ProFUSION
embedded systems

Tudo que você sempre quis saber sobre Bindings

PythonBrasil[7]
São Paulo

Gustavo Sverzut Barbieri

o que são bindings ou wrappers?

ProFUSION embedded systems

- binding - ligação, amarração
- wrapper - capa, envólucro
- expõe C/C++ em Python

exemplos bem conhecidos

ProFUSION embedded systems

- pygame (SDL, jogos)
- numpy (numérico, matrizes)
- sqlite, mysql, pypgsql (PostgreSQL)...
- pygtk, pyqt, pykde...
- stdlib: md5, sha*, zlib...

casos de uso

ProFUSION embedded systems

- integração com sistemas existentes (gtk, qt...)
- acesso a recursos de baixo nível (i/o, syscalls...)
- otimização de caminhos críticos (md5, sha*...)
- extensões em python para programas (games, kde...)

como funciona?

ProFUSION embedded systems

- um CPU só processa instruções nativas
- uma VM converte para instruções nativas
    - interpretando (CPython)
    - compilando na hora (JIT)
- implica em convenções:
    - chamadas de função
    - estrutura de dados

como CPython funciona?

ProFUSION embedded systems

convenções do CPython

ProFUSION embedded systems

- PyObject é a estrutura base
- PyObject *PyCFunction(PyObject *o, PyObject *args)
- PyCFunctionObject é criado
- PyEval_CallObject(), PyEval_CallFunction()...

como criar bindings em CPython?

ProFUSION embedded systems

- defina PyObject para suas estruturas
- implemente PyCFunction para suas funções
- PyCFunction convertem tipos de/para PyObject
- registra-se as funções e tipos

ferramentas

ProFUSION embedded systems

- manual
- introspeção: ctypes (ffi)
- descrição multi-linguagem: swig
- descrição exclusiva para python: cython/pyrex
- específicas: pygobject, pyside (qt)...
- não tem melhor, depende do caso!

manual

ProFUSION embedded systems

- uma biblioteca em C/C++ convencional
- API em Python.h
- permite código mais otimizado
- chato pra caramba de fazer e manter
- proibitivo para grandes projetos

manual - exemplo

ProFUSION embedded systems

```
#include <Python.h>
#include <time.h>
static PyObject *
pymymod_time(PyObject *self, PyObject *args){
  time_t t = time(NULL);
  return PyInt_FromLong(t);
}
PyMODINIT_FUNC initmypymod(void){
  static PyMethodDef methods[] = {
    {"time", pymymod_time, METH_NOARGS, NULL},
    {NULL, NULL, 0, NULL}
  };
  Py_InitModule("mypymod", methods);
}
```

ctypes (ffi)

ProFUSION embedded systems

- usa introspecção "foreign function interface"
- uso 100% python, em runtime
- fácil, porém mais lento
- estruturas ainda precisam ser convertidas
- trabalhoso para grandes projetos

ctypes - exemplo

ProFUSION embedded systems

```
import ctypes
 _libc = ctypes.CDLL("libc.so.6")
def time():
    return _libc.time(None)
```

swig

ProFUSION embedded systems

- usa arquivo de descrição próprio
- gera python, java, lua, ruby...
- não gera classes, apenas funções
- módulo gerado é pouco amigável
- ruim para fazer callbacks
- precisa pré-processar e compilar
- difícil de depurar o binding (gdb)

swig - exemplo

ProFUSION embedded systems

```
%module mypymod
%{
#include <time.h>
static time_t mymodpy_time(void) {
    return time(NULL);
}
%}
%rename(time) mymodpy_time;
time_t mymodpy_time(void);
```

cython/pyrex

ProFUSION embedded systems

- usa arquivo de descrição pythônico

- específico para python

- muito fácil de usar

- módulo gerado é muito amigável

... mas tem algumas pegadinhas!

- precisa pré-processar e compilar

- difícil de depurar o binding (gdb)

cython/pyrex - exemplo

ProFUSION embedded systems

```
cdef extern from "time.h":
    cdef extern int mymodpy_time "time"(int *)
def time():
    return mymodpy_time(NULL)
```

considerações finais

ProFUSION embedded systems

- use a ferramenta certa para o caso certo
- outras ferramentas: pygobject, pyside (qt)...
- faça apenas o necessário no binding/C/C++
- pondere API pythônica x original
- cuidado com gerência de memória!
- cuidado com o GIL! (Lock do Interpretador)

perguntas & respostas

ProFUSION embedded systems

Obrigado!

Gustavo Sverzut Barbieri

<barbieri@profusion.mobi>