MADAGASCAR

UWA Workshop

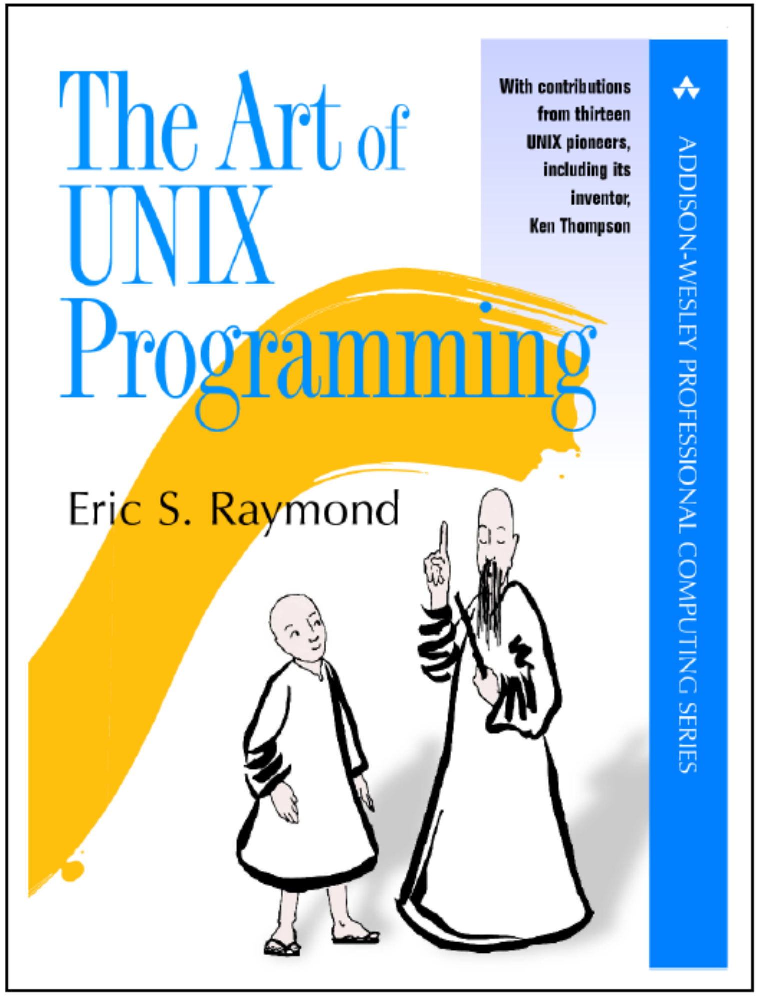
July 3rd 2015

Perth, Australia

Introduction to SConstruct and Python

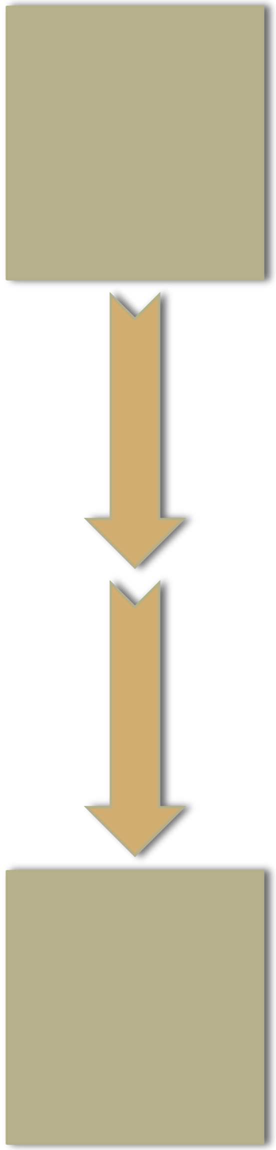
Adapted from a talk by Sergey Fomel

Madagascar Design Principle


The Art of UNIX Programming
Eric S. Raymond
With contributions from thirteen UNIX pioneers, including its inventor, Ken Thompson
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

- Data arrays are file objects on disk
- "Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface."

