Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу
«Операционные системы»

СРЕДСТВА ДИАГНОСТИКИ В ОС

Студент: Лукманова Аэлита Алимовна
Группа: М8О–301Б–19
Вариант: 4
Преподаватель: Миронов Е. С.
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2020.

Постановка задачи

Цель работы
Приобретение практических навыков диагностики работы программного обеспечения.

Задание
При выполнении последующих лабораторных работ необходимо продемонстрировать ключевые системные вызовы, которые в них используются и то, что их использование соответствует варианту ЛР.

По итогам выполнения всех лабораторных работ отчет по данной должен содержать краткую сводку по исследованию последующих ЛР.

Общие сведения о программе

Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса пишет имя файла, которое будет передано при создании дочернего процесса. Родительский и дочерний процесс должны быть представлены разными программами.

Пользователь вводит имена файлов в стандартный ввод в родительском процессе. Затем вводит строки. Четные и нечетные строки дочерний процесс отправляет в дочерний процесс, который в один файл записывает перевернутые полученный четные строки, в другой — нечетные.

Передача строк из родительского в дочерний происходит с помощью mmap, которая создает общую видимую для всех процессов область виртуальной памяти.

В программе используются следующие системные вызовы:

1. **mmap** — отражает *length* байтов, начиная со смещения *offset* файла (или другого объекта), определенного файловым дескриптором *fd*, в память, начиная с адреса *start*. При удачном выполнении mmap возвращает указатель на область с отраженными данными. При ошибке возвращается значение MAP_FAILED (-1), а переменная *errno* приобретает соответствующее значение.
2. **munmap** — удаляет все отражения из заданной области памяти, после чего все ссылки на данную область будут вызывать ошибку "неправильное обращение к памяти". При удачном выполнении munmap возвращаемое значение равно нулю. При ошибке возвращается -1, а переменная *errno* приобретает соответствующее значение.

3. **fork —** создает новый процесс, который является копией родительского процесса, за исключением разных process ID и parent process ID. В случае успеха fork() возвращает 0 для ребенка, число больше 0 для родителя – child ID, в случае ошибки возвращает -1.
4. **read —** предназначена для чтения какого-то числа байт из файла, принимает в качестве аргументов файловый дескриптор, буфер, в который будут записаны данные и число байт. В случае успеха вернет число прочитанных байт, иначе -1.
5. **write —** предназначена для записи какого-то числа байт в файл, принимает в качестве аргументов файловый дескриптор, буфер, из которого будут считаны данные для записи и число байт. В случае успеха вернет число записанных байт, иначе -1.

Общий метод и алгоритм решения.

Для реализации поставленной задачи необходимо:

1. Написать ЛР4.
2. Изучить работу утилиты strace.
3. Запустить strace с ключом -f и небольшим тестом.

Проанализируем вывод strace.

В родительском процессе мы отражаем разделяемую память для процесса-отца и процесса-ребенка для буфера данных размера 4096 байт и для буфера сигналов размера 4 байта. Адреса, по которым будут располагаться данные возвращаются из функций mmap.
`mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED|MAP_ANONYMOUS, -1, 0) = 0x7fc9a9d95000` `mmap(NULL, 4, PROT_READ|PROT_WRITE, MAP_SHARED|MAP_ANONYMOUS, -1, 0) = 0x7fc9a9d94000`
Системный вызов для создания дочернего процесса или потока.
`clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace: Process 5404 attached` `, child_tidptr=0x7fc9a9d84850) = 5404`
Открыть файл с именем “test_file_name” с правом на запись, если он существует, либо создать файл с правами на чтение и запись, если его нет. Данному файлу присвоен файловый дескриптор равный 3.

`[pid  5404] openat(AT_FDCWD, "test_file_name", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3`
Очистка отраженной разделяемой памяти по указанному адресу.

