****Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работ №4 по курсу**

**«Операционные системы»**

Группа: М80 – 207Б – 18

Студент: Син Денис Дмитриевич

Преподаватель: Миронов Евгений Сергеевич

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

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2019.

**Содержание**

1. Постановка задачи
2. Общие сведения о программе
3. Общий метод и алгоритм решения
4. Основные файлы программы
5. Демонстрация работы программы
6. Вывод

**Постановка задачи**.

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 с помощью отображения файла на память.

Вариант 11: написать целочисленный калькулятор с операциями +, -. В дочернем процессе должны происходить вычисления выражений. В родительском процессе должны происходить ввод и вывод.

**Общие сведения о программе**

В программе изначально создается временный файл с помощью системного вызова mkstemp, заполняется до определенного размера, данный временный файл отображается в оперативную память, затем создается дочерний процесс. Родительский процесс принимает данные со стандартного потока ввода в формате , где – целые числа, а – арифметическая операция . Затем, если данные введены корректно родительский процесс передает данные дочернему процессу через общедоступный им участок памяти, который мы получили, отобразив временный файл в оперативную память. Дочерний процесс производит вычисление и передает результат обратно в родительский процесс, после чего родительский процесс печатает результат на стандартный поток вывода и снова ждет ввода пользователя. Для синхронизации процессов используется примитив синхронизации семафор.

Системные вызовы, использующиеся в программе:

1. **mkstemp –** создает временный файл.
2. **unlink –** удаляет ссылку на временный файл.
3. **write –** записывает данные в файл, файловый дескриптор которого указан.
4. **exit –** немедленно завершает работу программы.
5. **mmap –** отображает файл в оперативную память.
6. **munmap –** снимает отображение файла.
7. **sem\_open –** устанавливает связь между именованным семафором и процессом
8. **sem\_unlink –** удаляется семафор, но семафор разрушается, как только все остальные процессы, у которых семафор открыт закроют его.
9. **sem\_post –** увеличивает значение семафора.
10. **sem\_wait** – уменьшает значение семафора, если значение семафора равно 0, то блокирует поток.
11. **sem\_close –** освобождает системные ресурсы, выделенные системой для использования этим процессом данного семафора.
12. **close –** закрывает файл.

**Общий метод и алгоритм решения**.

Для решения поставленной задачи необходимо наладить взаимодействие между двумя процессами, а затем синхронизовать их действия.

Чтобы наладить межпроцессорное взаимодействие используется отображение файлов. После того как процесс отобразил файл в оперативную память, к этому участку памяти имеют доступ другие процессы. Для отображения создадим временный файл системным вызовом mkstemp. Чтобы файл был удален после завершения работы программы необходимо удалить ссылку на него системным вызовом unlink, также необходимо что-то записать в файл, чтобы он был непустой и отобразился. Затем отобразим данный файл с помощью системного вызова mmap. Таким образом, получим область памяти, по которой могут общаться несколько процессов.

Чтобы синхронизировать действия двух процессов воспользуемся семафорами. Один семафор нужен, для того-чтобы родительский процесс оповещал дочерний, когда данные уже готовы для вычисления, а второй семафор нужен, для того-чтобы дочерний процесс оповещал родительский, когда вычисления уже произведены и доступны для родительского процесса.

**Основные файлы программы**.

**Файл main.c**

*/\* Integer multiproccessing calculator \*/*

*/\* Using file mapping and semaphores \*/*

*/\* Author: Sin Denis \*/*

#include <stdlib.h>

#include <string.h>

#include <stdio.h>

#include <pthread.h>

#include <sys/mman.h>

#include <unistd.h>

#include <fcntl.h>

#include <semaphore.h>

#include <errno.h>

#include <signal.h>

typedef unsigned char uc\_t;

typedef struct {

    int lhs;

    int rhs;

    char op;

} expr\_t;

int expr\_read(expr\_t\* *expr*)

{

    return scanf("%d %c %d", &expr->lhs, &expr->op, &expr->rhs) == 3;

}

void expr\_write(expr\_t\* *expr*)

{

    printf("%d %c %d\n", expr->lhs, expr->op, expr->rhs);

}

int expr\_sum(expr\_t\* *expr*)

{

    return expr->lhs + expr->rhs;

