Федеральное агентство связи

Сибирский государственный университет телекоммуникаций и

информатики

Кафедра прикладной математики и кибернетики

Отчёт по лабораторной работе №8

Выполнил: студент группы ИП–811 Адов А.С.

Проверил: Малков Е.А.

Новосибирск 2020

Цель: приобрести навыки разработки многопоточных приложений на основе различных реализаций потоков.

Задание 1: протестировать программы, разобранные в Лекции 8.

Задание 2: написать многопоточную программу, каждый поток которой увеличивает значение глобальной переменной на определенную величину. Проверить совпадает ли значение этой переменной с ожидаемым значением после выполнения всех потоков.

***lec\_8.c***

#include <windows.h>

#include <process.h>

#include <stdio.h>

typedef int (\*fun)(char \*, int \*p);

int g(char \*str, int \*p)

{

\*p = 0;

printf("%s\n", str);

return 0;

}

int q = 1;

void Thread(void \*pg)

{

int counter = 0;

while(q)

{

printf("child\n");

Sleep(10);

if((counter++) > 1000)

{

break;

}

}

((fun)pg)("thread\_is\_over!", &q);

}

int main(void)

{

\_beginthread(Thread, 0, (void \*)g);

while(q)

{

printf("parent\n", q);

Sleep(10);

}

return 0;

}

***lec\_8\_api.c***

#include <windows.h>

#include <stdio.h>

typedef int (\*fun)(char\*, int\* p);

int g(char\* str, int\* p)

{

\*p=0;

printf("%s\n", str);

return 0;

}

int q=1;

DWORD WINAPI Thread(void\* pg)

{

int counter = 0;

while(q)

{

printf("child\n");

Sleep(10);

if((counter++) > 1000)

{

break;

}

}

((fun)pg)("thread\_is\_over!", &q);

return 0;

}

int main(void)

{

DWORD dwThreadId;

CreateThread(NULL, 0, Thread, g, 0, &dwThreadId);

while(q)

{

printf("parent\n", q);

Sleep(10);

}

return 0;

}

***lab\_8.c***

#include <windows.h>

#include <process.h>

#include <stdio.h>

#define START\_GLOBAL\_VALUE 0

#define QUANTITY\_THREAD 4

#define MAX\_GLOBAL\_VALUE 10000

int global\_value = START\_GLOBAL\_VALUE;

DWORD WINAPI Increase\_Global\_Value() //because parametr NULL in create thread function

{

for(short int i = 0; i < MAX\_GLOBAL\_VALUE; i++)

{

global\_value++;//increment

}

return 0;

}

int main()

{

DWORD Array\_Thread\_Identifier[QUANTITY\_THREAD]; //array dword type for id threads

HANDLE Array\_Tread\_Handle[QUANTITY\_THREAD]; //array handle type for descriptors threads

for(short int i = 0; i < QUANTITY\_THREAD; i++)

{

Array\_Tread\_Handle[i] = CreateThread(NULL, 0, Increase\_Global\_Value, NULL, 0, &Array\_Thread\_Identifier[i]); //function for create thread

//first parametr == NULL: for handle couldn't be inherited child process

//second parametr == 0: first size stack(if == 0 then used standart size)

//third parametr == my function: pointer for function which should be executed by the thread

//fourth parametr == NULL: pointer for variable which should be to thread

//fifth parametr == 0: flags for control create thread(if == 0 then start thread immediately after creation)

//sixth parametr == &thread\_id: pointer for variable which get id thread

}

WaitForMultipleObjects(QUANTITY\_THREAD, Array\_Tread\_Handle, TRUE, INFINITE); //function for wait until one or all of the specified threads are the signaled state

//first parametr == quant\_thread: for number of object handles in the second parametr

//second parametr == thread\_handle: array of object handles

//third parametr == true: the function returns when the state of (if == true then all) objects in the second parameter

//fourth parametr == ifinite: interval time-out (if == infinite then fucntion returns only when objects are signaled)

for(short int i = 0; i < QUANTITY\_THREAD; i++)

{

CloseHandle(Array\_Tread\_Handle[i]); //function for close handle for open object

}

printf("Real\tExpected\n");

printf("%d\t%d\n", global\_value, MAX\_GLOBAL\_VALUE \* QUANTITY\_THREAD);

return 0;

}

***Выполнение программ :***

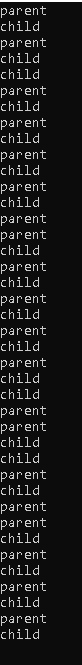
 

Рис.1 «Результаты выполнения программы ***lec\_8.c*** и ***lec\_8\_api.c***, являющимися тестами лекции»

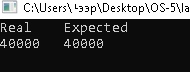
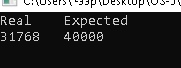
 

Рис.2 «Результаты двух запусков программы ***lab\_8.с*** »

***Заключение:***

Приобрел навыки разработки многопоточных приложений на основе различных реализаций потоков.