`[pid  5403] munmap(0x7fc9a9d95000, 4096 <unfinished ...>`
`[pid  5403] <... munmap resumed>)       = 0`
`[pid  5403] munmap(0x7fc9a9d94000, 4 <unfinished ...>`
`[pid  5403] <... munmap resumed>)       = 0`

Завершение работы процесса-отца и процесса-сына без ошибок.
`[pid  5403] exit_group(0)              = ?` `[pid  5404] exit_group(0)              = ?`
Загрузка и запуск исполняемой программы по указанному пути с передачей одного аргумента и переменных окружения.
`execve("./lab4", ["./lab4"], 0x7ffcb89522d8 /* 44 vars */) = 0`

Основные файлы программы

child.c:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <errno.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/mman.h>


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

    char * file_name = argv[1];

    int fd = open(file_name, O_RDWR | O_CREAT);
    if (fd < 0) {
        printf("can't open %s", file_name);
        return 0;
    }

    struct stat st;
    fstat(fd, &st);
    char *contents = mmap(NULL, st.st_size, PROT_READ | PROT_WRITE, MAP_SHARED, fd,
0);

    printf("\nHello, I'm %s\nI've recieved this contents of size %lld :\n%s", argv[0],
st.st_size, contents);

    int currentIndex = 0;
    char *enter = "\n";

    while (currentIndex < (int)st.st_size) {
        char tmpString[100] = {""};
        int ind = 0;
        while (contents[currentIndex] != *enter) {
            //printf("contents[%d]: %c\n", currentIndex, contents[currentIndex]);
            tmpString[ind] = contents[currentIndex];
            currentIndex++;
            ind++;
        }

        for (size_t i = currentIndex-ind; i < currentIndex; i++) {
            contents[i] = tmpString[currentIndex - i - 1];
        }
```

```
        currentIndex++;
    }

    printf("Then I reversed each string and rewrite them. This is result: \n%s\n",
contents);
    munmap(contents, st.st_size);
    close(fd);


    printf("%s say goodbye\n", argv[0]);
    return 0;
}
```

main.c:

```
#include <stdio.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <errno.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <signal.h>



#define NUMBER_OF_STRING 25
#define MAX_STRING_SIZE 200


//Вариант 21

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

    //MARK: -названия файлов на запись
    char firstChildFile[200];
    char secChildFile[200];
    printf("Введите имя файла на запись для первого дочернего процесса: ");
    fgets(firstChildFile, 200, stdin);
    firstChildFile[strlen(firstChildFile) - 1] = '\0';

    printf("Спасибо, введите имя файла на запись для второго дочернего процесса: ");
    fgets(secChildFile, 200, stdin);
    secChildFile[strlen(secChildFile) - 1] = '\0';


    //MARK: -инициализация
    int fd[2];
    int fd2[2];
    if (pipe(fd) == -1) {
        return 1;
    }
    if (pipe(fd2) == -1) {
        return 1;
    }

    pid_t pid_ch1 = fork();
    if (pid_ch1 == -1) {
        return 2;
    }


    if (pid_ch1 == 0) { //child1
        printf("child1");

        char *argv[10];
```

```
        argv[0] = "./child1";
        argv[1] = firstChildFile;
        argv[2] = NULL;
        wait(NULL);
        execv("./child", argv);
    }

    if (pid_ch1 > 0) { //parent

        pid_t pid_ch2 = fork();

        if (pid_ch2 == 0) { //child2
            printf("child2");

            char *argv[10];

            argv[0] = "./child2";
            argv[1] = secChildFile;
            argv[2] = NULL;
            wait(NULL);
            execv("./child", argv);

        } else {

            //MARK: -работа с pid
            printf("Hello, I'm a parent\n");

            //отсылаем детям сигналы стоп
            kill(pid_ch1, SIGSTOP);
            kill(pid_ch2, SIGSTOP);

            //заполняем два массива четными и нечентными строками
            char arrayOfString[50][200];
            printf("Input strings: \n");
            int arrayCount = 0;
            char line[200];
            while(fgets(line, 200, stdin)){
                strcpy(arrayOfString[arrayCount], line);
                arrayCount++;
            }

            char arrayOfString1[NUMBER_OF_STRING][MAX_STRING_SIZE];
            char arrayOfString2[NUMBER_OF_STRING][MAX_STRING_SIZE];
            char currentStr[200];

            int arrayCount1 = 0;
            int arrayCount2 = 0;
            for (int i = 0; i < arrayCount; i++) {
                strcpy(currentStr, arrayOfString[i]);
                if (i%2 == 0) {
                    strcpy(arrayOfString1[arrayCount1], currentStr);
                    arrayCount1++;
                } else {
                    strcpy(arrayOfString2[arrayCount2], currentStr);
                    arrayCount2++;
                }
            }


