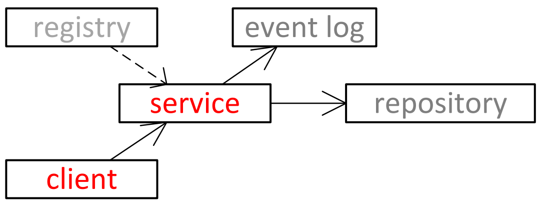
PB173 - Systémové programování Windows

Úkol 11



Rozšiřte projekt Service tak:

- aby mohl komunikovat s klientem běžícím s oprávněními standardního uživatele.

Nutné úpravy:

- Aby stejnou konfiguraci viděl klient i služba, je nutné ji přesunout z HKCU do HKLM (v registrech i v kódu)

Tipy:

- File Mapping Security and Access Rights http://msdn.microsoft.com/en-us/library/windows/desktop/aa366559(v=vs.85).aspx
- Synchronization Object Security and Access Rights http://msdn.microsoft.com/en-us/library/windows/desktop/ms686670(v=vs.85).aspx
- Služba zdroje vytváří, klient zdroje otevírá.

Příloha 1 - Creating a Security Descriptor for a New Object in C++

Zdroj: http://msdn.microsoft.com/en-us/library/windows/desktop/aa446595(v=vs.85).aspx

```
#pragma comment(lib, "advapi32.lib")

#include <windows.h>
#include <stdio.h>
#include <aclapi.h>
#include <tchar.h>

void main()
{

    DWORD dwRes, dwDisposition;
    PSID pEveryoneSID = NULL, pAdminSID = NULL;
    PACL pACL = NULL;
    PSECURITY_DESCRIPTOR pSD = NULL;
    EXPLICIT_ACCESS ea[2];
    SID_IDENTIFIER_AUTHORITY SIDAuthWorld =
            SECURITY_WORLD_SID_AUTHORITY;
    SID_IDENTIFIER_AUTHORITY SIDAuthNT = SECURITY_NT_AUTHORITY;
    SECURITY_ATTRIBUTES sa;
    LONG lRes;
    HKEY hkSub = NULL;
```

```
// Create a well-known SID for the Everyone group.
if(!AllocateAndInitializeSid(&SIDAuthWorld, 1,
                 SECURITY_WORLD_RID,
                 0, 0, 0, 0, 0, 0, 0,
                 &pEveryoneSID))
{
    _tprintf(_T("AllocateAndInitializeSid Error %u\n"), GetLastError());
    goto Cleanup;
}

// Initialize an EXPLICIT_ACCESS structure for an ACE.
// The ACE will allow Everyone read access to the key.
ZeroMemory(&ea, 2 * sizeof(EXPLICIT_ACCESS));
ea[0].grfAccessPermissions = KEY_READ;
ea[0].grfAccessMode = SET_ACCESS;
ea[0].grfInheritance= NO_INHERITANCE;
ea[0].Trustee.TrusteeForm = TRUSTEE_IS_SID;
ea[0].Trustee.TrusteeType = TRUSTEE_IS_WELL_KNOWN_GROUP;
ea[0].Trustee.ptstrName  = (LPTSTR) pEveryoneSID;

// Create a SID for the BUILTIN\Administrators group.
if(! AllocateAndInitializeSid(&SIDAuthNT, 2,
                 SECURITY_BUILTIN_DOMAIN_RID,
                 DOMAIN_ALIAS_RID_ADMINS,
                 0, 0, 0, 0, 0, 0,
                 &pAdminSID))
{
    _tprintf(_T("AllocateAndInitializeSid Error %u\n"), GetLastError());
    goto Cleanup;
}

// Initialize an EXPLICIT_ACCESS structure for an ACE.
// The ACE will allow the Administrators group full access to
// the key.
ea[1].grfAccessPermissions = KEY_ALL_ACCESS;
ea[1].grfAccessMode = SET_ACCESS;
ea[1].grfInheritance= NO_INHERITANCE;
ea[1].Trustee.TrusteeForm = TRUSTEE_IS_SID;
ea[1].Trustee.TrusteeType = TRUSTEE_IS_GROUP;
ea[1].Trustee.ptstrName  = (LPTSTR) pAdminSID;

// Create a new ACL that contains the new ACEs.
dwRes = SetEntriesInAcl(2, ea, NULL, &pACL);
if (ERROR_SUCCESS != dwRes)
{
    _tprintf(_T("SetEntriesInAcl Error %u\n"), GetLastError());
    goto Cleanup;
}

// Initialize a security descriptor.
pSD = (PSECURITY_DESCRIPTOR) LocalAlloc(LPTR,
                         SECURITY_DESCRIPTOR_MIN_LENGTH);
if (NULL == pSD)
{
    _tprintf(_T("LocalAlloc Error %u\n"), GetLastError());
    goto Cleanup;
}

if (!InitializeSecurityDescriptor(pSD,
        SECURITY_DESCRIPTOR_REVISION))
{
    _tprintf(_T("InitializeSecurityDescriptor Error %u\n"),
                            GetLastError());
    goto Cleanup;
}

// Add the ACL to the security descriptor.
if (!SetSecurityDescriptorDacl(pSD,
        TRUE,     // bDaclPresent flag
        pACL,
        FALSE))   // not a default DACL
{
    _tprintf(_T("SetSecurityDescriptorDacl Error %u\n"),
            GetLastError());
    goto Cleanup;
}
```

```
    // Initialize a security attributes structure.
    sa.nLength = sizeof (SECURITY_ATTRIBUTES);
    sa.lpSecurityDescriptor = pSD;
    sa.bInheritHandle = FALSE;

    // Use the security attributes to set the security descriptor
    // when you create a key.
    lRes = RegCreateKeyEx(HKEY_CURRENT_USER, _T("mykey"), 0, _T(""), 0,
            KEY_READ | KEY_WRITE, &sa, &hkSub, &dwDisposition);
    _tprintf(_T("RegCreateKeyEx result %u\n"), lRes );

Cleanup:

    if (pEveryoneSID)
        FreeSid(pEveryoneSID);
    if (pAdminSID)
        FreeSid(pAdminSID);
    if (pACL)
        LocalFree(pACL);
    if (pSD)
        LocalFree(pSD);
    if (hkSub)
        RegCloseKey(hkSub);

    return;
}
```

