УО «Белорусский государственный университет информатики и

радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №4

по предмету

Операционные системы и системное программирование

**Выполнил Каширский А.Е.**

**Проверил Деменковец Д. В.**

Группа:

**8**51005

Минск 2020

Код программы

#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\_STRING\_LENGTH 100

#define HELP\_MSG "1 - Find key\n2 - Read key string value\n3 - Write string value\n4 - Show subkeys\n5 - Read key flags\n6 - Create new key\n7 - Close programm\n"

#define INPUT\_KEY\_MSG "Enter key name: "

#define INPUT\_VALUE\_NAME\_MSG "Enter value name: "

#define INPUT\_VALUE\_MSG "Enter new value: "

#define ERROR\_MSG\_CANNOT\_OPEN\_KEY "Can not open key!"

#define ERROR\_MSG\_CANNOT\_READ\_VALUE "Can not read value!"

#define ERROR\_MSG\_CANNOT\_WRITE\_REG "Can not write registry!"

#define FIND\_REG\_KEY 1

#define READ\_REG\_STRING 2

#define WRITE\_REG\_STRING 3

#define SHOW\_SUBKEYS 4

#define READ\_KEY\_FLAG 5

#define CREATE\_REG\_KEY 6

#define CLOSE\_PROGRAMM 7

int findSubKey(HKEY, LPCSTR);

LPCSTR readStringValue(HKEY, LPCSTR, LPCSTR);

DWORD writeStringValue(HKEY, LPCSTR, LPCSTR, LPCSTR);

void showSubKeys(HKEY);

void showKeyFlags(HKEY);

int createKeyByName(HKEY, LPCSTR);

int findSubKey(HKEY key, LPCSTR subKeyName)

{

DWORD subKeysAmount, subKeyLen = 4096, currentSubKeyLen, result;

RegQueryInfoKey(key, NULL, 0, NULL, &subKeysAmount, &subKeyLen,

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

char\* bufferName = new char[subKeyLen];

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

{

currentSubKeyLen = 4096;

result = RegEnumKeyEx(key, i, bufferName, &currentSubKeyLen, NULL, NULL, NULL, NULL);

if (result == ERROR\_SUCCESS)

{

if (!strcmp(bufferName, subKeyName))

{

return 1;

}

HKEY subKey;

result = RegOpenKey(key, bufferName, &subKey);

if (result == ERROR\_SUCCESS)

{

result = findSubKey(subKey, subKeyName);

if (result)

{

RegCloseKey(subKey);

return result;

}

}

RegCloseKey(subKey);

}

}

return 0;

}

LPCSTR readStringValue(HKEY key, LPCSTR subKey, LPCSTR valueName)

{

HKEY openedKey;

char\* stringValue = new char[MAX\_STRING\_LENGTH];

if (ERROR\_SUCCESS != RegOpenKeyEx(key, subKey, NULL, KEY\_READ, &openedKey))

{

printf(ERROR\_MSG\_CANNOT\_OPEN\_KEY);

return NULL;

}

DWORD length = MAX\_STRING\_LENGTH;

if (ERROR\_SUCCESS != RegQueryValueEx(openedKey, valueName, NULL, NULL, (BYTE\*)stringValue, &length))

{

printf(ERROR\_MSG\_CANNOT\_READ\_VALUE);

return NULL;

}

RegCloseKey(openedKey);

return stringValue;

}

DWORD writeStringValue(HKEY key, LPCSTR subKey, LPCSTR valueName, LPCSTR value)

{

HKEY openedKey;

if (ERROR\_SUCCESS != RegOpenKeyEx(key, subKey, NULL, KEY\_ALL\_ACCESS, &openedKey))

{

printf(ERROR\_MSG\_CANNOT\_OPEN\_KEY);

return -1;

}

if (ERROR\_SUCCESS != RegSetValueEx(openedKey, valueName, 0, REG\_SZ, (BYTE\*)value, strlen(value)))

{

printf(ERROR\_MSG\_CANNOT\_WRITE\_REG);

return -1;

}

RegCloseKey(openedKey);

return 0;

}

void showSubKeys(HKEY key)

{

DWORD subKeysAmount;

DWORD maxSubKeyLength;

char\* subKeyName = new char[MAX\_STRING\_LENGTH];

RegQueryInfoKey(key, NULL, NULL, NULL, &subKeysAmount, &maxSubKeyLength, NULL, NULL, NULL, NULL, NULL, NULL);

maxSubKeyLength = MAX\_STRING\_LENGTH; // hot fix for max length because RegQueryInfo doesn't work correctly((((((

if (subKeysAmount > 0)

{

printf("Subkeys: \n");

}

else

{

printf("There are no subkeys for this key!\n");

}

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

{

maxSubKeyLength = MAX\_STRING\_LENGTH;

if (ERROR\_SUCCESS != RegEnumKeyEx(key, i, subKeyName, &maxSubKeyLength, NULL, NULL, NULL, NULL))