Doug McIlroy



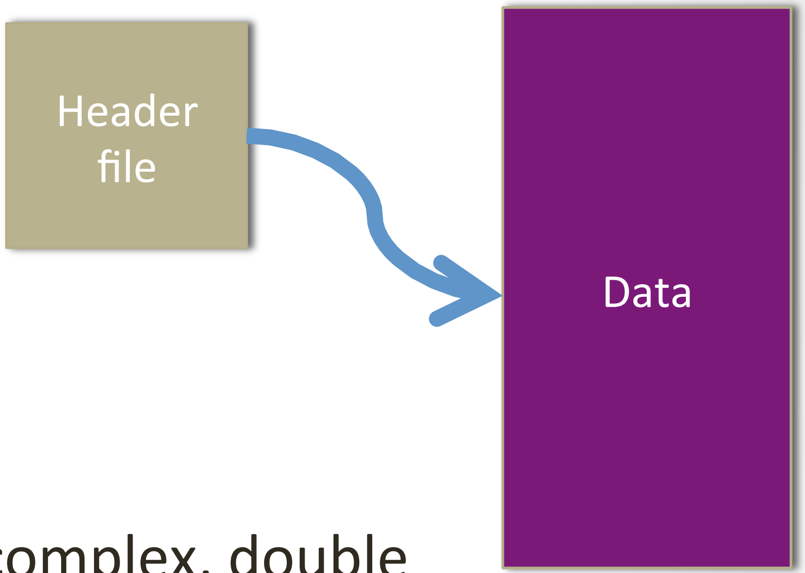
- “To design a perfect anti-Unix, make all file formats binary and opaque, and require heavyweight tools to read and edit them.”
- “If you feel an urge to design a complex binary file format, or a complex binary application protocol, it is generally wise to lie down until the feeling passes.”

Eric Raymond

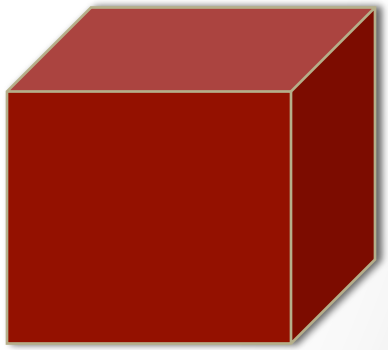
Outline

- **RSF file format**
- Introduction to SCons and Python
- Putting it all together, SConstruct

RSF File Format

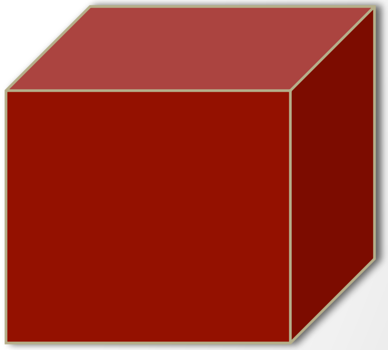


- Self-describing data format
  - Header file
  - Data file
- Supported data types
  - Uchar, char, short, int, float, complex, double
- Supported data file formats
  - Native binary, XDR, ASCII
- Can interface with other formats
  - SEPlib
  - SEGY
  - Seismic Unix (SU)



RSF format describes a Hypercube

```
1.6-svn sfput   dir:  user@machine    Thu Mar 12 16:37:00 2015

     label3=Distance
     unit1=m
     unit2=m
     unit3=m
     d1=5.000000
     d2=10.000000
     d3=10.000000
     o1=0.000000
     data_format="native_float"
     o2=3340000.000000
     o3=281616.000000
     in="binary_file.rsf@"
     label1=Depth
```



RSF language interface

Can read and write RSF files with other languages

- C
- C++
- Fortran
- Python
- Octave
- Java
- Matlab

Datapath determines where data file is put

- datapath= parameter on the command line.
- DATAPATH environmental variable
- .datapath file in the current directory.
- .datapath file in the user home directory.
- Current directory.
- Data file may be appended to header

Outline

- RSF file format
- **Introduction to SCons and Python**
- Putting it all together, SConstruct

What is SCons?

- Build system (**S**oftware **Cons**truction)
- Written in **Python**
  - Configuration files are Python scripts
- Built-in support for different languages
- Dependency analysis
- Parallel builds
- Cross-platform

Evolution of Build Systems

- **Make (1977)**
  - “Sendmail and **make** are two well known programs that are pretty widely regarded as originally being *debugged into existence*. That's why their command languages are so poorly thought out and difficult to learn. It's not just you - everyone finds them troublesome.”

    Peter van der Linden
- **GNU Make (1988)**
- **SCons (2000)**

SCONS

What is Python?

- **Dynamically typed programming language**
- **Clear, readable syntax**
  - "friendly and easy to learn"
- **Full modularity**
  - "batteries included"
- **Integrates with other languages**
  - "plays well with others"
- **Free and open-source**

Who uses Python?

