Белорусский государственный университет

информатики и радиоэлектроники

Кафедра ПОИТ

Отчет по лабораторной работе №4

по дисциплине

«Операционные системы и системное программирование»

Выполнил:

Гладкий М.Г.

группа 851005

Проверил:

Деменковец Д.В.

Минск, 2020

**Исходный код программы**

Main.cpp

#undef UNICODE

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <windows.h>

#include <stdio.h>

#include <AclAPI.h>

#include <sddl.h>

#include <string>

using namespace std;

#define MAX\_LINE 50

#define EXIT 0

#define CREATE\_KEY 1

#define FIND\_REG\_KEY 2

#define READ\_KEY\_FLAG 3

#define NOTIFY\_KEY\_CHANGED 4

#define READ\_REG\_DWORD 5

#define WRITE\_REG\_DWORD 6

#define READ\_REG\_STRING 7

#define WRITE\_REG\_STRING 8

HANDLE hKeyChangedEvent;

HANDLE hKeyChangedThread;

BOOL FindKey(HKEY currentKey, LPCSTR keyName);

const char\* ConvertAceStrToStr(char\* source);

int ReadKeyFlags(HKEY currentKey);

LPCSTR ReadStrFromReg(HKEY key, LPCSTR subkey, LPCSTR valueName);

void WriteStrInReg(HKEY key, LPCSTR subkey, LPCSTR valueName, LPCSTR value);

void WriteDwordInReg(HKEY key, LPCSTR subkey, LPCSTR valueName, DWORD value);

DWORD ReadDwordFromReg(HKEY key, LPCSTR subkey, LPCSTR valueName);

void CreateRegKey(HKEY key, LPCSTR subkey);

DWORD WINAPI OnKeyChanged(LPVOID lpParam);

void CreateRegKey(HKEY key, LPCSTR subkey)

{

DWORD dwDisposition;

HKEY hKey;

DWORD Ret;

Ret =

RegCreateKeyEx(key, subkey, 0,

NULL, REG\_OPTION\_NON\_VOLATILE, KEY\_ALL\_ACCESS, NULL, &hKey, &dwDisposition);

if (Ret != ERROR\_SUCCESS)

{

printf("Error opening (or creating new) key.\n");

}

else

printf("Success opening (or creating new) key.\n");

RegCloseKey(hKey);

}

int ReadKeyFlags(HKEY currentKey)

{

int isSuccess = 1;

DWORD securityDescriptorSize;

DWORD subkeysNumber;

RegQueryInfoKey(currentKey, NULL, 0, NULL, &subkeysNumber,

NULL, NULL, NULL, NULL, NULL, &securityDescriptorSize, NULL);

char\* buffer = new char[securityDescriptorSize];

DWORD result;

result = RegGetKeySecurity(currentKey, DACL\_SECURITY\_INFORMATION, buffer, &securityDescriptorSize);

if (result != ERROR\_SUCCESS)

{

printf("Can't get security.\n");

isSuccess = 0;

}

else

{

SECURITY\_DESCRIPTOR\* security = reinterpret\_cast<SECURITY\_DESCRIPTOR\*>(buffer);

LPSTR strDacl;

ConvertSecurityDescriptorToStringSecurityDescriptor(security, SDDL\_REVISION\_1, DACL\_SECURITY\_INFORMATION, &strDacl, NULL);

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

printf("%s\n", ConvertAceStrToStr(strDacl));

}

return isSuccess;

}

void NotifyKeyChanged(HKEY currentKey)

{

hKeyChangedEvent = CreateEvent(NULL, TRUE, FALSE, "KeyChanged");

DWORD threadId;

hKeyChangedThread = CreateThread(NULL, 0, OnKeyChanged, NULL, 0, &threadId);

RegNotifyChangeKeyValue(currentKey, TRUE, REG\_NOTIFY\_CHANGE\_LAST\_SET, hKeyChangedEvent, TRUE);

}

DWORD ReadDwordFromReg(HKEY key, LPCSTR subkey, LPCSTR valueName)

{

DWORD result;

