Developing your own programs in Madagascar

Jingwei Hu
Institute for Computational Engineering and Sciences (ICES)
The University of Texas at Austin

Madagascar School
Austin, TX
July 21, 2012

Writing and adding your own programs

- There are about 1000 programs already in Madagascar
- Include both seismic and generic data processing tools

Writing and adding your own programs

- There are about 1000 programs already in Madagascar
- Include both seismic and generic data processing tools

BUT

- Some tasks are not (easily) doable with available tools; some tools may not exist at all
- Need to develop programs for your own purpose

What is the main goal of this tutorial?

After this presentation you should know how to put your own programs into madagascar

What is the main goal of this tutorial?

After this presentation you should know how to put your own programs into madagascar

How are we going to do it?

What is the main goal of this tutorial?

After this presentation you should know how to put your own programs into madagascar

How are we going to do it?

- Introduce the program design philosophy in Madagascar
- Give an overview of the Madagascar APIs
- Use a concrete example to show how to write, add, and test your own programs in Madagascar

Before we start

Program architecture

- Madagascar programs are task-centric: ONE task per program
- Programs are constructed to run in a pipeline with input from standard in and output to standard out:

```
sfwindow < in.rsf | sf_my_program | sffft > out.rsf
```

- Pass parameters from command line or SConstruct file

Before we start

Where to begin

- Break the problem into several pieces: each performs a single task
- For each task, make sure to check out

  `sfdoc -k keyword`

  or List of programs on ahay.org

Do not waste time reinventing things !

Common programs to use:
e.g. data set manipulation (add, multiply, concatenate...), FFT, bandpass filtering, etc.

Sample problem

Xiaoming wants to apply the newest XYZ filter in the frequency domain, but his RSF data is in the time domain, how should he design his new Madagascar program?

Sample problem

Xiaoming wants to apply the newest XYZ filter in the frequency domain, but his RSF data is in the time domain, how should he design his new Madagascar program?

Possible solutions:

- Write new code that applies FFT, then the filter, and then the inverse FFT
- Write new code that applies the filter, and call an existing library for FFT and its inverse
- Only write the filter program, and use Madagascar programs for FFT and its inverse

Sample problem

Xiaoming wants to apply the newest XYZ filter in the frequency domain, but his RSF data is in the time domain, how should he design his new Madagascar program?

Not task-centric and Xiaoming wastes time researching/ writing/debugging a FFT code

Possible solutions:

- Write new code that applies FFT, then the filter, and then the inverse FFT
- Write new code that applies the filter, and call an existing library for FFT and its inverse
- Only write the filter program, and use Madagascar programs for FFT and its inverse

Sample problem

Xiaoming wants to apply the newest XYZ filter in the frequency domain, but his RSF data is in the time domain, how should he design his new Madagascar program?

Possible solutions:

- Write new code that applies FFT, then the filter, and then the inverse FFT
- Write new code that applies the filter, and call an existing library for FFT and its inverse
- Only write the filter program, and use Madagascar programs for FFT and its inverse

Sample problem

Xiaoming wants to apply the newest XYZ filter in the frequency domain, but his RSF data is in the time domain, how should he design his new Madagascar program?

Not task-centric but Xiaoming uses existing libraries to shorten the development time

Possible solutions:

- Write new code that applies FFT, then the filter, and then the inverse FFT
- Write new code that applies the filter, and call an existing library for FFT and its inverse
- Only write the filter program, and use Madagascar programs for FFT and its inverse

Sample problem

Xiaoming wants to apply the newest XYZ filter in the frequency domain, but his RSF data is in the time domain, how should he design his new Madagascar program?

Possible solutions:

- Write new code that applies FFT, then the filter, and then the inverse FFT
- Write new code that applies the filter, and call an existing library for FFT and its inverse
- Only write the filter program, and use Madagascar programs for FFT and its inverse

Sample problem

Xiaoming wants to apply the newest XYZ filter in the frequency domain, but his RSF data is in the time domain, how should he design his new Madagascar program?