            //создадим отображение файлов детей в виртуальной памяти
            char * file_name_ch1 = firstChildFile;
            char * file_name_ch2 = secChildFile;

//            O_RDWR нужно писать даже если просто записываем, потому что
            int fd_ch1 = open(file_name_ch1, O_RDWR | O_CREAT | O_TRUNC);
            int fd_ch2 = open(file_name_ch2, O_RDWR | O_CREAT | O_TRUNC);
            if (fd_ch1 < 0) {
                printf("can't open %s for reading", file_name_ch1);
                return 0;
            }
            if (fd_ch2 < 0) {
```

```
                printf("can't open %s for reading", file_name_ch2);
                return 0;
            }

            //зададим размеры каждому отображению:
            size_t page_size_ch1 = sizeof(arrayOfString1);
            size_t page_size_ch2 = sizeof(arrayOfString2);

//            MAP_PRIVATE - изменения видны только моему процессу, то есть даже не
попадут в файл, не то что в дети. хотя главное в файл, это как раз то, что нам нужно
            char *contents_ch1 = mmap(NULL, page_size_ch1, PROT_READ | PROT_WRITE,
MAP_SHARED, fd_ch1, 0);
            char *contents_ch2 = mmap(NULL, page_size_ch2, PROT_READ | PROT_WRITE,
MAP_SHARED, fd_ch2, 0);

            if (contents_ch1 == MAP_FAILED){
                    close(fd_ch1);
                    perror("Error mmapping the file");
                    exit(EXIT_FAILURE);
            }
            if (contents_ch2 == MAP_FAILED){
                    close(fd_ch2);
                    perror("Error mmapping the file");
                    exit(EXIT_FAILURE);
            }


            //заполним отображения одного файла нечетными строками, другого файла --
четными
            for (int i = 0; i < arrayCount1; i++) {
                write(fd_ch1, arrayOfString1[i], strlen(arrayOfString1[i]));
            }
            for (int i = 0; i < arrayCount2; i++) {
                write(fd_ch2, arrayOfString2[i], strlen(arrayOfString2[i]));
            }

            //покажем результат
            printf("File contents_ch1:\n%s\n", contents_ch1);
            printf("File contents_ch2:\n%s\n", contents_ch2);

            // Write it now to disk
            if (msync(contents_ch1, page_size_ch1, MS_SYNC) == -1) {
                   perror("Could not sync the file to disk");
            }
            if (msync(contents_ch2, page_size_ch2, MS_SYNC) == -1) {
                   perror("Could not sync the file to disk");
            }

            //удалим отображения
            munmap(contents_ch1, page_size_ch1);
            munmap(contents_ch2, page_size_ch2);

            //закроем файлы
            close(fd_ch1);
            close(fd_ch2);

            //отсылаем детям сигнал продолжать выполнение их процессов
            kill(pid_ch1, SIGCONT);
            kill(pid_ch2, SIGCONT);

            waitpid(pid_ch2, NULL, 0);
            waitpid(pid_ch1, NULL, 0);

