



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
КАФЕДРА

Информатика и системы управления
Информационная безопасность (ИУ8)

Безопасность Систем Баз Данных

Отчет
по Лабораторной работе №4
“Получение потоков запускаемого приложения (C++)”

Выполнил:
Евула А. С.,
студент группы ИУ8-63

Проверил:
Зенькович С. А.,
старший преподаватель
кафедры ИУ8

ОГЛАВЛЕНИЕ

Цель работы	3
Основная часть	3
1. Запускаемое приложение.....	3
2. Запускающее приложение	3
1. Конкатенация аргументов в одну строку.....	3
2. Использование WEXITSTATUS:.....	3
3. Разветвление процесса и запуск скрипта.....	4
Выводы	5
Приложение А	6
Приложение Б	7

ЦЕЛЬ РАБОТЫ

Необходимо реализовать приложение, запускающее произвольное приложение с получением его потоков на C++.

ОСНОВНАЯ ЧАСТЬ

1. ЗАПУСКАЕМОЕ ПРИЛОЖЕНИЕ

Создадим скрипт `adder.sh`, который будет считать сумму длин поданных аргументов. Будем считать, что отсутствие аргументов (т.е. сумма длин аргументов равна нулю) - ошибка, т.е., если аргументов нет – скрипт выводит ошибку “argument not given”.

Исходный код в Приложение А.

2. ЗАПУСКАЮЩЕЕ ПРИЛОЖЕНИЕ

Дальше напишем приложение, запускающий данный скрипт. Оно получает потоки запускаемого приложения - `stdout` и `stderr`, выводит в консоль и записывает в файл (`out.log` и `err.log` соответственно) Также получает код завершения. выводит его в консоль и записывает в файл (`exc.log`). При возникновении ошибки в запускающем приложении, она выводится в файл `sys.log`

1. КОНКАТЕНАЦИЯ АРГУМЕНТОВ В ОДНУ СТРОКУ

```
std::string temp = "./";
for (unsigned int i = 0; i < argc - 1; ++i)
    temp += std::string(argv[i + 1]) + " ";
// remove extra space from end
temp.erase(temp.length() - 1);
const char *path = temp.c_str();
```

2. ИСПОЛЬЗОВАНИЕ WEXITSTATUS:

```
STATUS = system(path);
return WEXITSTATUS(STATUS);
```

это обусловлено тем, `int` что содержит больше, чем просто код выхода - он также хранит информацию о том, как процесс завершился (н.п. для `exit`, `WIFSIGNALED...`).

3. РАЗВЕТВЛЕНИЕ ПРОЦЕССА И ЗАПУСК СКРИПТА

```
// fork the process
int STATUS;
pid_t pid = fork();
// child
if (pid == 0)
{
    close(out_pipe[0]);
    close(err_pipe[0]);
    dup2(out_pipe[1], 1);
    dup2(err_pipe[1], 2);
    close(out_pipe[1]);
    close(err_pipe[1]);

    // run the script
    STATUS = system(path);
    return WEXITSTATUS(STATUS);
}
// parent
else
{
    waitpid(pid, &STATUS, 0);
}
STATUS = WEXITSTATUS(STATUS);
close(out_pipe[1]);
close(err_pipe[1]);
```

