Primeros pasos en la programación

Guillem Borrell Nogueras

31 de agosto de 2007

Resumen

Durante el primer contacto con la programación uno suele recibir un peligroso exceso de información. Cada entorno de trabajo exige una adaptación puesto que obliga a un método de trabajo particular. Los entornos de desarrollo o IDEs están orientados a facilitar y dinamizar el proceso de creación de programas pero no lo son todo. El proceso esencial sigue siendo el mismo que cuando se creó el arte de la programación: el diseño del algoritmo, le escritura del programa y la depuración de la ejecución.

1. El primer paso es siempre diseñar el algoritmo

Existe una diferencia **esencial** entre las aplicaciones propias del uso del ordenador (sistema operativo, entorno de escritorio...) y los programas de simulación. En el primer caso no existe ningún algoritmo, el programa es una serie de comandos de comunicación con el sistema operativo, una sucesión de llamadas a una biblioteca con unos determinados interfaces. En los programas de simulación el entorno es completamente distinto; el lenguaje de comunicación son las matemáticas y el entorno es mucho más cerrado.

¿Qué es lo más importante de una aplicación? La estructura; porque no existe algoritmo alguno, todo son llamadas a funciones. ¿Qué es lo más importante de un programa de simulación? El algoritmo. Los programas de simulación no son más que algoritmos puros implementados utilizando un lenguaje de programación. Estos dos caracteres implican una diferencia esencial. Los programas de simulación o los que realizan tareas puramente matemáticas no dependen del lenguaje de programación. Las diferencias existen sólo en la manera de escribirlo pero nunca en qué es lo que está haciendo el programa.

Un programa de simulación no va a ser mejor o peor dependiendo del lenguaje de programación. De hecho se habla de lenguajes más o menos adecuados para implementar un algoritmo. Si escribimos aplicaciones puramente matriciales que muevan grandes cantidades de memoria FORTRAN es el lenguaje adecuado. Si queremos crear una aplicación sencilla para probar un algoritmo optaremos por un lenguaje interpretado. En el caso de necesitar una aplicación muy sofisticada es mejor escoger un lenguaje polivalente y estructurado como C o C++.

Que nada nos haga perder de vista que lo más importante es el algoritmo. Ningún lenguaje de programación es capaz de convertir en bueno un algoritmo mediocre, al igual que es muy complicado estropear un algoritmo brillante. Por desgracia, los algoritmos brillantes suelen basarse en *ideas felices* que ahorran memoria, tiempo de

desarrollo y tiempo de implementación a la vez. La capacidad de encontrar ideas felices está muy influida por la experiencia que tengamos en el diseño de algoritmos, nunca en la pericia que tengamos programando[1].

2. Sólo necesitamos un editor y un compilador

Lo métodos de trabajo de Windows han contaminado peligrosamente el desarrollo de aplicaciones. En los albores de la informática no existían las interfaces gráficas. ¿Cómo se trabajaba en aquél entonces? ¿Cómo era posible crear aplicaciones tan sofisticadas como los primeros sistemas operativos sin la ayuda de nigún soporte gráfico?.

Cuando creamos un proyecto, sea en el lenguaje que sea, la primera pregunta siempre es la misma... ¿Es una aplicación con o sin interfaz gráfica? Esta primera pregunta genera otra ¿Si estamos construyendo una aplicación sin interfaz gráfica, por qué nos servimos de una para crearla? Todos los compiladores, editores, gestores de procesos, debuggers... existen independientemente de la interfaz gráfica. Las ventanas son *añadidas* a una aplicación que no las necesita para nada. En muchos casos son beneficiosas, en otros no hacen más que distraernos de lo que es verdaderamente esencial: el compilador.

