Apuntes de Fortran

Samuel Gómez

Índice

Boiler plate

```
program Boiler_plate
    use iso_fortran_env
    implicit none

    module mod_hola_mundo
    ! Datos globales
    end module mod_hola_mundo

    contains

    subroutine hola_mundo
    ! Test
    end subroutine hola_mundo
end program Boiler_plate
```

Makefile

Este es un ejemplo de un Makefile

```
FC = gfortran
SRC = $(wildcard *.f90)
TARGET = $(addprefix app/,$(SRC:.f90=))
ARGS =-ffree-form -fimplicit-none -fcheck -fbacktrace
OPEN_BLAS=-L/opt/OpenBLAS/lib

all: $(TARGET)

# Caso genérico
app/%: %.f90
    $(FC) $(ARGS) -o $@ $<

clean:
    @rm -f $(TARGET)
    @rm -f *.mod *.o
```

Este otro lo he usado más y me da buenos resultados

```
FC = gfortran
CAF = caf
SRC = $(wildcard src/*.f)
FILES_NO_EXT = $(notdir $(SRC))
TARGETS = $(addprefix app/,$(FILES_NO_EXT:.f=))

# Modules
MOD_SRC = $(wildcard src/mod/*.f)
MOD_TARGETS = $(addprefix app/mod/,$(notdir $(MOD_SRC:.f=.mod)))

# Arguments
#ARGS = -Wall -Wextra -std=f2018 -O3 -pedantic -fdec-math -ffree-form -Imod
ARGS = -std=f2018 -ffree-form -Imod


all: app mod $(TARGETS)

# Make target directory
app mod:
    @mkdir -p app
    @mkdir -p mod

# Caso genérico
app/%: src/%.f
    $(FC) $(ARGS) $< -o $@

clean:
    @rm -f $(TARGETS)
    @rm -f mod/*.mod mod/*.o
```

Language Server

Se encuentra en el paquete `fortran-language-server`. Puede ser arrancado mediante la orden `fortls`

El paquete contiene estos archivos

```
samuel@hp-i5:~$ apt-file list fortran-language-server

fortran-language-server: /usr/bin/fortls
fortran-language-server: /usr/lib/python3/dist-packages/fortls/__init__.py
fortran-language-server: /usr/lib/python3/dist-packages/fortls/intrinsic_funs.json
fortran-language-server: /usr/lib/python3/dist-packages/fortls/intrinsic_mods.json
fortran-language-server: /usr/lib/python3/dist-packages/fortls/intrinsics.py
fortran-language-server: /usr/lib/python3/dist-packages/fortls/jsonrpc.py
fortran-language-server: /usr/lib/python3/dist-packages/fortls/keywords.json
fortran-language-server: /usr/lib/python3/dist-packages/fortls/langserver.py
fortran-language-server: /usr/lib/python3/dist-packages/fortls/objects.py
fortran-language-server: /usr/lib/python3/dist-packages/fortls/parse_fortran.py
fortran-language-server: /usr/lib/python3/dist-packages/fortls/statements.json
fortran-language-server: /usr/lib/python3/dist-packages/fortran_language_server-
1.10.3.egg-info/PKG-INFO
fortran-language-server: /usr/lib/python3/dist-packages/fortran_language_server-
1.10.3.egg-info/dependency_links.txt
fortran-language-server: /usr/lib/python3/dist-packages/fortran_language_server-
1.10.3.egg-info/entry_points.txt
fortran-language-server: /usr/lib/python3/dist-packages/fortran_language_server-
1.10.3.egg-info/requires.txt
fortran-language-server: /usr/lib/python3/dist-packages/fortran_language_server-
1.10.3.egg-info/top_level.txt
fortran-language-server: /usr/share/doc/fortran-language-server/README.rst.gz
fortran-language-server: /usr/share/doc/fortran-language-server/changelog.Debian.gz
fortran-language-server: /usr/share/doc/fortran-language-server/copyright
fortran-language-server: /usr/share/man/man1/fortls.1.gz
```

Tipos

Tipos ISO

```
program Tipos_iso
    use iso_fortran_env
    integer, parameter:: N = 3
    real(real64):: A(N,N)
    integer(int64):: B(N,N)

    A = .0_real64       ! Matriz de ceros reales
    B = 0_int64     ! Matriz de ceros enteros

    write (*,"(3f6.1)") A   ! Mostrar la matriz en tres filas y tres columnas
    write (*,"(3i6)") B     ! Mostrar la matriz en tres filas y tres columnas
end program Tipos_iso
```

Tipos derivados

```
type Persona
    character(len=6):: id
    character(len=10):: nombre
    real::              edad
end type Persona
```

Variable compleja

```
program coseno
    complex w, z


    ! Entrada
    write (*, "(a)", advance="no") "Valor real de z: "
    read *, z%re
    write (*, "(a)", advance="no") "Valor imaginario de z: "
    read *, z%im

    ! También se puede usar la función intrínseca
    ! z = cmplx(a, b)

    ! Cálculos
    w = cos(z)

    ! Salida
    write (*,"(a)") "w=cos(z)"
    write (*,"(a,f4.2,sp,f5.2,a)") "z:", real(z), aimag(z), "i"
    write (*,"(a,f4.2,sp,f5.2,a)") "w:", real(w), aimag(w), "i"

end program coseno
```

Strings

Se declaran así

```
program Strings
    use iso_fortran_env
    implicit none

    character(len=5):: s            ! Una cadena
    character(len=:):: s_array(:)   ! Un array de cadenas

    ! SETUP -----------
    ! Reserva espacio para la 10 cadenas de 5 caractéres cada una
    allocate(character(len=5):: s_array(10))

end program Strings
```

Estructuras de datos

Estructura

```
type :: persona
    character(len=12):: nombre, apellido
    integer:: edad
end type
```

Estructura paramétrica o template

Los parámetros de una estructura de este tipo pueden tener el atributo **len** o **kind** para especificar la longitud de una cadena o la de un tipo de datos como las **template** de C++.

```
program parametric_structure
    type :: persona(n)
        integer, len:: n = 10     ! Parámetro de la estructura con valor por defecto

        character(len=n):: nombre, apellidos
        integer:: edad
    end type

    type(persona(20)) :: p1

    p1 % nombre = "Samuel"

end program parametric_structure
```

Array allocatable

Se trata de un array que reserva memoria en tiempo de ejecución. Tampoco es necesario especificar su tamaño en el código.