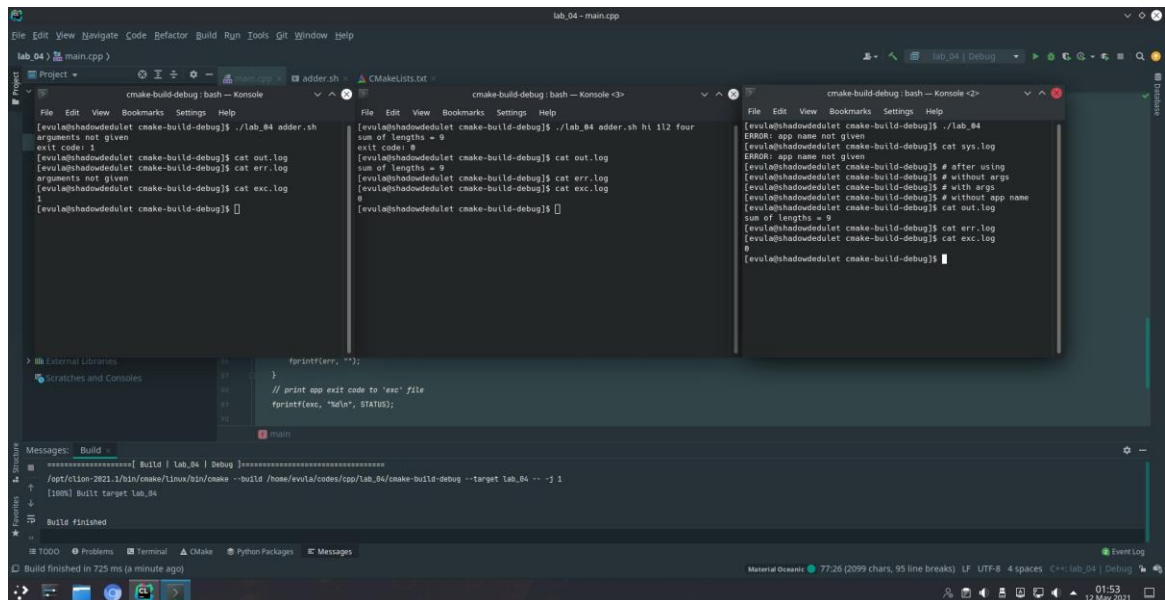


Рисунок 1 – пример использования приложения с различными аргументами

Исходный код в Приложение Б.

ВЫВОДЫ

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

ПРИЛОЖЕНИЕ А

```
#!/bin/sh
# greeting script

if [ $# -gt 0 ]; then
    SUM=0;
    for word in "$@"
    do
        SUM=$((SUM + ${#word}));
    done

    echo "sum of lengths = $SUM";
    exit 0;
fi

echo "arguments not given" 1>&2;
exit 1;
```

ПРИЛОЖЕНИЕ Б

```
#include <cstdlib>
#include <stdio>
#include <string>
#include <unistd.h>
#include <sys/wait.h>

int main(int argc, char *argv[])
{
    FILE *log;
    FILE *out;
    FILE *err;
    FILE *exc;
    log = fopen("sys.log", "w");
    out = fopen("out.log", "w");
    err = fopen("err.log", "w");
    exc = fopen("exc.log", "w");

    // no path to script
    if (argc < 2)
    {
        fprintf(log, "ERROR: app name not given\n");
        printf("ERROR: app name not given\n");
        exit(1);
    }

    // concatenate all arguments to 'path-string'
    // ./ + app_name + [args]
    std::string temp = "./";
    for (unsigned int i = 0; i < argc - 1; ++i)
        temp += std::string(argv[i + 1]) + " ";
    // remove extra space from end
    temp.erase(temp.length() - 1);
    const char *path = temp.c_str();

    // separate pipes for stdout and stderr
    int out_pipe[2], err_pipe[2];
    if (pipe(out_pipe) < 0 || pipe(err_pipe) < 0)
    {
        fprintf(log, "ERROR: FD\n");
        printf("ERROR: FD\n");
        exit(1);
    }

    // fork the process
    int STATUS;
    pid_t pid = fork();
    // child
    if (pid == 0)
    {
        close(out_pipe[0]);
```

```

        close(err_pipe[0]);
        dup2(out_pipe[1], 1);
        dup2(err_pipe[1], 2);
        close(out_pipe[1]);
        close(err_pipe[1]);

        // run the script
        STATUS = system(path);
        return WEXITSTATUS(STATUS);
    }
    // parent
    else
    {
        waitpid(pid, &STATUS, 0);
    }
    STATUS = WEXITSTATUS(STATUS);
    close(out_pipe[1]);
    close(err_pipe[1]);

    // read collected data
    char buff_out[1024], buff_err[1024];
    buff_out[read(out_pipe[0], buff_out, sizeof(buff_out))] = '\0';
    buff_err[read(err_pipe[0], buff_err, sizeof(buff_err))] = '\0';

    // print received stdout
    fprintf(out, "%s", buff_out);
    printf("%s", buff_out);
    // print received stderr
    fprintf(err, "%s", buff_err);
    printf("%s", buff_err);
    // print received exit code
    fprintf(exc, "exit code: %d\n", STATUS);
    printf("exit code: %d\n", STATUS);

    // close files
    fclose(log);
    fclose(out);
    fclose(err);
    fclose(exc);

    exit(0);
}

```