МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

Факультет радіоелектроніки, комп'ютерних систем та інфокомунікацій

Кафедра комп'ютерних систем, мереж і кібербезпеки

**Лабораторна робота**

з Системного програмування

(назва дисципліни)

на тему:

«Вивчення системних викликів Win32 API для роботи з процесами та потоками»

Виконав: студент 3-го курсу групи № 535а

напряму підготовки (спеціальності)

123-«Комп’ютерна інженерія»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр і назва напряму підготовки (спеціальності))

Скринник Є. І.

(прізвище й ініціали студента)

Прийняв: асистент каф.503

Мозговий М.В.

(посада, науковий ступінь, прізвище й ініціали)

Національна шкала: \_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_

Харків – 2020

**Цель работы:**

1. Изучение системных вызовов Win32 API работы с процессами, создание дочерних процессов.
2. Изучение системных вызовов по работе с потоками. Использование TLS памяти потока.

**Постановка задачи:**

1. Написать программу, реализующую упаковку и распаковку zip архивов. Программа должна использовать утилиту 7z.exe, которая будет непосредственно выполнять упаковку и распаковку файлов путем запуска в дочернем процессе. Программа должна поддерживать такие операции как:

1. Распаковка архива в папку

2. Упаковка одного файла в новый архив

Для получения максимальной оценки необходимо выполнить обработку ошибок от дочернего процесса путем перенаправления потока вывода. Это позволит родительскому процессу получить содержимое консоли, сформированное программой 7z.exe и по этому тексту определить была ошибка или нет.

1. Написать программу, которая может создавать 2 и более потоков (кол-во задается в командной строке). Перед запуском потоков программа заполняет для каждого потока исходный массив целочисленных значений (5-10 элементов) от 10 до 100. Каждый поток должен найти для каждого элемента массива его наибольший делитель, сохраняя полученные значения в TLS память. После нахождения всех значений он должен вывести сумму всех полученных значений и напечатать свой идентификатор. Расчет наибольшего делителя и вычисление конечной суммы должны реализовываться двумя отдельными функциями.

**Код первой программы:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <Windows.h>

#include <stdio.h>

#include <locale.h>

using namespace std;

// Функция разархивации архива

bool unpack\_files(LPSTR packed\_file, LPSTR result\_file);

// Функция архивации архива

bool pack\_files(LPSTR unpacked\_file, LPSTR result\_file);

// Функция запуска дочернего процесса

bool launch\_process(LPSTR command\_line);

// Функция Вывода ошибки

void print\_error();

int main()

{

setlocale(LC\_ALL, "Russian");

int choice;

for(;;)

{

system("cls");

printf("Лабораторная работа № 4\nПо системному программированию\nВыполнил студент 535а группы Скрынник Егор\nПрограмма для работы с архивами\n\n");

printf("Меню:\n");

printf("1 - Упаковка архива\n");

printf("2 - Распаковка архива\n");

printf("3 - Выход\n");

printf("Введите пункт меню:");

scanf("%i", &choice);

switch (choice)

{

case 1:

{

LPSTR unpacked = new CHAR[MAX\_PATH];

LPSTR result\_file = new CHAR[MAX\_PATH];

printf("Путь к файлу или директории:");

scanf("%s", unpacked);

printf("Путь к результату архивации:");

scanf("%s", result\_file);

if(pack\_files(unpacked, result\_file))

{

printf("Упаковка архива завершена успешно!\n");

}

break;

}

case 2:

{

LPSTR packed\_file = new CHAR[MAX\_PATH];

LPSTR result\_file = new CHAR[MAX\_PATH];

printf("Введите путь к архиву:");

scanf("%s", packed\_file);

printf("Введите директорию куда распаковывать архив:");

scanf("%s", result\_file);

if(unpack\_files(packed\_file, result\_file))

{

printf("Распаковка архива завершена успешно!\n");

}

break;

}

case 3: return 0; break;

default:printf("Выбран неверный пункт меню попробуйте еще раз!\n");

}

system("pause");

system("cls");

}

}

bool unpack\_files(LPSTR packed\_file, LPSTR result\_file)

{

LPSTR command\_line = new CHAR[MAX\_PATH];

ZeroMemory(command\_line, MAX\_PATH);

// Установка пути к 7z and и объкта архивирования .zip

strncpy(command\_line, "C:\\Program Files\\7-Zip\\7z.exe e", MAX\_PATH - strlen(command\_line));

strncat(command\_line, packed\_file, MAX\_PATH - strlen(command\_line));

// -o set установка директории для распаковки

strncat(command\_line, " -o", MAX\_PATH - strlen(command\_line));

strncat(command\_line, result\_file, MAX\_PATH - strlen(command\_line));

