

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Вычислительной техники

ОТЧЁТ
по лабораторной работе №5
по дисциплине «Организация процессов и программирования в среде Linux»
Тема: ОБРАБОТКА СИГНАЛОВ

Студент гр. 9308

Преподаватель

Соболев М.С.

Разумовский Г.В.

Санкт-Петербург,

2022

Оглавление

1. Введение.....	3
1.1. Введение.....	3
1.2. Порядок выполнения работы.....	3
1.3. Содержание отчёта.....	3
2. Тексты программы.....	4
3. Скриншоты экрана результатов работы программы при каждом запуске.....	7
4. Вывод.....	9
5. Список использованных источников.....	10

1. Введение

1.1. Введение

Тема работы: Обработка сигналов.

Цель работы: Знакомство с механизмом сигналов и способами их обработки.

1.2. Порядок выполнения работы

1. Написать программу, которая реагирует на ошибки при выполнении операции деления и неверном использовании указателя (деление на ноль, нарушение защиты памяти). При обнаружении ошибки программа должна передать управление функции, которая выведет сообщение и завершит работу программы с кодом ошибки (1 или 2). Тип ошибки, который должна зафиксировать программа, задаётся как параметр при её запуске.

2. Откомпилировать программу и дважды запустить её с разными значениями типа ошибки.

1.3. Содержание отчёта

Отчёт по лабораторной работе должен содержать:

1. Цель и задание.
2. Тексты программы.
3. Скриншоты экрана результатов работы программы при каждом запуске.

2. Тексты программы

```
// start program
// ./main (signal | sigaction | <other=default>) (1 | 2 | <other=default>)
// -----
// 1st
// signal -- signal function usage
// sigaction -- sigaction function usage
// <other=default> -- default (signal) function usage
// -----
// 2nd
// 1 -- dividing by zero operation usage
// 2 -- adresssing to nullptr operation usage
// <other=default> -- default (dividing by zero) operation usage

#include <signal.h>
#include <unistd.h>
#include <iostream>
#include <string>

using namespace std;

void ErrorHandler (int local);

int main (int argc, char *argv[])
{
    string type_func = argv[1];
    int type_err = atoi(argv[2]);

    // https://man7.org/linux/man-pages/man2/sigaction.2.html
    struct sigaction sig; // struct for handling sigaction

    sig.sa_handler = &ErrorHandler; // error (sigaction) handle function set

    // https://www.ibm.com/docs/en/zos/2.1.0?topic=functions-sigaction-examine-change-signal-action
    // http://fkn.ktu10.com/?q=node/666
    // no flags added, because "the... ..mask with signal stays in effect
    // until the signal handler returns, or..."
```

```

// https://stackoverflow.com/questions/45477254/how-sigaction-differs-from-signal
// https://stackoverflow.com/questions/231912/what-is-the-difference-between-sigaction-and-signal
if (type_func == "signal") // signal
{
    signal(SIGFPE, ErrorHandler); // SIGFPE = 8, invalid operation (overflow, dividing by 0)
    signal(SIGSEGV, ErrorHandler); // SIGSEGV = 11, memory protection breach
}
else if (type_func == "sigaction") // sigaction
{
    sigaction(SIGFPE, &sig, NULL); // SIGFPE = 8, invalid operation (overflow, dividing by 0)
    sigaction(SIGSEGV, &sig, NULL); // SIGSEGV = 11, memory protection breach
}
else // signal by default
{
    signal(SIGFPE, ErrorHandler); // SIGFPE = 8, invalid operation (overflow, dividing by 0)
    signal(SIGSEGV, ErrorHandler); // SIGSEGV = 11, memory protection breach
}

switch(type_err)
{
    case 1: // invalid operation (overflow, dividing by 0): dividing by zero
    {
        int number = 1;
        int zero = 0;
        int res = 0;
        res = number/zero;
        break;
    }
    case 2: // memory protection breach: trying to adress to nullptr pointer
    {
        char *c = nullptr;
        *c = 'z';
        break;
    }
    default: // dividing by zero by default
    {
        int number = 1;