}

int expr\_sub(expr\_t\* *expr*)

{

    return expr->lhs - expr->rhs;

}

void expr\_share(expr\_t\* *expr*, void\* *memory*, int *start*)

{

    int\* lhs = (int\*)((uc\_t\*)(memory) + start);

    int\* rhs = (int\*)((uc\_t\*)(memory) + start + 5);

    char\* op = (char\*)((uc\_t\*)(memory) + start + 4);

    \*lhs = expr->lhs;

    \*rhs = expr->rhs;

    \*op = expr->op;

}

void expr\_get\_from\_mem(expr\_t\* *expr*, void\* *memory*, int *pos*)

{

    int\* lhs = (int\*)((uc\_t \*)(memory) + pos);

    int\* rhs = (int\*)((uc\_t \*)(memory) + pos + 5);

    char\* op = (char\*)((uc\_t \*)(memory) + pos + 4);

    expr->lhs = \*lhs;

    expr->rhs = \*rhs;

    expr->op = \*op;

}

void expr\_write\_res(void\* *memory*, int *pos*, int *res*)

{

    int \*place = (int\*)((uc\_t\*)(memory) + pos);

    \*place = res;

}

int expr\_get\_res(void\* *memory*, int *pos*)

{

    int\* res = (int\*)((uc\_t\*)(memory) + pos);

    return \*res;

}

int get\_temp\_file(void)

{

    char \*tempFileName = strdup("/tmp/tmpFile.XXXXXX");

    int fd = mkstemp(tempFileName);

    unlink(tempFileName);

    free(tempFileName);

    write(fd, "\0\0\0\0\0\0\0\0\0\0\0\0\0", 13);

    return fd;

}

void\* get\_map\_memory(int *fd*, int *start*, int *size*)

{

    uc\_t\* mem = (uc\_t\*)mmap(NULL, size, PROT\_WRITE | PROT\_READ, MAP\_SHARED, fd, start);

    if (mem == MAP\_FAILED) {

        perror("Memmory mapping failed");

        exit(1);

    }

    return mem;

}

int main()

{

    int fd = get\_temp\_file();

    uc\_t\* mem = get\_map\_memory(fd, 0, 13);

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

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

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

        perror("Semaphores doesn't create");

        exit(1);

    }

    sem\_unlink("/calculus");

    sem\_unlink("/output");

    int err = fork();

    if (err == -1) {

        perror("Can't fork child");

        exit(1);

    } else if (err > 0) {

        expr\_t expr;

        int res;

        printf("> ");

        int read\_flag;

        while ((read\_flag = expr\_read(&expr))) {

            if (expr.op != '+' && expr.op != '-') {

                printf("This operation doesn't support\n> ");

                continue ;

            }

            expr\_share(&expr, mem, 0);

            sem\_post(sem\_calc);

            sem\_wait(sem\_output);

            res = expr\_get\_res(mem, 9);

            printf("%d\n> ", res);

        }

        mem[12] = 1;

        sem\_post(sem\_calc);

        sem\_close(sem\_calc);

        sem\_close(sem\_output);

        close(fd);

    } else {

        expr\_t expr;

        int res;

        while (1) {

            sem\_wait(sem\_calc);

            if (mem[12] == 1)

                break;

            expr\_get\_from\_mem(&expr, mem, 0);

            switch (expr.op) {

                case '+':

                    res = expr\_sum(&*expr*);

                    break;

                case '-':

                    res = expr\_sub(&*expr*);

                    break;

            }

            expr\_write\_res(mem, 9, res);

            sem\_post(sem\_output);

        }

        sem\_close(sem\_calc);

        sem\_close(sem\_output);

        munmap(mem, 13);

        close(fd);

    }

    return 0;

}

**Демонстрация работы программы.**

~/CSinDenis/university/sem3/OperatingSystems/lab04(master\*) » ./calc

> 90 + 12

102

> 5 + 5

10

> 0 - 1

-1

> 32 - 43

-11

> 873 + 23

896

> 0-9

-9

> 7888888888 - 12

-701045716

> 32 e 3

This operation doesn't support

> 23 \* 32

This operation doesn't support

> 12^ 4

This operation doesn't support

> %

**Вывод**

Отображение файлов дает удобство при работе с файлами, так как позволяет работать с областью файла как с обычным участком памяти. Другими словами, мы имеем доступ к каждому байту области памяти, которую мы отобразили и для этого не надо использовать lseek, также количество системных вызовов по чтению и записи сводится к нулю, так как мы работаем с оперативной памятью. Но также отображение файлов дает нам возможность в межпроцессорном взаимодействии. При отображении файла на участок памяти, этой память могут разделять несколько процессов, но в отличии от pipe, теперь синхронизация остается на разработчике.