Possible solutions:

- Write new code that applies FFT, then the filter, and then the inverse FFT
- Write new code that applies the filter, and call an existing library for FFT and its inverse
- Only write the filter program, and use Madagascar programs for FFT and its inverse

Task-centric coding that can be used in a pipeline, and be applied to any frequency domain data set

Sample problem

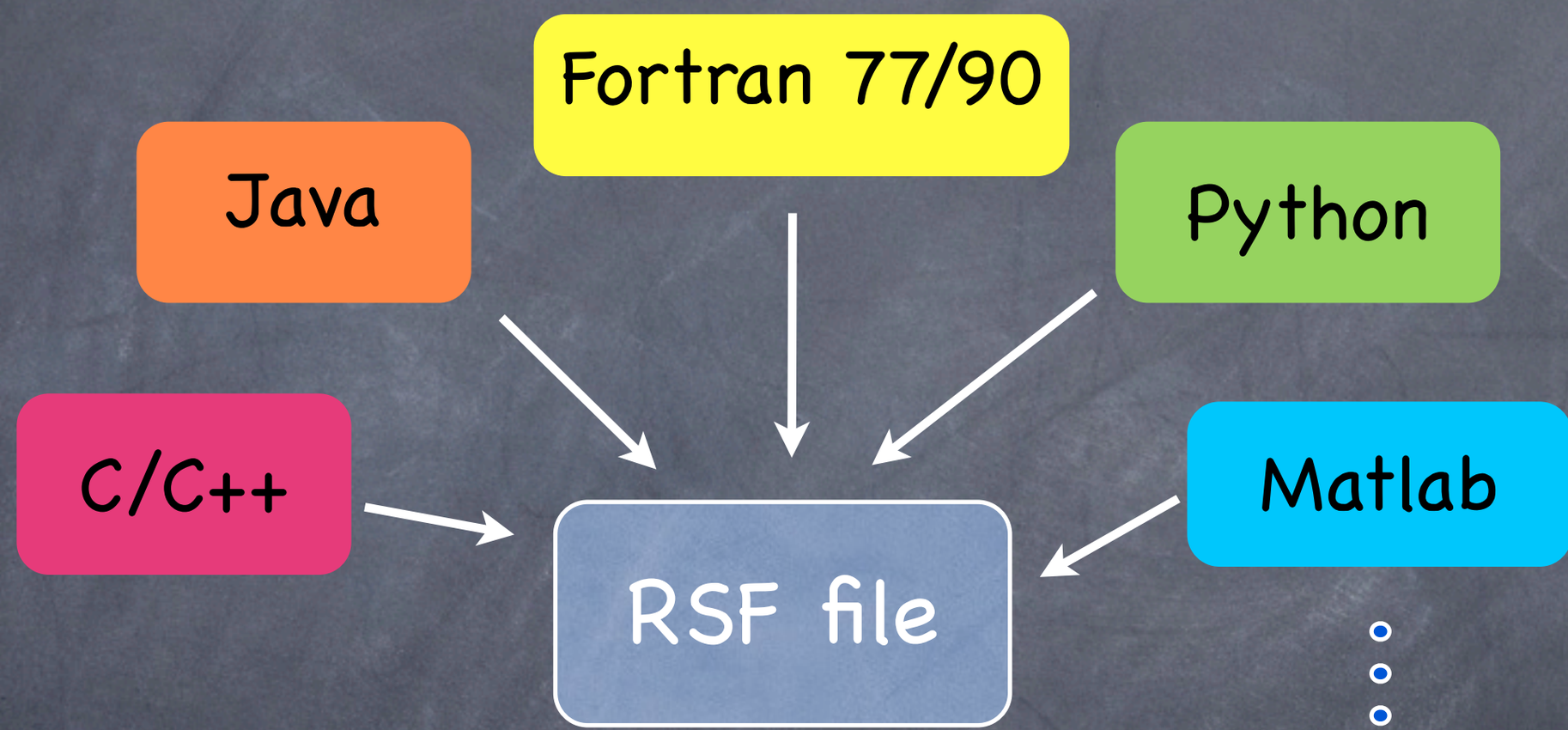
Xiaoming wants to apply the newest XYZ filter in the frequency domain, but his RSF data is in the time domain, how should he design his new Madagascar program?

