Babel Support for FORTRAN 90: Status

Tammy Dahlgren, Tom Epperly, Scott Kohn, and Gary Kumfert

Center for Applied Scientific Computing

Common Component Architecture Working Group

October 3, 2002

This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

UCRL-PRES-150102

Overview

- Initial Goals
- Modifications
  - Command Line
  - SIDL Runtime
  - Back-end
    - Directory structure
    - Fortran
    - Writer
  - Regression Tests
  - Build System
- Example
- Future Work

A minimalist approach has been taken for quicker turn-around.

Feature	F77	F90	Comment
File extension	.f	.f90	Standard
Format	Fixed	Free	Although F90 handles both, the Impls are generated in free-form
Comment style	C	!	
Subroutine termination	end	end subroutine	
Use statement	---	New splicer block	
Subroutine name lengths	---	---	*See Tom Epperly's talk.*

Initial modifications focus on maximizing code re-use within the Babel compiler.

- Support new language option at command line
- Generalize existing back end code
  - Directory change (i.e., f77 ➔ fortran)
  - Common
  - Fortran
  - Writers
- *Utilize tweaked F77 regression tests for F90*
- *Modify build system to support "standard" autoconf macros for F90/95*

~~Modifying the build in a general way has been a *very painful* process!~~

- Documentation

Only 7 compiler files were impacted by the modifications.

Compiler Source	Change(s)
UserOptions.java	Recognizes F90 as a valid language.
backend/ CodeGenerationFactory.java	Registers F90 server, client, and makefile generators.
backend/ CodeConstants.java	Has F90 comment character and impl file extension.
backend/fortran/ Fortran.java	Returns ".f90" impl extension when target language is F90.
backend/fortran/ ImplSource.java	Generates sub's splicer block, indents arg declarations, and ends subroutine properly for F90.
backend/fortran/ StubDoc.java	Indents arg declarations and ends subroutine for F90.
backend/writers/ LanguageWriterForFortran.java	Initializes the first tab stop, tab spacing, line break, and comment for F90.

Command line changes simply involve adding the new language identifier.

Usage babel [-h | --help]
or babel [-v | --version]
or babel *option(s) sidlfilename1 ... sidlfilenameN*
where help, version, and option(s) are

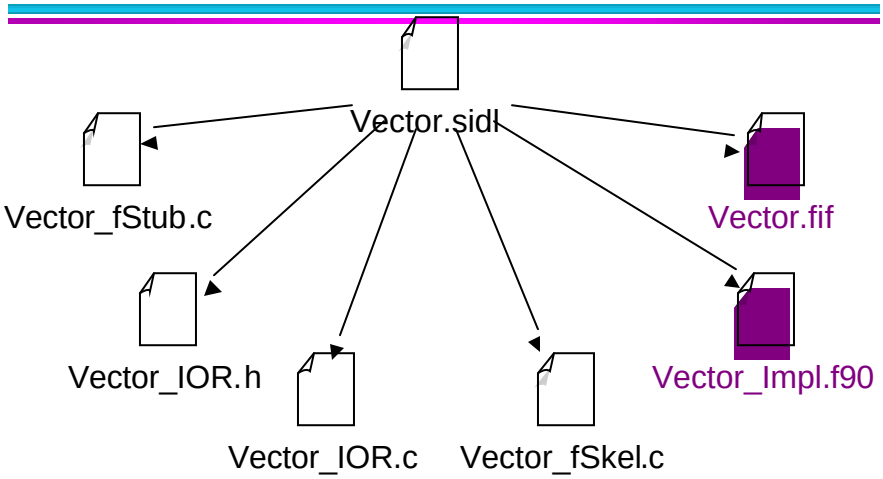
-h	| --help	Display usage information and exit.
-v	| --version	Display version and exit.
-p	| --parse-check	Parse the sidl file but do not generate code.
-x	| --xml	Generate only SIDL XML (for repository update).
-c*lang*	| --client=*lang*	Generate only client code in specified language (C | C++ | F77 | F90 | Java | Python).
-s*lang*	| --server=*lang*	Generate server (and client) code in specified language (C | C++ | F77 | F90 | Python).
-o*dir*	| --output-directory=*dir*	Set Babel output directory ('.' default).
-R*path*	| --repository-path=*path*	Set semicolon-separated URL list used to resolve symbols.
-g	| --generate-subdirs	Generate code in subdirs matching package hierarchy.
--no-default-repository		Prohibit use of default to resolve symbols.
--suppress-timestamp		Suppress timestamps in generated files.
--generate-sidl-stdlib		Regenerate only the SIDL standard library.

As an example, suppose we have a vector spec that includes a norm interface.

```
interface Vector {
   double norm ();
...
}
```

Vector.sidl

Currently generated F90 files are very similar to their F77 counterparts.



Would like to see some examples of real F90/95 code to help define the module file.

The resulting Impl file snippet below illustrates the generated code.

```
subroutine Vector_norm_impl(self, retval)
  ! DO-NOT-DELETE splicer.begin(Vector.norm.use)
  !    Insert use statements here…
  ! DO-NOT-DELETE splicer.end(Vector.norm.use)
  implicit none
  integer*8 self
  double precision retval

! DO-NOT-DELETE splicer.begin(Vector.norm)
!    Insert the implementation here…
! DO-NOT-DELETE splicer.end(Vector.norm)
end subroutine
```

Vector_Impl.f90

Future Work

- Near term
  - Complete build changes
  - Complete F90 regression tests
  - Update the User's Guide
- Long term
  - Address Fortran 90 array descriptors
  - Generate module files
- *Anything else?*