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: Верменников М.В.

Преподаватель: Миронов Е.С.

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

Дата: 04.10.24

Москва, 2023

**Постановка задачи**

**Вариант 4.**

Пользователь вводит команды вида: «число число число<endline>». Родительский процесс считывает эти числа, а затем порождает дочерний процесс. Далее между процессами осуществляется обмен данными не через каналы (pipe), а с использованием технологии отображаемых в память файлов (memory-mapped files). Родительский процесс записывает данные (количество чисел и сами числа) в совместно отобразимую область памяти. Дочерний процесс, получив доступ к тем же данным, выполняет деление первого числа на последующие. Результат деления (или сигнал о возникшей ошибке деления на ноль) также помещается в эту область памяти. Если хотя бы одно из последующих чисел равно нулю, дочерний процесс устанавливает флаг ошибки, и оба процесса завершают свою работу. В противном случае результаты деления доступны родительскому процессу, который выводит их на экран. Числа имеют тип float. Количество чисел может быть произвольным.

**Общий метод и алгоритм решения**

Использованные системные вызовы:

* **pid\_t fork(void);** — создает дочерний процесс.
* **int execl(const char \*path, const char \*arg, ...);** — загружает в текущий процесс новый образ программы (дочерний процесс загружает программу для обработки чисел).
* **int open(const char \*pathname, int flags, mode\_t mode);** — открытие или создание файла, который в дальнейшем будет отображен в память.
* **int ftruncate(int fd, off\_t length);** — изменение размера файла. Используется для задания нужного объема под общую память.
* **void \*mmap(void \*addr, size\_t length, int prot, int flags, int fd, off\_t offset);** — отображение файла в память. Основной механизм межпроцессного взаимодействия в данной работе.
* **int munmap(void \*addr, size\_t length);** — отмена отображения файла в память по окончании работы.
* **int waitpid(pid\_t pid, int \*status, int options);** — ожидание завершения дочернего процесса и получение кода завершения.
* **ssize\_t read(int fd, void \*buf, size\_t count);, ssize\_t write(int fd, const void \*buf, size\_t count);** — используются в дочернем процессе для возможной записи результатов в файл-лог или чтения данных при необходимости.

**Алгоритм работы программы:**

1. **Инициализация в родительском процессе**:
   1. Считывание введенных пользователем чисел.
   2. Создание или открытие файла, предназначенного для общей памяти.
   3. Установка нужного размера файла с помощью ftruncate().
   4. Отображение файла в память (mmap()) и запись туда введенных чисел и вспомогательных полей (количество, флаг ошибки).
2. **Создание дочернего процесса**:
   1. Вызов fork() порождает дочерний процесс.
   2. Родительский процесс переходит в режим ожидания завершения дочернего, а дочерний процесс выполняет свои действия.
3. **Дочерний процесс**:
   1. Открывает тот же файл, что и родитель.
   2. Отображает файл в память (mmap()), получая доступ к тем же данным.
   3. Читает из памяти входные данные, выполняет деление первого числа на остальные. При попытке деления на 0 устанавливает флаг ошибки. Иначе записывает результат последнего деления обратно в разделяемую память.
   4. Завершает работу, снимая отображение (munmap).
4. **Родительский процесс после завершения дочернего**:
   1. waitpid() дожидается завершения дочернего процесса.
   2. Проверяет флаг ошибки в общей памяти. Если установлен, выводит сообщение об ошибке.
   3. Если ошибки нет, читает результат и выводит его на экран.
   4. Освобождает ресурсы (munmap), может удалить временный файл.
5. **Завершение работы**:
   1. Родительский процесс завершает свое выполнение после успешного чтения результата или при обнаружении ошибки деления на ноль.

