Sourcery Overview & Virtual Machine Installation

Damian Rouson, Ph.D., P.E.
Sourcery, Inc.
www.sourceryinstitute.org
About Us

Sourcery, Inc., is a software consultancy founded by and for computational scientists, engineers, and mathematicians.

We are involved in software development, training, and consulting centered around modern Fortran, including:

OpenCoarrays

Sourcery, Inc., was launched in 2013 through the Entrepreneurial Separation to Transfer Technology program of Sandia National Laboratories.
Sourcery Curriculum

Course lengths of 1-5 days (usually 3) with content selected from a menu:

- **Foundation**: a narrated evolution of Fortran 77 code to Fortran 2008.

- **Programming Paradigms in Modern Fortran**
  - Object-Oriented Programming (OOP) for computational science, engineering, and mathematics.
  - Functional Programming using user-defined operators for asynchronous expression evaluation.
  - Parallel Programming, including
    - Single Program Multiple Data (SPMD) programming via the coarray Fortran (CAF)
    - Many-core accelerator and GPU programming.
    - Mixing CAF and MPI
    - Breaking the bulk synchronization bottleneck with Fortran 2015 CAF EVENTs.
  - Loop concurrency
Sourcery Curriculum (cont.)

Software Engineering in Modern Fortran

- Object-Oriented Design Patterns: best practices for choosing program structure and behavior.
- Agile development and Test-Driven Development (TDD)
- Literate programming with documentation generators.
- Programming by contract, formal design constraints, and runtime assertions.
- Tools, Libraries and Frameworks: performance tuning, building, testing, documentation generation, reverse engineering of class diagrams, parallel sparse linear algebra.

Case Studies

- Parallelizing legacy Fortran.
- Mixing Fortran with C/C++.
- Unit testing.


GCC 6.0: Fortran, C, & C++ compilers, including support for CAF parallel programming.

OpenCoarrays: transport layer using MPI or GASNet communication libraries to support CAF compilers.

MPICH, OpenMPI: Message Passing Interface implementations.

GASNet: a communication library for Partitioned Global Address Space (PGAS) languages, including CAF (which is part of Fortran 2008) and Unified Parallel C (UPC)

PSBLAS: modern Fortran Parallel Sparse Basic Linear Algebra Subroutines for CPUs and GPUs.

pFUnit: a unit testing framework for modern Fortran.

Eclipse/Photran: an integrated development environment (IDE) and refactoring tool for modern Fortran.
TAU: parallel performance tuning and analysis utilities.

CMake/CTest: Cross-platform Makefile generator for automated software building and testing.

ForUML: Reverse engineering tool for generating object-oriented design diagrams of modern Fortran codes.

ROBODoc: Automated documentation generator.

Git: distributed version control.

Octave: a MATLAB-like interpreted language for numerical computation and visualization.

Doxygen: Automated documentation generator and reverse engineering tool.
## Compiler Support

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GNU</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>Cray</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>Intel</td>
<td>x</td>
<td>-</td>
</tr>
<tr>
<td>IBM</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Portland Group (Nvidia)</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>NAG</td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>

- **x** supports all Fortran 2003 features employed in our courses
- **o** supports all Fortran 2008 & 2015 features we employ
- **-** supports all Fortran 2008 features we employ & some 2015 features
Virtual Machine Installation


2. Launch VirtualBox and select “File->Import Appliance” and import the “SourceryLubuntuYYYYMMDD.ova” file*.

3. Double-click the imported file to boot Linux in a window.

4. Log in using **Account Name**: sourcery student  **Password**: U-Guest-It (Your username is “guest”)

5. The sourcery student account has superuser privileges via the sudoers list.

6. The virtual machine contains environment modules for loading and unloading a few of the installed packages, including various versions of GCC, MPICH, etc. Type “module avail” to see the available modules and see http://modules.sourceforge.net for more details.

*If you have less than 12 GB of free storage space on your computer, see the next slide.
For Machines with Limited Storage

If you have less than approximately 12 GB of free space on your computer, you can store the virtual disk on a flash drive. The screenshot below shows where to specify the path to the virtual disk:

Placing the virtual disk on the flash drive might affect its performance and will require that the flash drive be connected to the computer every time you use the virtual machine.
Sourcery Courses Leverage Modern Fortran’s Support for Multiple Programming Paradigms

Parallel programming (Fortran 2008 and 2015)
Object-oriented programming (Fortran 2003)
Mixed-Language Programming (Fortran 2003)
Functional Programming (Fortran 95)

User-defined, purely functional, asynchronous operators

\[ u_t = -\frac{(\text{grad}.p)}{\rho} + \nu^2(\text{laplacian}.u) - (u\cdot\text{grad}.u) \]
Live Coding “Jam Sessions” Motivate Modern Fortran

Why write pure procedures?
- To facilitate optimizations, including vectorization & parallelization.

Why use coarrays?
- To write parallel algorithms that operate on distributed data.

Why use modules?
- To hide information, increase type safety, and document dependencies.

Why define derived types?
- To encapsulate data and move to a higher level of abstraction.

Why extend types?
- To facilitate code reuse and polymorphism.

Why use abstract types?
- To facilitate design reuse and make code more flexible.
Sourcery Courses Incorporate Open-Source HPC Tools

In this PDE solver, TAU identifies synchronizations as the bottleneck.
Sourcery Courses Teach Techniques that Scale to Massively Parallel Platforms

87% parallel efficiency in weak scaling on 16,384 cores
We Employ Examples and Hands-On Exercises Relevant to Scientific Programming

Starting with “stars” initially located randomly around a unit sphere with velocities oriented radially outward, we will develop a parallel simulation of the stars’ motion using a linear drag law:

\[
\frac{dx}{dt} = \vec{v} \\
\frac{dv}{dt} = \frac{1}{\tau_p} [\vec{v}_{wind}(z) - \vec{v}] \quad \tau_p = \frac{m}{6\pi a \mu (T_{air}(z; T_0))}
\]

Object-oriented design patterns give us the flexibility to extend our initial code and design. We progressively enhance the simulation’s fidelity by adding the option to include fireworks colors, a turbulent wind velocity profile, and variable air properties while offloading much of the new logic to the compiler in the form of dynamic dispatching of algorithms.
Course Attendees Receive a Copy of Scientific Software Design: The Object-Oriented Way

Image of book cover

http://www.cambridge.org/Rouson

Textbook for ShortCourses at Wright Patterson Air Force Base, SC14, Bettis Atomic Power Laboratory, BP, NASA LaRC, ARL, NRL (DC), NRL (CA), Knolls Atomic Power Laboratory, NASA GSFC, Oak Ridge, SC12, NCAR, NERSC, Hector (UK)
Sourcery Short Courses Employ Material Developed for University Graduate Courses

Stanford University (2013, 2014)
University of Cyprus (2006)