{

printf("Can not get subkey\n");

}

else

{

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

}

}

}

int getSecurityFlags(HKEY key, DWORD securityDescriptorSize, long securityCode, char\* msg)

{

bool isOK = true;

char\* securityInformation = new char[securityDescriptorSize];

if (ERROR\_SUCCESS != RegGetKeySecurity(key, securityCode, securityInformation, &securityDescriptorSize))

{

printf("Can not get security\n");

isOK = false;

}

else

{

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

LPSTR strSecurity;

ConvertSecurityDescriptorToStringSecurityDescriptor(security, SDDL\_REVISION\_1, securityCode, &strSecurity, NULL);

printf(msg);

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

}

return isOK;

}

int readKeyFlags(HKEY key)

{

int isSuccess = 1;

DWORD securityDescriptorSize;

RegQueryInfoKey(key, NULL, 0, NULL, NULL,

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

char msgOwner[] = "Owner security flags\n";

char msgGroup[] = "Group security flags\n";

char msgDACL[] = "DACL security flags\n";

char msgSACL[] = "SACL security flags\n";

getSecurityFlags(key, securityDescriptorSize, OWNER\_SECURITY\_INFORMATION, msgOwner);

getSecurityFlags(key, securityDescriptorSize, GROUP\_SECURITY\_INFORMATION, msgGroup);

getSecurityFlags(key, securityDescriptorSize, DACL\_SECURITY\_INFORMATION, msgDACL);

getSecurityFlags(key, securityDescriptorSize, SACL\_SECURITY\_INFORMATION, msgSACL);

return isSuccess;

}

int createKeyByName(HKEY key, LPCSTR keyName)

{

DWORD disposition;

HKEY openedKey;

if (ERROR\_SUCCESS != RegCreateKeyEx(HKEY\_CURRENT\_USER, keyName, NULL, REG\_OPTION\_NON\_VOLATILE, NULL, KEY\_ALL\_ACCESS, NULL, &openedKey, &disposition))

{

printf("Can not create key\n");

return -1;

}

else

{

if (REG\_OPENED\_EXISTING\_KEY == disposition)

{

printf("Such key already exists!\n");

}

else

{

printf("Key sucessfully created\n");

}

}

RegCloseKey(openedKey);

}

int main()

{

char keyName[MAX\_STRING\_LENGTH];

char valueName[MAX\_STRING\_LENGTH];

char newValue[MAX\_STRING\_LENGTH];

int command;

bool isNotClosing = true;

printf(HELP\_MSG);

while (isNotClosing)

{

scanf("%d", &command);

switch (command)

{

case FIND\_REG\_KEY:

{

printf(INPUT\_KEY\_MSG);

scanf("%s", keyName);

if (findSubKey(HKEY\_CURRENT\_USER, keyName))

{

printf("Key exists!\n");

}

else

{

printf("Key doesn't exist!\n");

}

}

break;

case READ\_REG\_STRING:

{

printf(INPUT\_KEY\_MSG);

scanf("%s", keyName);

printf(INPUT\_VALUE\_NAME\_MSG);

scanf("%s", valueName);

LPCSTR strResult = readStringValue(HKEY\_CURRENT\_USER, keyName, valueName);

if (strResult != NULL)

{

printf("String value is: %s\n", strResult);

}

}

break;

case WRITE\_REG\_STRING:

{

printf(INPUT\_KEY\_MSG);

scanf("%s", keyName);

printf(INPUT\_VALUE\_NAME\_MSG);

scanf("%s", valueName);

printf(INPUT\_VALUE\_MSG);

scanf("%s", newValue);

writeStringValue(HKEY\_CURRENT\_USER, keyName, valueName, newValue);

}

break;

case SHOW\_SUBKEYS:

{

HKEY key;

printf(INPUT\_KEY\_MSG);

scanf("%s", keyName);

if (ERROR\_SUCCESS != RegOpenKey(HKEY\_CURRENT\_USER, keyName, &key))

{

printf(ERROR\_MSG\_CANNOT\_OPEN\_KEY);

}

else

{

showSubKeys(key);

RegCloseKey(key);

}

}

break;

case READ\_KEY\_FLAG:

{

HKEY key;

printf(INPUT\_KEY\_MSG);

scanf("%s", keyName);

if (ERROR\_SUCCESS != RegOpenKey(HKEY\_CURRENT\_USER, keyName, &key))

{

printf(ERROR\_MSG\_CANNOT\_OPEN\_KEY);

}

else

{

readKeyFlags(key);

RegCloseKey(key);

}

}

break;

case CREATE\_REG\_KEY:

{

printf(INPUT\_KEY\_MSG);

scanf("%s", keyName);

createKeyByName(HKEY\_CURRENT\_USER, keyName);

}

break;

case CLOSE\_PROGRAMM:

{

isNotClosing = false;

}

break;

}

}

return 0;

}

Скриншоты работы программы