**Код программы**

**Main.cpp**

#include<iostream>

#include<vector>

#include<unistd.h>

#include<sys/wait.h>

#include<sys/mman.h>

#include<sys/stat.h>

#include<fcntl.h>

#include<cstring>

#include<cstdlib>

structSharedData{

intlen; // Кол-во чисел

interror\_flag;// Флаг ошибки: 0 - нет ошибки, 1 - деление на ноль

// Далее идут числа и результат. Структура памяти:

// [int len][int error\_flag][float numbers[len]][float results[len-1]]

// однако мы не можем заранее задать массив переменной длины в структуре,

// поэтому будем обращаться к ним по смещению

};

intmain() {

std::cout<<"Введите числа в формате: «число число число<endline>»."<<std::endl;

std::vector<float>numbers;

floatnumber;

while(std::cin>>number) {

numbers.push\_back(number);

if(std::cin.peek() =='\n') break;

}

intlen=(int)numbers.size();

if(len<2) {

std::cerr<<"Нужно как минимум 2 числа."<<std::endl;

return1;

}

// Рассчитаем размер, который нам нужен под mmap:

// SharedData + numbers[len] + results[len-1]

// Размер: sizeof(SharedData) + len\*sizeof(float) + (len-1)\*sizeof(float)

size\_tsize=sizeof(SharedData) +len\*sizeof(float) +(len-1) \*sizeof(float);

constchar\*filename="shared.bin";

intfd=open(filename, O\_RDWR|O\_CREAT|O\_TRUNC, 0666);

if(fd==-1) {

std::cerr<<"Ошибка при открытии файла для mmap"<<std::endl;

return1;

}

if(ftruncate(fd, size) ==-1) {

std::cerr<<"Ошибка при изменении размера файла"<<std::endl;

close(fd);

return1;

}

void\*addr=mmap(nullptr, size, PROT\_READ|PROT\_WRITE, MAP\_SHARED, fd, 0);

if(addr==MAP\_FAILED) {

std::cerr<<"Ошибка при mmap"<<std::endl;

close(fd);

return1;

}

close(fd);// Дескриптор файла больше не нужен, т.к. мы уже сделали mmap

// Запишем данные в общую память

SharedData\*shared=(SharedData\*)addr;

shared->len=len;

shared->error\_flag=0;

// Область в памяти после SharedData для чисел и результатов

float\*numbers\_ptr=(float\*)((char\*)addr+sizeof(SharedData));

float\*results\_ptr=numbers\_ptr+len;

// Записываем числа

memcpy(numbers\_ptr, numbers.data(), len\*sizeof(float));

// Создаем дочерний процесс

pid\_tpid=fork();

if(pid<0) {

std::cerr<<"Ошибка при создании процесса"<<std::endl;

munmap(addr, size);

return1;

}

if(pid>0) {

// Родительский процесс

intstatus;

waitpid(pid, &status, 0);

if(WIFEXITED(status)) {

if(shared->error\_flag==1) {

std::cerr<<"Деление на 0. Завершение работы."<<std::endl;

} else{

// Читаем результат

intres\_len=len-1;

std::vector<float>results(res\_len);

memcpy(results.data(), results\_ptr, res\_len\*sizeof(float));

std::cout<<"Результат деления: "<<results.back() <<std::endl;

}

} else{

std::cerr<<"Дочерний процесс завершился нештатно."<<std::endl;

}

munmap(addr, size);

} else{

// Дочерний процесс: запускаем child

execl("./child", "child", filename, (char\*)NULL);

std::cerr<<"Ошибка при вызове execl"<<std::endl;

munmap(addr, size);

return1;

}

return0;

}

**Child.cpp**

#include<unistd.h>

#include<vector>

#include<fcntl.h>

#include<sys/types.h>

#include<sys/stat.h>

#include<iostream>

#include<cstdio>

#include<sys/mman.h>

#include<cstring>

#include<cstdlib>

structSharedData{

intlen;

interror\_flag;

};

intmain(intargc, char\*argv[]) {

if(argc<2) {

std::cerr<<"Не передано имя файла"<<std::endl;

return1;

}

constchar\*filename=argv[1];

intfd=open(filename, O\_RDWR);

if(fd==-1) {

std::cerr<<"Ошибка при открытии файла"<<std::endl;

return1;

}

// Определим размер файла

structstatst;

if(fstat(fd, &st) ==-1) {

std::cerr<<"Ошибка fstat"<<std::endl;

close(fd);

return1;