Supongamos que estamos delante de un programa de prueba escrito en FORTRAN, en concreto en Fortran 95, última variante del lenguaje publicada. El código es el siguiente:

```
program test
  real :: rndn

  call random_number(rndn)
  write(*,*) 'esto es un numero aleatorio',rndn
end program test
```

Si estamos utilizando un entorno de desarrollo como el Microsoft Visual Studio, o que es lo mismo, el compilador Intel-Compaq-Digital[2], primero tendremos que crear un Workspace, luego añadirle un archivo... Demasiados pasos. Una tarea tan sencilla no puede requerir tantos artificios. ¿Y si intentamos simplificar más el proceso?. ¿Qué sistema operativo estamos utilizando? Supongamos que esa bazofia con colorines y un Emule llamada Windows. Primero instalaremos un compilador por consola[3] y encenderemos cualqier editor; en el presente caso Scite. En el editor escribiremos el código fuente del programa ejemplo como apreciamos en la figura 1. Cualquier editor es perfectamente válido aunque es recomendable utilizar alguno con soporte para el lenguaje en el que queremos programar.

[1] Matlab es una pequeña excepción, un algoritmo brillante puede estropearse con una programación torpe en el uso de la notación matricial

[2] Uno de los compiladores más populares para MS Windows es el compilador de Digital, el Visual Fortran. Digital utilizaba el entorno de desarrollo Visual Studio de Microsoft para añadir una interfaz gráfica a la criatura. El compilador era derivado de la plataforma Alpha, de 64 bits; una arquitectura de gran calidad. Degital desapareció cuando fue comprada por Compaq. La siguiente versión del compilador se llamaba ya Compaq visual Fortran. El siguiente paso de la historia fue la compra de Compaq por parte de Hewlett Packard que a diferencia de Compaq no siguió vendiendo el compilador. El equipo de desarrolladores de uno de los mejores compiladores fue contratado casi en pleno por Intel que actualmente cuenta con el compilador de Fortran más exitoso de la actualidad

[3] Hay un apéndice sobre cómo instalar la versión para Windows de gfortran

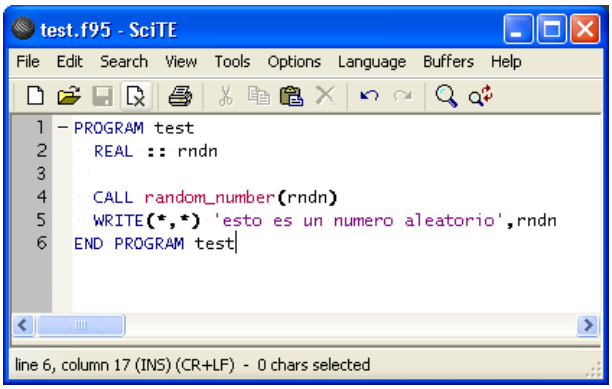

Figura 1: Scite con el programa ejemplo

Lo siguiente es llamar al compilador... ¿Como? ¡Si ha desaparecido el entorno gráfico! Ahora todo parece en el limbo informático, sólo tenemos un archivo en un directorio con la capacidad de convertirse en un programa. El gran defecto de Windows es que ha convertido la interfaz gráfica en una necesidad que nunca ha existido realmente. Los compiladores son programas presentes desde los albores de la informática. Además, Fortran tiene el privilegio de ser el primer lenguaje de programación de alto nivel de la historia, su nacimiento se remonta a finales de los años sesenta.

2.1. Compilar mediante la consola y Salford FTN95

Vamos a compilar un programa como se ha hecho durante décadas y como se sigue haciendo en cualquier sistema operativo que no sea Windows, mediante la consola del sistema.

El compilador de Fortran 95 de Salford no solo tiene una buena interfaz gráfica para Windows sino que además es una aplicación para su intèrprete de comandos. Su nombre para Windows es ftn95. Estemos en el directorio que estemos, tecleando `ftn95` accederemos al compilador. Pero no a la interfaz gráfica del compilador, el IDE o Integrated Development Environment sino al núcleo en sí del compilador completamente independiente del resto de herramientas. Los compiladores suelen tener centenares de opciones de compilación cuando se usan por consola, a nosotros sólo nos interesará la opción `/LINK` que nos permitirá crear el ejecutable final del programa.

