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»

Виконала: студентка 3-го курсу групи №525ст2
напряму підготовки (спеціальності)
123-«Комп'ютерна інженерія»
(шифр і назва напряму підготовки (спеціальності))
Коваленко Я.О.
(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:

Программа 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.

Программа 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.

Ход работы:

Код программы:

```
#define _CRT_SECURE_NO_WARNINGS
#include <Windows.h>
```

```
#include <stdio.h>
#include <locale.h>

#define ACTION_UNPACK 1
#define ACTION_PACK 2

LPCSTR unpack_7z = "C:\\Program Files\\7-Zip\\7z.exe e ";
LPCSTR pack_7z = "C:\\Program Files\\7-Zip\\7z.exe a -tzip ";

void unpack_files(LPSTR unpackFile, LPSTR resultFile);
void pack_files(LPSTR unpackFile, LPSTR resultFile);
void print_error();

int main()
{	int i;
while (true)
{
	printf("1 - Unpack file\n");
	printf("2 - Pack files\n");
	printf(">>");
	scanf_s("%i", &i);

	switch (i)
	{
	case ACTION_UNPACK:
	{
		LPSTR unpackFile = new CHAR[MAX_PATH];
		LPSTR resultFile = new CHAR[MAX_PATH];

		printf("Path to zip\n");
		printf(">> ");
		scanf("%s", unpackFile);

		printf("Path to result\n");
		printf(">> ");
		scanf("%s", resultFile);

		unpack_files(unpackFile, resultFile);
		break;
	}
	case ACTION_PACK:
	{
		LPSTR packFile = new CHAR[MAX_PATH];
		LPSTR resultFile = new CHAR[MAX_PATH];

		printf("Path to File/Directory\n");
		printf(">> ");
		scanf("%s", packFile);

		printf("Path to result (.zip)\n");
		printf(">> ");
		scanf("%s", resultFile);
```

```
                pack_files(packFile, resultFile);
                break;
            }

            default: printf("Invalid input!\n"); break;
            }

            system("pause");
            system("cls");
        }


        }

        void unpack_files(LPSTR unpackFile, LPSTR resultFile)
        {
        LPSTR commandLine = new CHAR[MAX_PATH];
        ZeroMemory(commandLine, MAX_PATH);

        // set path to 7z and .zip archive (-e)
        strncpy(commandLine, unpack_7z, MAX_PATH - strlen(commandLine));
        strncat(commandLine, unpackFile, MAX_PATH - strlen(commandLine));

        // -o set result directory
        strncat(commandLine, " -o", MAX_PATH - strlen(commandLine));
        strncat(commandLine, resultFile, MAX_PATH - strlen(commandLine));

        // -y auto answer YES
        strncat(commandLine, " -y", MAX_PATH - strlen(commandLine));

        //Create pipes
        HANDLE hReadPipe;
        HANDLE hWritePipe;

        //security attributes for pipes
        SECURITY_ATTRIBUTES saAttr;
        saAttr.nLength = sizeof(SECURITY_ATTRIBUTES);
        saAttr.bInheritHandle = TRUE;
        saAttr.lpSecurityDescriptor = NULL;

        CreatePipe(&hReadPipe, &hWritePipe, &saAttr, 0);
        if (hReadPipe == INVALID_HANDLE_VALUE || hWritePipe ==
INVALID_HANDLE_VALUE)
            exit(1);

        if (!SetHandleInformation(hReadPipe, HANDLE_FLAG_INHERIT, 0))
            exit(1);

        STARTUPINFOA si;
        ZeroMemory(&si, sizeof(STARTUPINFOA));
        si.cb = sizeof(si);
        //Need only errors
```

```
	//si.hStdOutput = hWritePipe;
	si.hStdError = hWritePipe;
	si.dwFlags |= STARTF_USESTDHANDLES;

	PROCESS_INFORMATION pi;
	ZeroMemory(&pi, sizeof(pi));

	if (!CreateProcessA(NULL, commandLine, NULL, NULL, TRUE, 0, NULL, NULL, &si,
&pi))
	{
		print_error();
	}
	else
	{
		WaitForSingleObject(pi.hProcess, INFINITE);
		DWORD readed = 0;
		LPSTR result = new CHAR[1024];
		ZeroMemory(result, 1024);
		OVERLAPPED overlapped;
		while (ReadFile(hReadPipe, result, 1024, &readed, &overlapped))
		{
			printf("%s", result);
		}
	}

	CloseHandle(hReadPipe);
	CloseHandle(hWritePipe);
	CloseHandle(pi.hProcess);
	CloseHandle(pi.hThread);
	}
	void pack_files(LPSTR packFile, LPSTR resultFile)
	{

	LPSTR commandLine = new CHAR[MAX_PATH];
	ZeroMemory(commandLine, MAX_PATH);
	// set path to 7z and result archive (a)
	strncpy(commandLine, pack_7z, MAX_PATH - strlen(commandLine));
	strncat(commandLine, resultFile, MAX_PATH - strlen(commandLine));

	strncat(commandLine, " ", MAX_PATH - strlen(commandLine));
	strncat(commandLine, packFile, MAX_PATH - strlen(commandLine));

	//Create pipes
	HANDLE hReadPipe;
	HANDLE hWritePipe;

	//security attributes for pipes
	SECURITY_ATTRIBUTES saAttr;
	saAttr.nLength = sizeof(SECURITY_ATTRIBUTES);
	saAttr.bInheritHandle = TRUE;
	saAttr.lpSecurityDescriptor = NULL;
```

```
	CreatePipe(&hReadPipe, &hWritePipe, &saAttr, 0);
	if (hReadPipe == INVALID_HANDLE_VALUE || hWritePipe == INVALID_HANDLE_VALUE)
		exit(1);

	if (!SetHandleInformation(hReadPipe, HANDLE_FLAG_INHERIT, 0))
		exit(1);

	STARTUPINFOA si;
	ZeroMemory(&si, sizeof(STARTUPINFOA));
	si.cb = sizeof(si);
	//Need only errors
	//si.hStdOutput = hWritePipe;
	si.hStdError = hWritePipe;
	si.dwFlags |= STARTF_USESTDHANDLES;

	PROCESS_INFORMATION pi;
	ZeroMemory(&pi, sizeof(pi));


	if (!CreateProcessA(NULL, commandLine, NULL, NULL, TRUE, NORMAL_PRIORITY_CLASS, NULL, NULL, &si, &pi)) {
		print_error();
	}
	else
	{
		WaitForSingleObject(pi.hProcess, INFINITE);
		DWORD readed = 0;
		LPSTR result = new CHAR[1024];
		ZeroMemory(result, 1024);
		OVERLAPPED overlapped;
		while (ReadFile(hReadPipe, result, 1024, &readed, &overlapped))
		{
			printf("%s", result);
		}
	}
	CloseHandle(hReadPipe);
	CloseHandle(hWritePipe);
	CloseHandle(pi.hProcess);
	CloseHandle(pi.hThread);
	}

	void print_error()
	{
	printf("\nSomething went wrong(\n");
	LPVOID e_mess;
	DWORD e_code = GetLastError();
	FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM, NULL,
		e_code, MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), (LPTSTR)&e_mess, 0, NULL);
	char* err_mess = (char*)e_mess;
```

```
wprintf(L"ERROR code: 0x%x\n", e_code);
wprintf(L"ERROR message: %s\n", err_mess);
}
#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;
void launch_threads(int count);
DWORD WINAPI thread_function(LPVOID arr);
void compute_array(int* main_array, int* result_array);
int find_largest_divisor(int num);
int array_sum(int* array);
void print_array(int* arr);

int main()
{
int count;
printf("Input count streams:");
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("Stream id: %u\n", GetCurrentThreadId());
int* array = (int*)param;
TlsSetValue(tls_index, (LPVOID)(new int[ARRAY_MAX_SIZE]));
int sum = 0;

printf("Stream array: ");
print_array(array);

compute_array(array, (int*)TlsGetValue(tls_index));

printf("\nCounted array: ");
print_array((int*)TlsGetValue(tls_index));

sum = array_sum((int*)TlsGetValue(tls_index));
printf("\nFinal amount: %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]);
}
}
```