}

size\_tsize=st.st\_size;

void\*addr=mmap(nullptr, size, PROT\_READ|PROT\_WRITE, MAP\_SHARED, fd, 0);

if(addr==MAP\_FAILED) {

std::cerr<<"Ошибка при mmap в дочернем процессе"<<std::endl;

close(fd);

return1;

}

close(fd);

SharedData\*shared=(SharedData\*)addr;

intlen=shared->len;

float\*numbers\_ptr=(float\*)((char\*)addr+sizeof(SharedData));

float\*results\_ptr=numbers\_ptr+len;

std::vector<float>numbers(len);

memcpy(numbers.data(), numbers\_ptr, len\*sizeof(float));

floatfirst\_number=numbers[0];

intres\_len=len-1;

std::vector<float>results(res\_len);

for(inti=1; i<len; ++i) {

if(numbers[i]==0) {

// Деление на ноль

shared->error\_flag=1;

munmap(addr, size);

exit(0);

}

results[i-1]=first\_number/numbers[i];

}

// Записываем результаты обратно

memcpy(results\_ptr, results.data(), res\_len\*sizeof(float));

munmap(addr, size);

return0;

}

**Протокол работы программы**

**Тестирование:**

$ ./main

Введите числа в формате: «число число число<endline>».

4 4 5 5 6

Результат деления: 0.666667

**Strace:**

1858 execve("./main", ["./main"], 0x7ffd697e62a8 /\* 26 vars \*/) = 0

1858 brk(NULL) = 0x26f04000

1858 mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f090aa78000

1858 access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

1858 openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

1858 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=25258, ...}, AT\_EMPTY\_PATH) = 0

1858 mmap(NULL, 25258, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f090aa71000

1858 close(3) = 0

1858 openat(AT\_FDCWD, "/usr/local/lib64/libstdc++.so.6", O\_RDONLY|O\_CLOEXEC) = 3

1858 read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

1858 newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2530008, ...}, AT\_EMPTY\_PATH) = 0

1858 mmap(NULL, 2543808, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f090a803000

1858 mmap(0x7f090a8a8000, 1216512, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xa5000) = 0x7f090a8a8000

1858 mmap(0x7f090a9d1000, 581632, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ce000) = 0x7f090a9d1000

1858 mmap(0x7f090aa5f000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x25c000) = 0x7f090aa5f000

1858 mmap(0x7f090aa6d000, 12480, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f090aa6d000

1858 close(3) = 0

1858 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libm.so.6", O\_RDONLY|O\_CLOEXEC) = 3

1858 read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

1858 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=907784, ...}, AT\_EMPTY\_PATH) = 0

1858 mmap(NULL, 909560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f090a724000

1858 mmap(0x7f090a734000, 471040, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x10000) = 0x7f090a734000

1858 mmap(0x7f090a7a7000, 368640, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x83000) = 0x7f090a7a7000

1858 mmap(0x7f090a801000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xdc000) = 0x7f090a801000

1858 close(3) = 0

1858 openat(AT\_FDCWD, "/usr/local/lib64/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC) = 3

1858 read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

1858 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=906528, ...}, AT\_EMPTY\_PATH) = 0

1858 mmap(NULL, 181160, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f090a6f7000

1858 mmap(0x7f090a6fb000, 143360, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4000) = 0x7f090a6fb000

1858 mmap(0x7f090a71e000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x27000) = 0x7f090a71e000

1858 mmap(0x7f090a722000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2b000) = 0x7f090a722000

1858 close(3) = 0

1858 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

1858 read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20t\2\0\0\0\0\0"..., 832) = 832

1858 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

1858 newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=1922136, ...}, AT\_EMPTY\_PATH) = 0

1858 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

1858 mmap(NULL, 1970000, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f090a516000

1858 mmap(0x7f090a53c000, 1396736, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x26000) = 0x7f090a53c000

1858 mmap(0x7f090a691000, 339968, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x17b000) = 0x7f090a691000

1858 mmap(0x7f090a6e4000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ce000) = 0x7f090a6e4000

1858 mmap(0x7f090a6ea000, 53072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f090a6ea000

