МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут **ІКНІ**
Кафедра **ПЗ**

ЗВІТ

До лабораторної роботи № 2
На тему: *"**Робота з процесами в ОС Windows**"*
З дисципліни: *"Операційні системи."*

Лектор:
доцент каф. ПЗ
Яковина В.С.

Виконав:
ст. гр. ПЗ-23
Горєліков А.М.

Прийняв:
ст. викл. каф. ПЗ
Гасько Р. Т.

« ____ » _____ 2017 р.

∑= ____ _____.

Львів – 2017

Тема: Робота з процесами в ОС Windows.
Мета: Навчитися працювати с процессами за допомогою функцій WinAPI.

Завдання для лабораторної роботи

1. Створити окремий процес, і здійснити в ньому табулювання функції, задану розкладом в ряд Тейлора, в області її визначення на відрізку від A до B (кількість кроків не менше 100 000). Функцію взяти з у відповідності до номера функції та порядкового номера у журнальному списку.

2. Реалізувати табулювання функцій у 2-ох, 4-ох, 8-ох процесах. Виміряти час роботи процесів за допомогою функцій WinAPI. Порівняти результати роботи в одному і в багатьох процесах.

3. Для кожного процесу реалізувати можливість його запуску, зупинення, завершення та примусове завершення («вбиття»).

4. Реалізувати можливість зміни пріоритету виконання процесу.

5. Продемонструвати результати виконання роботи, а також кількість створених процесів у “Диспетчері задач”, або подібних утилітах (н-д, ProcessExplorer).

Хід роботи:

Код:

server_main.cpp

```
#include <iostream>
#include <Windows.h>

#define EXECUTABLE_LOCATION "sub_process.exe"

int main(int argc, char** argv)
{
   int number_of_processes = 0;
   int process_to_suspend = 0;
   int process_to_kill = 0;
   int process_to_realtime = 0;
   std::cout << "Enter number of processes: ";
   std::cin >> number_of_processes;
   std::cout << "Enter process to suspend (or -1 for not suspending): ";
   std::cin >> process_to_suspend;
   std::cout << "Enter process to kill (that's bad, i know, but what can i do / -1 for not killing): ";
   std::cin >> process_to_kill;
   std::cout << "Enter process, whose priority will be changed to realtime: ";
   std::cin >> process_to_realtime;

   PROCESS_INFORMATION *process_info = new PROCESS_INFORMATION[number_of_processes]();
   STARTUPINFO info = {};
   info.cb = sizeof(info);

   for (int i = 0; i < number_of_processes; ++i)
   {
      std::cout << "Creating process, number=" << i << std::endl;
      if (!CreateProcess(EXECUTABLE_LOCATION, //exe location
                  NULL,          //cmd command
                  NULL,          //security attributes for process
                  NULL,          //security attributes for thread
                  FALSE,          //inherit handles?
                  CREATE_NO_WINDOW,    //creation flags
                  NULL,          //enviroment varibles block
                  NULL,          //process directory
                  &info,
                  &process_info[i]))
      {
         std::cout << "Process creation failed, code : " << (int)GetLastError() << std::endl;
         return 0;
      }
```

```
        if (process_to_suspend == i + 1)
        {
            std::cout << "Suspending process..." << std::endl;
            SuspendThread(process_info[i].hThread);
        }
        if (process_to_realtime == i + 1)
        {
            std::cout << "Increasing priority..." << std::endl;
            SetPriorityClass(process_info[i].hProcess, REALTIME_PRIORITY_CLASS);
        }
    }

    std::cout << "Resuming process num=" << process_to_suspend << std::endl;
    ResumeThread(process_info[process_to_suspend].hThread);

    std::cout << "Killing thread num=" << process_to_kill << std::endl;
    TerminateProcess(process_info[process_to_kill].hThread, 0);

    for (int i = 0; i < number_of_processes; ++i)
    {
        WaitForSingleObject(process_info[i].hProcess, INFINITE);

        int time_elapsed;
        GetExitCodeProcess(process_info[i].hProcess, (LPDWORD)&time_elapsed);

        std::cout << "Process " << i << " : time elapsed - " << time_elapsed / 1e4 << " msec" << std::endl;
    }

    system("pause");
    return 0;
}
```