28084/0xe0f92: thread\_selfid(0x0, 0x0, 0x0) = 921490 0

28084/0xe0f92: open(".\0", 0x0, 0x1) = 3 0

28084/0xe0f92: fstat64(0x3, 0x7FFEE6873DD0, 0x0) = 0 0

28084/0xe0f92: fcntl(0x3, 0x32, 0x7FFEE68744A0) = 0 0

28084/0xe0f92: close(0x3) = 0 0

28084/0xe0f92: stat64("/Users/sindenis/CSinDenis/university/sem3/OperatingSystems/lab04\0", 0x7FFEE6873D40, 0x0) = 0 0

28084/0xe0f92: \_\_mac\_syscall(0x115CBAC5F, 0x5A, 0x7FFEE6873FA0) = 0 0

28084/0xe0f92: shared\_region\_check\_np(0x7FFEE6873E08, 0x0, 0x0) = 0 0

28084/0xe0f92: stat64("/private/var/db/dyld/dyld\_shared\_cache\_x86\_64h\0", 0x7FFEE6873D50, 0x0) = 0 0

28084/0xe0f92: csrctl(0x0, 0x7FFEE6873B1C, 0x4) = -1 Err#1

28084/0xe0f92: stat64("/Users/sindenis/CSinDenis/university/sem3/OperatingSystems/lab04/./calc\0", 0x7FFEE68744A0, 0x0) = 0 0

28084/0xe0f92: getpid(0x0, 0x0, 0x0) = 28084 0

28084/0xe0f92: proc\_info(0x2, 0x6DB4, 0x16) = 1272 0

28084/0xe0f92: stat64("/usr/lib/libSystem.B.dylib\0", 0x7FFEE68731A0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libcache.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libcommonCrypto.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libcompiler\_rt.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libcopyfile.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libcorecrypto.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libdispatch.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libdyld.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libkeymgr.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/liblaunch.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libmacho.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libquarantine.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libremovefile.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libsystem\_asl.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libsystem\_blocks.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libsystem\_c.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libsystem\_configuration.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libsystem\_coreservices.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libsystem\_darwin.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libsystem\_dnssd.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libsystem\_info.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libsystem\_m.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libsystem\_malloc.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libsystem\_networkextension.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libsystem\_notify.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libsystem\_sandbox.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libsystem\_secinit.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libsystem\_kernel.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libsystem\_platform.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libsystem\_pthread.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libsystem\_symptoms.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libsystem\_trace.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libunwind.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/system/libxpc.dylib\0", 0x7FFEE6872DD0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/libobjc.A.dylib\0", 0x7FFEE68720D0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/libc++abi.dylib\0", 0x7FFEE68717B0, 0x0) = 0 0

28084/0xe0f92: stat64("/usr/lib/libc++.1.dylib\0", 0x7FFEE68717B0, 0x0) = 0 0

28084/0xe0f92: open("/dev/dtracehelper\0", 0x2, 0xFFFFFFFFE6873F30) = 3 0

28084/0xe0f92: ioctl(0x3, 0x80086804, 0x7FFEE6873D40) = 0 0

28084/0xe0f92: close(0x3) = 0 0

28084/0xe0f92: access("/AppleInternal/XBS/.isChrooted\0", 0x0, 0x0) = -1 Err#2

28084/0xe0f92: bsdthread\_register(0x7FFF692A1400, 0x7FFF692A13F0, 0x2000) = 1073742047 0

28084/0xe0f92: sysctlbyname(kern.bootargs, 0xD, 0x7FFEE6873190, 0x7FFEE6873180, 0x0) = 0 0

