Informática

Clase 5: Entrada y salida de información

Mario Merino Martínez
mario.merino@upm.es

Escuela de Ingeniería Aeronáutica y del Espacio
Universidad Politécnica de Madrid

8 de noviembre de 2011

Índice

Abrir y cerrar archivos

- **Abrir un fichero** en Fortran consiste en asignarle una etiqueta numérica (**número de unidad**) entre 1 y 99 (salvo 5 y 6, reservados), con la orden **open**:

```
open(17,FILE='misdatos.txt')
```

- Tras escribir/leer del archivo, hemos de cerrarlo con la orden **close** y el **número de unidad**:

```
close(17)
```

Open y close poseen múltiples opciones adicionales (podéis buscarlas en la ayuda de Plato)

Escribir y leer de archivos

- Para **escribir** a archivo en lugar de por pantalla, simplemente indicamos el número de unidad de un **archivo abierto** como primer argumento en `write` (`print` no sirve ahora):

```
write(17,*) x, y, z
```

- Para **leer** datos de un archivo en lugar de por teclado, igualmente usamos `read` con el número de unidad:

```
read(17,*) x, A(1,1), A(1,2)
```

Hasta ahora, usábamos la unidad por defecto (**asterisco** "*"): pantalla o teclado, respectivamente.

*Imprescindibles I

- `open`, `close`, `read`, `write` permiten trabajar con archivos
- Cada archivo recibe un número de unidad
- **No escribimos y leemos a la vez** de un mismo archivo: los resultados pueden ser inesperados. Normalmente, abrimos un archivo con idea de leer o escribir solamente.
- Al abrir un archivo con **open**, el **cursor de lectura/escritura** se sitúa al principio por defecto.
- **Cada ejecución** de `read` y `write` avanza una linea en el archivo: **hemos de leer/escribir filas enteras cada vez**

Lectura avanzada de archivos

El argumento opcional **iostat=*variable*** (una variable integer) en orden **read** permite conocer si:

- **Se leyó correctamente** la línea (devuelve *variable* = 0):
- Se llegó al **final del archivo** (*variable* = -1):

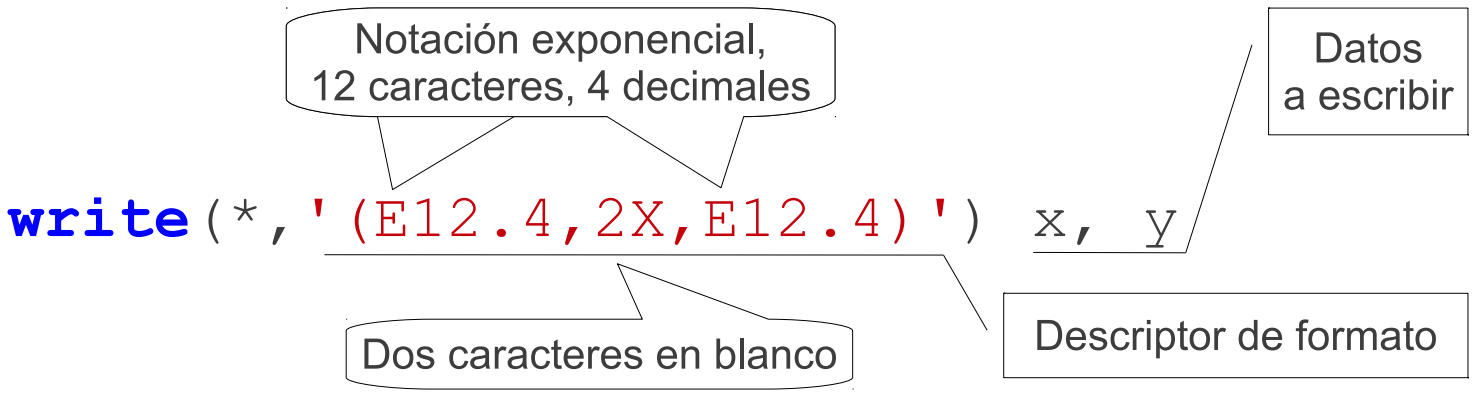
```
integer :: estado
open(17,file='archivo_existente.txt')
do
  read(17,*,iostat=estado) x, y, z, w
  print*, estado ! valdrá -1 al final del archivo
  pause
enddo
```

