, 0x53053053) = 0

[pid 30518] mprotect(0x7ef86a442000, 16384, PROT\_READ) = 0

[pid 30518] mprotect(0x5b3b715d7000, 4096, PROT\_READ) = 0

[pid 30518] mprotect(0x7ef86a4aa000, 8192, PROT\_READ) = 0

[pid 30518] prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

[pid 30518] munmap(0x7ef86a452000, 121891) = 0

[pid 30518] openat(AT\_FDCWD, "/dev/shm/shared\_memory", O\_RDWR|O\_NOFOLLOW|O\_CLOEXEC) = 3

[pid 30518] mmap(NULL, 256, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 3, 0) = 0x7ef86a46f000

[pid 30518] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=1, tv\_nsec=0}, test.0x7ffe5af1fe60) = 0

[pid 30518] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=1, tv\_nsec=0}, txt

<unfinished ...>

[pid 30517] <... read resumed>"test.txt\n", 1024) = 9

[pid 30517] write(1, "Enter a message (exit for exit)\n", 32Enter a message (exit for exit)

) = 32

[pid 30517] read(0, <unfinished ...>

[pid 30518] <... clock\_nanosleep resumed>0x7ffe5af1fe60) = 0

[pid 30518] getrandom("\xc5\x10\x72\x31\x59\xd1\x69\x43", 8, GRND\_NONBLOCK) = 8

[pid 30518] brk(NULL) = 0x5b3b8209b000

[pid 30518] brk(0x5b3b820bc000) = 0x5b3b820bc000

[pid 30518] openat(AT\_FDCWD, "test.txt", O\_WRONLY|O\_CREAT|O\_TRUNC, 0666) = 4

[pid 30518] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=1, tv\_nsec=0}, 0x7ffe5af1fe60) = 0

[pid 30518] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=1, tv\_nsec=0}, 0x7ffe5af1fe60) = 0

[pid 30518] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=1, tv\_nsec=0}, hel0x7ffe5af1fe60) = 0

[pid 30518] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=1, tv\_nsec=0}, low0x7ffe5af1fe60) = 0

[pid 30518] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=1, tv\_nsec=0}, ;

<unfinished ...>

[pid 30517] <... read resumed>"hellow;\n", 1024) = 8

[pid 30517] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=0, tv\_nsec=100000000}, NULL) = 0

[pid 30517] write(1, "Status: Not over in ';' or '.'\n", 31Status: Not over in ';' or '.'

) = 31

[pid 30517] write(1, "Enter a message (exit for exit)\n", 32Enter a message (exit for exit)

) = 32

[pid 30517] read(0, <unfinished ...>

[pid 30518] <... clock\_nanosleep resumed>0x7ffe5af1fe60) = 0

[pid 30518] fstat(4, {st\_mode=S\_IFREG|0644, st\_size=0, ...}) = 0

[pid 30518] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=1, tv\_nsec=0}, 0x7ffe5af1fe60) = 0

[pid 30518] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=1, tv\_nsec=0}, 0x7ffe5af1fe60) = 0

[pid 30518] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=1, tv\_nsec=0}, yo0x7ffe5af1fe60) = 0

[pid 30518] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=1, tv\_nsec=0}, oo

<unfinished ...>

[pid 30517] <... read resumed>"yooo\n", 1024) = 5

[pid 30517] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=0, tv\_nsec=100000000}, NULL) = 0

[pid 30517] write(1, "Status: Success\n", 16Status: Success

) = 16

[pid 30517] write(1, "Enter a message (exit for exit)\n", 32Enter a message (exit for exit)

) = 32

[pid 30517] read(0, <unfinished ...>

[pid 30518] <... clock\_nanosleep resumed>0x7ffe5af1fe60) = 0

[pid 30518] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=1, tv\_nsec=0}, l0x7ffe5af1fe60) = 0

[pid 30518] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=1, tv\_nsec=0},0x7ffe5af1fe60) = 0

[pid 30518] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=1, tv\_nsec=0}, 0x7ffe5af1fe60) = 0

[pid 30518] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=1, tv\_nsec=0}, n0x7ffe5af1fe60) = 0

[pid 30518] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=1, tv\_nsec=0}, aaaa;0x7ffe5af1fe60) = 0

[pid 30518] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=1, tv\_nsec=0},

<unfinished ...>

[pid 30517] <... read resumed>"naaaa;\n", 1024) = 7

[pid 30517] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=0, tv\_nsec=100000000}, NULL) = 0

[pid 30517] write(1, "Status: Not over in ';' or '.'\n", 31Status: Not over in ';' or '.'

) = 31

[pid 30517] write(1, "Enter a message (exit for exit)\n", 32Enter a message (exit for exit)

) = 32

[pid 30517] read(0, e <unfinished ...>

[pid 30518] <... clock\_nanosleep resumed>0x7ffe5af1fe60) = 0

[pid 30518] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=1, tv\_nsec=0}, xit0x7ffe5af1fe60) = 0

[pid 30518] clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=1, tv\_nsec=0},

<unfinished ...>

[pid 30517] <... read resumed>"exit\n", 1024) = 5

[pid 30517] wait4(30518, <unfinished ...>

[pid 30518] <... clock\_nanosleep resumed>0x7ffe5af1fe60) = 0

[pid 30518] write(4, "hellow;\nhellow;\nhellow;\nhellow;\n"..., 46) = 46

[pid 30518] close(4) = 0

[pid 30518] exit\_group(0) = ?

[pid 30518] +++ exited with 0 +++

<... wait4 resumed>NULL, 0, NULL) = 30518

--- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=30518, si\_uid=1000, si\_status=0, si\_utime=0, si\_stime=0} ---

unlink("/dev/shm/shared\_memory") = 0

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

Программа успешно демонстрирует использование разделяемой памяти для межпроцессного взаимодействия, позволяя родительскому и дочернему процессам обмениваться данными. Родительский процесс отправляет имя файла и сообщения дочернему процессу, который проверяет их и записывает в файл, если они соответствуют заданным условиям. Родительский процесс также получает статус выполнения от дочернего процесса, что позволяет ему отслеживать успешность операций.