28084/0xe0f92: issetugid(0x0, 0x0, 0x0) = 0 0

28084/0xe0f92: ioctl(0x2, 0x4004667A, 0x7FFEE6872994) = 0 0

28084/0xe0f92: mprotect(0x109392000, 0x1000, 0x0) = 0 0

28084/0xe0f92: mprotect(0x109399000, 0x1000, 0x0) = 0 0

28084/0xe0f92: mprotect(0x10939A000, 0x1000, 0x0) = 0 0

28084/0xe0f92: mprotect(0x1093A1000, 0x1000, 0x0) = 0 0

28084/0xe0f92: mprotect(0x109390000, 0x90, 0x1) = 0 0

28084/0xe0f92: mprotect(0x1093A2000, 0x1000, 0x1) = 0 0

28084/0xe0f92: mprotect(0x109390000, 0x90, 0x3) = 0 0

28084/0xe0f92: mprotect(0x109390000, 0x90, 0x1) = 0 0

28084/0xe0f92: getentropy(0x7FFEE6872AE0, 0x20, 0x0) = 0 0

28084/0xe0f92: getpid(0x0, 0x0, 0x0) = 28084 0

28084/0xe0f92: stat64("/AppleInternal\0", 0x7FFEE6873600, 0x0) = -1 Err#2

28084/0xe0f92: csops(0x6DB4, 0x7, 0x7FFEE6873130) = -1 Err#22

28084/0xe0f92: proc\_info(0x2, 0x6DB4, 0xD) = 64 0

28084/0xe0f92: csops(0x6DB4, 0x7, 0x7FFEE6872970) = -1 Err#22

28084/0xe0f92: fstatat64(0xFFFFFFFFFFFFFFFE, 0x7FF2E0C00620, 0x7FFEE6874370) = 0 0

28084/0xe0f92: openat\_nocancel(0xFFFFFFFFFFFFFFFE, 0x7FF2E0C00620, 0xA02) = 3 0

28084/0xe0f92: unlink("/tmp/tmpFile.NrO6EB\0", 0x0, 0x0) = 0 0

dtrace: error on enabled probe ID 2172 (ID 170: syscall::write:return): invalid kernel access in action #13 at DIF offset 68

28084/0xe0f92: mmap(0x0, 0xD, 0x3, 0x1, 0x3, 0x0) = 0x1093A3000 0

28084/0xe0f92: sem\_open(0x10938BF42, 0x200, 0x309) = 4 0

28084/0xe0f92: sem\_open(0x10938BF4C, 0x200, 0x309) = 5 0

28084/0xe0f92: sem\_unlink(0x10938BF42, 0x0, 0x0) = 0 0

28084/0xe0f92: sem\_unlink(0x10938BF4C, 0x0, 0x0) = 0 0

28084/0xe0f92: fork() = 28085 0

28085/0xe0f96: thread\_selfid(0x0, 0x0, 0x0) = 921494 0

28085/0xe0f96: bsdthread\_register(0x7FFF692A1400, 0x7FFF692A13F0, 0x2000) = -1 Err#22

28084/0xe0f92: getrlimit(0x1008, 0x7FFEE68745C0, 0x0) = 0 0

28084/0xe0f92: fstat64(0x1, 0x7FFEE68745A8, 0x0) = 0 0

28084/0xe0f92: ioctl(0x1, 0x4004667A, 0x7FFEE68745F4) = 0 0

28084/0xe0f92: fstat64(0x0, 0x7FFEE68741D8, 0x0) = 0 0

28084/0xe0f92: ioctl(0x0, 0x4004667A, 0x7FFEE6874224) = 0 0

28085/0xe0f96: getpid(0x0, 0x0, 0x0) = 28085 0

28085/0xe0f96: \_\_mac\_syscall(0x7FFF692ADA17, 0x4, 0x7FFEE68743E8) = -1 Err#45

28085/0xe0f96: csops(0x6DB5, 0xB, 0x7FFEE6874428) = -1 Err#22

28084/0xe0f92: sem\_post(0x4, 0x0, 0x0) = 0 0

28085/0xe0f96: sem\_wait(0x4, 0x0, 0x0) = 0 0

28085/0xe0f96: sem\_post(0x5, 0x0, 0x0) = 0 0

28084/0xe0f92: sem\_wait(0x5, 0x0, 0x0) = 0 0

dtrace: error on enabled probe ID 2173 (ID 956: syscall::write\_nocancel:return): invalid kernel access in action #13 at DIF offset 68

dtrace: error on enabled probe ID 2173 (ID 956: syscall::write\_nocancel:return): invalid kernel access in action #13 at DIF offset 68

dtrace: error on enabled probe ID 2175 (ID 954: syscall::read\_nocancel:return): invalid kernel access in action #13 at DIF offset 68

