Правительство Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Отчёт к лабораторной работе №14

по дисциплине

«Языки программирования»

Работу выполнила

Студент группы СКБ222 _____ М. Х. Халимов
подпись, дата

Работу проверил _____ С. А. Булгаков
подпись, дата

Содержание

Постановка задачи

Разработать консольную утилиту позволяющую выполнять проверку целостности файлов на основе механизма контрольных сумм. При запуске программа проверяет наличие файла *Checksum.ini*. При его наличия выполняется проверка контрольных сумм для файлов, указанных в нем, иначе если поток ввода не пуст,

то из него считывается имя файла(ов) и, отделенные символом табуляции, их контрольные суммы (см. ключ *-a*). Если поток ввода пуст, проводится разбор параметров командной строки и, в зависимости от входных данных, выполняется действие либо выводится *usage*.

1. Описание вспомогательных функций программы Lab-14

1.1. Функция *calculate_checksum*

Функция получает на вход ссылку на два значения типа *string*. Функция производит вычисление контрольной суммы файла путем выполнения команд, получения вывода, его обработку и соответственно получение контрольной суммы.

1.2. Функция *read_checksums*

Функция выполняет чтение контрольных сумм из файла *Checksum.ini*.

1.3. Функция *async_calculate_checksum*

Функция получает на вход ссылку на два значения типа *string*. Производит асинхронное вычисление контрольных сумм файла.

2. Описание функции main программы Lab-14

Функция производит обработку параметров командной строки, чтение контрольных сумм из файла с помощью *read_checksums* либо из *stdin*. Также выполняет асинхронное вычисление контрольных сумм с помощью *async_calculate_checksum*. Заканчивает работу проверкой контрольных сумм и выводом результата.

3. Листинг

Без явного указания алгоритма:

```
All files are valid
```

С указанием алгоритма (на примере *crc32*):

```
1.txt 0x7929DAE1
All files are valid
```

Приложение А

Исходный код Lab-14

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <vector>
#include <map>
#include <cstdlib>
#include <cstring>
#include <algorithm>
#include <array>
#include <future>
#include <filesystem>

namespace fs = std::filesystem;

// Функция для вычисления контрольной суммы файла
std::string calculate_checksum(const std::string& file_path,
const std::string& algorithm) {
    std::string cmd;
    if (algorithm == "crc32") {
        cmd = "crc32 " + file_path;
    } else if (algorithm == "md5") {
        cmd = "md5sum " + file_path;
    } else if (algorithm == "sha256") {
        cmd = "sha256sum " + file_path;
    } else {
          std::cerr << "Unknown algorithm: " << algorithm <<
std::endl;
        return "";
    }
    // Выполнение команды и получение вывода
    std::array<char, 128> buffer{};
    std::string result;
    FILE* pipe = popen(cmd.c_str(), "r");
    if (!pipe) {
          std::cerr << "Error executing command: " << cmd <<
std::endl;
        return "";
    }
    while (fgets(buffer.data(), 128, pipe) != nullptr) {
        result += buffer.data();
    }
    pclose(pipe);

    // Обработка вывода и получение контрольной суммы
    std::stringstream ss(result);
    std::string checksum;
    ss >> checksum;
    return checksum;
```

```
}

// Функция для чтения контрольных сумм из файла Checksum.ini
std::map<std::string, std::string> read_checksums() {
    std::map<std::string, std::string> checksums;
    std::ifstream ini_file("Checksum.ini");
    if (!ini_file.is_open()) {
        return checksums;
    }

    std::string line;
    std::string current_section;
    while (std::getline(ini_file, line)) {
        // Удаление комментариев
           line.erase(std::find(line.begin(),  line.end(),  ';'),
line.end());
           line.erase(std::find(line.begin(),  line.end(),  '#'),
line.end());

        // Обработка секций
        if (line.front() == '[') {
            int ind = line.find(']');
            current_section = line.substr(1, ind - 1);
            line = line.substr(ind + 2, line.length() - ind - 2);

