exporting Ada software to Python and Julia

applying GPRbuild to make shared object files

Jan Verschelde

University of Illinois at Chicago
Department of Mathematics, Statistics, and Computer Science
`http://www.math.uic.edu/~jan`
`janv@uic.edu`

Ada devroom, FOSDEM 2022, 6 February, online

Outline

1. Introduction
   - motivation and problem statement
   - mixed language development with GPRbuild

2. Interface Development
   - giving control to the application
   - demonstration package
   - applying GPRbuild

3. an application: PHCpack
   - lessons learned
   - github repositories

make Ada software available to Python and Julia

Two goals when exporting Ada software:

1. Make the build process as simple as possible.
2. Give control to as many functionality as possible.

Jupyter = Julia, Python, R, and many others . . .

- The Jupyter notebook is popular for interactive computing.
- Used in SageMath, an open source mathematical software.
- Not tied to any particular programming language.

GPRbuild is the project manager of the gnu-ada compiler GNAT.

GPRbuild enables mixed-language development,
combining Ada, C, and C++ software.

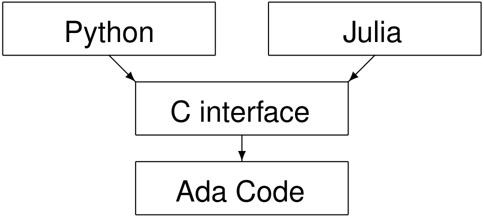
The interfacing in this talk is mainly intended for programmers.

Julia, Python, R, and many others

The Jupyter notebook comes with many kernels.

- Python is a widely used scripting language.
- Julia is a new programming for scientific computing.

Both Python and Julia interface well with C code.



The main point is to automate the build process with GPRbuild.

mixed language development with GPRbuild

GPRbuild recognizes Ada, C, and C**++** as languages.

C is a some kind of least common multiple:

- widely available on almost all computers,
- most languages interface to C.

Therefore, if your software can be used by a C programmer,
then applications in other languages are also likely to benefit.

Library projects build shared object files, files with the extension `.so` (Linux), `.dll` (Windows), `.dylib` (Mac OS X).

developing an interface

Two types of interfaces:

1. The Ada program `main` remains in control.
2. The interface package gives control to a C program.

Example: program that swaps the characters in a string.

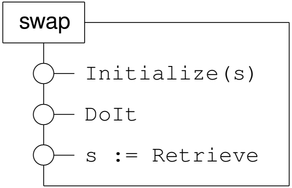
Two types of interfaces:

1. The Ada `main` reads the string, swaps, and writes.
2. The C program has control:
   - passes data to some Ada package
   - calls a procedure exported by the Ada package to swap
   - extracts the data from the Ada package

A string in this context is an array of ASCII codes (32-bit integers).

a demonstration package, and its C interface

Ada package:




```
with C_Integer_Arrays;                      use C_Integer_Arrays;

function call_swap ( jobnbr : integer;
                     sizedata : integer;
                     swapdata : C_intarrs.Pointer;
                     verbose : integer ) return integer;
```

where `C_Integer_Arrays` defines `C_Integer_Array` as an array of `Interfaces.C.int`, and it contains

```
  package C_intarrs is
    new Interfaces.C.Pointers(Interfaces.C.size_T,
                              Interfaces.C.int,
                              C_Integer_Array,0);
```

testing the C interface

```
sizeword = strlen(word);

for(int idx = 0; idx < sizeword; idx++)
   dataword[idx] = (int) word[idx];

adainit();
fail = _ada_call_swap(0,sizeword,dataword,1);
fail = _ada_call_swap(1,sizeword,dataword,1);
fail = _ada_call_swap(2,sizeword,dataword,1);
adafinal();

for(int idx = 0; idx < sizeword; idx++)
   word[idx] = (char) dataword[idx];
```

applying GPRbuild — the file `demo.gpr`

```
project Demo is

    for Languages use ("Ada", "C");

    for Source_Dirs use ("src");

    for Main use
    (
        "hello_world.adb",
        "main.adb",
        "test_call_swap.c"
    );

    for Object_Dir use "obj";

    for Exec_Dir use "bin";

end Demo;
```

a library project — essentials of `demolib.gpr`

```
for Library_Dir use "lib";
for Library_Name use "demo";
for Library_Kind use "dynamic";
for Library_Auto_Init use "true";
for Library_Interface use
(
    "hello_world", "main", "swap", "call_swap", "c_integer_arrays"
);
for Library_Standalone use "encapsulated";

package Compiler is

    for Switches ("call_swap.adb") use ("-c");

end Compiler;

package Binder is

  -- the "-Lada" is needed for the adainit and adafinal functions
  for Default_Switches ("Ada") use ("-n", "-Lada");

end Binder;
```

the Julia `ccall()` function

The Julia code below calls the `call_swap` procedure.

```
LIBRARY = "../Ada/lib/libdemo"

word = [Cint('h'), Cint('e'), Cint('l'), Cint('l'), Cint('o')]
println(word)
ptr2word = pointer(word, 1)
p = ccall((:_ada_call_swap, LIBRARY), Cint,
          (Cint, Cint, Ref{Cint}, Cint), 0, 5, ptr2word, 1)
p = ccall((:_ada_call_swap, LIBRARY), Cint,
          (Cint, Cint, Ref{Cint}, Cint), 1, 5, ptr2word, 1)
p = ccall((:_ada_call_swap, LIBRARY), Cint,
          (Cint, Cint, Ref{Cint}, Cint), 2, 5, ptr2word, 1)
println(word)
```

The string `"hello"` is represented by `Int32[104, 101, 108, 108, 111]`.

The last `println(word)` shows `Int32[111, 108, 108, 101, 104]`.

extending Python

To make code available to Python:

1. Define an extension module in C or C**++**.
2. Define `setup.py`, adding

   ```
   extra_objects=['../Ada/lib/libdemo.a', \
       ADALIB + 'libgnat_pic.a', \
       ADALIB + 'libgnarl_pic.a']
   ```

   where `ADALIB` is the location of the Ada libraries.
3. Run `python setup.py build_ext`, which compiles the extension module and makes the shared object.

The shared object can be imported in a Python session.

an application: PHCpack

PHCpack is software for Polynomial Homotopy Continuation, to solve systems of polynomial equations.

- Mostly written in Ada, developed over almost 30 years.
- Contains `DEMiCs`, written in C++ by Mizutani and Takeda.
- `phcpy` is an interface to Python, for Linux and Mac OS X.
- `phcpy` is motivated by the open source software SageMath.
- A Julia interface is under development.

From the `Julia` folder of the PHCpack source distribution:

```
$ julia version.jl
-> in use_c2phc4c.Handle_Jobs ...
PHCv2.4.85 released 2021-06-30
$
```

`ccall()` uses the `libPHCpack` shared object, made with GPRbuild.

free and open source software

Pointers to github repositories (GPL-3.0 License):

- `github.com/janverschelde/PHCpack`
- `github.com/janverschelde/ExportAdaGPRbuild`

The `ExportAdaGPRbuild` contains the demo code for this talk.

Thanks for your interest in this work.