Reservar memoria dos veces para el mismo array genera un error, por eso es conveniente comprobar antes de reservar la memoria si ya ha sido previamente reservada.

```
program allocatable_array
    integer, allocatable:: a(:)
    real, allocatable::    b(:)
    real, parameter:: pi = 3.141592
    integer:: i

    a = [(i, i=1,10)]                    ! a = [1, 2, .., 9, 10]
    b = [(sin(2*pi*i/1000.), i=0,1000)]  ! b = [0, 6.28e-3, ...]

    if (.not. allocated(a)) then         ! Para evitar error de doble asignación
        allocate(a(10))
    end if
end program allocatable_array
```

Array de estructuras

```
type :: body
  character(len =4) :: units
  real :: mass
  real :: pos(3), vel(3)
end type body
```

y la forma de usarlo es

```
type(body), allocatable :: vector(:)
allocate(vector(n))
```

Estructura de arrays

Permite vectorizar y optimizar código. En este caso se usa una estructura paramétrica

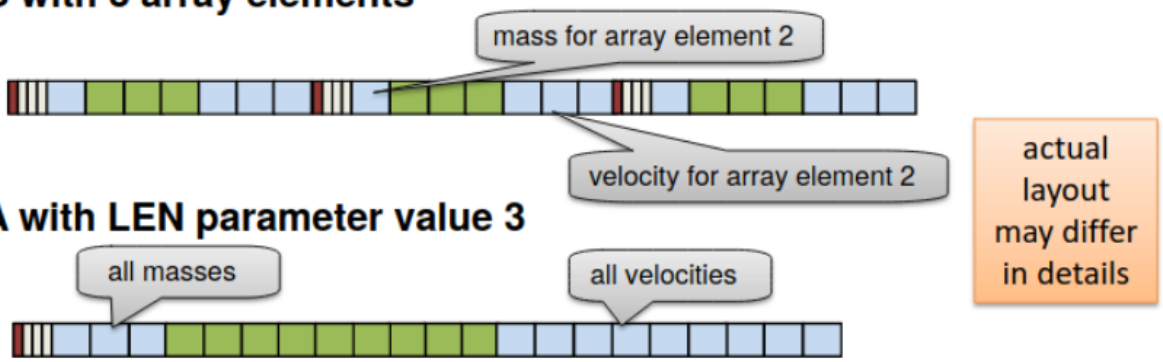
```
type :: body_p(k, n)                ! Estructura paramétrica template
  integer, kind :: k = kind(1.0)    ! Tipo
  integer, len ::   n = 1           ! Número

  character(len=4) :: units
  real(kind=k) :: mass(n)
  real(kind=k) :: pos(n,3),   vel (n,3)
end type body_p
```

y se usa de esta manera

```
type(body_p(n=:)), allocatable ::  vector
allocate(body_p(n=20) :: vector)
```

- **AoS with 3 array elements**



- **SoA with LEN parameter value 3**

Entrada y salida

Table 1. Formatos de entrada y salida

Modificador	Descripción
A	Cadena
Fa,b	Real de anchura total a y número de decimales b
SP	Forzar signo positivo

Coarrays

Varias instancias de ejecución en paralelo

```
program pi_sum
    integer, parameter:: limit = 1000
    integer:: i
    real:: pi[*]

    do i = this_image(), limit, num_images()
        pi = pi + (-1)**(i+1) / real( 2*i-1)
    end do
    sync all

    ! global barrier
    if (this_image() == 1) then
        do i = 2, num_images()
            pi = pi + pi[i]
        end do
        pi = pi * 4.0
        print *, "Result", pi
    end if

end program pi_sum
```

Tools

Medida del tiempo en ejecución cpu_time

```
real(REAL64):: t(2)

call cpu_time(t(1))
call test
call cpu_time(t(2))
print '("time iamax: ",G0)', t2-t1
```

Resources

- Librerías
- Entrada y salida con formato
- Format
- Coarrays
- Open Coarrays

OpenBlas

Para instalar OpenBlas debes seguir los siguientes pasos

Descargar fuente de Mpich y después

1. ./configure
2. make
3. sudo make install

Plugin de identación para VIM

Tiene algunas teclas asignadas como son:

- \= para identar todo el fichero fuente.
- \c comentar la línea
- \f cambiar flags del plugin de identado
- \w cambiar (toggle) identación de línea o fichero completo

Para más instrucciones puedes hacer '$ findent --vim_help'

He seguido las instrucciones de Source Forge que es el proyecto de Willem Vermin, pero parece que también existe otro proyecto del mismo autor en Identación con FIndent

```
vimroot=$HOME/.vim
mkdir -p $vimroot/plugin
findent --vim_findent > $vimroot/plugin/findent.vim
mkdir -p $vimroot/after/indent
findent --vim_fortran > $vimroot/after/indent/fortran.vim
```