Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

#### ФАКУЛЬТЕТ Информатика и системы управления КАФЕДРА Информационная безопасность (ИУ8)

**Безопасность Систем Баз Данных**

Отчет

по Лабораторной работе №4

**“Получение потоков запускаемого приложения (С++)”**

###### Выполнил:

#### Овсепян А.Н.,

#### студент группы ИУ8-63

###### Проверил:

#### Зенькович С. А.,

#### старший преподаватель кафедры ИУ8

#### Москва, 2021

**Оглавление**

[Цель работы 3](#_Toc75830307)

[Ход работы 3](#_Toc75830308)

[Практическая часть 3](#_Toc75830309)

[Вывод 4](#_Toc75830310)

[Приложение А. 5](#_Toc75830311)

[Приложение Б. 6](#_Toc75830312)

# Цель работы

Реализовать приложение, запускающее произвольное приложение с получением его потоков и кода завершения на C++

# Ход работы

## Практическая часть

Для выполнения задания мною был реализован shell скрипт. Данный скрипт возводит число base в степень power (оба числа подаются в качестве аргументов). Скрипт проверяет введенные данные на:

* правильное количество (2)
* правильное тип (целые числа)

Исходный код скрипта script.sh в [приложении А](#_Приложение_А.).

C++ приложение, которое запускает вышеописанный скрипт, получает его стандартные потоки (stdout, stderr) и код завершения. Выводит полученные данные в соответствующие поток и файл. Ошибки в ходе работы самого приложения выводятся в stderr и записываются в отдельный файл.

Исходный код С++ приложения в [приложении Б](#_Приложение_Б.).

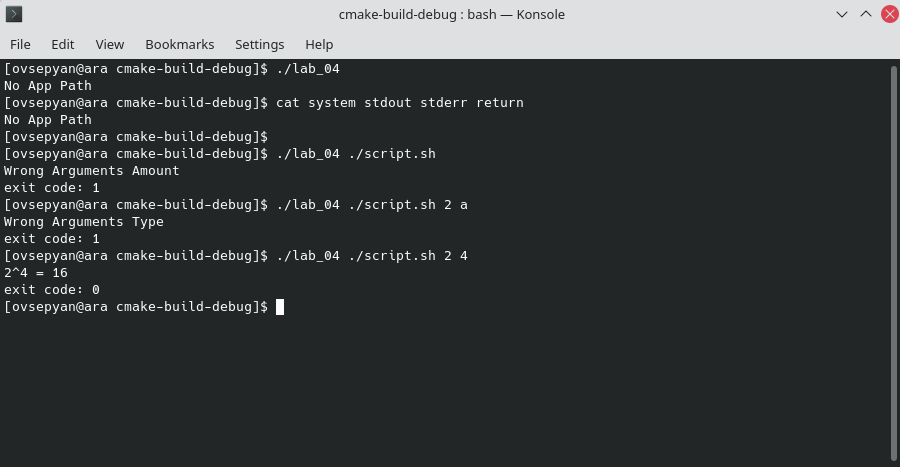


Рисунок 1 - пример использования приложения

# Вывод

Были получены знания о потоках и реализовано приложение для работы с потоками запускаемого приложения.

# Приложение А.

#!/bin/bash

**if** [ $# -eq 2 ]

**then**

re='^[0-9]+$'

**if** **!** [[ $1 =~ $re ]] || **!** [[ $2 =~ $re ]]

**then**

**echo** "Wrong Arguments Type">&2;

**exit** 1;

**fi**

res=1

base=$1

power=$2

**for** ((i=0;i<power;i++))

{

res=$((res \* base))

}

**echo** "$1^$2 = $res">&1

**exit** 0

**fi**

**echo** "Wrong Arguments Amount">&2

**exit** 1;

# Приложение Б.

#include <iostream>

#include <unistd.h>

#include <sys/wait.h>

**template**<**typename**... ARGS>

**void** print(FILE \*file, **int** code, **char** \*format, ARGS... args)

{

fprintf(file, format, args...);

**if** (code)

{

fprintf(stdout, format, args...);

} **else**

{

fprintf(stderr, format, args...);

}

}

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

{

// file for logging app errors

FILE \*f\_log;

f\_log = fopen("system", "w");

// files for streams and exit code

FILE \*f\_stdout;

FILE \*f\_stderr;

FILE \*f\_return;

f\_stdout = fopen("stdout", "w");

f\_stderr = fopen("stderr", "w");

f\_return = fopen("return", "w");

**if** (argc < 2)

{

print(f\_log, 1, "No App Path\n");

**return** 1;

}

// full path to app with args

std::string string\_path = "./";

**for** (**int** i = 1; i < argc; i++)

{

string\_path += std::string(argv[i]) + " ";

}

**const** **char** \*full\_path = string\_path.erase(string\_path.length() - 1).c\_str();

**int** stdout\_pipe[2], stderr\_pipe[2];

**if** (pipe(stdout\_pipe) < 0 || pipe(stderr\_pipe) < 0)

{

print(f\_log, 1, "Pipe Error\n");

**return** 1;

}

**int** exitcode;

pid\_t pid = fork();

**if** (pid == 0)

{

close(stdout\_pipe[0]);

close(stderr\_pipe[0]);

dup2(stdout\_pipe[1], 1);

dup2(stderr\_pipe[1], 2);

close(stdout\_pipe[1]);

close(stderr\_pipe[1]);

exitcode = system(full\_path);

**return** WEXITSTATUS(exitcode);

} **else**

{

waitpid(pid, &exitcode, 0);

}

exitcode = WEXITSTATUS(exitcode);

close(stdout\_pipe[1]);

close(stderr\_pipe[1]);

// store streams data to buffers

**char** stdout\_buffer[128], stderr\_buffer[128];

stdout\_buffer[read(stdout\_pipe[0], stdout\_buffer, **sizeof**(stdout\_buffer))] = '\0';

stderr\_buffer[read(stderr\_pipe[0], stderr\_buffer, **sizeof**(stderr\_buffer))] = '\0';

// print data

print(f\_stdout, 0, stdout\_buffer);

print(f\_stderr, 1, stderr\_buffer);

print(f\_return, 0, "exit code: %d\n", exitcode);

**return** 0;

}