

## Práctica 0

# Ciclo de desarrollo de un programa Ada

### 0.1 Objetivos

Con esta práctica dirigida se pretende que el alumno:

- conozca el entorno de desarrollo de aplicaciones con lenguaje Ada que hay instalado en el Laboratorio de Programación de Sistemas en Tiempo Real,
- practique con este entorno y aprenda el ciclo de implementación de un programa.

### 0.2 Desarrollo teórico

Designamos por ciclo de creación de un programa al conjunto de pasos que se deben realizar para implementarlo. La implementación es la cuarta de las cinco actividades estudiadas para desarrollar una aplicación de tiempo real:

1. Especificación de requisitos.
2. Análisis.
3. Diseño.
4. Implementación.
5. Pruebas.

La etapa de implementación involucra tres acciones principales:

1. Edición.
2. Compilación, encuadernación (*binding*) y enlazado.
3. Corrección de los errores de compilación y enlazado.

Para realizar estas acciones el alumno dispone en el laboratorio del compilador de Ada GNAT para LINUX que posee las siguientes características:

- GNAT es un compilador de Ada 95 integrado en el sistema de desarrollo GCC.
- GNAT está disponible para una amplia variedad de plataformas hardware y sistemas operativos y puede ser usado como compilador cruzado (*cross-compiler*) desde cualquier máquina origen hacia cualquier máquina destino.
- GNAT dispone de una interfaz excelente para realizar desarrollo multilenguaje: Ada 95, C, C++, Fortran, etc.

- GNAT (fuentes y ejecutables) se distribuyen gratuitamente a través de Internet.
- GCC es un sistema de desarrollo altamente transportable con múltiples frentes (*front-ends*): C, C++, Objective-C, Fortran77, Pascal, Chill y Ada 95.
- GCC produce código para muchos tipos de procesadores:
  - CISC: Intel x86, Motorola 68k.
  - RISC: PowerPC, DEC Alpha, HP-PA RISC, RS600, MIPS.
- GCC (fuentes y ejecutables) también se distribuyen gratuitamente a través de Internet.

Como primer ejemplo para familiarizarnos con el entorno podemos editar el programa siguiente:<sup>1</sup>

```
$ gedit numeros.adb
```

---

**Programa 1:** Primer programa de ejemplo (numeros.adb)

---

```

1  -- Este programa pretende ilustrar la forma de manejo de las herramientas
2  -- del entorno de desarrollo en Ada.
3
4  -- Paquete estándar que vamos a manejar.
5  with Ada.Text_IO; use Ada.Text_IO;
6
7  -- Procedimiento principal.
8  procedure numeros is
9
10     -- Creación de ejemplares, para la entrada/salida de números, a partir
11     -- de los paquete genéricos "Ada.Text_IO.Integer_IO" y
12     -- "Ada.Text_IO.Float_IO".
13     package Ent_Es is new Ada.Text_IO.Integer_IO(Integer);
14     package Real_Es is new Ada.Text_IO.Float_IO(Float);
15
16
17     -- Declaración de variables locales.
18     I, J: Integer;
19     X, Y: Float;
20
21     -- Cuerpo del procedimiento "Numeros".
22     begin
23         -- Entrada/salida de números enteros.
24         -- Lectura de datos
25         New_Line;
26         Put ("Introduce un número entero: ");
27         Ent_Es.Get (I);
28         Put ("Introduce otro número entero: ");
29         Ent_Es.Get (J);
30         New_Line;
31         -- Presentación de los resultados en el sistema decimal
32         Ent_Es.Put (I); Put ('+');
33         Ent_Es.Put (J); Put ('=');
34         Ent_Es.Put (I+J);
35         New_Line;
36

