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: М80-206Б-22

Студент: Свиридов С. Ю.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 17.11.23

Москва, 2023

Постановка задачи

Группа вариантов 2.

Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия файла с таким именем на чтение. Стандартный поток ввода дочернего процесса переопределяется открытым файлом. Дочерний процесс читает команды из стандартного потока ввода. Родительский процесс выводит результат в свой стандартный поток вывода. Родительский и дочерний процесс должны быть представлены разными программами. Взаимодействие между процессами осуществляется через системные сигналы/события и/или через отображаемые файлы (memory-mapped files).

Вариант 9.

В файле записаны команды вида:«число число число<endline>». Дочерний процесс производит деление первого числа команда, на последующие числа в команде, а результат “кладет” в map-file. Если происходит деление на 0, то дочерний (и родительский) процесс завершают свою работу с ошибкой. Проверка деления на 0 должна осуществляться на стороне дочернего процесса. 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 **arg0*, ... /*, (char *)0 */); - заменяет текущий образ процесса новым образом
- pid_t wait(int *status); – приостаналивает выполнение текущего процесса до тех пор, пока дочерний процесс не завершится.
- void* mmap(void *, size_t, int, int, int, off_t) - выделяет память или отображает файлы (или устройства) в памяти.
- int munmap(void*, size_t) - удаляет сопоставление с выделенной памятью.
- int ftruncate(int, off_t) - приводит файл к заданному размеру.
- int shm_open(const char *, int, ...) - инициализирует область памяти
- int shm_unlink(const char* name) — разрывает связь между областью памяти и заданным ей именем

Сначала пользователь в качестве аргумента командной строки пишет имя файла, которое будет использоваться для открытия файла с таким же именем на чтение. Если строка введена корректно, и файл с таким именем существует, то создается дочерний процесс. После чего для дочернего процесса подменятся стандартный ввод, которым теперь является открытый файл. В дочернем процессе инициализируется область памяти именем «laba3» и туда записываются обработанные им данные. После завершения дочернего процесса родительский процесс читает данные из этой же области памяти и печатает их в терминал. В случае, если родительский процесс прочитает значение -1, на экран будет выведено сообщение «Attempt to divide by zero» и выполнение программы завершится ошибкой.

Код программы

parent.c

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include "stddef.h"
#include <fcntl.h>
#include <string.h>
#include <stdbool.h>
#include <semaphore.h>

#define MEMORY_NAME "laba3"
#define DATA_SIZE 256
#define MEMORY_SIZE 8192

void check_error(bool expression, char* message) {
    if (expression) {
        write(STDOUT_FILENO, message, strlen(message) * sizeof(char));
        write(STDOUT_FILENO, "\n", 1);
        exit(-1);
    }
}

typedef struct {
    size_t size;
    float data[DATA_SIZE];
} res;

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

```
    pid_t pid;
    FILE *fp = NULL;
    if (argc != 2) {
        write(1, "Wrong arguments\n", 17);
        exit(EXIT_FAILURE);
    }
    pid = fork();
    if (pid == -1) {
        perror("fork");
        return -1;
    }
    else if (pid == 0) {
        fp = freopen(argv[1], "r", stdin);
        check_error(fp == NULL, "Can't open file");
        execl("./child", "/.child", NULL);
        perror("execl");
        return 1;
    }
    else {
        wait(0);
        int fd = shm_open(MEMORY_NAME, O_RDONLY, S_IRUSR | S_IWUSR);
        check_error(fd == -1, "Can't oped shared memory file");
        res *addr = mmap(NULL, MEMORY_SIZE, PROT_READ, MAP_SHARED, fd, 0);
        check_error(addr == (void*) -1, "Mmap error");
        for (int i = 0; i < addr->size; i++) {
            if (addr->data[i] == -1) {
                printf("Attempt to divide by zero\n");
                break;
            }
            printf("%f\n", addr->data[i]);
        }
        munmap(addr, MEMORY_SIZE);
        shm_unlink(MEMORY_NAME);
        close(fd);
    }
    return 0;
}
```

child.c

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
```

```
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include "stddef.h"
#include <fcntl.h>
#include <string.h>
#include <stdbool.h>
#include <semaphore.h>
#include <math.h>

#define MEMORY_NAME "laba3"
#define DATA_SIZE 256
#define MEMORY_SIZE 8192

void check_error(bool expression, char* message) {
    if (expression) {
        write(STDOUT_FILENO, message, strlen(message) * sizeof(char));
        write(STDOUT_FILENO, "\n", 1);
        exit(-1);
    }
}

typedef struct {
    size_t size;
    float data[DATA_SIZE];
} res;

int main() {
    int fd = shm_open(MEMORY_NAME, O_EXCL | O_CREAT | O_RDWR, S_IRUSR |
S_IWUSR);
    check_error(fd == -1, "Can't open shared memory file");
    if (ftruncate(fd, MEMORY_SIZE) == -1) {
        perror("ftruncate");
    }
    res *addr = mmap(NULL, MEMORY_SIZE, PROT_WRITE, MAP_SHARED, fd, 0);
    check_error(addr == (void*)-1, "Mmap error");
    addr->size = 0;

    char c;
    bool not_end = true;
    float nmbr = 0;
    float result = 0;
    float first = 0;
```

