Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей Кафедра информатики

Дисциплина: Операционные среды и системное программирование

ОТЧЕТ

к лабораторной работе № 4

на тему

**УПРАВЛЕНИЕ ПРОЦЕССАМИ И ПОТОКАМИ (WINDOWS) ПОРОЖДЕНИЕ, ЗАВЕРШЕНИЕ, ИЗМЕНЕНИЕ ПРИОРИТЕТОВ ПРОЦЕССОВ И ПОТОКОВ, ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ.**

Выполнил:

студент гр. 153503

Вергасов В.М.

Проверил:

Гриценко Н.Ю.

Минск 2023

**СОДЕРЖАНИЕ**

[1 Цели работы 3](#_Toc146883343)

[2 Краткие теоретические сведения 4](#_Toc146883344)

[3 Полученные результаты 11](#_Toc146883345)

[Выводы 12](#_Toc146883346)

[Список литературы 13](#_Toc146883347)

# 1 ЦЕЛИ РАБОТЫ

1. Изучение основных понятий и концепций управления процессами и потоками в операционной системе Windows.
2. Освоение методов порождения процессов и потоков в Windows.
3. Изучение возможностей и инструментов для завершения процессов и потоков.
4. Исследование принципов и алгоритмов изменения приоритетов процессов и потоков для оптимизации производительности.

**2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

В операционных системах *Windows* процессы и потоки являются основными единицами исполнения кода. Процесс представляет собой контейнер для одного или нескольких потоков, каждый из которых выполняет определенную часть кода.

Порождением процесса или потока называется создание нового экземпляра процесса или потока. В операционной системе Windows это осуществляется с помощью системных вызовов *CreateProcess* и *CreateThread* соответственно [1].

Завершение процесса или потока означает его удаление из системы. Это осуществляется с помощью системного вызова *TerminateProcess* или *ExitThread* соответственно. Важно отметить, что частое и неконтролируемое завершение процессов и потоков может негативно сказаться на стабильности и производительности системы.

Изменение приоритетов процессов и потоков позволяет управлять порядком выполнения задач в системе. В *Windows* приоритеты процессов и потоков могут быть установлены с использованием функций *SetPriorityClass* и *SetThreadPriority* соответственно. Правильно настроенные приоритеты могут оптимизировать использование системных ресурсов и повысить общую производительность системы.

Исследование эффективности процессов и потоков включает в себя анализ различных метрик, таких как время выполнения, использование памяти, загрузка процессора и т.д. Это помогает определить, насколько эффективно используются системные ресурсы и выявить возможные проблемы производительности.

# 3 ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ

Результат работы программы после запуска показан на рисунке 3.1.



Рисунок 3.1 – Результат работы программы после запуска

После нажатия клавиши *Start* появляются три линии, которые с разной скорости увеличиваются. Пример можно увидеть на рисунке 3.2.



Рисунок 3.2 – Нарисованные прямые в окне

После нажатия на кнопку *Pause* потоки приостанавливают выполнение и линии прекращают увеличиваться. Результат виден на рисунке 3.3. Исходный код программы отображен в листинге 1.



Рисунок 3.3 – приостановленная программа

Также остается возможность возобновить исполнение потоков, нажав кнопку *Start*. Т.е. после нажатия на кнопку прямые продолжать увеличиваться по циклу.

Листинг 1 **–** Код исходной программы:

#ifndef UNICODE

#define UNICODE

#define UNICODE\_WAS\_UNDEFINED

#endif

#include <Windows.h>

#include <TlHelp32.h>

#ifdef UNICODE\_WAS\_UNDEFINED

#undef UNICODE

#endif

#define ID\_BUTTON\_START 228

#define ID\_BUTTON\_PAUSE 229

#define ID\_THREAD\_1 1001

#define ID\_THREAD\_2 2002

#define ID\_THREAD\_3 3003

HWND hWND;

#define GENERATE\_DRAW\_FUNCTION(name, sleep\_time, \

bottom\_shift, left\_shift, right\_shift, R, G, B \

) \

DWORD WINAPI draw\_func\_##name(LPVOID lpParam) { \

PAINTSTRUCT ps; \

HPEN hPen; \

RECT rect; \

GetClientRect(hWND, &rect); \

rect.bottom = rect.bottom - bottom\_shift; \

rect.left = rect.left + left\_shift; \

rect.right = rect.right - right\_shift; \

POINT coords = \*((POINT\*)lpParam); \

HDC hdc = GetDC(hWND); \

hPen = CreatePen(PS\_SOLID, 5, RGB(R, G, B)); \

SelectObject(hdc, hPen); \

MoveToEx(hdc, coords.x, coords.y, NULL); \

while (true) { \

coords.y += 10; \

if (coords.y > 350) { \

coords.y = 5; \

MoveToEx(hdc, coords.x, coords.y, NULL); \

FillRect(hdc, &rect, (HBRUSH)(COLOR\_WINDOW + 1)); \

} \

else { \

LineTo(hdc, coords.x, coords.y); \

} \

Sleep(sleep\_time); \

DeleteObject(hPen); \

} \

return 0; \

}

HANDLE threads[3];