// -y Авто-ответ на все вопросы которые может спросить система да

strncat(command\_line, " -y", MAX\_PATH - strlen(command\_line));

return launch\_process(command\_line);

}

bool pack\_files(LPSTR unpacked\_file, LPSTR result\_file)

{

LPSTR command\_line = new CHAR[MAX\_PATH];

ZeroMemory(command\_line, MAX\_PATH);

// Указание пути к 7z и результирующему архиву

strncpy(command\_line, "C:\\Program Files\\7-Zip\\7z.exe a -tzip ", MAX\_PATH - strlen(command\_line));

strncat(command\_line, result\_file, MAX\_PATH - strlen(command\_line));

strncat(command\_line, " ", MAX\_PATH - strlen(command\_line));

strncat(command\_line, unpacked\_file, MAX\_PATH - strlen(command\_line));

return launch\_process(command\_line);

}

bool launch\_process(LPSTR command\_line){

// Создание пайпов

HANDLE read\_pipe;

HANDLE write\_pipe;

// Установка атрибутов безопасности

SECURITY\_ATTRIBUTES security\_attributes;

security\_attributes.nLength = sizeof(SECURITY\_ATTRIBUTES);

security\_attributes.bInheritHandle = TRUE;

security\_attributes.lpSecurityDescriptor = NULL;

CreatePipe(&read\_pipe, &write\_pipe, &security\_attributes, 0);

if (read\_pipe == INVALID\_HANDLE\_VALUE || write\_pipe == INVALID\_HANDLE\_VALUE)

{

print\_error();

return false;

}

if (!SetHandleInformation(read\_pipe, HANDLE\_FLAG\_INHERIT, 0))

{

print\_error();

return false;

}

STARTUPINFOA startup\_info;

ZeroMemory(&startup\_info, sizeof(STARTUPINFOA));

startup\_info.cb = sizeof(startup\_info);

startup\_info.hStdError = write\_pipe;

startup\_info.dwFlags |= STARTF\_USESTDHANDLES;

PROCESS\_INFORMATION process\_information;

ZeroMemory(&process\_information, sizeof(process\_information));

if (!CreateProcessA(NULL, command\_line, NULL, NULL, TRUE, NORMAL\_PRIORITY\_CLASS, NULL, NULL, &startup\_info, &process\_information)) {

print\_error(); // Выводим ошибку если не удалось создать процесс.

return false;

}

WaitForSingleObject(process\_information.hProcess, INFINITE);

DWORD readed = 0;

LPSTR result = new CHAR[1024];

ZeroMemory(result, 1024);

OVERLAPPED overlapped;

while (ReadFile(read\_pipe, result, 1024, &readed, &overlapped))

{

printf("%s", result);

}

CloseHandle(read\_pipe);

CloseHandle(write\_pipe);

CloseHandle(process\_information.hProcess);

CloseHandle(process\_information.hThread);

return true;

}

void print\_error()

{

printf("\n\*\*\*Ошибка!\*\*\*\nТекст ошибки:");

void\* cstr; //В эту переменную будет записано сообщение

FormatMessage(

FORMAT\_MESSAGE\_ALLOCATE\_BUFFER | FORMAT\_MESSAGE\_FROM\_SYSTEM,

NULL,

GetLastError(),

MAKELANGID(LANG\_NEUTRAL, SUBLANG\_DEFAULT),

(LPTSTR)&cstr,

0,

NULL

);

PTCHAR res((char\*)cstr);

printf("%s", res);

}

**Результат работы:**

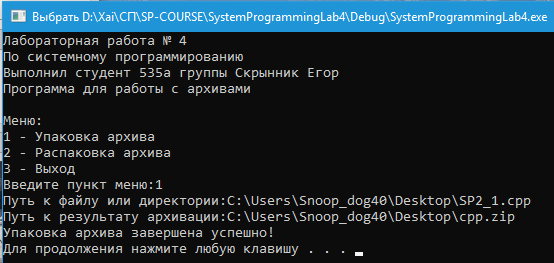


Рисунок 1 – Упаковка файла в архив.

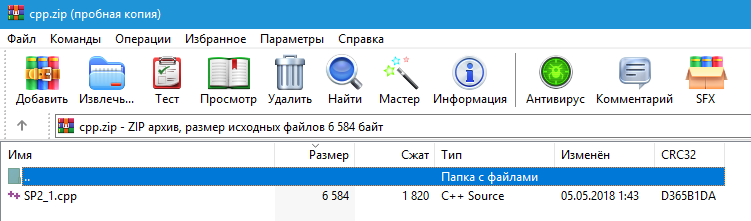
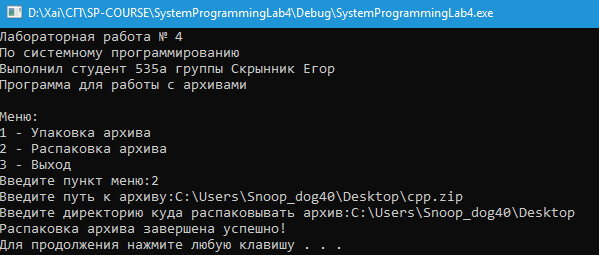


Рисунок 2 – Упакованный архив.