HKEY openedKey;

result = RegOpenKeyEx(key, subkey, 0, KEY\_READ, &openedKey);

if (result != ERROR\_SUCCESS)

{

printf("Can't open key.\n");

return -1;

}

DWORD buffer;

DWORD len = sizeof(buffer);

result = RegQueryValueEx(openedKey, valueName, NULL, NULL, (BYTE\*)&buffer, &len);

if (result != ERROR\_SUCCESS)

{

printf("Can't read value.\n");

return -1;

}

RegCloseKey(openedKey);

return buffer;

}

void WriteDwordInReg(HKEY key, LPCSTR subkey, LPCSTR valueName, DWORD value)

{

HKEY openedKey;

DWORD result;

result = RegOpenKeyEx(key, subkey, 0, KEY\_WRITE, &openedKey);

if (result != ERROR\_SUCCESS)

{

printf("Can't open key.\n");

}

result = RegSetValueEx(openedKey, valueName, 0, REG\_DWORD, (BYTE\*)&value, sizeof(DWORD));

if (result != ERROR\_SUCCESS)

{

printf("Can't write to registry.\n");

}

RegCloseKey(openedKey);

printf("Ready!\n");

}

LPCSTR ReadStrFromReg(HKEY key, LPCSTR subkey, LPCSTR valueName)

{

DWORD result;

HKEY openedKey;

result = RegOpenKeyEx(key, subkey, 0, KEY\_READ, &openedKey);

if (result != ERROR\_SUCCESS)

{

printf("Can't open key.\n");

return NULL;

}

char\* buffer = new char[MAX\_LINE];

DWORD len = MAX\_LINE;

result = RegQueryValueEx(openedKey, valueName, NULL, NULL, (BYTE\*)buffer, &len);

if (result != ERROR\_SUCCESS)

{

printf("Can't read value.\n");

return NULL;

}

RegCloseKey(openedKey);

return buffer;;

}

void WriteStrInReg(HKEY key, LPCSTR subkey, LPCSTR valueName, LPCSTR value)

{

HKEY openedKey;

DWORD result;

result = RegOpenKeyEx(key, subkey, 0, KEY\_WRITE, &openedKey);

if (result != ERROR\_SUCCESS)

printf("Can't create key.\n");

result = RegSetValueEx(openedKey, valueName, 0, REG\_SZ, (BYTE\*)value, strlen(value));

if (result != ERROR\_SUCCESS)

printf("Can't write in registry.\n");

RegCloseKey(openedKey);

printf("Ready!\n");

}

BOOL FindKey(HKEY currentKey, LPCSTR keyName)

{

DWORD subkeysAmount;

DWORD maxSubkeyLen, currentSubkeyLen;

BOOL result;

RegQueryInfoKey(currentKey, NULL, 0, NULL, &subkeysAmount, &maxSubkeyLen,

NULL, NULL, NULL, NULL, NULL, NULL);

char\* bufferName = new char[maxSubkeyLen];

for (int i = 0; i < subkeysAmount; i++)

{

currentSubkeyLen = maxSubkeyLen;

result = RegEnumKeyEx(currentKey, i, bufferName, &currentSubkeyLen, NULL, NULL, NULL, NULL);

if (result == ERROR\_SUCCESS)

{

if (!strcmp(bufferName, keyName))

{

return TRUE;

}

HKEY innerKey;

result = RegOpenKey(currentKey, bufferName, &innerKey);

if (result == ERROR\_SUCCESS)

{

result = FindKey(innerKey, keyName);

if (result)

{

RegCloseKey(innerKey);

return result;

}

}

RegCloseKey(innerKey);

}

}

return FALSE;

}

const char\* ConvertAceStrToStr(char\* source)

{

string strSource(source);

string\* strKeyAccess = new string("");

int strLen = strSource.length();

int i = 0;

int semicolonCounter = 0;

while (i < strLen)

{

if (source[i] == ')')

{

semicolonCounter = 0;

}

if (source[i] == ';')

{

semicolonCounter++;

}

if (semicolonCounter == 2)