- "**Python** has been an important part of Google since the beginning, and remains so as the system grows and evolves. Today dozens of Google engineers use Python, and we're looking for more people with skills in this language." *Peter Norvig*

- "**Python** is fun, free, runs on a broad range of platforms and has a large library of sophisticated modules, including numerical. It meets all our criteria for a first language." *John Scales & Hans Ecke*

Python in 6 Easy Steps

1. Variables and strings
2. Lists and dictionaries
3. For loop
4. If/else, indentation
5. Functions and modules
6. Multidimensional arrays

1. Variables and Strings

```
>>> a='Melbourne'
>>> a[0]
'M'
>>> a[:3]
'Mel'
>>> b = a + " rocks"
>>> print b
'Melbourne rocks'
>>> a+2
Traceback (most recent call last):
  File "<stdin>", line 1, module <module>
TypeError: cannot concatenate 'str' and 'int' objects
>>> a+str(2013)
'Melbourne2013'
```

2. Lists and Dictionaries

```
>>> a = [‘Melbourne’, ‘ASEG’]
>>> a[0]
‘Melbourne’
>>> len(a)
9
>>> a.append(20)
>>> a
[‘Melbourne’,‘ASEG’,20]
>>> b = (‘Melbourne’,‘ASEG’)
>>> b.append(20)
Traceback (most recent call last):
  File “<stdin>”, line 1, in <module>
AttributeError: ‘tuple’ has no attribute ‘append’
>>> c = {‘city’: ‘Melbourne’, ‘year’: 2013}
```

3. For loop

```
>>> a = (‘Melbourne’, ‘Perth’)
>>> for city in a:
…       print city, len(city)
Melbourne 9
Perth 5
>>> for k in range(2):
…    print k, a[k]
0 Melbourne
1 Perth
>>> c = {‘city’: ‘Perth’, ‘year’: 2015}
for key in c.keys():
…    print c[key]
‘Perth’
2015
```

4. If/else, indentation

```
>>> for k in range(4):
>>>     if k < 2:
…           print k
…       else:
…           print ‘no’
0
1
no
no
>>> try:
…       a = ‘Perth’ + 2015
except:
…       print ‘error’
error
```

5. Functions and modules

```
>>> def add_5(a):
…        ‘Add 5 to input’
…        return 5+a
>>> a = add_5(3)
>>> a
8
>>> def add_b(a,b=5):
         ‘Add b to a’
…        return b+a
>>> add_b(a)
13
>>> import math
>>> math.sqrt(add_b(a,8))
4
```