Possible solutions:

- Write new code that applies FFT, then the filter, and then the inverse FFT
- Write new code that applies the filter, and call an existing library for FFT and its inverse
- Only write the filter program, and use Madagascar programs for FFT and its inverse

Task-centric coding that can be used in a pipeline, and be applied to any frequency domain data set

Madagascar APIs



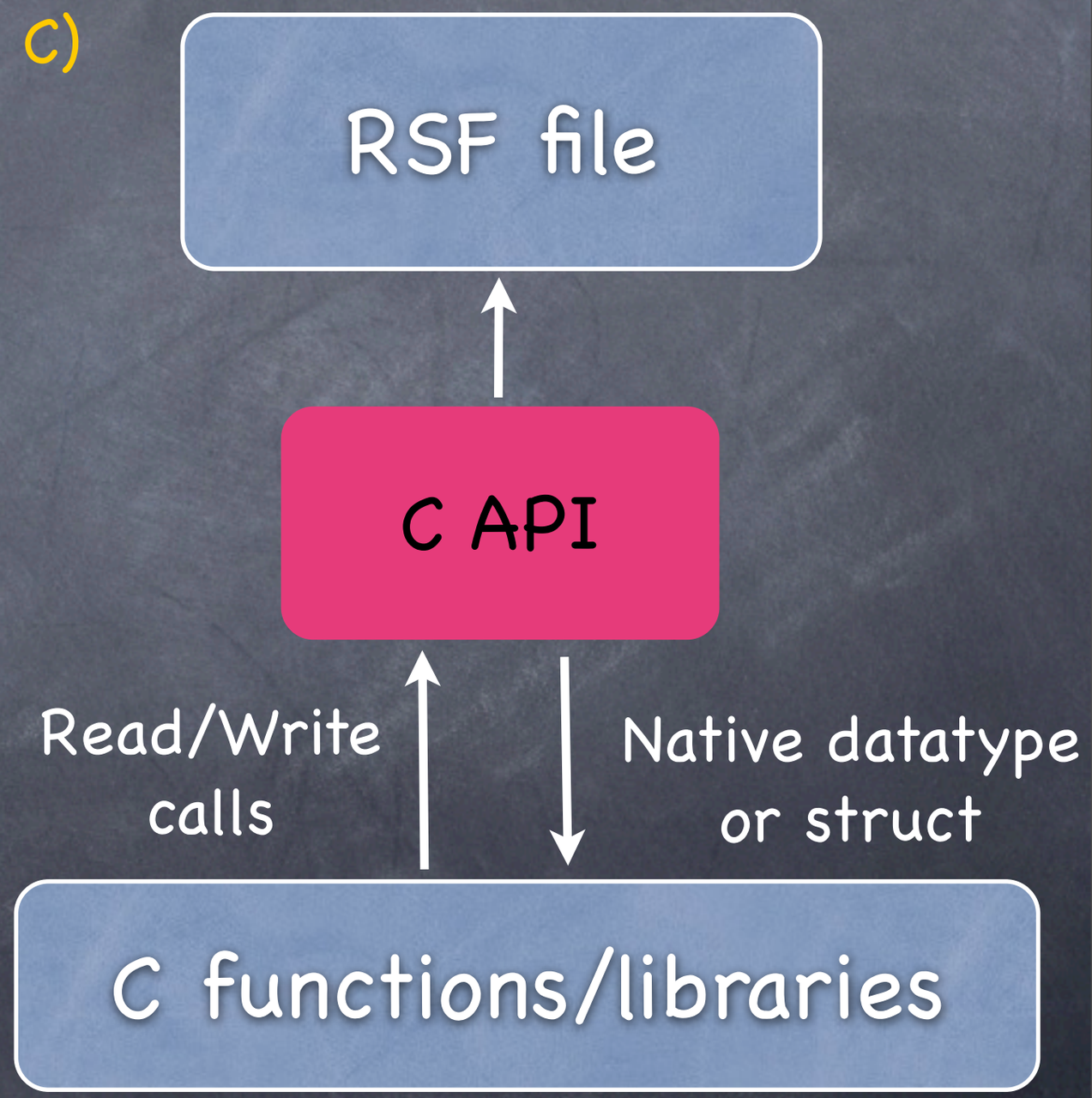
Application Programming Interface (API)