```

---

<sup>1</sup>El alumno no debe preocuparse si no entiende el texto del programa en su totalidad ya que algunas partes del mismo serán explicadas con posterioridad y el objetivo de esta práctica es otro.

```

37  -- Presentación de los resultados en binario
38  Ent_Es.Put (I, Base => 2);
39  Put ('+');
40  Ent_Es.Put (J, Base => 2);
41  Put ('=');
42  Ent_Es.Put (I+J, Base => 2);
43  New_Line;
44
45  -- Presentación de los resultados en hexadecimal
46  Ent_Es.Put (I, Base => 16);
47  Put ('+');
48  Ent_Es.Put (J, Base => 16);
49  Put ('=');
50  Ent_Es.Put (I+J, Base => 16);
51  New_Line;
52
53  -- Presentación de los resultados con formato
54  Ent_Es.Put (I);
55  Put ('+');
56  Ent_Es.Put (J, Width => 0);
57  Put ('=');
58  Ent_Es.Put (I+J, Width => 6);
59  New_Line; New_Line;
60
61  -- Entrada/salida de números reales.
62  -- Lectura de los datos
63  Put ("Introduce un número real: ");
64  Real_Es.Get (X);
65  Put ("Introduce otro número real: ");
66  Real_Es.Get (Y);
67  New_Line;
68  -- Presentación de los resultados
69  Real_Es.Put (X); Put ('/'); Real_Es.Put (Y); Put ('=');
70  Real_Es.Put (X/Y);
71  New_Line;
72  -- Presentación de los resultados con formato
73  Real_Es.Put (X, Fore => 2, Aft => 2, Exp => 3);
74  Put ('/');
75  Real_Es.Put (Y, Fore => 0, Aft => 0, Exp => 0);
76  Put ('=');
77  Real_Es.Put (X/Y, Fore => 10, Aft => 4, Exp => 0);
78  New_Line; New_Line;
79  end numeros;

```

---

- Para crear, a partir de este fichero, un programa ejecutable, tenemos que compilarlo y enlazarlo con los elementos de la biblioteca.
- La orden para compilar es gcc:

```
$ gcc -c numeros.adb
```

- Esto genera los ficheros intermedios: `numeros.ali` y `numeros.o`.
- El parámetro `-c` le indica a `gcc` que compile pero no enlace.

- `numeros.o` es el fichero objeto.
- `numeros.ali` contiene información sobre las relaciones que hay entre las distintas unidades de compilación (dependencias entre unidades).

*ALI* → *Ada Library Information*.

Las dependencias son usadas por el encuadernador (*binder*) para verificar la consistencia de las unidades y determinar el orden de elaboración de las mismas.

- La orden para encuadernar es `gnatbind`:

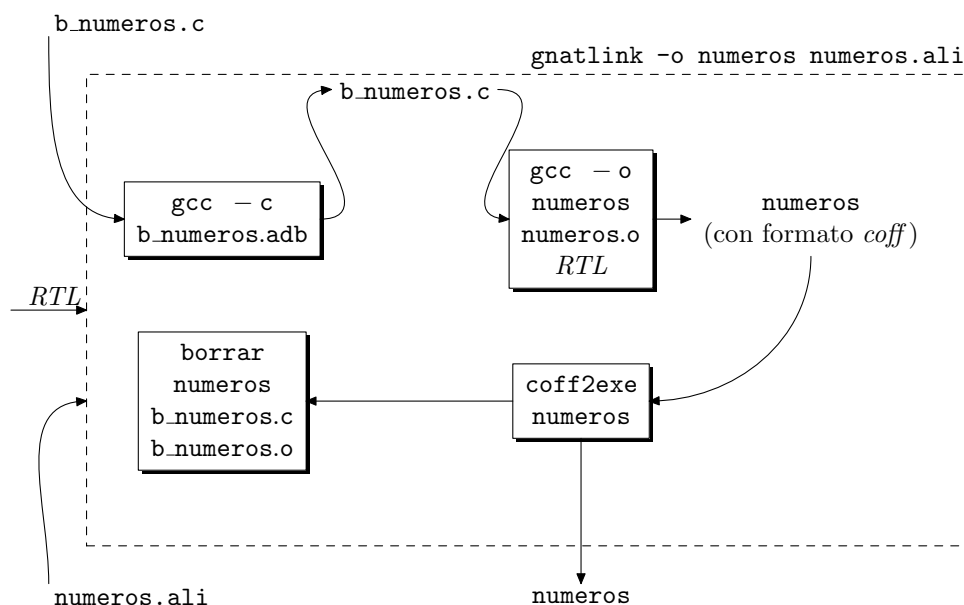
```
$ gnatbind -x numeros.ali
```

- Esto genera el fichero intermedio `B_numeros.c`.
- `B_numeros.c` contiene las llamadas necesarias a los procedimientos de elaboración. Al final de `B_numeros.c` aparece una lista de ficheros objeto dependientes entre sí.
- El parámetro `-x` le indica a `gnatbind` que verifique sólo la consistencia de los objetos sin tener en cuenta los ficheros fuente.