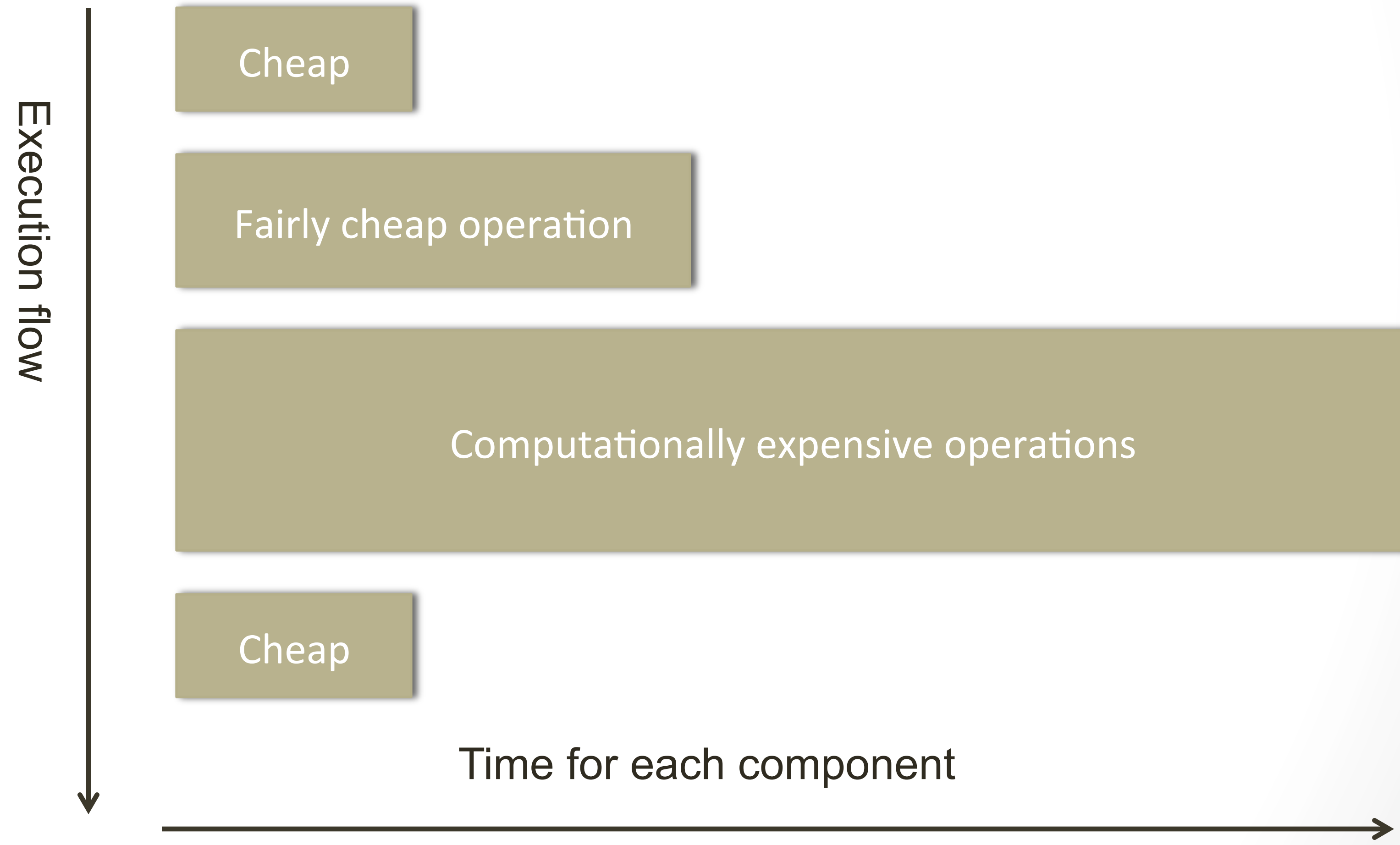
6. Multidimensional arrays

```
>>> import numpy
>>> arr=numpy.zeros((2,2), dtype=numpy.float32)

array([[ 0.,  0.],
       [ 0.,  0.]])

>>> arr[0,1]=1.0;

array([[ 0.,  1.],
       [ 0.,  0.]])

>>> arr.ravel()
array([ 0.,  1.,  0.,  0.])

>>> arr*2+1
array([[ 1.,  3.],
       [ 1.,  1.]])
```