subprocess_main.cpp

```
#include <iostream>

#include <cmath>
#include <cstdlib>
#include <chrono>

#include <Windows.h>

const double EPS_VALUE = 0.01;
const double X_VALUE = 2;
const double B_VALUE = 3;

int main(int argc, char** argv)
{
    double b = 0,
        log_n = 0,
        x = 0,
        eps = 0,
        result = 0,
        step = 0;

    x = X_VALUE;
    b = B_VALUE;
    eps = EPS_VALUE;
    SYSTEMTIME sys_time;
    FILETIME temp_ft;
    ULARGE_INTEGER time_before,
                   time_after;

    GetSystemTime(&sys_time);
    SystemTimeToFileTime(&sys_time, &temp_ft);
    time_before.LowPart = temp_ft.dwLowDateTime;
    time_before.HighPart = temp_ft.dwHighDateTime;
    while (x <= b)
    {
        result = 0;
        log_n = log(x);
        for (int n(0); fabs(log(x) - result) > fabs(eps); n++)
        {
            step = (1 / (2 * n + 1));
            step *= pow((x + 1) / (x - 1), 2 * n + 1);
```

```
            step *= 2;
            result += step;
        }
        std::cout << "log_n(x): " << log_n << "\t" << "teyl.: " << result << std::endl;
        x += 0.1;
    }
    GetSystemTime(&sys_time);
    SystemTimeToFileTime(&sys_time, &temp_ft);
    time_after.LowPart = temp_ft.dwLowDateTime;
    time_after.HighPart = temp_ft.dwHighDateTime;
    int time = int(time_after.QuadPart - time_before.QuadPart);
    return time;
}
```

Результат виконання

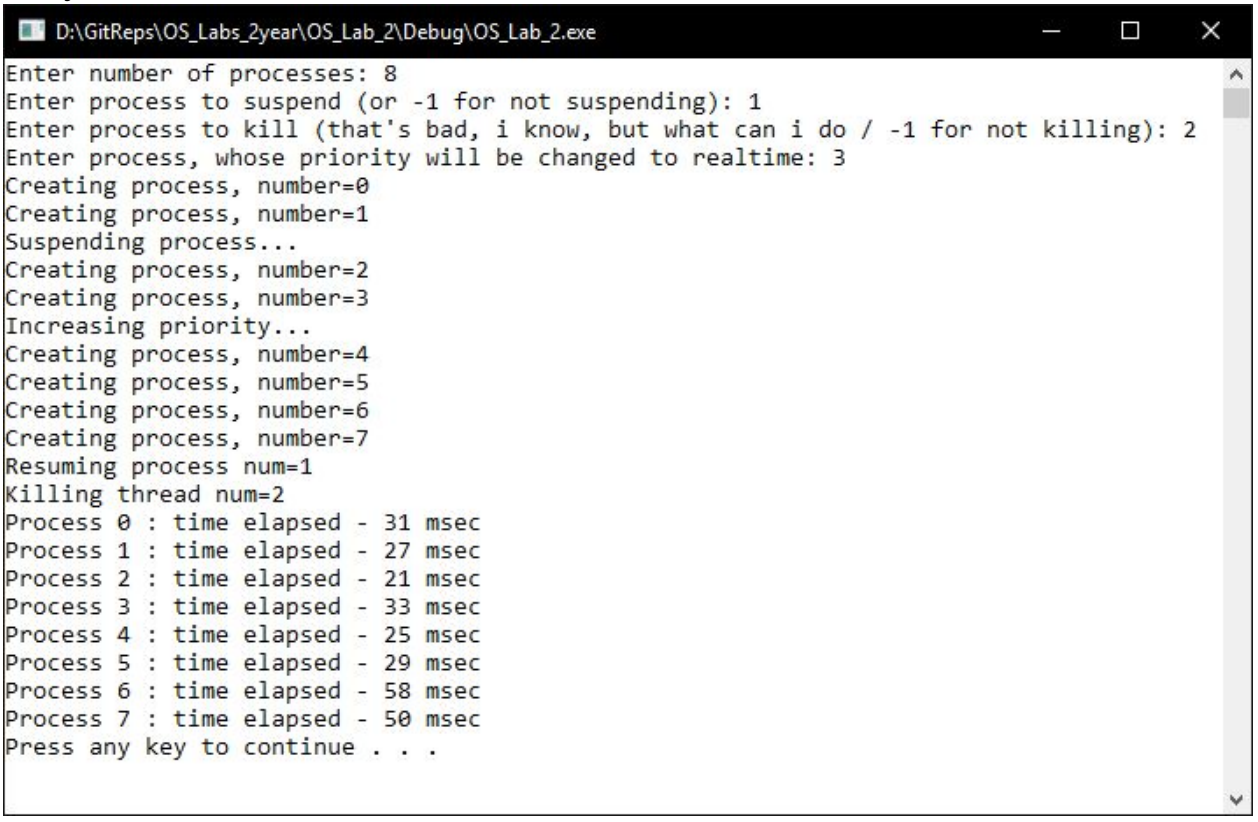

Рис.1. Результат виконання

Висновки

Під час роботи над цією лабораторною я навчився працювати з процессами за допомогою функцій WinAPI, а саме створювати, змінювати пріорітет, призупиняти, дізнаватися код закінчення та вбивати процесси.