Lo único que tenemos que hacer es llamar al compilador por su nombre y pasarle como argumento el nombre del archivo. Encontramos un esquema de la operación en la figura **??** Primero nos hemos dirigido al directorio donde guardamos el archivo con el código fuente del programa y luego hemos tecleado:

```
C:\...\Mis Documentos> ftn95 test.f95 /LINK
```

Esto ha generado un archivo llamado `test.exe`, un archivo que podemos ejecutar sólo escribiendo su nombre:

```
C:\...\Mis Documentos> test.exe
 esto es un numero aleatorio 0.673070
```

No necesitamos nada más en absoluto.

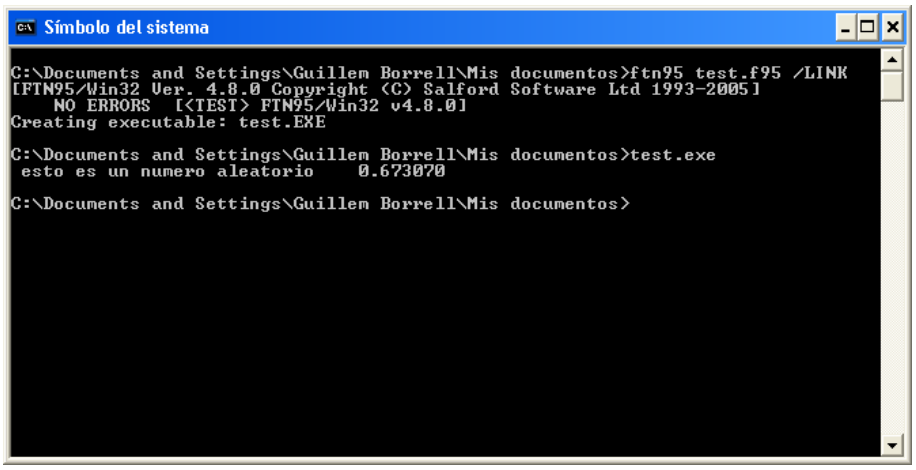

Figura 2: El proceso de compilación mediante la consola del sistema

2.2. Gfortran, el compilador libre

La colección de compiladores GCC (The GNU Compiler Collection) dispone desde la versión 4.0 de un compilador de Fortran 95. La versión anterior del compilador, la 3.4, disponía del g77 que era sólo capaz de compilar código en fortran 77 con algunas extensiones. Uno de los motivos del progresivo abandono de fortran para uso científico en favor de C ha sido la inexistencia de un compilador de uso libre y portable. Gfortran se lleva fatal con Windows, sin embargo en el apéndice teneis un manual bastante conciso de cómo instalarlo.

El manual de instrucciones se parece bastante al del ftn95 aunque para hacer un ejecutable no necesitamos ni el `/LINK`, es lo suficientemente listo como para darse cuenta de que el código contiene un archivo de programa y no una función o una subrutina. La figura 2 contiene toda la información necesaria.