- La orden para enlazar es `gnatlink`:

```
$ gnatlink -o numeros numeros.ali
```

- Esto genera el fichero ejecutable `numeros`.
- `gnatlink` compila `B_numeros.c` y enlaza el fichero objeto `B_numeros.o` con el resto de ficheros enumerados en el fichero de encuadernación (`B_numeros.c`).
- Después de enlazar, los ficheros `B_numeros.c` y `B_numeros.o` son borrados.
- El enlazado se completa llamando a `gcc` y pasándole los ficheros objeto y las bibliotecas necesarias para crear el fichero ejecutable resultante. En la figura 1 podemos ver las órdenes que se ejecutan de forma automática al llamar a `gnatlink`.



**Fig. 1:** Proceso de enlazado.

- Para ejecutar el programa basta con escribir:

```
$ ./numeros
```

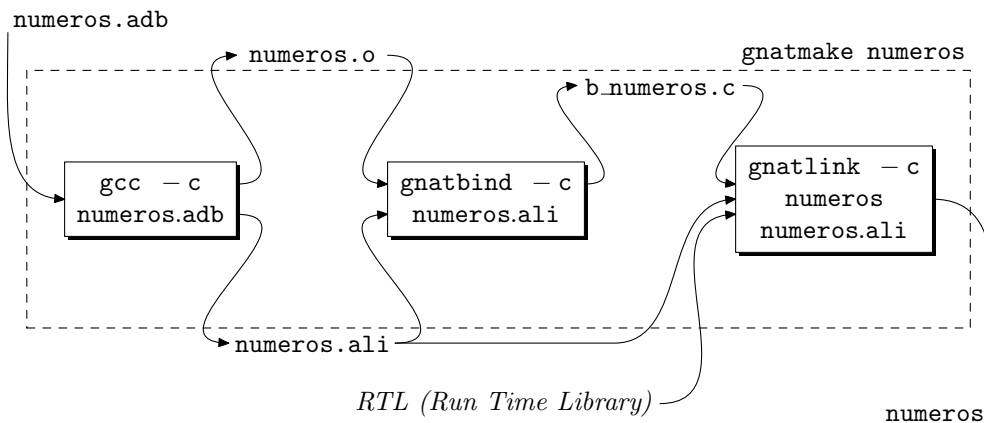
- Una forma automática de compilar, encuadernar y enlazar es con la orden **gnatmake**. Esta orden es especialmente útil a la hora de compilar aplicaciones multimodulares ya que la compilación y recompilación se realiza en función de las necesidades según los ficheros que se hayan modificado y las dependencias que hay entre ellos:

```
$ gnatmake numeros.adb
```

```
o
```

```
$ gnatmake numeros (numeros es el nombre del procedimiento de entrada a la aplicación, procedi-
miento principal)
```

En la figura 2 podemos ver las órdenes que se ejecutan de forma automática al llamar a **gnatmake**.



**Fig. 2:** Proceso de compilación, encuadernación y enlazado.

### 0.3 Tareas a realizar

En esta práctica se tienen que realizar las tareas siguientes:

1. Editar un fichero de nombre **numeros.adb** con un texto como el indicado en el desarrollo teórico anterior.
2. Compilar, encuadernar y enlazar **numeros.adb**.
3. Ejecutar el programa **numeros**.
4. Cuando llamamos al procedimiento Put de la línea 26 ¿en qué paquete se encuentra este procedimiento?
5. Si quisiéramos poder reescribir la línea 34 para que quedara Put(I+J) ¿qué deberíamos añadir en el código y donde? Deja la línea 34 como se indica en esta cuestión y haz los cambios necesarios para que el programa funcione.
6. En general ¿cómo crees que el lenguaje sabe a qué procedimiento Put tiene que llamar si no se lo indicamos en el código?

### 0.4 Obtención del compilador

El compilador del lenguaje Ada que hay instalado en el laboratorio forma parte de la distribución UBUNTU y se puede instalar en cualquier distribución con la orden:

```
$ sudo apt-get install gnat-7
```

Pueden instalarse múltiples versiones de gnat (gnat-X.X) siendo recomendable utilizar una versión superior a la 4. También puede instalarse con apt:

```
$ sudo apt install gnat-5
```

También existen compiladores para WINDOWS. En [www.macada.org](http://www.macada.org) está disponible el compilador para OS X.