POINT coordinates[3] = {

{300, 5}, {600, 5}, {900, 5}

};

HWND start\_button, pause\_button;

bool pause\_press = true, threads\_suspend = false;

GENERATE\_DRAW\_FUNCTION(R, 20, 100, 290, 500, 255, 0, 0);

GENERATE\_DRAW\_FUNCTION(G, 40, 100, 590, 300, 0, 255, 5);

GENERATE\_DRAW\_FUNCTION(B, 60, 100, 690, 0, 0, 0, 255);

LRESULT CALLBACK WndProc(

HWND hwnd,

UINT msg,

WPARAM wParam,

LPARAM lParam

) {

switch (msg) {

case WM\_CREATE:

{

start\_button = CreateWindow(L"Button", L"Start",

WS\_CHILD | WS\_VISIBLE | WS\_BORDER, 50, 400, 100, 30,

hwnd, (HMENU)ID\_BUTTON\_START, NULL, NULL

);

pause\_button = CreateWindow(

L"Button", L"Pause",

WS\_CHILD | WS\_VISIBLE | WS\_BORDER, 175, 400, 100, 30,

hwnd, (HMENU)ID\_BUTTON\_PAUSE, NULL, NULL

);

break;

}

case WM\_COMMAND:

{

switch (LOWORD(wParam)) {

case ID\_BUTTON\_START:

{

if (pause\_press) {

if (threads\_suspend) {

for (size\_t i = 0; i < 3; ++i) {

ResumeThread(threads[i]);

}

} else {

threads[0] = CreateThread(NULL, NULL, draw\_func\_R, &coordinates[0], NULL, NULL);

threads[1] = CreateThread(NULL, NULL, draw\_func\_G, &coordinates[1], NULL, NULL);

threads[2] = CreateThread(NULL, NULL, draw\_func\_B, &coordinates[2], NULL, NULL);

threads\_suspend = true;

}

pause\_press = false;

}

break;

}

case ID\_BUTTON\_PAUSE:

{

if (!pause\_press) {

for (size\_t i = 0; i < 3; ++i) {

SuspendThread(threads[i]);

}

pause\_press = true;

}

break;

}

}

break;

}

case WM\_GETMINMAXINFO:

{

MINMAXINFO \*mmi = (MINMAXINFO \*)lParam;

mmi->ptMinTrackSize.x = 1000;

mmi->ptMinTrackSize.y = 500;

mmi->ptMaxTrackSize.x = 1000;

mmi->ptMaxTrackSize.y = 500;

break;

}

case WM\_CLOSE:

{

DestroyWindow(hwnd);

break;

}

default:

{

return DefWindowProc(hwnd, msg, wParam, lParam);

}

}

}

int WINAPI WinMain(

HINSTANCE hInstance,

HINSTANCE hPrevInstance,

LPSTR lpCmdLine,

int nCmdShow

) {

MSG msg;

HWND hwnd;

WNDCLASSEX wc;

wc.cbSize = sizeof(WNDCLASSEX);

wc.style = 0;

wc.lpfnWndProc = WndProc;

wc.cbClsExtra = 0;

wc.cbWndExtra = 0;

wc.hInstance = hInstance;

wc.hIcon = LoadIcon(NULL, IDI\_APPLICATION);

wc.hCursor = LoadCursor(NULL, IDC\_ARROW);

wc.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wc.lpszMenuName = NULL;

wc.lpszClassName = L"VadVergasovWindowClass";

wc.hIconSm = LoadIcon(NULL, IDI\_APPLICATION);

if (!RegisterClassEx(&wc)) {

MessageBox(NULL, L"Window registration is failed!", L"Error!", MB\_ICONWARNING | MB\_OK);

return 0;

}

hwnd = CreateWindowEx(

WS\_EX\_CLIENTEDGE,

L"VadVergasovWindowClass",

L"Operating Environment and System Programming, Lab 4",

WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, CW\_USEDEFAULT, 800, 600,

NULL, NULL, hInstance, NULL

);

if (hwnd == NULL) {

MessageBox(NULL, L"Window creation is failed!", L"Error!", MB\_ICONWARNING | MB\_OK);

return 0;

}

hWND = hwnd;

ShowWindow(hwnd, nCmdShow);

UpdateWindow(hwnd);

while (GetMessage(&msg, NULL, 0, 0)) {

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return msg.wParam;

}

# ВЫВОДЫ

В результате выполнения работ были освоены основные принципы и методы работы с процессами и потоками. Были изучены способы порождения, завершения и изменения приоритетов процессов и потоков.

Была сделана программа, которая рисует прямые в разных потоках на одной поверхности. Предусмотрена возможность приостанавливать потоки и возобновлять их выполнение.

# СПИСОК ЛИТЕРАТУРЫ

1. Programming reference for the Win32 API [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/en-us/windows/win32/api/ – Дата доступа: 21.09.2023.