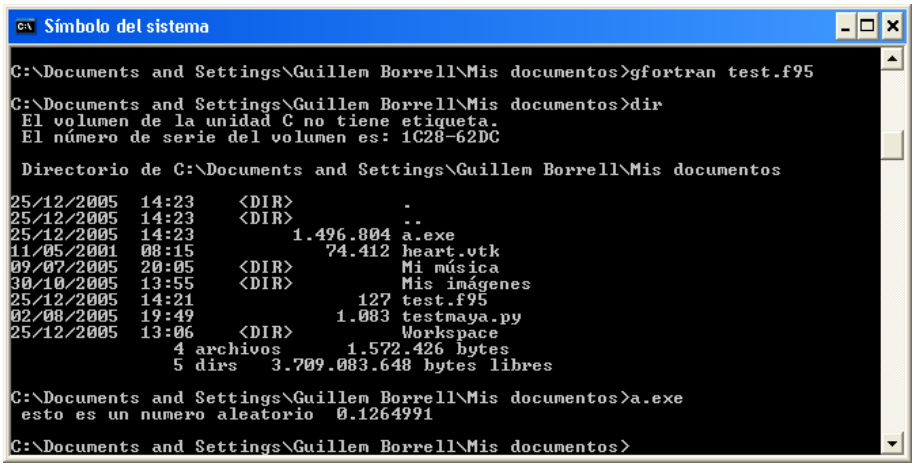

Figura 3: El proceso de compilación mediante la consola del sistema

3. ¿Qué está haciendo el compilador?

Otro defecto importante del uso de la interfaz gráfica es que se nos está escondiendo el verdadero proceso de compilación. Detrás de las ventanas todo parece que suceda como por arte de magia. Nada más lejos de la verdad, el compilador es una herramienta cuyos pasos son más que conocidos. ¿Cuál es el primer paso de un compilador con un archivo de código fuente? Transformarlo en un archivo objeto ¿Qué es un archivo objeto? Un archivo intermedio que un *linker* puede convertir en una unidad de programa o en una parte de él. Ahora hay que explicar estos conceptos.

Los compiladores se dividen en dos subprogramas, el compilador propiamente dicho y el linker. El compilador convierte código en código objeto como ya hemos dicho. ¿Por qué existe entonces el linker? Supongamos que tenemos nuestro programa esparcido en decenas de archivos. El compilador no creará un único archivo objeto sino que creará uno para cada archivo de código. El compilador es mucho más dedicado de lo que parece, sólo se encarga de procesar el código. El linker es el programa que junta todos estos archivos objeto, distingue cuál es la unidad principal del programa y los convierte en un archivo ejecutable.

Es capaz además de acoplar otros archivos objeto que no hayamos escrito nosotros mismos presentes en librerías estáticas. Existen dos tipos de librerías, las estáticas y las dinámicas según se utilicen en tiempo de compilación o de ejecución. Las estáticas se acoplan al programa en tiempo de compilación, es decir; no necesitamos la librería para ejecutar el programa porque ya está dentro del programa.

Estos pasos no son ni mucho menos confusos, como todo en la vida requiere un aprendizaje. Sin embargo creo que es un error olvidarnos de qué es lo que hace el compilador simplemente porque somos capaces de hacerlo todo con un par de clicks. Debemos acostumbrarnos a trabajar del modo más universal posible y depender de una interfaz gráfica no lo es.

Apéndice. Instalar gfortran en Windows

Para descargar gfortran sólo tenemos que descargar el ejecutable que encontraremos en la página del proyecto: `http://gcc.gnu.org/wiki/GFortran#download`. En él encontraremos un instalador con una versión bastante reciente del compilador, instalarlo es trivial. Una vez tengamos el compilador en nuestro ordenador es muy importante que la consola sea capaz de llamarlo desde cualquier directorio, para ello debemos hacer lo siguiente:

1. En el Panel de Control cambiaremos a vista clásica y entraremos en el menú de Configuración del sistema (figura **??**)

2. Seleccionaremos la pestaña de Opciones Avanzadas

3. Abriremos el menú de Variables de entorno

4. En el menú de Variables de entorno (figura **??**) nos aseguraremos de que en la variable del usuario `PATH` tenga como valor `C:\Archivos de Programa\gfortran\bin`. Si esto no fuera así la añadiremos con el diálogo de Nueva. Si despues de todo lo anterior abrimos un intérprete de comandos y despues de teclear `gfortran` obtenemos lo siguiente:

```
C:\...\Mis Documentos> gfortran
gfortran: no input files
```

es que el proceso de instalación es perfecto. Ahora ya podemos probar un compilador que sólo existe en la consola.

