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 № 5

на тему

**РЕЕСТР И ЖУРНАЛЫ (WINDOWS) ДОСТУП К РЕЕСТРУ WINDOWS. РАБОТА С ЖУРНАЛАМИ WINDOWS. ДРУГИЕ ВСПОМОГАТЕЛЬНЫЕ СРЕДСТВА УПРАВЛЕНИЯ.**

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 5](#_Toc146883345)

[Заключение 10](#_Toc146883346)

[Список литературы 11](#_Toc146883347)

# 1 ЦЕЛИ РАБОТЫ

1. Ознакомиться со структурой реестра *Windows* и его основными разделами.
2. Научиться работать с журналами *Windows*, такими как *Event Viewer* и *System Log*.
3. Изучить возможности использования сторонних программ для работы с реестром и журналами *Windows*.
4. Написать программу, проверяющий *TTL* и, если он не равен 65 – выставляющим его в значение 65, логи программы должны быть в журнале событий.

**2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Реестр Windows — это иерархическая база данных, содержащая информацию о настройках и конфигурациях системы, приложений и пользователей. Он состоит из нескольких разделов, таких как *HKEY\_CURRENT\_USER* (для текущего пользователя), *HKEY\_LOCAL\_MACHINE* (для всех пользователей) и *HKEY\_USERS* (для каждого пользователя).

Журналы *Windows* — это средства сбора и хранения информации о событиях, происходящих в системе. Они включают *Event Viewer* (для просмотра событий) и *System Log* (для отслеживания системных событий).

Вспомогательные средства управления Windows — это инструменты, предоставляющие доступ к настройкам и параметрам системы. К ним относятся Панель управления (для управления настройками системы) и Службы (для управления службами операционной системы).

Для работы с реестром *Windows* можно использовать стандартные инструменты, такие как *Regedit.exe* или сторонние программы, например, *Registry Editor* от *Microsoft*. Для работы с журналами *Windows* доступны такие инструменты, как *Event Viewer*, *System Log* и другие.

Для доступа к реестру *Windows* можно использовать *API*, такие как *RegOpenKeyEx*, *RegQueryValueEx*, *RegSetValueEx*, и другие функции [1].

Для доступа к журналам *Windows* можно использовать *API*, такие как функции *RegisterEventSource*, *ReportEvent*, и другие.

# 3 ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ

Результат работы программы после запуска, если на компьютере установлен *TTL* со значением 65 показан на рисунке 3.1.

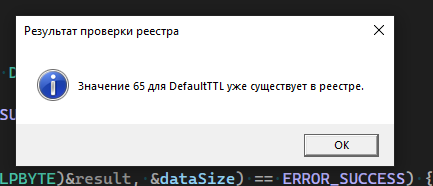


Рисунок 3.1 – Результат работы программы после запуска

Результат работы программы после запуска, если на компьютере не установлен *TTL* со значением 65 показан на рисунке 3.2.

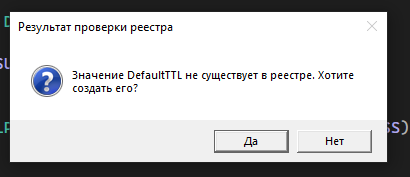


Рисунок 3.2 – Результат работы программы после запуска

После нажатия на кнопку «Да» программа пытается создать параметр в реестре со значением 65, если произойдет ошибка, то программа сделает окно с сообщением об ошибке, пример показан на рисунке 3.3.

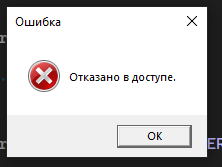


Рисунок 3.3 – Произошла ошибка

При успешном выполнении операции – будет сообщение, что параметр задан успешно, результат можно увидеть на рисунке 3.4.

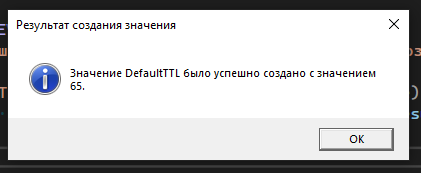


Рисунок 3.4 – Программа выполнилась успешно

Все логи программы пишутся в события *Windows*, пример событий можно увидеть на рисунке 3.5. Исходный код программы отображен в листинге 1.

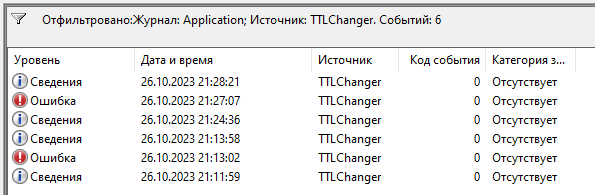


Рисунок 3.5 – Пример событий программы

Листинг 1 **–** Код исходной программы:

#include <windows.h>

#include <iostream>

#include <string>

#include <tchar.h>

HINSTANCE hInstance;

HWND hMainWindow;

std::wstring GetErrorMessage(DWORD errorCode) {

LPVOID errorMsg = nullptr;

FormatMessage(

FORMAT\_MESSAGE\_ALLOCATE\_BUFFER | FORMAT\_MESSAGE\_FROM\_SYSTEM,

NULL,

errorCode,

0,

(LPWSTR)&errorMsg,

0,

NULL

);

if (errorMsg != NULL) {

std::wstring message(static\_cast<LPCWSTR>(errorMsg));

LocalFree(errorMsg);

return message;