```
int k = 0;
float dot = 0;
int count = 0;
float numbers[100];

do {
    if (not_end) {
        if (c <= '9' && c >= '0') {
            if (nmbr != 0 && (floor(nmbr) != nmbr)) {
                if (c == '0') {
                    k += 1;
                    nmbr += 0;
                }
                else {
                    dot = c - '0';
                    nmbr = (nmbr + 0.1) + (dot / pow(10, k + 1));
                }
            }
            else {
                nmbr = nmbr * 10 + c - '0';
            }
        }
        if (c == '.') {
            nmbr = nmbr - 0.1;
        }
        if (c == ' ' || c == '\n' || c == EOF) {
            numbers[count] = nmbr;
            nmbr = 0;
            count++;
            if (c == '\n' || c == EOF) {
                first = numbers[0];
                for (int i = 1; i < count; i++) {
                    if (numbers[i] == 0) {
                        addr->data[addr->size++] = -1;
                    }
                    result = first / numbers[i];
                    first = result;
                }
                first = 0;
                count = 0;
            }
        }
    }
    if (c == '\n' || c == EOF) {
```

```
            addr->data[addr->size++] = result;
            result = 0;
            k = 0;
            dot = 0;
            not_end = true;
        }
    } while((scanf("%c", &c)) > 0);

    return 0;
}
```

Протокол работы программы

Тестирование:

```
stepan@stepan-ASUS:~/Рабочий стол/учеба/prog 3 sem/OSI/laba3/src$ gcc -o
parent parent.c

stepan@stepan-ASUS:~/Рабочий стол/учеба/prog 3 sem/OSI/laba3/src$ gcc -o child
child.c -lm

stepan@stepan-ASUS:~/Рабочий стол/учеба/prog 3 sem/OSI/laba3/src$ ./parent
file.txt