{

int start = i + 1;

do

{

i++;

} while (source[i] != ';');

strKeyAccess->append(strSource.substr(start, i - start));

semicolonCounter++;

}

if (semicolonCounter == 5)

{

int start = i + 1;

while (source[i] != ')')

{

i++;

}

strKeyAccess->append(" <-" + strSource.substr(start, i - start) + "\n");

semicolonCounter = 0;

}

i++;

}

return strKeyAccess->c\_str();

}

void CloseEvents()

{

CloseHandle(hKeyChangedEvent);

}

DWORD WINAPI OnKeyChanged(LPVOID lpParam)

{

WaitForSingleObject(hKeyChangedEvent, INFINITE);

printf("Key changed.\n");

return 0;

}

int main()

{

string start = "0 - Close\n1 - Create key\n2 - Find key\n3 - Read key flags\n4 - Key change notify\n5 - Read dword value\n6 - Write dword value\n7 - Read string value\n8 - Write string value\n";

printf("%s\n", start.c\_str());

int isContinue = 1;

char buffer[1024];

char secondBuffer[1024];

DWORD dwValue;

DWORD dwRes;

LPCSTR strRes;

char subkey[1024];

while (isContinue)

{

int comm;

scanf("%d", &comm);

switch (comm)

{

case EXIT:

isContinue = 0;

CloseEvents();

break;

case CREATE\_KEY:

printf("Enter key name:\n");

scanf("%s", buffer);

CreateRegKey(HKEY\_CURRENT\_USER, buffer);

break;

case FIND\_REG\_KEY:

printf("Enter key to find:\n");

scanf("%s", buffer);

if (FindKey(HKEY\_CURRENT\_USER, buffer))

printf("Found!\n");

else

printf("Not found.\n");

break;

case READ\_KEY\_FLAG:

{

HKEY currKey;

printf("Enter key name:\n");

scanf("%s", buffer);

dwRes = RegOpenKey(HKEY\_CURRENT\_USER, buffer, &currKey);

if (dwRes != ERROR\_SUCCESS)

printf("Can't open key.\n");

else

{

ReadKeyFlags(currKey);

RegCloseKey(currKey);

}

}

break;

case NOTIFY\_KEY\_CHANGED:

{

printf("Enter key name:\n");

scanf("%s", buffer);

HKEY currKey;

dwRes = RegOpenKey(HKEY\_CURRENT\_USER, buffer, &currKey);

if (dwRes != ERROR\_SUCCESS)

printf("Can't open key.\n");

else

NotifyKeyChanged(currKey);

}

break;

case READ\_REG\_DWORD:

printf("Enter subkey name:\n");

scanf("%s", subkey);

printf("Enter dword name:\n");

scanf("%s", buffer);

dwRes = ReadDwordFromReg(HKEY\_CURRENT\_USER, subkey, buffer);

if (dwRes != -1)

printf("Value: %d\n", dwRes);

break;

case WRITE\_REG\_DWORD:

printf("Enter subkey name:\n");

scanf("%s", subkey);

printf("Enter dword name:\n");

scanf("%s", buffer);

printf("Enter value:\n");

scanf("%d", &dwValue);

WriteDwordInReg(HKEY\_CURRENT\_USER, subkey, buffer, dwValue);

break;

case READ\_REG\_STRING:

printf("Enter subkey name:\n");

scanf("%s", subkey);

printf("Enter string name:\n");

scanf("%s", buffer);

strRes = ReadStrFromReg(HKEY\_CURRENT\_USER, subkey, buffer);

if (strRes != NULL)

printf("Value: %s\n", strRes);

break;

case WRITE\_REG\_STRING:

printf("Enter subkey name:\n");

scanf("%s", subkey);

printf("Enter string name:\n");

scanf("%s", buffer);

printf("Enter value:\n");

scanf("%s", secondBuffer);

WriteStrInReg(HKEY\_CURRENT\_USER, subkey, buffer, secondBuffer);

break;

default:

break;

}

}

return 0;

}

**Скриншоты выполнения программы**