        }

        // Обработка ключей и значений
        std::stringstream ss(line);
        std::string key;
        std::string value;
       if (std::getline(ss, key, '=') && std::getline(ss, value))
{
            std::string file_path = key;
            std::string checksum = value;

            // Преобразование строки контрольной суммы
             if (checksum.size() >= 2 && checksum.substr(0, 2) ==
"0x") {
                checksum = checksum.substr(2);
            }

            // Добавление записи в карту контрольных сумм
            if (!current_section.empty()) {
                file_path = file_path;
            }

            checksums[file_path] = checksum;
        }
    }

    return checksums;
}
```

```
// Функция для асинхронного вычисления контрольной суммы файла
std::string      async_calculate_checksum(const      std::string&
file_path, const std::string& algorithm) {
    std::string checksum;
    try {
        checksum = calculate_checksum(file_path, algorithm);
    } catch (const std::exception& e) {
         std::cerr << "Error calculating checksum for file: " <<
file_path << std::endl;
        std::cerr << "Error message: " << e.what() << std::endl;
    }

    return checksum;
}

int main(int argc, char* argv[]) {
    std::string algorithm = "md5";
    std::vector<std::string> file_paths;
    // Обработка параметров командной строки
    for (int i = 1; i < argc; i++) {
        std::string arg = argv[i];
        if (arg == "-a") {
            if (i + 1 < argc) {
                algorithm = argv[i + 1];
                i++;
            } else {
                   std::cerr << "Missing algorithm parameter" <<
std::endl;
                return 1;
            }
        } else if (arg.front() == '-') {
            std::cerr << "Unknown option: " << arg << std::endl;
            return 1;
        } else {
            file_paths.push_back(arg);
        }
    }

    // Чтение контрольных сумм из файла Checksum.ini
    std::map<std::string, std::string> checksums;
    if (argc == 1) {
        checksums = read_checksums();
    }

    // Если файлы не указаны явно, читаем из stdin
    if (file_paths.empty() && checksums.empty()) {
        std::string line;
        while (std::getline(std::cin, line)) {
            std::stringstream ss(line);
            std::string file_path;
            std::string checksum;
```

```
                    if (std::getline(ss, file_path, '\t') &&
std::getline(ss, checksum)) {
                checksums[file_path] = checksum;
            }
        }
    }

    // Асинхронное вычисление контрольных сумм файлов
    std::vector<std::future<std::pair<std::string, std::string>>>
futures;
    for (const std::string& file_path : file_paths) {
        futures.push_back(std::async(std::launch::async, [](const
std::string& path, const std::string& algo) {
                                    return    std::make_pair(path,
async_calculate_checksum(path, algo));
        }, file_path, algorithm));
    }

    // Проверка контрольных сумм файлов
    bool is_valid = true;
    for (auto& future : futures) {
        auto result = future.get();
        const std::string& file_path = result.first;
                  const   std::string&   expected_checksum   =
checksums[file_path];
        const std::string& actual_checksum = result.second;

        if (expected_checksum.empty()) {
           std::cerr << "No checksum found for file: " << file_path
<< std::endl;
            is_valid = false;
            continue;
        }

        if (actual_checksum.empty()) {
             std::cerr << "Error calculating checksum for file: "
<< file_path << std::endl;
            is_valid = false;
            continue;
        }

        if (actual_checksum != expected_checksum) {
           std::cerr << "Checksum mismatch for file: " << file_path
<< std::endl;
              std::cerr << "Expected: " << expected_checksum <<
std::endl;
               std::cerr << "Actual:    " << actual_checksum <<
std::endl;
            is_valid = false;
        }
    }
    // Вывод результата
    if (is_valid) {
```

```
        std::cout << "All files are valid" << std::endl;
        return 0;
    } else {
        std::cerr << "Some files are invalid" << std::endl;
        return 1;
    }
}
```