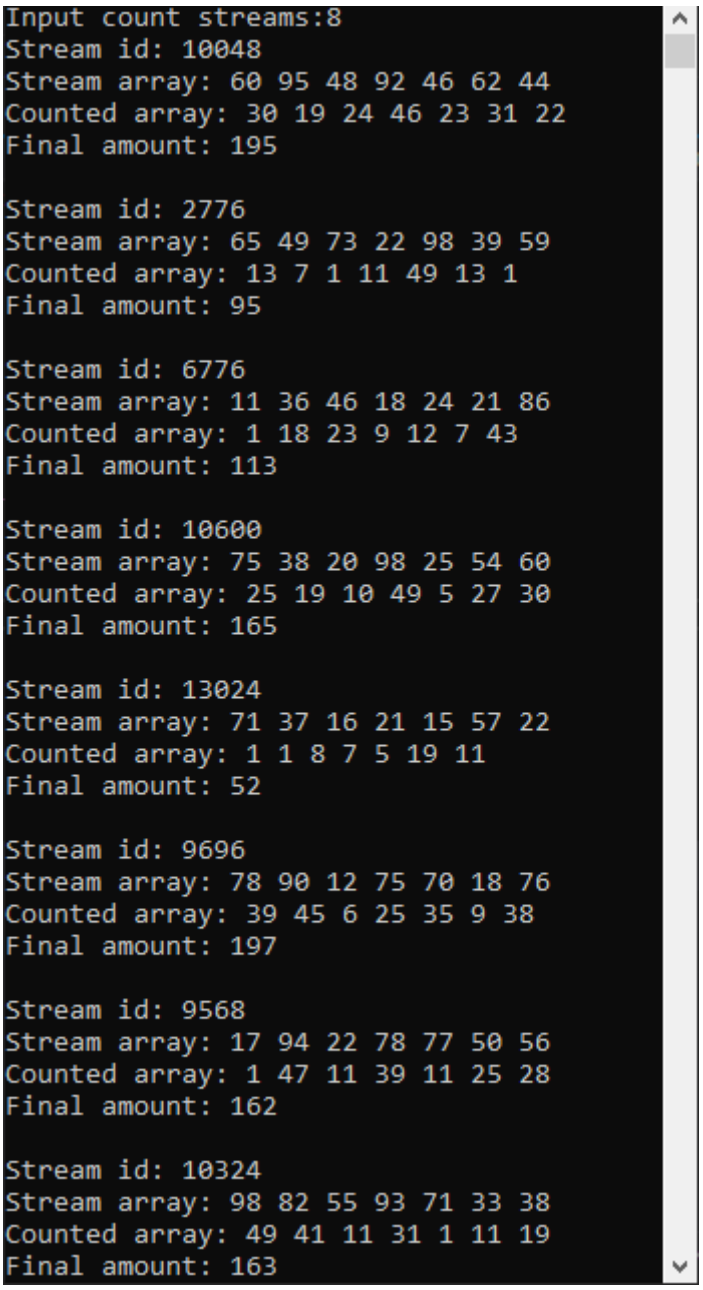
Результат работы:

```
1 - Unpack file
2 - Pack files
>>2
Path to File/Directory
>> E:\SP\lab0\lab0\Debug\lab0.exe
Path to result (.zip)
>> E:\SP\lab3\lab3\lab3\Debug\lab0.zip
Press any key to continue . . .
```