            printf("\nparent say goodbye\n");
        }
    }
}
```

Пример работы

```
aelitalukmanova@MacBook-Pro-Aelita os-4 % strace -f ./lab4 < test.txt
execve("./lab4", ["./lab4"], 0x7ffcb89522d8 /* 44 vars */) = 0
brk(NULL)                               = 0x564315f2f000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe1673b210) = -1 EINVAL (Недопустимый аргумент)
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
fstat(3, {st_mode=S_IFREG|0644, st_size=67778, ...}) = 0
mmap(NULL, 67778, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fc9a9d85000
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\0\360\215\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\20\0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0", 32,
848) = 32
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0~\303\347M\250B\312<j\233\242\v!0<\341"...,
68, 880) = 68
fstat(3, {st_mode=S_IFREG|0755, st_size=1995896, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fc9a9d83000
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, 2004064, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fc9a9b99000
mprotect(0x7fc9a9bbf000, 1810432, PROT_NONE) = 0
mmap(0x7fc9a9bbf000, 1495040, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENY-
WRITE, 3, 0x26000) = 0x7fc9a9bbf000
mmap(0x7fc9a9d2c000, 311296, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x193000) = 0x7fc9a9d2c000
mmap(0x7fc9a9d79000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1df000) = 0x7fc9a9d79000
mmap(0x7fc9a9d7f000, 13408, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7fc9a9d7f000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fc9a9b97000
arch_prctl(ARCH_SET_FS, 0x7fc9a9d84580) = 0
mprotect(0x7fc9a9d79000, 12288, PROT_READ) = 0
mprotect(0x564314b22000, 4096, PROT_READ) = 0
mprotect(0x7fc9a9dc4000, 4096, PROT_READ) = 0
munmap(0x7fc9a9d85000, 67778)           = 0
read(0, "t", 1)                         = 1
read(0, "e", 1)                         = 1
read(0, "s", 1)                         = 1
read(0, "t", 1)                         = 1
read(0, "_", 1)                         = 1
read(0, "f", 1)                         = 1
read(0, "i", 1)                         = 1
read(0, "l", 1)                         = 1
read(0, "e", 1)                         = 1
read(0, "_", 1)                         = 1
read(0, "n", 1)                         = 1
read(0, "a", 1)                         = 1
read(0, "m", 1)                         = 1
read(0, "e", 1)                         = 1
read(0, "\n", 1)                        = 1
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_SHARED|MAP_ANONYMOUS, -1, 0) =
0x7fc9a9d95000
mmap(NULL, 4, PROT_READ|PROT_WRITE, MAP_SHARED|MAP_ANONYMOUS, -1, 0) = 0x7fc9a9d94000
clone(child_stack=NULL, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLDstrace:
Process 5404 attached
, child_tidptr=0x7fc9a9d84850) = 5404
[pid  5403] read(0, "4", 1)             = 1
[pid  5403] read(0, "2", 1)             = 1
[pid  5403] read(0, "\n", 1)            = 1
[pid  5404] brk(NULL)                   = 0x564315f2f000
[pid  5404] brk(0x564315f50000)         = 0x564315f50000
[pid  5404] openat(AT_FDCWD, "test_file_name", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
[pid  5404] fstat(3, {st_mode=S_IFREG|0664, st_size=0, ...}) = 0
```

```
[pid  5403] read(0, "4", 1)             = 1
[pid  5403] read(0, " ", 1)             = 1
[pid  5403] read(0, "2", 1)             = 1
[pid  5403] read(0, " ", 1)             = 1
[pid  5403] read(0, "2", 1)             = 1
[pid  5403] read(0, "\n", 1)            = 1
[pid  5403] read(0, "3", 1)             = 1
[pid  5403] read(0, " ", 1)             = 1
[pid  5403] read(0, "1", 1)             = 1
[pid  5403] read(0, "\n", 1)            = 1
[pid  5403] read(0, "6", 1)             = 1
[pid  5403] read(0, "4", 1)             = 1
[pid  5403] read(0, " ", 1)             = 1
[pid  5403] read(0, "2", 1)             = 1
[pid  5403] read(0, " ", 1)             = 1
[pid  5403] read(0, "2", 1)             = 1
[pid  5403] read(0, " ", 1)             = 1
[pid  5403] read(0, "2", 1)             = 1
[pid  5403] read(0, " ", 1)             = 1
[pid  5403] read(0, "2", 1)             = 1
[pid  5403] read(0, "\n", 1)            = 1
[pid  5403] read(0, "", 1)              = 0
[pid  5403] munmap(0x7fc9a9d95000, 4096 <unfinished ...>
[pid  5404] write(3, "42.000000\n1.000000\n3.000000\n4.00"..., 37 <unfinished ...>
[pid  5403] <... munmap resumed>)       = 0
[pid  5403] munmap(0x7fc9a9d94000, 4 <unfinished ...>
[pid  5404] <... write resumed>)        = 37
[pid  5403] <... munmap resumed>)       = 0
[pid  5404] close(3 <unfinished ...>
[pid  5403] exit_group(0)               = ?
[pid  5404] <... close resumed>)        = 0
[pid  5404] exit_group(0)               = ?
[pid  5403] +++ exited with 0 +++
+++ exited with 0 +++
```

Вывод

Выполняя данную лабораторную работу, я познакомилась с утилитой strace. Где она может пригодиться? Эту лабораторную я делала последней, и мне она не пригодилась. Но на каких-то проектах она может помочь находить ошибки.