Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по Системному программированию

Практическая работа «Разработка программного кода для работы с процессами и потоками в Windows»

Вариант 10-15

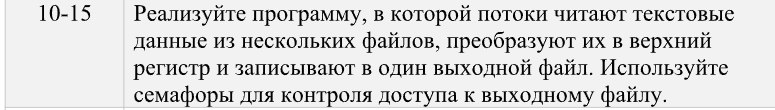
Выполнил: Плоских А.П

Группа: ПР-31

Преподаватель: Морозов В.В

2025 год

1. **Описание задачи**



1. **Результаты действия программы:**

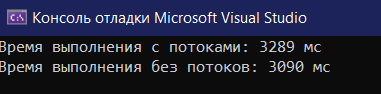


Рис. 1: Работа программы

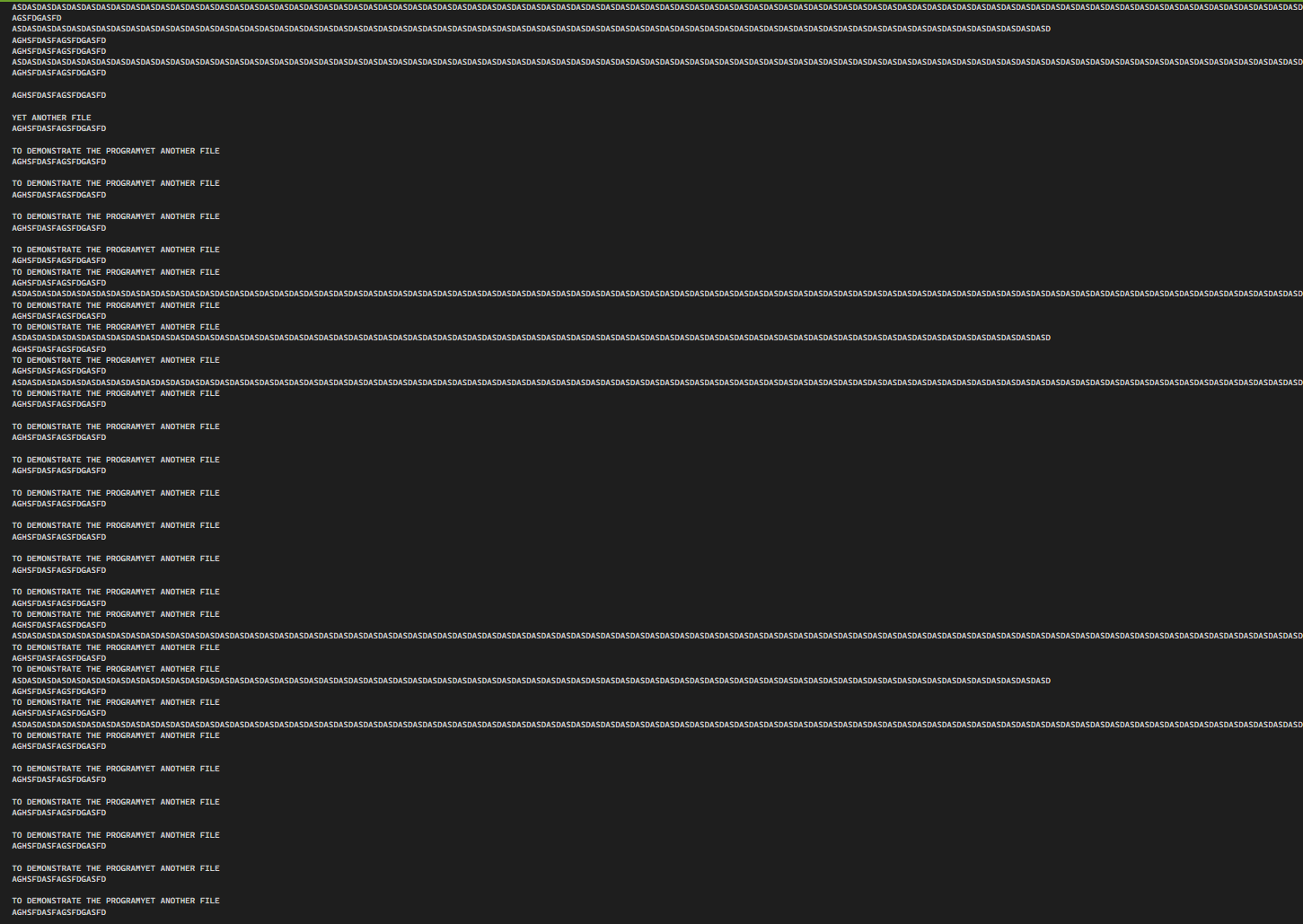


Рис 2: Итоговый файл

1. **Текст программы:**

#include <iostream>

#include <fstream>

#include <string>

#include <windows.h>

#include <vector>

#include <chrono> // Для измерения времени

std::string toUpperCase(const std::string& str) {

std::string result;

for (char c : str) {

if (c >= 'a' && c <= 'z') {

result += c - ('a' - 'A');

