Advanced Programming

Talk 8: Language Binding

Saumitra Joshi

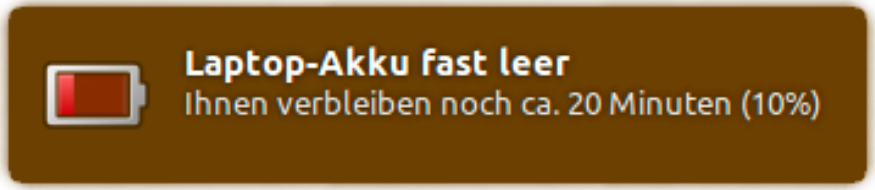
Overview

- What and why?
- Binding examples
- Hands-on: Interfacing C++ on Matlab
- Conclusion

First week at TUM

First week at TUM

Hmm.. my laptop ran out of charge.



First week at TUM

Hmm.. my laptop ran out of charge.

Problem: Plug doesn't fit the socket :(

First week at TUM

Hmm.. my laptop ran out of charge.

Problem: Plug doesn't fit the socket :(

Solution: An adapter!

First week at TUM

Hmm.. my laptop ran out of charge.

Problem: Plug doesn't fit the socket :(

Solution: An adapter!

:)

Binders: What and why?

- Glue code
- Lets you use a library/service in a different programming language

Binders: What and why?

- Glue code
- Lets you use a library/service in a different programming language

- Need:
  - For speed AND ease (use multiple languages)

Binders: What and why?

- Glue code
- Lets you use a library/service in a different programming language

- Need:
  - For speed AND ease (use multiple languages)
  - To use old packages (they are efficient, but tedious)

Binders: What and why?

- Glue code
- Lets you use a library/service in a different programming language

- Need:
  - For speed AND ease (use multiple languages)
  - To use old packages (they are efficient, but tedious)
  - Outdated software for large projects (maintainability and extentability)

Example: SWIG

- "Simplified Wrapper and Interface Generator"

Example: SWIG

- "Simplified Wrapper and Interface Generator"
- Open-source tool connecting C/C++ code with:
  - Scripting languages: Lua, Perl, PHP, Python ..
  - XML
  - Lisp
  - Other

Example: Extend Python in C/C++

```
#include <python.h>
```

- Takes reference to self and to list of arguments
- Objects refered to as `PyObject`

Example: Interfacing C++-Python

- Boost libraries: hipster C++
- Binaries for BLAS, pseudorandom numbers, multithreading, img-proc, unit-testing ..
- You can find a nice tutorial here.

Hands-on: Interfacing C++ on Matlab

- Matlab is easy, but a bit slower
- Execute expensive operations on C++

Hands-on: Interfacing C++ on Matlab

- Matlab is easy, but a bit slower
- Execute expensive operations on C++
  - Pointers
  - Libraries

Hands-on: Interfacing C++ on Matlab

- Matlab is easy, but a bit slower
- Execute expensive operations on C++
  - Pointers
  - Libraries

- Use the `mex` functionality!
  - "`m`atlab `ex`ecutable"
  - compiles and links C, C++, or Fortran source files into binary file **callable from Matlab**

Interfacing C++ on Matlab (1)

- Create `.cpp` file with desired function name

Interfacing C++ on Matlab (1)

- Create `.cpp` file with desired function name
- Instead of `main()`, use this function structure:

```
void mexFunction (
        int nlhs,
        mxArray *plhs[],
        int nrhs,
        const mxArray *prhs[]
);
```

Interfacing C++ on Matlab (2)

- On Matlab, call

  ```
  mex functionName.cpp
  ```

- Generates a `.mexa64` file (arch-dependent!)
- Use as a typical Matlab function

Interfacing C++ on Matlab (2)

- On Matlab, call

  ```
  mex functionName.cpp
  ```

- Generates a `.mexa64` file (arch-dependent!)
- Use as a typical Matlab function

Time for a demo!

Interfacing C++ on Matlab (2)

- On Matlab, call

  ```
  mex functionName.cpp
  ```

- Generates a `.mexa64` file (arch-dependent!)
- Use as a typical Matlab function

Time for a demo!

- More info here.

That's all :)

Questions?