Міністерство освіти и науки України Національний

аерокосмічний уіиверситет ІМ. Н. Е. ЖУКОВСЬКОГО

Кафедра 503

Лабораторна робота № 4

з дисципліни

«**Системе програмування**»

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

Виконав: ст. гр. 535Б

Жовнір В.Е

Перевірив: асистент каф. 503

Мозговий М.В.

Харків 2020

Задание 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.

Текст программы

```
// Archiever.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#define MAX_BUF 200
#define STCHAR sizeof(TCHAR)
#define WAITING_TIME 2000

int _tmain(INT argc, TCHAR** argv)
{
	DWORD realC = 1;
	LPCTSTR utilPath = TEXT("7z.exe");
	LPTSTR act = argv[1];
	LPTSTR archName, dirOrFile;

	//making a pipe
	HANDLE meReads, childWrites;
	SECURITY_ATTRIBUTES saAttr =
		{sizeof(SECURITY_ATTRIBUTES), NULL, TRUE};

	if (!CreatePipe(&meReads, &childWrites, &saAttr, MAX_BUF*STCHAR))
		{ _tprintf(TEXT("Pipe hasn't been created\n"));
		return 0;
	}

	if (!SetHandleInformation(meReads, HANDLE_FLAG_INHERIT, 0))
		{ _tprintf(TEXT("Not inherited handle\n"));
		return 0;
	}

	//creating process
	LPTSTR cmdLine = new TCHAR[MAX_BUF];
	memset(cmdLine, 0, MAX_BUF*STCHAR);

	//making command line
	archName = new TCHAR[MAX_BUF];
	dirOrFile = new TCHAR[MAX_BUF];
	GetFullPathName(argv[2], MAX_BUF, archName, NULL);
	GetFullPathName(argv[3], MAX_BUF, dirOrFile, NULL);

	switch (act[0]) {
	case TEXT('a'):
		_stprintf(cmdLine, TEXT("%s %s %s %s"),
			utilPath, act, archName, dirOrFile);
		break;

	case TEXT('e'):
		_stprintf(cmdLine, TEXT("%s %s %s"),
			utilPath, act, archName);

		//set cwd for extraction
		if(!SetCurrentDirectory(dirOrFile)){
			_tprintf(TEXT("Wrong path"));
			return 0;
		}
		break;

	default :
		_tprintf(TEXT("Wrong command"));
		return 0;
	}

	STARTUPINFO si = {};
	PROCESS_INFORMATION pi = {};

	GetStartupInfo(&si);
```

```
        si.cb = sizeof(STARTUPINFO);
        si.hStdOutput = si.hStdError = childWrites;
        si.hStdInput = NULL;
        si.dwFlags = STARTF_USESTDHANDLES;

        if (!CreateProcess(NULL,
               cmdLine,
               NULL, NULL, TRUE, 0, NULL, NULL,
               &si, &pi)) {

               _tprintf(TEXT("Can't create process"));
               return 0;
        }


        //get result of process

        realC = 1;
        LPSTR buf = new CHAR[MAX_BUF+1];
        memset(buf, 0, MAX_BUF+1);
        BOOL r = TRUE;
        BOOL error = FALSE;
        LPSTR
               needle1 = "WARNING",
               needle2 = "ERROR";

        WaitForSingleObject(pi.hThread, WAITING_TIME);
        //define if operation succeeded
        do {
               r = ReadFile(meReads, (LPVOID)buf, MAX_BUF, &realC,
               NULL); buf[realC] = '\0';

               //error search
               if (!error) {
                      error = strstr(buf, needle1)
                             || strstr(buf, needle2);
               }

               printf("%s", buf);
        } while ((realC | r != 0) && realC == MAX_BUF);

        if (!error) {
               _tprintf(TEXT("\n\n\tEverithing is ok.\n"));
        }
        else {
               _tprintf(TEXT("\n\n\tSorry, error occured.\n"));
        }

        //close process
        CloseHandle(pi.hThread);
        CloseHandle(pi.hProcess);
        CloseHandle(childWrites);

    return 0;
}
```