- A set of rules or interface that software programs follow to communicate with each other
- Specifies routines, data structures and the protocols used for communicate between the consumer program and the implementer program of the API

Overview of C API

Strength of Madagascar API (here C)

- Interoperable
  - Common RSF file structure
  - Defines standard for data exchange
  - Enables pipelining with other programs
- Improves development efficiency
  - Access RSF C function/libraries
  - Encapsulate many tasks (e.g. predefined data I/O subroutines)
- Enhances usability
  - Common program documentation style
  - Helps other people use your code
  - Helps you use other people's code



In what follows

- Write a simple program in C++
- Compile and install it in Madagascar
- Test it with various SConstruct Flow() and Plot() rules or command lines

In what follows

- Write a simple program in C++
- Compile and install it in Madagascar
- Test it with various SConstruct Flow() and Plot() rules or command lines

```
RSFSRC/book/rsf/school2012/cpp_code

RSFSRC/book/rsf/school2012/test
```

Task

Apply the soft thresholding to an 1-D data set

Task

Apply the soft thresholding to an 1-D data set

Soft thresholding function:

$$S(x) = \begin{cases} x - \mu & x \geq \mu \\ 0 & |x| < \mu \\ x + \mu & x \leq -\mu \end{cases}$$

used for denoising. μ is the thresholding parameter.

Apply $S(\cdot)$ to the data componentwisely

Madagascar program in C++

~/cpp_code/Mtest.cc

```
// Soft thresholding for 1-D data

#include<rsf.hh>

int main(int argc, char** argv)
{
  // Initialize RSF
  sf_init(argc,argv);


  // Get input
  iRSF input;
  int n1;
  input.get("n1",n1);

  // Read data
  std::valarray<float> fdata(n1);
  input >> fdata;
```

Madagascar program in C++

~/cpp_code/Mtest.cc

```
// Soft thresholding for 1-D data

#include<rsf.hh>

int main(int argc, char** argv)
{
  // Initialize RSF
  sf_init(argc,argv);


  // Get input
  iRSF input;
  int n1;
  input.get("n1",n1);

  // Read data
  std::valarray<float> fdata(n1);
  input >> fdata;
```

access the C++ interface

Madagascar program in C++

~/cpp_code/Mtest.cc

```
// Soft thresholding for 1-D data

#include<rsf.hh>

int main(int argc, char** argv)
{
  // Initialize RSF
  sf_init(argc,argv);


  // Get input
  iRSF input;
  int n1;
  input.get("n1",n1);

  // Read data
  std::valarray<float> fdata(n1);
  input >> fdata;
```

access the C++ interface

initialize the internally stored table of command-line arguments

Madagascar program in C++

~/cpp_code/Mtest.cc

```
// Soft thresholding for 1-D data

#include<rsf.hh>

int main(int argc, char** argv)
{
  // Initialize RSF
  sf_init(argc,argv);


  // Get input
  iRSF input;
  int n1;
  input.get("n1",n1);

  // Read data
  std::valarray<float> fdata(n1);
  input >> fdata;
```