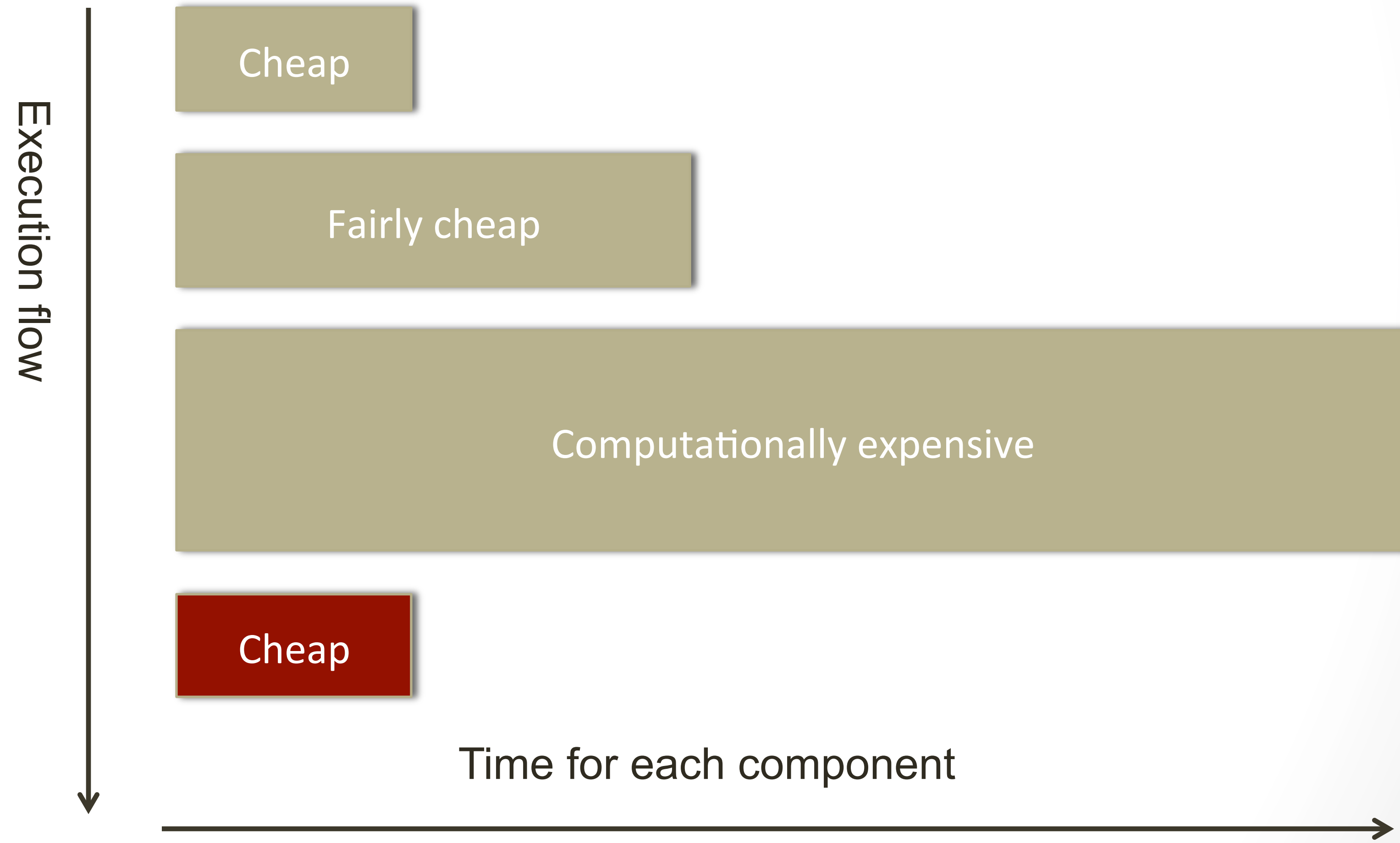
Outline

- RSF file format
- Introduction to SCons and Python
- **Putting it all together, SConstruct**

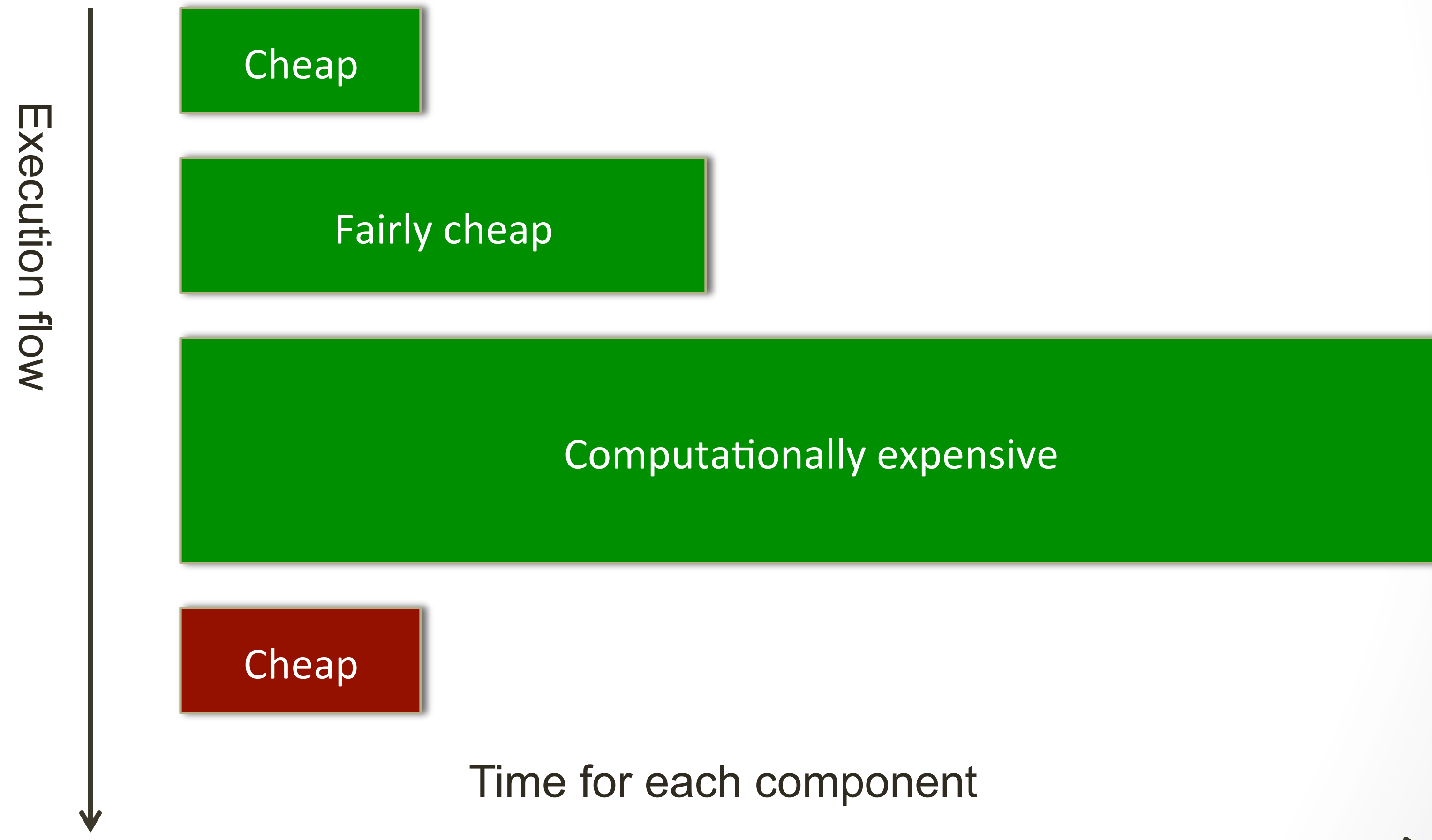
The problem with scripts for data processing…