Тесты

Цель	Вх. Данные	Ожидаемый результат	Результат
Архивация файла	C:\Users\Dell\Desktop\lab4\Archiever\Debug>Archiever.exe a arch.7z Archiever.ilk	Создание архива по имени *arch.7z* в папке с проэктом, содержащим файл *Archiever.ilk*	7-Zip [64] 16.04 : Copyright (c) 1999-201 Pavlov : 2016-10-04 Items to compress: 1 Scanning the drive: 1 file, 515596 bytes (504 KiB) Creating archive: E:\work\sysProg\lab4\Archiever\Debug\ar Items to compress: 1 Files read from disk: 1 Archive size: 50828 bytes (50 KiB) Everything is Ok
Распаковка архива в папку	e arch.7z E:\work\sysProg\lab4\Archiever	Появление файла *Archiever.ilk* в папке E:\work\sysProg\lab4\Archiever	Scanning the drive for archives: 1 file, 48171 bytes (48 KiB) Extracting archive: C:\Users\Dell\Desktop\lab4\Archiever\Debug\arch.7z -- Path = C:\Users\Dell\Desktop\lab4\Archiever\Debug\arch.7z Type = 7z Physical Size = 48171 Headers Size = 130 Method = LZMA2:19 Solid = - Blocks = 1 Everything is Ok Size: 405024 Compressed: 48171 Everithing is ok.

Задание 2

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.

Текст программы

```
// ThreadsCalcs.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#define NVALS 7
#define MAX_VAL 100
#define MIN_VAL 10

#define MIN_COUNT 5
#define MAX_COUNT 10

INT addrIndex, countIndex;


void BiggestDivider(INT* arr, INT count, INT* bds)
	{ //const INT SINT = 4;

	for (size_t i = 0; i < count; i++)
	{
		INT curr = arr[i];
		//bd seems to have no sense...
		bds[i] = curr;
	}
}

INT Sum() {

	//get array and his size
	INT* arr = (INT*)TlsGetValue(countIndex);
	INT count = (INT)TlsGetValue(addrIndex);

	INT r = 0;
	for (size_t i = 0; i < count; i++)
	{
		r += arr[i];
	}
	return r;
}

VOID ThreadFunc(VOID) {

	//making generator random
	DWORD currId = GetCurrentThreadId();
	srand((unsigned int)currId);

	//get count
	INT count = rand() % (MAX_COUNT - MIN_COUNT) + MIN_COUNT;

	//make values
	int* vals = new int[count];
	for (size_t j = 0; j < count; j++)
	{
		vals[j] = rand() % (MAX_VAL - MIN_VAL) + MIN_VAL;
	}

	INT* rArray = new INT[count];
	BiggestDivider(vals, count, rArray);

	//save to TLS
	if (!TlsSetValue(countIndex, (LPVOID)rArray)) {
		_tprintf(TEXT("Value (1) has't been set\n"));
	}

	if (!TlsSetValue(addrIndex, (LPVOID)count)) {
		_tprintf(TEXT("Value (2) has't been set\n"));
	}
```

```
	_tprintf(TEXT("\nID: %d, Sum: %d\n\n"), currId, Sum());

	delete[] vals;
	delete[] rArray;
}

int _tmain(INT argc, TCHAR** argv)
{
	if (argc != 2) {
		_tprintf(TEXT("Wrong n of args\n"));
		return 0;
	}

	//get count of threads
	INT nThreads = _ttoi(argv[1]);

	HANDLE* threads = new HANDLE[nThreads];

	//allocate tls for all the threads addrIndex
	= TlsAlloc(); //array pointer countIndex =
	TlsAlloc(); // count of elements

	//launch threads with their values
	for (size_t i = 0; i < nThreads; i++)
	{
		DWORD IDThread;
		threads[i] = CreateThread(NULL, // default security attributes
			0,                          // use  default stack size
			(LPTHREAD_START_ROUTINE)ThreadFunc, // thread function
			NULL,                    // no thread function argument
			0,                       // use  default creation flags
			&IDThread);              // returns  thread identifier
	}

	for (size_t i = 0; i < nThreads; i++)
		WaitForSingleObject(threads[i], INFINITE);

	TlsFree(addrIndex);
	delete[] threads;

	return 0;
}
```

Тесты

Цель	Вх. Данные	Ожидаемый результат	Результат
Запуск программы	10	Вывод результатов вычислений из 10 потоков.	ID: 14772, Sum: 504 ID: 15908, Sum: 585 ID: 16780, Sum: 432 ID: 4796, Sum: 204 ID: 18240, Sum: 354 ID: 11968, Sum: 490 ID: 15408, Sum: 316 ID: 18320, Sum: 402 ID: 18180, Sum: 535 ID: 18308, Sum: 307