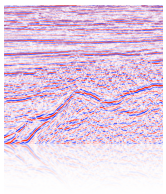
Basic processing flows with SCons

Gilles Hennenfent
PhD student

Seismic Laboratory for Imaging & Modeling
Department of Earth & Ocean Sciences
The University of British Columbia



Reproducible Research in Computational Geophysics
Vancouver, BC
August 30, 2006

References

- on the Web
  http://rsf.sourceforge.net/wiki/index.php/SCons
- in the Madagascar source code tree
  /path/to/madagascar/book/rsf/scons/paper.tex

Madagascar

- multi-level architecture
  - **low-level**
    - main programs: typically developed in C/C++ (fortran) programming language
  - **mid-level**
    - processing flows: written using Python and SCons
  - **high-level**
    - documentation: written using LaTeX and SCons

Python*

- what is Python?
  - dynamic object-oriented programming language
- main features
  - multi-paradigm language
    - object orientation, structured programming, functional programming, and aspect-oriented programming supported
  - uses automatic memory management
    - garbage collection
  - **very clear, readable syntax**
  - strong support for integration with other languages and tools
    - extension modules can be written in C/C++ and wrapped with SWIG
  - comes with extensive standard libraries
  - cross-platform
    - Windows, Linux/Unix, Mac OS X, and OS/2
  - distributed under an OSI-approved open source license

* from http://www.python.org/

Python

- who recommends it?
  - Scales, J. A. and H. Ecke, 2002, *What programming languages should we teach our undergraduates?*: The Leading Edge, 21, 260–267.

    "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. It can be learned quickly, a necessary requirement if we want faculty involved."
  - Peter Norvig (Google)

    "Python has been an important part of Google since the beginning, and remains so as the system grows and evolves."

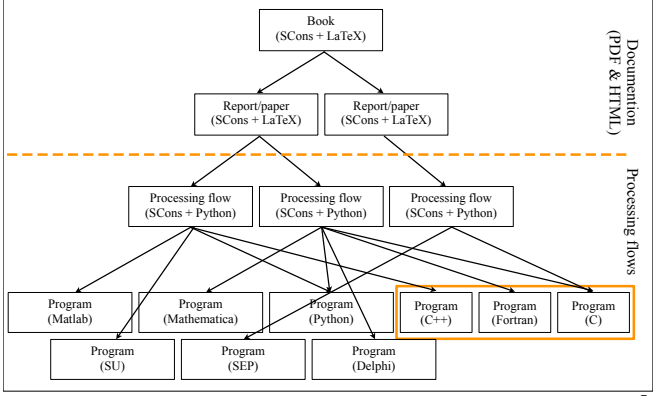
SCons*

- what is SCons?
  - next-generation, cross-platform, **build tool**
  - alternative to Make
- main features
  - configuration files are Python scripts
  - reliable, automatic dependency analysis
    - no need of make depend step
  - built-in support for multiple languages
    - C, C++, Java, Fortran, Qt, SWIG, (La)TeX among others...
  - cross-platform
    - Linux, POSIX systems, Windows (NT, 2000, XP), Mac OS X, and OS/2
  - reliable detection of file changes using MD5 signatures (and/or time stamp)
  - global view of dependencies
  - parallel builds
  - released under MIT license

* from http://scons.tigris.org

command line

3 levels of usage



Processing flows

- SCons and Madagascar
  - processing flows are written in files named SConstruct
  - 4 main Madagascar commands
    - **Fetch**()
    - **Flow**()
    - **Plot**()
    - **Result**()
  - main SCons commands
    - scons, scons lock
    - scons file.rsf, scons graph.vpl
    - scons graph.view
  - CAUTION: Python (and thus SCons) use indentation, rather than curly braces, to delimit blocks
    - an increase in indentation comes after certain statements
    - a decrease in indentation signifies the end of the current block
- SCons, Madagascar, and Python

Madagascar commands for SConstruct

- **Fetch**(data_file,dir[,ftp_server_info])
  - a rule to download <data_file> from a specific directory <dir> of an FTP server <ftp_server_info>.
- **Flow**(target[s],source[s],command[s][,stdin][,stdout])
  - a rule to generate <target[s]> from <source[s]> using <command[s]>.
- **Plot**(intermediate_plot[,source],plot_command) or **Plot**(intermediate_plot,intermediate_plots,combination)
  - a rule to generate <intermediate_plot> in the working directory.
- **Result**(plot[,source],plot_command) or **Result**(plot,intermediate_plots,combination)
  - a rule to generate a final <plot> in the special Fig folder of the working directory.
- **End**()
  - a rule to collect default targets.

"Hello world" SConstruct

```
from rsfproj import *

# Download the input data file
Fetch('lena.img','imgs')

# Create RSF header
Flow('lena.hdr','lena.img',
     'echo n1=512 n2=513 d1=1 d2=1 in=$SOURCE data_format=native_uchar',
     stdin=0)

# Convert to floating point and window out first trace
Flow('lena','lena.hdr','sfdd type=float | sfwindow f2=1')

# Display
Result('lena',
       '''
       sfgrey title="Hello, World!" transp=n color=b bias=128
       clip=100 screenratio=1
       ''')

# Wrap up
End()
```


Demo...

SCons, Madagascar, and Python

```
fft2 = 'sffft1 sym=y | sffft3 sym=y'
ifft2 = 'sffft3 sym=y inv=y | sffft1 sym=y inv=y '
Flow('fnlena','nlena',fft2)




def grey(title='',transp='n',label1='',label2=''):
    return '''
    sfgrey title=%s transp=%s minval=%f maxval=%f clip=%f
    screenratio=1 label1=%s label2=%s titlesz=14
    ''' % (title,transp,minval,maxval,clip,label1,label2)

Plot('lplena',grey('Noisy Lena LP filtered'))
```

SCons, Madagascar, and Python

```
# define name of mask
maskloc = '/users/slic/hegilles/research/data/masks/'
maskname = 'mask_ntr300_keep%d.rsf'%(100-perc*100)
mask = maskloc+maskname

# check if access to prepared mask with maximum gap constraint. If not
# construct a new one
if os.access(mask,os.R_OK):
    Flow('mask',mask,'sfcp $SOURCE $TARGET',stdin=0,stdout=-1)
else:
    Flow('mask','model','sfshotholes perc=%f'%perc)
```


SCons, Madagascar, and Python

```
interp = WhereIs('interpolation.py')

[...]

for itr_nbout in range(5,105,5):
    crsi2.append('crsi2_res_%d'%(nbin*itr_nbout))
    Flow(crsi2[-1],['data2','mask2',interp],
         interp + '''
         mask=${SOURCES[1]} data=$SOURCE output=$TARGET verbose=0
         angconst=%f,%f solverparams=%d,%d transparams=%d,%d,%d
         thrparams=%f,%f
         '''%(angmaxL2,angmaxR2,itr_nbout,nbin,nbs,nba,ac,thrmax,thrmin),
         stdin=0,stdout=-1)
    Result(crsi2[-1],mygrey("CRSI2: %d itr"%(nbin*itr_nbout)) )
```


Conclusions

- Madagascar, SCons and Python for processing flows
  - very clear, readable syntax
  - flexible
  - few commands to remember
    - **Fetch**()
    - **Flow**()
    - **Plot**()
    - **Result**()
  - parallel builds
  - reliable detection of file changes using MD5 signatures (and/or time stamp)

Acknowledgments

- Sergey Fomel for sharing Madagascar with us more than 1 year ago
- Felix Herrmann for pointing me to Madagascar and all the SLIM team for testing my reproducible codes