Is that you have to run the entire script again if something goes wrong..



Storing the results at each step and resuming from the problem saves time



SConstruct for data processing

- **An SConstruct file is a Python script with functionality to construct causal dependencies between data processing blocks**
  - **Fetch(‘filename’, ’dirname’)**
    - A rule for downloading files from a server
  - **Flow(‘target’, ‘source’, ‘command’)**
    - A rule for making target from source
  - **Plot(‘target’, ’source’, ‘command’)**
    - Like Flow but generates a figure file
  - **Result(‘target’, ‘source’, ‘command’)**
    - Like Plot but generates a final result

SConstruct for data processing

```
from rsf.proj import *
import sys

# Fetch a shot gather from the default Madagascar data server
Fetch('wz.35.H','wz')

# Convert the shot gather to native format and window it
Flow('shot_gather','wz.35.H','dd form=native | window n1=400 j1=2 |
smooth rect1=3')

# Plot the data
Result('shot_gather','pow pow1=2 | grey')

# Example flows, note that sf is missing from program calls, this Flow
generates "spikes.rsf"
Flow("spikes", None,'''spike nsp=3 n1=1000 k1=200,600,800
mag=1,-0.5,0.5''')
```

SConstruct for data processing

```
# sprayed.rsf is now the target of or causally related to the source spikes.rsf
Flow("sprayed","spikes",'''bandpass fhi=5 | spray axis=2 n=100''')

# This takes sprayed.rsf and puts a finalised plot in the Fig directory, note the
pipe from bandpass to grey
Result("spray_plot","sprayed",'''bandpass fhi=5 | grey color=e title="Yes we
can with SConstruct!" ''')

# This puts a plot in the current directory
Plot("spray_temp","sprayed",'''bandpass fhi=5 | grey color=e title="Yes we can
with SConstruct!" ''')

# You can only print to standard error in SConstructs and Madagascar
programs.
# Standard output and input is reserved for IO.
print >> sys.stderr, "Python code in the SConstruct is executed before Flows"

End()
```

Using SConstruct for program compilation

```
env=Environment()
env.Program(“program”, “file.c”)
```

```
scons –h  # Print help message
scons –n  # Dry run
scons –c  # Clean the build
scons –Q # Suppress messages
scons –j  # Build jobs in parallel
```

Parallel Processing with pscons

- Figures out the number of nodes/CPUs and runs **scons –j**
- Use **split=** and **reduce=** to split data for simple data-parallel processing

Summary

- **RSF** file format is simple, represents a multidimensional array as text + binary
- Madagascar programs are building blocks, can run on the command line and pipe Unix-style
- **SCons** is a powerful and convenient build system adopted by Madagascar for data processing and reproducible documents
  - Configuration files are **Python** scripts

MADAGASCAR

UWA Workshop

July 3rd 2015

Perth, Australia