     900.000122

     1.000000

     14.273480

     0.666667

     90.000000

     38.136364

     4329.239258

     2880.000000

     Attempt to divide by zero
```

Strace:

stepan@stepan-ASUS:~/Рабочий стол/учеба/labs 3 sem/OSI/laba3/src$ strace -f ./parent file.txt

execve("./parent", ["./parent", "file.txt"], 0x7ffdcfaa45f0 /* 60 vars */) = 0
brk(NULL) = 0x55dad5123000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffedcb8eb40) = -1 EINVAL (Недопустимый аргумент)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fbe162ff000
access("/etc/ld.so.preload", R_OK) = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=88411, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 88411, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fbe162e9000
close(3) = 0
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\0P\237\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
pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48

pread64(3,
"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\244;\374\204(\337f#\315I\214\234\f\256\271\32"..., 68,
896) = 68
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2216304, ...}, AT_EMPTY_PATH) = 0
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, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0x7fbe16000000
mmap(0x7fbe16028000, 1658880, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7fbe16028000
mmap(0x7fbe161bd000, 360448, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7fbe161bd000
mmap(0x7fbe16215000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x214000) = 0x7fbe16215000
mmap(0x7fbe1621b000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fbe1621b000
close(3) = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fbe162e6000
arch_prctl(ARCH_SET_FS, 0x7fbe162e6740) = 0
set_tid_address(0x7fbe162e6a10) = 8428
set_robust_list(0x7fbe162e6a20, 24) = 0
rseq(0x7fbe162e70e0, 0x20, 0, 0x53053053) = 0
mprotect(0x7fbe16215000, 16384, PROT_READ) = 0
mprotect(0x55dad3ba6000, 4096, PROT_READ) = 0
mprotect(0x7fbe16339000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7fbe162e9000, 88411) = 0
clone(child_stack=NULL,
flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace:
Process 8429 attached
, child_tidptr=0x7fbe162e6a10) = 8429
[pid 8428] wait4(-1, <unfinished ...>
[pid 8429] set_robust_list(0x7fbe162e6a20, 24) = 0
[pid 8429] openat(AT_FDCWD, "file.txt", O_RDONLY) = 3
[pid 8429] dup3(3, 0, 0) = 0
[pid 8429] close(3) = 0
[pid 8429] execve("./child", ["/.child"], 0x7ffedcb8ed20 /* 60 vars */) = 0
[pid 8429] brk(NULL) = 0x55a65f328000
[pid 8429] arch_prctl(0x3001 /* ARCH_??? */, 0x7fff56e715b0) = -1 EINVAL
(Недопустимый аргумент)
[pid 8429] mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fdf6d2f1000

[pid 8429] access("/etc/ld.so.preload", R_OK) = -1 ENOENT (Нет такого файла или каталога)
[pid 8429] openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
[pid 8429] newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=88411, ...}, AT_EMPTY_PATH) = 0
[pid 8429] mmap(NULL, 88411, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fdf6d2db000
[pid 8429] close(3) = 0
[pid 8429] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY|O_CLOEXEC) = 3
[pid 8429] 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"..., 832) = 832
[pid 8429] newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=940560, ...}, AT_EMPTY_PATH) = 0
[pid 8429] mmap(NULL, 942344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fdf6d1f4000
[pid 8429] mmap(0x7fdf6d202000, 507904, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe000) = 0x7fdf6d202000
[pid 8429] mmap(0x7fdf6d27e000, 372736, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8a000) = 0x7fdf6d27e000
[pid 8429] mmap(0x7fdf6d2d9000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe4000) = 0x7fdf6d2d9000
[pid 8429] close(3) = 0
[pid 8429] openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
[pid 8429] read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832) = 832
[pid 8429] 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 8429] pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48
[pid 8429] pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\244;\374\204(\337f#\315I\214\234\f\256\271\32"..., 68, 896) = 68
[pid 8429] newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2216304, ...}, AT_EMPTY_PATH) = 0
[pid 8429] 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 8429] mmap(NULL, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fdf6ce00000
[pid 8429] mmap(0x7fdf6ce28000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7fdf6ce28000
[pid 8429] mmap(0x7fdf6cfbd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7fdf6cfbd000

[pid 8429] mmap(0x7fdf6d015000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x214000) = 0x7fdf6d015000
[pid 8429] mmap(0x7fdf6d01b000, 52816, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fdf6d01b000
[pid 8429] close(3) = 0
[pid 8429] mmap(NULL, 12288, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fdf6d1f1000
[pid 8429] arch_prctl(ARCH_SET_FS, 0x7fdf6d1f1740) = 0
[pid 8429] set_tid_address(0x7fdf6d1f1a10) = 8429
[pid 8429] set_robust_list(0x7fdf6d1f1a20, 24) = 0
[pid 8429] rseq(0x7fdf6d1f20e0, 0x20, 0, 0x53053053) = 0
[pid 8429] mprotect(0x7fdf6d015000, 16384, PROT_READ) = 0
[pid 8429] mprotect(0x7fdf6d2d9000, 4096, PROT_READ) = 0
[pid 8429] mprotect(0x55a65e375000, 4096, PROT_READ) = 0
[pid 8429] mprotect(0x7fdf6d32b000, 8192, PROT_READ) = 0
[pid 8429] prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0
[pid 8429] munmap(0x7fdf6d2db000, 88411) = 0
[pid 8429] **openat(AT_FDCWD, "/dev/shm/laba3",
O_RDWR|O_CREAT|O_EXCL|O_NOFOLLOW|O_CLOEXEC, 0600) = 3**
[pid 8429] ftruncate(3, 8192) = 0
[pid 8429] **mmap(NULL, 8192, PROT_WRITE, MAP_SHARED, 3, 0) =
0x7fdf6d2ef000**
[pid 8429] newfstatat(0, "", {st_mode=S_IFREG|0664, st_size=85, ...},
AT_EMPTY_PATH) = 0
[pid 8429] getrandom("\xbc\x69\xf0\x43\xe1\x05\xf1\xc1", 8, GRND_NONBLOCK) = 8
[pid 8429] brk(NULL) = 0x55a65f328000
[pid 8429] brk(0x55a65f349000) = 0x55a65f349000
[pid 8429] read(0, "9 00.01 1\n10 2.5 4\n1000 7.006 2 "..., 4096) = 85
[pid 8429] read(0, "", 4096) = 0
[pid 8429] exit_group(0) = ?
[pid 8429] +++ exited with 0 +++
<... wait4 resumed>NULL, 0, NULL) = 8429
--- SIGCHLD {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=8429, si_uid=1000,
si_status=0, si_utime=0, si_stime=0} ---
**openat(AT_FDCWD, "/dev/shm/laba3", O_RDONLY|O_NOFOLLOW|O_CLOEXEC)
= 3**
mmap(NULL, 8192, PROT_READ, MAP_SHARED, 3, 0) = 0x7fbe162fd000
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...},
AT_EMPTY_PATH) = 0
getrandom("\x73\x26\xaa\x91\x46\x23\xf3\x2a", 8, GRND_NONBLOCK) = 8
brk(NULL) = 0x55dad5123000
brk(0x55dad5144000) = 0x55dad5144000
write(1, "900.000122\n", 11900.000122

```
)           = 11
write(1, "1.000000\n", 91.000000
)             = 9
write(1, "14.273480\n", 1014.273480
)           = 10
write(1, "0.666667\n", 90.666667
)             = 9
write(1, "90.000000\n", 1090.000000
)           = 10
write(1, "38.136364\n", 1038.136364
)           = 10
write(1, "4329.239258\n", 124329.239258
)          = 12
write(1, "2880.000000\n", 122880.000000
)          = 12
write(1, "Attempt to divide by zero\n", 26Attempt to divide by zero
) = 26
munmap(0x7fbe162fd000, 8192)          = 0
unlink("/dev/shm/laba3")              = 0
close(3)                              = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

Выполнив данную лабораторную работу, я понял, что существует хорошая альтернатива pipe. File mapping весьма интересная тема, но нужно более подробно изучать ее, потому что существует множество подводных камней и инетерсных вещей, который могут быть непонятными.