28084/0xe0f92: sem\_post(0x4, 0x0, 0x0) = 0 0

28085/0xe0f96: sem\_wait(0x4, 0x0, 0x0) = 0 0

28085/0xe0f96: sem\_post(0x5, 0x0, 0x0) = 0 0

28084/0xe0f92: sem\_wait(0x5, 0x0, 0x0) = 0 0

dtrace: error on enabled probe ID 2173 (ID 956: syscall::write\_nocancel:return): invalid kernel access in action #13 at DIF offset 68

dtrace: error on enabled probe ID 2173 (ID 956: syscall::write\_nocancel:return): invalid kernel access in action #13 at DIF offset 68

dtrace: error on enabled probe ID 2175 (ID 954: syscall::read\_nocancel:return): invalid kernel access in action #13 at DIF offset 68

28084/0xe0f92: sem\_post(0x4, 0x0, 0x0) = 0 0

28085/0xe0f96: sem\_wait(0x4, 0x0, 0x0) = 0 0

28085/0xe0f96: sem\_post(0x5, 0x0, 0x0) = 0 0

28084/0xe0f92: sem\_wait(0x5, 0x0, 0x0) = 0 0

dtrace: error on enabled probe ID 2173 (ID 956: syscall::write\_nocancel:return): invalid kernel access in action #13 at DIF offset 68

dtrace: error on enabled probe ID 2173 (ID 956: syscall::write\_nocancel:return): invalid kernel access in action #13 at DIF offset 68

dtrace: error on enabled probe ID 2175 (ID 954: syscall::read\_nocancel:return): invalid kernel access in action #13 at DIF offset 68

28084/0xe0f92: sem\_post(0x4, 0x0, 0x0) = 0 0

28085/0xe0f96: sem\_wait(0x4, 0x0, 0x0) = 0 0

28085/0xe0f96: sem\_post(0x5, 0x0, 0x0) = 0 0

28084/0xe0f92: sem\_wait(0x5, 0x0, 0x0) = 0 0

28084/0xe0f92: sem\_post(0x4, 0x0, 0x0) = 0 0

28085/0xe0f96: sem\_wait(0x4, 0x0, 0x0) = 0 0

28085/0xe0f96: sem\_post(0x5, 0x0, 0x0) = 0 0

28084/0xe0f92: sem\_wait(0x5, 0x0, 0x0) = 0 0

28084/0xe0f92: sem\_post(0x4, 0x0, 0x0) = 0 0

28085/0xe0f96: sem\_wait(0x4, 0x0, 0x0) = 0 0

28085/0xe0f96: sem\_post(0x5, 0x0, 0x0) = 0 0

28084/0xe0f92: sem\_wait(0x5, 0x0, 0x0) = 0 0

28084/0xe0f92: sem\_post(0x4, 0x0, 0x0) = 0 0

28085/0xe0f96: sem\_wait(0x4, 0x0, 0x0) = 0 0

28085/0xe0f96: sem\_post(0x5, 0x0, 0x0) = 0 0

28084/0xe0f92: sem\_wait(0x5, 0x0, 0x0) = 0 0

28084/0xe0f92: sem\_post(0x4, 0x0, 0x0) = 0 0

28085/0xe0f96: sem\_wait(0x4, 0x0, 0x0) = 0 0

28085/0xe0f96: sem\_close(0x4, 0x0, 0x0) = 0 0

28084/0xe0f92: sem\_close(0x4, 0x0, 0x0) = 0 0

28084/0xe0f92: sem\_close(0x5, 0x0, 0x0) = 0 0

28085/0xe0f96: sem\_close(0x5, 0x0, 0x0) = 0 0

28084/0xe0f92: close(0x3) = 0 0

28085/0xe0f96: munmap(0x1093A3000, 0xD) = 0 0

28085/0xe0f96: close(0x3) = 0 0