Metody připojování periferií

BI-MPP

Přednáška 13

Ing. Miroslav Skrbek, Ph.D.

Katedra číslicového návrhu
Fakulta informačních technologií
České vysoké učení technické v Praze

ZS2021/22

Evropský sociální fond
Praha & EU: Investujeme do vaší budoucnosti

Agenda

- Přístup k periferiím z virtuálních strojů

Přístup k periferiím z virtuálních strojů

Nativní metody v technologii .NET

Nativní metody jsou napsané v jazyce C nebo C++ a zapouzdřené do DLL knihoven. Jazyky jako je Visual Basic nebo C# mají rozšíření pro volání nativních metod.

Visual Basic

```
Imports System.Runtime.InteropServices

<DllImport("c:\my.dll", CallingConvention:=CallingConvention.Cdecl)> _
Private Shared Function add(a As Integer, b As Integer) As Integer
End Sub

…
c = add(10, 10)
```

C#

```
using System.Runtime.InteropServices;

[DllImport("c:\my.dll")]
private static extern int add(int a, int b);

…
c = add(10, 10)
```

Nativní DLL knihovna pro .NET

použije se C jmenná konvence (naming convention) uvnitř C++

funkce se exportuje z DLL

```
extern "C" {
	__declspec(dllexport) int add(int a, int b) {
		return a + b;
	}
}
```

Nativní metody v technologii JAVA

- Java používá pro volání nativních funkcí speciální rozhraní zvané Java Native Interface (JNI)
- Kromě volání nativních metod, rozhraní dovoluje manipulaci s javovskými objekty a volání metod
- V Linuxu jsou nativní metody zapouzdřeny do sdílených knihoven (lib*.so)
- Ve Windows jsou nativní metody zapouzdřeny do DLL knihoven
- Java poskytuje nástroje pro automatické generování hlavičkových souborů pro jazyk C a C++

Deklarace nativních metod v Javě

```
package mypackage;
class MyClass {
	{	System.loadLibrary("calc");
	}
	public native int add(int a, int b);
}
```

Generování hlavičkových souborů

```
javah –classpath . –jni mypackage.MyClass
```

Předpokládá se, že soubor MyClass.class je umístěn v adresáři ./mypackage

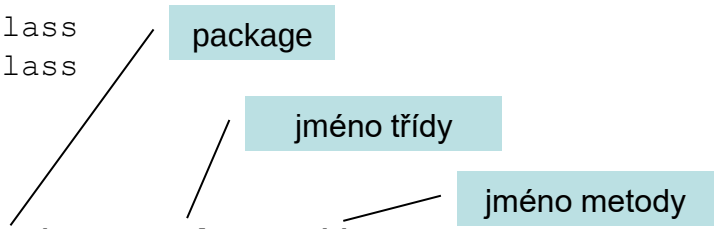
Hlavičkový soubor mypackage_MyClass.h

```
#include <jni.h>
#ifndef _Included_mypackage_MyClass
#define _Included_mypackage_MyClass
#ifdef __cplusplus
extern "C" {
#endif

JNIEXPORT jint JNICALL Java_mypackage_MyClass_add
          (JNIEnv *, jobject, jint, jint);

#ifdef __cplusplus
}

#endif
#endif
```



Implementace (calc.dll)

```
#include "mypackage_MyClass.h"

JNIEXPORT jint JNICALL Java_mypackage_MyClass_add
  (JNIEnv *env, jobject self, jint a, jint b) {
      return a + b;
}
```

reprezentuje objekt, který obsahuje nativní metodu

Ukazatel na datovou strukturu, která obsahuje ukazatele na funkce JNI

Řetězce v JNI

```
JNIEXPORT jint JNICALL Java_mypackage_MyClass_search
          (JNIEnv *env, jobject self, jchar c, jstring s) {

          int i, len;

          jchar* str = (*env)->GetStringChars(env, s, NULL);

          if (str == NULL) {
                    return NULL;
          }

          len = (*env)->GetStringLength(env, s);

          for(i = 0; i < len; i++) {

                    if (str[i] == c) {

                              (*env)->ReleaseStringChars(env, s, str);

                              return i;

                    }
          }

          (*env)->ReleaseStringChars(env, s, str);

          return -1;

}
```

Rozšíření Pythonu v C/C++

```
#define PY_SSIZE_T_CLEAN
#include <Python.h>

// definice metod
static PyMethodDef MyModuleMethods[] = {
    {"add", add, METH_VARARGS, "add two integers"},
    {NULL, NULL, 0, NULL}
};
// definice modulu
static struct PyModuleDef mymodule = {
    PyModuleDef_HEAD_INIT,
    "mymodule",   /* jméno modulu */
    NULL,
    -1,
    MyModuleMethods
};
```

Pokračování na dalším slidu

Metody a inicializační funkce

```
// Inicializační metoda
PyMODINIT_FUNC PyInit_mymodule(void)
{
    return PyModule_Create(&mymodule);
}

// Metoda ADD
static PyObject *add(PyObject *self, PyObject *args)
{
    int a, b;
    if (!PyArg_ParseTuple(args, “ii", &a, &b))
        return NULL;
    return PyLong_FromLong(a+b);
}
```

Překlad a použití

Vložení CFLAGS pro Python

```
gcc -o mymodule.so `pkg-config --cflags python3` \
      mymodule.c -shared `pkg-config --libs python3`
```

Vložení knihoven pro Python

test.py

```
import mymodule as mm
print(mm.add(3,5))
```

Testování

PYTHONPATH=. python3 test.py