access the C++ interface

initialize the internally stored table of command-line arguments

declare the input file
n1 is the data size

Madagascar program in C++

~/cpp_code/Mtest.cc

```
// Soft thresholding for 1-D data

#include<rsf.hh>

int main(int argc, char** argv)
{
  // Initialize RSF
  sf_init(argc,argv);


  // Get input
  iRSF input;
  int n1;
  input.get("n1",n1);

  // Read data
  std::valarray<float> fdata(n1);
  input >> fdata;
```

access the C++ interface

initialize the internally stored table of command-line arguments

declare the input file
n1 is the data size

data is stored using the valarray template class from the standard C++ library

Madagascar program in C++

~/cpp_code/Mtest.cc

```
// Get parameter
iRSF par(0);
float mu;
par.get("mu",mu);
// threshold value


// Soft thresholding
for (int i=0; i<n1; i++) {
  if (fdata[i]<=-mu)
    fdata[i]=fdata[i]+mu;
  else if (fdata[i]>=mu)
    fdata[i]=fdata[i]-mu;
  else
    fdata[i]=0;
}
```

Madagascar program in C++

~/cpp_code/Mtest.cc

```
// Get parameter
iRSF par(0);
float mu;
par.get("mu",mu);
// threshold value


// Soft thresholding
for (int i=0; i<n1; i++) {
  if (fdata[i]<=-mu)
    fdata[i]=fdata[i]+mu;
  else if (fdata[i]>=mu)
    fdata[i]=fdata[i]-mu;
  else
    fdata[i]=0;
}
```

the command-line parameter is also handled as iRSF object, initialized to zero

Madagascar program in C++

~/cpp_code/Mtest.cc

```
// Get parameter
iRSF par(0);
float mu;
par.get("mu",mu);
// threshold value


// Soft thresholding
for (int i=0; i<n1; i++) {
  if (fdata[i]<=-mu)
    fdata[i]=fdata[i]+mu;
  else if (fdata[i]>=mu)
    fdata[i]=fdata[i]-mu;
  else
    fdata[i]=0;
}
```

the command-line parameter is also handled as iRSF object, initialized to zero

the main part of the program loop over all components to apply the soft thresholding function

Madagascar program in C++

~/cpp_code/Mtest.cc

```
  // Set output
  oRSF output;

  // Write data
  output << fdata;

  exit(0);
}
```

Madagascar program in C++

~/cpp_code/Mtest.cc

```
  // Set output
  oRSF output;

  // Write data
  output << fdata;

  exit(0);
}
```

declare the output file
if size different from input
output.put("n1",n1)...

Madagascar program in C++

~/cpp_code/Mtest.cc

```
// Set output
oRSF output;

// Write data
output << fdata;

exit(0);
}
```

declare the output file
if size different from input
output.put("n1",n1)...

Basic flow of a Madagascar program

read data and parameters

↓

process data

↓

write data to output

Build programs to Madagascar 1

- Create a directory YourName under RSFSRC/user
- cd to YourName and copy Mtest.cc and SConstruct files under cpp_code to the directory YourName

this SConstruct file is for compiling (like a makefile), different from the one for managing data processing flows

Build programs to Madagascar 2

- `cd RSFSRC/user/YourName` and `scons` to compile locally with debugging flags, generating an executable file `sftest`
- `cd RSFSRC` and `scons install` to compile globally with optimization flags and install the program in Madagascar

Need to install the C++ interface to complete these steps

```
./configure API=c++
scons install
```

Test programs in Madagascar

- Now the program should be installed in Madagascar, type `sftest` in terminal to see its documentation
- `cd RSFSRC/book/rsf/school2012/test` and test the program using the `SConstruct` file there or command line

Use what you learned yesterday to view and modify the results

Further information

- http://ahay.org/wiki/Guide_to_madagascar_API
  a simple example written in different APIs
- http://ahay.org/wiki/Adding_new_programs_to_Madagascar
- http://ahay.org/wiki/Contributing_new_programs_to_Madagascar
- http://www.ahay.org/RSF/book/rsf/manual/manual_html/
  a full reference of the C API