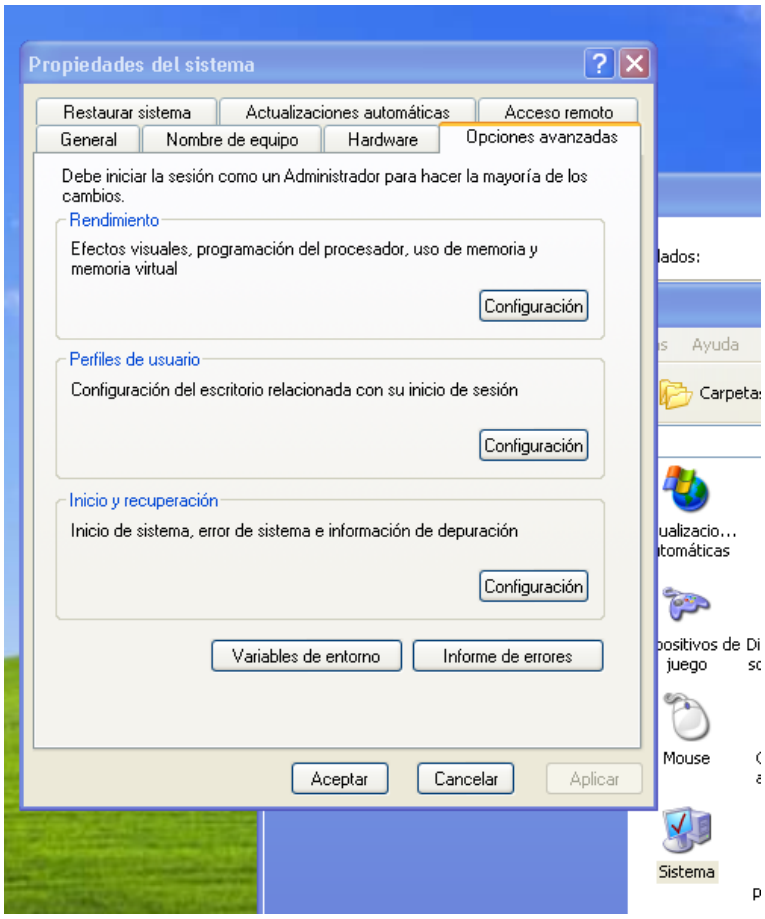

Figura 4: Diálogo de opciones del sistema.

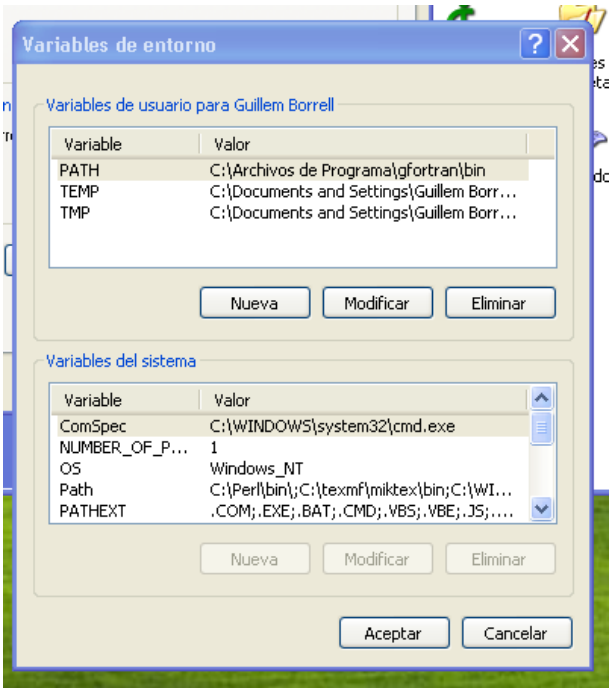

Figura 5: Configuración de las variales de entorno.