}

else {

result += c;

}

}

return result;

}

struct FileParams {

std::string inputFile;

std::string outputFile;

HANDLE outputMutex;

};

FileParams\* createParams(const std::string& input, const std::string& output, HANDLE mutex) {

return new FileParams{ input, output, mutex };

}

DWORD WINAPI processFile(LPVOID lpParam) {

FileParams\* params = (FileParams\*)lpParam;

std::ifstream inFile(params->inputFile);

if (!inFile.is\_open()) {

std::cerr << "Не удалось открыть файл: " << params->inputFile << std::endl;

return 1;

}

std::ofstream outFile(params->outputFile, std::ios::app);

if (!outFile.is\_open()) {

std::cerr << "Не удалось открыть выходной файл: " << params->outputFile << std::endl;

return 1;

}

std::string line;

while (std::getline(inFile, line)) {

std::string upperLine = toUpperCase(line);

WaitForSingleObject(params->outputMutex, INFINITE);

outFile << upperLine << std::endl;

ReleaseMutex(params->outputMutex);

}

inFile.close();

outFile.close();

delete params;

return 0;

}

void processFilesSequentially(const std::vector<std::string>& inputFiles, const std::string& outputFile, HANDLE outputMutex) {

std::ofstream outFile(outputFile, std::ios::app);

if (!outFile.is\_open()) {

std::cerr << "Не удалось открыть выходной файл: " << outputFile << std::endl;

return;

}

for (const auto& inputFile : inputFiles) {

std::ifstream inFile(inputFile);

if (!inFile.is\_open()) {

std::cerr << "Не удалось открыть файл: " << inputFile << std::endl;

continue;

}

std::string line;

while (std::getline(inFile, line)) {

std::string upperLine = toUpperCase(line);

WaitForSingleObject(outputMutex, INFINITE);

outFile << upperLine << std::endl;

ReleaseMutex(outputMutex);

}

inFile.close();

}

outFile.close();

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

const std::string inputFile1 = "1.txt";

const std::string inputFile2 = "2.txt";

const std::string inputFile3 = "3.txt";

const std::string outputFile = "o.txt";

HANDLE outputMutex = CreateMutex(NULL, FALSE, NULL);

if (outputMutex == NULL) {

std::cerr << "Не удалось создать мьютекс." << std::endl;

return 1;

}

auto startWithThreads = std::chrono::high\_resolution\_clock::now();

FileParams\* params1 = createParams(inputFile1, outputFile, outputMutex);

FileParams\* params2 = createParams(inputFile2, outputFile, outputMutex);

FileParams\* params3 = createParams(inputFile3, outputFile, outputMutex);

HANDLE thread1 = CreateThread(NULL, 0, processFile, params1, 0, NULL);

HANDLE thread2 = CreateThread(NULL, 0, processFile, params2, 0, NULL);

HANDLE thread3 = CreateThread(NULL, 0, processFile, params3, 0, NULL);

if (thread1 == NULL || thread2 == NULL || thread3 == NULL) {

std::cerr << "Не удалось создать поток." << std::endl;

return 1;

}

WaitForSingleObject(thread1, INFINITE);

WaitForSingleObject(thread2, INFINITE);

WaitForSingleObject(thread3, INFINITE);

CloseHandle(thread1);

CloseHandle(thread2);

CloseHandle(thread3);

auto endWithThreads = std::chrono::high\_resolution\_clock::now();

auto durationWithThreads = std::chrono::duration\_cast<std::chrono::milliseconds>(endWithThreads - startWithThreads).count();

auto startWithoutThreads = std::chrono::high\_resolution\_clock::now();

std::vector<std::string> inputFiles = { inputFile1, inputFile2, inputFile3 };

processFilesSequentially(inputFiles, outputFile, outputMutex);

auto endWithoutThreads = std::chrono::high\_resolution\_clock::now();

auto durationWithoutThreads = std::chrono::duration\_cast<std::chrono::milliseconds>(endWithoutThreads - startWithoutThreads).count();

CloseHandle(outputMutex);

// Вывод результатов

std::cout << "Время выполнения с потоками: " << durationWithThreads << " мс" << std::endl;

std::cout << "Время выполнения без потоков: " << durationWithoutThreads << " мс" << std::endl;

return 0;

}

1. **GitHub**

https://github.com/Mumlul/potokiwh.git

1. **Вывод**

С помощью этой программы как работают потоки и ка их использовать. Так же была разработана программа для смены регистра текста.