Esto es muy útil para terminar el bucle **do** con una condición **if** (estado == -1) **exit** **cuando hemos llegado al final del archivo**, *e.g. cuando se desconoce cómo de largo es.*

Formato de escritura/lectura

- "Formato de los datos" quiere decir **cómo se van a escribir o leer**: cuántos caracteres ocupa cada número, qué notación utilizar (exponencial o no, etc), cuantos espacios en blanco entre medias, etc.
- Las funciones `print _` , `write(*,_)` y `read(*,_)` admiten **como argumento** "_" una cadena de texto que especifique el formato a utilizar.
- Hasta ahora, no hemos utilizado formatos: **El asterisco "*" indicaba formato por defecto.**
- Salvo para leer archivos especiales, no se suele usar formato con read (se usa formato por defecto, "*")

Descriptores de formato I



- Los descriptores de formato son **cadenas de texto** (usar comillas).
- Contienen una **lista de especificadores** separados por comas. **Cada dato** a escribir requiere **su especificador**
- Los formatos empiezan y acaban **con paréntesis**.

Descriptores de formato II

Los **especificadores** consisten en una letra, un número n (número total de caracteres **incluyendo punto decimal, signo y exponente)**. Donde aparezca, d es el número de cifras decimales:

`I`n	Entero	`A`n	Caracteres
`F`$n.d$	Real ($n \geq d+3$ ó más)	`G`$n.d$	Cualquier tipo*
`E`$n.d$	Exponencial ($n \geq d+7$)*	`X`	Espacio en blanco
`D`$n.d$	Exp. doble precisión ($n \geq d+7$)*	/	Nueva linea

Es importante que n **sea suficientemente grande**. Si el número de caracteres es insuficiente, fortran **escribirá asteriscos en lugar de un número** (i.e., ******).
Útil: $n = 0$ calcula n automáticamente.

---

* *También se puede especificar el número de dígitos e del exponente:* `D`$n.d$`E`e

Descriptores de formato III

- **Un número** delante de un especificador o un grupo de ellos entre paréntesis **lo multiplica:** `5F8.4` ó `5(F5.2,2X)`
- Se puede introducir **texto literal** con dobles comillas “”, que será escrito junto con los datos:
  `'(F12.4, "x vale:", E12.4)'`
- Útil: declarar variable `character` (hay que indicar longitud) con nuestro **formato**:

```
character(80) :: fmt = '(F6.2,E12.4,D18.10)'
print fmt, x,y,z
```

- También existe la orden `format` con etiqueta numérica:

```
print 100, x,y,z
100 format (F6.2,E12.4,D18.10) ! Sin comillas
```

Ejemplos

$(x = 0.123456789;\ n = 123)$

```
print '("Resultado: x = ",F12.6,", n = ",I4)',x,n
```

⇒`Resultado: x = ____0.123457, n = _123`

$(x = (/1.2, 2.3, 3.4/);\ z = 12.345678)$

```
print '(A4,3(F8.4,2X), E12.5)','hola',x,z
```

⇒`hola__1.2000  __2.3000  __3.4000  _0.12346E+02`

Si n es excesivo, los caracteres extra se rellenan con espacios en blanco. Los decimales se redondean.

*Imprescindibles II

- Los **descriptores de formato** son cadenas de texto: **listas de especificadores entre paréntesis**.
- Ha de haber **tantos especificadores como datos** a escribir
- Importante: usar n elevado (o poner 0)
- **Multiplicadores**, **texto literal**, **variables character** y orden `format` facilitan el uso de formatos