1858 close(3) = 0

1858 mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f090a514000

1858 arch\_prctl(ARCH\_SET\_FS, 0x7f090a515480) = 0

1858 set\_tid\_address(0x7f090a515750) = 1858

1858 set\_robust\_list(0x7f090a515760, 24) = 0

1858 rseq(0x7f090a515da0, 0x20, 0, 0x53053053) = 0

1858 mprotect(0x7f090a6e4000, 16384, PROT\_READ) = 0

1858 mprotect(0x7f090a722000, 4096, PROT\_READ) = 0

1858 mprotect(0x7f090a801000, 4096, PROT\_READ) = 0

1858 mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f090a512000

1858 mprotect(0x7f090aa5f000, 45056, PROT\_READ) = 0

1858 mprotect(0x404000, 4096, PROT\_READ) = 0

1858 mprotect(0x7f090aaaa000, 8192, PROT\_READ) = 0

1858 prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

1858 munmap(0x7f090aa71000, 25258) = 0

1858 futex(0x7f090aa6d73c, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

1858 getrandom("\x87\xac\x5a\x2d\x29\x23\xde\x37", 8, GRND\_NONBLOCK) = 8

1858 brk(NULL) = 0x26f04000

1858 brk(0x26f25000) = 0x26f25000

1858 newfstatat(1, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}, AT\_EMPTY\_PATH) = 0

1858 write(1, "\320\222\320\262\320\265\320\264\320\270\321\202\320\265 \321\207\320\270\321\201\320\273\320\260 \320\262 \321\204\320"..., 92) = 92

1858 newfstatat(0, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0), ...}, AT\_EMPTY\_PATH) = 0

1858 read(0, "1 2 3\n", 1024) = 6

**1858 openat(AT\_FDCWD, "shared.bin", O\_RDWR|O\_CREAT|O\_TRUNC, 0666) = 3**

**1858 ftruncate(3, 28) = 0**

**1858 mmap(NULL, 28, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 3, 0) = 0x7f090aa77000**

**1858 close(3) = 0**

1858 clone(child\_stack=NULL, flags=CLONE\_CHILD\_CLEARTID|CLONE\_CHILD\_SETTID|SIGCHLD, child\_tidptr=0x7f090a515750) = 1888

1888 set\_robust\_list(0x7f090a515760, 24 <unfinished ...>

1858 wait4(1888, <unfinished ...>

1888 <... set\_robust\_list resumed>) = 0

1888 execve("./child", ["child", "shared.bin"], 0x7ffdcb3d2df8 /\* 26 vars \*/) = 0

1888 brk(NULL) = 0x2de47000

1888 mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f8ddf688000

1888 access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

1888 openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

1888 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=25258, ...}, AT\_EMPTY\_PATH) = 0

1888 mmap(NULL, 25258, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f8ddf681000

1888 close(3) = 0

1888 openat(AT\_FDCWD, "/usr/local/lib64/libstdc++.so.6", O\_RDONLY|O\_CLOEXEC) = 3

1888 read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

1888 newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2530008, ...}, AT\_EMPTY\_PATH) = 0

1888 mmap(NULL, 2543808, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f8ddf413000

1888 mmap(0x7f8ddf4b8000, 1216512, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xa5000) = 0x7f8ddf4b8000

1888 mmap(0x7f8ddf5e1000, 581632, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ce000) = 0x7f8ddf5e1000

1888 mmap(0x7f8ddf66f000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x25c000) = 0x7f8ddf66f000

1888 mmap(0x7f8ddf67d000, 12480, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f8ddf67d000

1888 close(3) = 0

1888 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libm.so.6", O\_RDONLY|O\_CLOEXEC) = 3

1888 read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

1888 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=907784, ...}, AT\_EMPTY\_PATH) = 0

1888 mmap(NULL, 909560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f8ddf334000

1888 mmap(0x7f8ddf344000, 471040, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x10000) = 0x7f8ddf344000

1888 mmap(0x7f8ddf3b7000, 368640, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x83000) = 0x7f8ddf3b7000

1888 mmap(0x7f8ddf411000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xdc000) = 0x7f8ddf411000

1888 close(3) = 0

1888 openat(AT\_FDCWD, "/usr/local/lib64/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC) = 3

1888 read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

1888 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=906528, ...}, AT\_EMPTY\_PATH) = 0