```

```

        int zero = 0;
        int res = 0;
        res = number/zero;
        break;
    }
}
return 0;
}

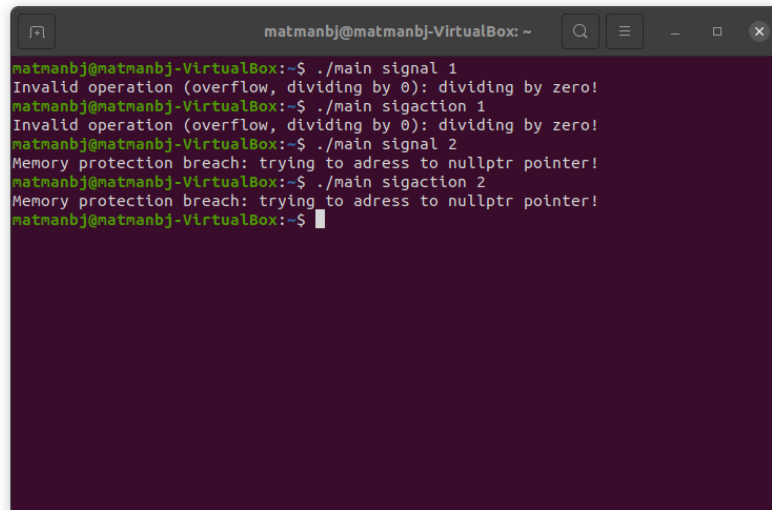
void ErrorHandler (int local)
{
    switch(local)
    {
        case SIGFPE: // dividing by zero, SIGFPE function
        {
            puts("Invalid operation (overflow, dividing by 0): dividing by zero!");
            exit(1);
        }
        case SIGSEGV: // memory protection breach, SIGSEGV function
        {
            puts("Memory protection breach: trying to adress to nullptr pointer!");
            exit(2);
        }
        default: // unable to recognize signal
        {
            puts("Unable to recognize signal!");
            exit(3);
        }
    }
}

```

3. Скриншоты экрана результатов работы программы при каждом запуске

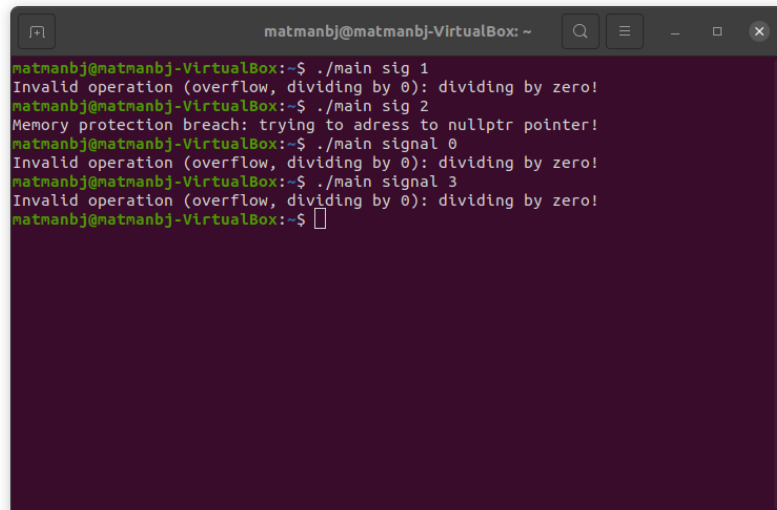
Запустим программу с различными ошибками (деления на 0 и адресации на несуществующий участок памяти) с различными функциями (signal и sigaction).

Также запустим программу с данными по умолчанию.



```
matmanbj@matmanbj-VirtualBox: ~  
matmanbj@matmanbj-VirtualBox:~$ ./main signal 1  
Invalid operation (overflow, dividing by 0): dividing by zero!  
matmanbj@matmanbj-VirtualBox:~$ ./main sigaction 1  
Invalid operation (overflow, dividing by 0): dividing by zero!  
matmanbj@matmanbj-VirtualBox:~$ ./main signal 2  
Memory protection breach: trying to adress to nullptr pointer!  
matmanbj@matmanbj-VirtualBox:~$ ./main sigaction 2  
Memory protection breach: trying to adress to nullptr pointer!  
matmanbj@matmanbj-VirtualBox:~$
```

Рисунок 1. Ошибки деления на 0 и адресации на несуществующий участок памяти с функциями signal и sigaction



```
matmanbj@matmanbj-VirtualBox: ~  
matmanbj@matmanbj-VirtualBox:~$ ./main sig 1  
Invalid operation (overflow, dividing by 0): dividing by zero!  
matmanbj@matmanbj-VirtualBox:~$ ./main sig 2  
Memory protection breach: trying to adress to nullptr pointer!  
matmanbj@matmanbj-VirtualBox:~$ ./main signal 0  
Invalid operation (overflow, dividing by 0): dividing by zero!  
matmanbj@matmanbj-VirtualBox:~$ ./main signal 3  
Invalid operation (overflow, dividing by 0): dividing by zero!  
matmanbj@matmanbj-VirtualBox:~$
```

Рисунок 2. Запуск программы с ошибкой (деление на 0) и функцией (signal) по умолчанию

4. Вывод

В ходе выполнения лабораторной работы №5 «Обработка сигналов» были изучены системные функции, которые выводили сообщение о соответствующей ошибке и завершали программу с нужным кодом. Была написана программа, которая генерировала различные ошибки (деление на 0 и адресация на несуществующий участок памяти) с различными функциями (signal и sigaction), в программе были предусмотрены обрабатывающая функция и генерируемая ошибка по умолчанию. Таким образом и было произведено ознакомление с механизмом сигналов и способами их обработки.

5. Список использованных источников

1. Онлайн-курс «Организация процессов и программирование в среде Linux» в LMS Moodle [сайт]. URL: <https://vec.etu.ru/moodle/course/view.php?id=9703>.

2. Разумовский Г.В. Организация процессов и программирование в среде Linux: учебно-методическое пособие. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2018. 40с.