  
Рисунок 3 – Распаковка архива.

**Код второй программы:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <Windows.h>

#include <time.h>

#include <locale.h>

#define ARRAY\_MAX\_SIZE 7

CRITICAL\_SECTION criticalSection;

HANDLE\* threads;

int tls\_index;

// Функция запуска потоков

void launch\_threads(int count);

// Функция потоков

DWORD WINAPI thread\_function(LPVOID arr);

// Функция подсчета массивов

void compute\_array(int\* main\_array, int\* result\_array);

// Функция нахождения наибольшего делителя числа

int find\_largest\_divisor(int num);

// Функция нахождения суммы элементов массива

int array\_sum(int\* array);

// Функция вывода массивов

void print\_array(int\* arr);

int main()

{

setlocale(LC\_ALL, "Russian");

int count;

printf("Лабораторная работа № 4\nПо системному программированию\nВыполнил студент 535а группы Скрынник Егор\nПрограмма для работы с потоками\n\n");

printf("Введите количество потоков:");

scanf("%i", &count);

threads = new HANDLE[count];

srand(time(0));

InitializeCriticalSection(&criticalSection);

launch\_threads(count);

WaitForMultipleObjects(count, threads, TRUE, INFINITE);

DeleteCriticalSection(&criticalSection);

system("pause");

return 0;

}

void launch\_threads(int count)

{

InitializeCriticalSection(&criticalSection);

int\*\* arrays = new int\*[count];

tls\_index = TlsAlloc();

for (int i = 0; i < count; i++)

{

arrays[i] = new int[ARRAY\_MAX\_SIZE];

for (int j = 0; j < ARRAY\_MAX\_SIZE; j++)

arrays[i][j] = rand() % 90 + 10;

threads[i] = CreateThread(NULL, 0, thread\_function, arrays[i], NULL, NULL);

}

}

DWORD WINAPI thread\_function(LPVOID param)

{

EnterCriticalSection(&criticalSection);

printf("Идентификатор потока: %u\n", GetCurrentThreadId());

int\* array = (int\*)param;

TlsSetValue(tls\_index, (LPVOID)(new int[ARRAY\_MAX\_SIZE]));

int sum = 0;

printf("Массив потока: ");

print\_array(array);

compute\_array(array, (int\*)TlsGetValue(tls\_index));

printf("\nПодсчитанный массив: ");

print\_array((int\*)TlsGetValue(tls\_index));

sum = array\_sum((int\*)TlsGetValue(tls\_index));

printf("\nКонечная сумма: %d\n\n", sum);

LeaveCriticalSection(&criticalSection);

return 0;

}

void compute\_array(int\* main\_array, int\* result\_array)

{

for (int i = 0; i < ARRAY\_MAX\_SIZE; i++)

{

result\_array[i] = find\_largest\_divisor(main\_array[i]);

}

}

int find\_largest\_divisor(int num)

{

int j = num / 2;

for (int i = j; i >= 2; i--)

{

if (num % i == 0)

return i;

}

}

int array\_sum(int\* array)

{

int result = 0;

for (int i = 0; i < ARRAY\_MAX\_SIZE; i++)

{

result += array[i];

}

return result;

}

void print\_array(int\* arr)

{

for (int i = 0; i < ARRAY\_MAX\_SIZE; i++) {

printf("%i ", arr[i]);

}

}

**Результат работы:**

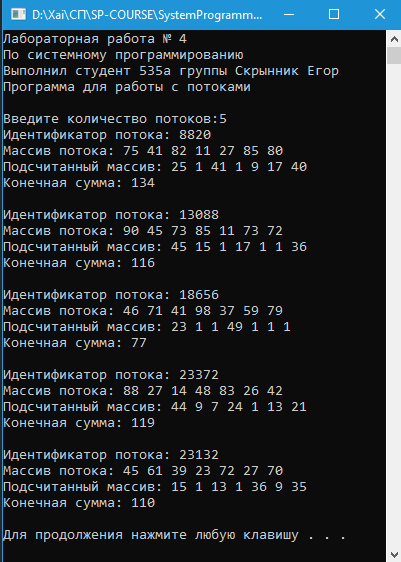


Рисунок 4 – Результат работы потоков.