Příloha 2 - Creating a DACL

Zdroj: http://msdn.microsoft.com/en-us/library/windows/desktop/ms717798(v=vs.85).aspx

```
#define _WIN32_WINNT 0x0500

#include <windows.h>
#include <sddl.h>
#include <stdio.h>

#pragma comment(lib, "advapi32.lib")

BOOL CreateMyDACL(SECURITY_ATTRIBUTES *);

void main()
{
     SECURITY_ATTRIBUTES  sa;

     sa.nLength = sizeof(SECURITY_ATTRIBUTES);
     sa.bInheritHandle = FALSE;

     // Call function to set the DACL. The DACL is set in the SECURITY_ATTRIBUTES
     // lpSecurityDescriptor member.
     if (!CreateMyDACL(&sa))
     {  // Error encountered; generate message and exit.
         printf("Failed CreateMyDACL\n");
         exit(1);
     }

     // Use the updated SECURITY_ATTRIBUTES to specify security attributes for
     // securable objects. This example uses security attributes during
     // creation of a new directory.
     if (0 == CreateDirectory(TEXT("C:\\MyFolder"), &sa))
     {   // Error encountered; generate message and exit.
         printf("Failed CreateDirectory\n");
         exit(1);
     }

     // Free the memory allocated for the SECURITY_DESCRIPTOR.
     if (NULL != LocalFree(sa.lpSecurityDescriptor))
     {   // Error encountered; generate message and exit.
         printf("Failed LocalFree\n");
         exit(1);
     }
}


// CreateMyDACL.
//    Create a security descriptor that contains the DACL you want.
//    This function uses SDDL to make Deny and Allow ACEs.
//
// Parameter:
//    SECURITY_ATTRIBUTES * pSA
//    Pointer to a SECURITY_ATTRIBUTES structure. It is your responsibility
//    to properly initialize the structure and to free the structure's
//    lpSecurityDescriptor member when you have finished using it. To free the structure's
//    lpSecurityDescriptor member, call the LocalFree function.
//
// Return value:
//    FALSE if the address to the structure is NULL. Otherwise, this function
//    returns the value from the ConvertStringSecurityDescriptorToSecurityDescriptor
//    function.
BOOL CreateMyDACL(SECURITY_ATTRIBUTES * pSA)
{
     // Define the SDDL for the DACL. This example sets the following access:
     //     Built-in guests are denied all access.
     //     Anonymous logon is denied all access.
     //     Authenticated users are allowed
     //     read/write/execute access.
     //     Administrators are allowed full control.
     // Modify these values as needed to generate the proper DACL for your application.

     TCHAR * szSD = TEXT("D:")       // Discretionary ACL
        TEXT("(D;OICI;GA;;;BG)")     // Deny access to built-in guests
        TEXT("(D;OICI;GA;;;AN)")     // Deny access to anonymous logon
        TEXT("(A;OICI;GRGWGX;;;AU)") // Allow read/write/execute to authenticated users
        TEXT("(A;OICI;GA;;;BA)");    // Allow full control to administrators

    if (NULL == pSA)
        return FALSE;

     return ConvertStringSecurityDescriptorToSecurityDescriptor(
                szSD,
                SDDL_REVISION_1,
                &(pSA->lpSecurityDescriptor),
                NULL);
}
```