}

return L"Unknown Error";

}

void LogEvent(const std::wstring &message, WORD eventType) {

HANDLE hEventLog = RegisterEventSource(NULL, L"TTLChanger");

if (hEventLog != NULL) {

const wchar\_t \*messageStrings[1] = { message.c\_str() };

ReportEvent(hEventLog, eventType, 0, 0, NULL, 1, 0, messageStrings, NULL);

DeregisterEventSource(hEventLog);

}

}

bool GetRegistryValue(HKEY hKey, LPCWSTR subKey, LPCWSTR valueName, DWORD &result) {

HKEY hSubKey;

if (RegOpenKeyEx(hKey, subKey, 0, KEY\_READ, &hSubKey) == ERROR\_SUCCESS) {

DWORD valueType;

DWORD dataSize = sizeof(result);

if (RegQueryValueEx(hSubKey, valueName, NULL, &valueType, (LPBYTE)&result, &dataSize) == ERROR\_SUCCESS) {

RegCloseKey(hSubKey);

return true;

}

RegCloseKey(hSubKey);

}

return false;

}

void CreateValue(HWND hWnd, HKEY hKey, LPCWSTR subKey, LPCWSTR valueName, DWORD value) {

int result = MessageBox(hWnd, L"Значение DefaultTTL не равно 65. Хотите создать его?", L"Результат проверки реестра", MB\_ICONQUESTION | MB\_YESNO);

if (result == IDYES) {

HKEY hKey;

auto result\_code = RegOpenKeyEx(HKEY\_LOCAL\_MACHINE, subKey, 0, KEY\_WRITE, &hKey);

if (result\_code == ERROR\_SUCCESS) {

DWORD newValue = 65;

result\_code = RegSetValueEx(hKey, valueName, 0, REG\_DWORD, (const BYTE \*)&newValue, sizeof(newValue));

if (result\_code == ERROR\_SUCCESS) {

LogEvent(L"Настройка создана!", EVENTLOG\_SUCCESS);

MessageBox(hWnd, L"Значение DefaultTTL было успешно создано с значением 65.", L"Результат создания значения", MB\_ICONINFORMATION);

} else {

LogEvent(L"Ошибка при создании значения DefaultTTL: " + GetErrorMessage(result\_code), EVENTLOG\_ERROR\_TYPE);

MessageBox(hWnd, (L"Ошибка при создании значения DefaultTTL: " + GetErrorMessage(result\_code)).c\_str(), L"Результат создания значения", MB\_ICONERROR);

}

RegCloseKey(hKey);

} else {

LogEvent(L"Ошибка при создани: " + GetErrorMessage(result\_code), EVENTLOG\_ERROR\_TYPE);

MessageBox(hWnd, GetErrorMessage(result\_code).c\_str(), L"Ошибка", MB\_ICONERROR);

}

}

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam) {

LPCWSTR registryPath = L"SYSTEM\\CurrentControlSet\\Services\\Tcpip\\Parameters";

LPCWSTR valueName = L"DefaultTTL";

DWORD value;

switch (message) {

case WM\_CREATE:

if (GetRegistryValue(HKEY\_LOCAL\_MACHINE, registryPath, valueName, value)) {

if (value == 65) {

LogEvent(L"Настройка уже существует", EVENTLOG\_INFORMATION\_TYPE);

MessageBox(hWnd, L"Значение 65 для DefaultTTL уже существует в реестре.", L"Результат проверки реестра", MB\_ICONINFORMATION);

} else {

CreateValue(hWnd, HKEY\_LOCAL\_MACHINE, registryPath, valueName, 65);

}

} else {

CreateValue(hWnd, HKEY\_LOCAL\_MACHINE, registryPath, valueName, 65);

}

PostQuitMessage(0);

break;

case WM\_CLOSE:

DestroyWindow(hWnd);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

INT WINAPI WinMain(\_In\_ HINSTANCE hInst, \_In\_opt\_ HINSTANCE hPrevInstance, \_In\_ LPSTR lpCmdLine, \_In\_ INT nCmdShow) {

WNDCLASSEX wc = { sizeof(WNDCLASSEX), CS\_CLASSDC, WndProc, 0L, 0L, GetModuleHandle(NULL), NULL, NULL, NULL, NULL, \_T("RegistryCheckApp"), NULL };

RegisterClassEx(&wc);

hInstance = hInst;

hMainWindow = CreateWindow(wc.lpszClassName, L"Проверка реестра", WS\_OVERLAPPEDWINDOW, 100, 100, 400, 200, NULL, NULL, wc.hInstance, NULL);

ShowWindow(hMainWindow, nCmdShow);

UpdateWindow(hMainWindow);

MSG msg;

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

return 0;

}

# ЗАКЛЮЧЕНИЕ

Работа с реестром и журналами *Windows* является важной составляющей управления системой и обеспечения ее стабильности и надежности. Использование вспомогательных средств управления *Windows* позволяет оптимизировать и настраивать параметры системы, а также устранять возможные проблемы.

Была сделана программа, которая проверяет *TTL* компьютера и при необходимости меняет его. Предусмотрена обработка ошибок, а также запись в журнал событий.

# СПИСОК ЛИТЕРАТУРЫ

1. Programming reference for the Win32 API [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/en-us/windows/win32/api/ – Дата доступа: 10.10.2023.