1888 mmap(NULL, 181160, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f8ddf307000

1888 mmap(0x7f8ddf30b000, 143360, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4000) = 0x7f8ddf30b000

1888 mmap(0x7f8ddf32e000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x27000) = 0x7f8ddf32e000

1888 mmap(0x7f8ddf332000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2b000) = 0x7f8ddf332000

1888 close(3) = 0

1888 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

1888 read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20t\2\0\0\0\0\0"..., 832) = 832

1888 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

1888 newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=1922136, ...}, AT\_EMPTY\_PATH) = 0

1888 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

1888 mmap(NULL, 1970000, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f8ddf126000

1888 mmap(0x7f8ddf14c000, 1396736, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x26000) = 0x7f8ddf14c000

1888 mmap(0x7f8ddf2a1000, 339968, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x17b000) = 0x7f8ddf2a1000

1888 mmap(0x7f8ddf2f4000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ce000) = 0x7f8ddf2f4000

1888 mmap(0x7f8ddf2fa000, 53072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f8ddf2fa000

1888 close(3) = 0

1888 mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f8ddf124000

1888 arch\_prctl(ARCH\_SET\_FS, 0x7f8ddf125480) = 0

1888 set\_tid\_address(0x7f8ddf125750) = 1888

1888 set\_robust\_list(0x7f8ddf125760, 24) = 0

1888 rseq(0x7f8ddf125da0, 0x20, 0, 0x53053053) = 0

1888 mprotect(0x7f8ddf2f4000, 16384, PROT\_READ) = 0

1888 mprotect(0x7f8ddf332000, 4096, PROT\_READ) = 0

1888 mprotect(0x7f8ddf411000, 4096, PROT\_READ) = 0

1888 mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f8ddf122000

1888 mprotect(0x7f8ddf66f000, 45056, PROT\_READ) = 0

1888 mprotect(0x403000, 4096, PROT\_READ) = 0

1888 mprotect(0x7f8ddf6ba000, 8192, PROT\_READ) = 0

1888 prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

1888 munmap(0x7f8ddf681000, 25258) = 0

1888 futex(0x7f8ddf67d73c, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

1888 getrandom("\xdd\xcb\x8a\xca\x7d\x8a\x5c\xfe", 8, GRND\_NONBLOCK) = 8

1888 brk(NULL) = 0x2de47000

1888 brk(0x2de68000) = 0x2de68000

**1888 openat(AT\_FDCWD, "shared.bin", O\_RDWR) = 3**

1888 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=28, ...}, AT\_EMPTY\_PATH) = 0

**1888 mmap(NULL, 28, PROT\_READ|PROT\_WRITE, MAP\_SHARED, 3, 0) = 0x7f8ddf687000**

**1888 close(3) = 0**

**1888 munmap(0x7f8ddf687000, 28) = 0**

1888 exit\_group(0) = ?

1888 +++ exited with 0 +++

1858 <... wait4 resumed>[{WIFEXITED(s) && WEXITSTATUS(s) == 0}], 0, NULL) = 1888

1858 --- SIGCHLD {si\_signo=SIGCHLD, si\_code=CLD\_EXITED, si\_pid=1888, si\_uid=0, si\_status=0, si\_utime=0, si\_stime=1 /\* 0.01 s \*/} ---

1858 write(1, "\320\240\320\265\320\267\321\203\320\273\321\214\321\202\320\260\321\202 \320\264\320\265\320\273\320\265\320\275\320\270\321"..., 44) = 44

**1858 munmap(0x7f090aa77000, 28) = 0**

1858 lseek(0, -1, SEEK\_CUR) = -1 ESPIPE (Illegal seek)

1858 exit\_group(0) = ?

1858 +++ exited with 0 +++

**Вывод**

В результате выполнения лабораторной работы №3 я познакомился с новым способом организации межпроцессного взаимодействия — использованием отображаемых файлов (memory-mapped files). В отличие от традиционных методов передачи данных, таких как каналы (pipe) или сокеты, memory mapping позволяет процессам совместно использовать одну и ту же область памяти, отразив файл в своё виртуальное адресное пространство.