Name	Date modified	Type	Size
lab3.tlog	5/7/2020 2:34 AM	File folder	
lab0.zip	5/7/2020 2:45 AM	Архив ZIP - WinR...	12 KB
lab3.log	5/7/2020 2:34 AM	Text Document	1 KB
vc141.idb	5/7/2020 2:34 AM	VC++ Minimum R...	27 KB
vc141.pdb	5/7/2020 2:34 AM	Program Debug D...	68 KB

```
1 - Unpack file
2 - Pack files
>>1
Path to zip
>> E:\SP\lab3\lab3\lab3\Debug\lab0.zip
Path to result
>> E:\SP\lab3\lab3\lab3\Debug\lab0.exe
Press any key to continue . . .
```

Name	Date modified	Type	Size
lab0.exe	5/7/2020 2:47 AM	File folder	
lab3.tlog	5/7/2020 2:34 AM	File folder	
lab0.zip	5/7/2020 2:45 AM	Архив ZIP - WinR...	12 KB
lab3.log	5/7/2020 2:34 AM	Text Document	1 KB
vc141.idb	5/7/2020 2:34 AM	VC++ Minimum R...	27 KB
vc141.pdb	5/7/2020 2:34 AM	Program Debug D...	68 KB



Выводы:

В результате выполнения данной лабораторной работы были изучены системные вызовы Win32 API работы с процессами; создание дочерних процессов, а также системных вызовов по работе с потоками. и использование TLS памяти потока.