Computación

Comandos de entrada y salida de datos, parte 2

Habíamos visto que para abrir un archivo, usábamos la sentencia OPEN() de la forma

open(22,file='Mi_archivo')

Esta orden conecta mi programa con el SO para trabajar con cierto archivo en particular

En la orden OPEN() hay muchos más parámetros y por lo tanto más opciones que las hasta ahora pudimos haber usado

Comandos de entrada y salida de datos, parte 2

Los datos que se escriben o leen siempre están en forma de **archivos**:

Hay dos tipos de archivos:

- De acceso **Secuencial** (los que estamos usando hasta ahora).
- De acceso **Directo**.

Secuenciales

Son los más comunes, usados para casi todo tipo de datos. La información va un Byte detrás del otro. En un archivo de texto las bajadas de renglón se deben a caracteres que no vemos pero que existen provocando comportamiento.

Problemas: para leer información útil pero que está al final de archivo hay que leerlo entero.
No es fácil modificar un valor en el medio de un archivo.

Acceso Directo

El archivo tiene un tamaño fijo de renglón, los renglones pueden leerse o escribirse independientemente uno del otro. Son mucho menos usados o conocidos, se usan con frecuencia para guardar datos astronómicos. La base del formato FITS se basa en archivos con un tamaño de 2880 Bytes

Sentencia open

Forma general

OPEN(UNIT=’número’, file=*variable de caracteres*, ERR= *entero*, IOSTAT=*variable entera*, STATUS= parámetro, ACCESS= parámetro, RECL= *entero*)

Hay más parámetros de los que estudiamos en la introducción que hicimos hace unas clases.
Pero estos no son obligatorios, ya hay tomado un “Default”
Algunos tienen un valor previamente elegido y otros son indicaciones de control (pueden informarnos de un error)

Los únicos parámetros obligatorios son la Unidad Lógica y el nombre del archivo

Hay que prestar atención que estás órdenes le permiten al Fortran “hablar” con el sistema operativo (SO) de la máquina donde corre y estos comandos pueden funcionar ligeramente diferente según el SO en el que está trabajando.

Los resultados y la información de problemas que podrían haber sucedido dependen del SO.

Sentencia OPEN()

OPEN(UNIT='número', file=*variable de caracteres*, ERR= *entero*, IOSTAT=*variable entera*, STATUS= parámetro, ACCESS= parámetro, RECL= *entero*)

UNIT: (unidad Lógica), es el número que se le asigna al archivo, para el uso en los READ, WRITE, etc. Algunos UNIT pueden ser hardware asignado a la computadora (ejemplo: 6 es la impresora).

FILE: Nombre del archivo o de la estructura de directorios y el archivo con el fin de navegar hasta él

../../home/Jorge/datos.txt Navego en forma relativa

/home/Carlos/datos2.txt Navego en forma absoluta. La estructura de directorios comienza en / (root <-> raíz)

OPEN(UNIT='número', file=*variable de caracteres*, ERR= *entero*, IOSTAT=*variable entera*, STATUS= parámetro, ACCESS= parámetro, RECL= *entero*)

ERR: Apunta a un número de sentencia a la que se salta en el caso de ocurrir un error, es como si fuese un "Goto". Si no está puesta y se produce un error el programa termina, si existe el programa continua en la sentencia indicada.

IOSTAT: Es un número que se guarda en una variable entera con el que puedo buscar en el manual del Fortran cuál fue el error que se produjo. Es muy útil en combinación con el **err** para resolver problemas con programas que se dejan mucho tiempo corriendo sin una supervisión humana. Si el IOSTAT es igual a 0, la orden se ejecutó sin problemas.

Gfortran Error Codes
When using READ in fortran error codes can returned in the value IOSTAT. These values are

-3 LIBERROR_FIRST = -3,
-2 LIBERROR_EOR = -2,
-1 LIBERROR_END = -1,
0 LIBERROR_OK = 0,
5000 LIBERROR_OS = 5000,
5001 LIBERROR_OPTION_CONFLICT
5002 LIBERROR_BAD_OPTION
5003 LIBERROR_MISSING_OPTION
5004 LIBERROR_ALREADY_OPEN
5005 LIBERROR_BAD_UNIT
5006 LIBERROR_FORMAT
5007 LIBERROR_BAD_ACTION
5008 LIBERROR_ENDFILE
5009 LIBERROR_BAD_US
5010 LIBERROR_READ_VALUE
5011 LIBERROR_READ_OVERFLOW
5012 LIBERROR_INTERNAL
5013 LIBERROR_INTERNAL_UNIT
5014 LIBERROR_ALLOCATION
5015 LIBERROR_DIRECT_EOR
5016 LIBERROR_SHORT_RECORD
5017 LIBERROR_CORRUPT_FILE
5018 LIBERROR_INQUIRE_INTERNAL_UNIT

as defined in gcc/fortran/libgfortran.h. (These values are from gfortran 4.6)

OPEN(UNIT='número', file=*variable de caracteres*, ERR= *entero*, IOSTAT=*variable entera*, STATUS= parámetro, ACCESS= parámetro, RECL= *entero*)

Status: puede ser **OLD**, **NEW**, **UNKNOWN** o **SCRATCH**. Si es OLD, el archivo tiene que existir, si no da error. Si dice NEW el archivo NO debe existir. Si dice UNKNOWN da lo mismo que esté o no, pero si está puede ser sobrescrito, perdiéndose la información anterior. Si dice SCRATCH se lo considera un borrador y desaparecerá al terminar de correr el programa. El default es 'UNKNOWN'.

ACCESS: Es un parámetro que indica si el archivo es '**DIRECT**' es decir de acceso directo o del tipo secuencial y se le indica: '**SEQUENTIAL**'. El default es 'SEQUENTIAL'.

RECL: Si el archivo es de acceso directo hay que indicar el tamaño de renglón en este comando. Ejemplo RECL=2880.

Sentencia CLOSE()

CLOSE(UNIT=’número’, STATUS=parámetro, ERR= *entero*, IOSTAT=*variable entera*)

UNIT es el único parámetro obligatorio. STATUS, ERR y IOSTAT funcionan igual que como se indicó anteriormente. Pero el STATUS tiene ahora dos parámetros ‘KEEP’ (default) para conservar el archivo y ‘DELETE’ para que se borre cuando se ejecuta el comando.

Pero ojo, que existe cache de disco!!!

Sentencias READ() y WRITE()

READ(UNIT= *Variable entera*, FMT=*Formato*, ERR= *Variable entera*, END=*Variable entera*, IOSTAT=*Variable entera*, REC=*Variable entera*) Lista de variables

WRITE(UNIT= *Variable entera*, FMT=*Formato*, ERR= *Variable entera*, END=*Variable entera*, IOSTAT=*Variable entera*, REC=*Variable entera*) Lista de variables

Hasta ahora hacíamos

read(*,*) x,y,x

write(*,*) x,y,z

Tenemos el Formato (el uso de los formatos lo verán en la práctica y como apéndice de los apuntes).

Lo que se agrega nuevo es la sentencia END, que nos indica un salto si el archivo se encuentra al final y ya no se puede leer o escribir.

REC en cambio es el número de renglón que escribo o leo y sólo se usa en el caso de archivos de acceso directo.

Ejemplo para usar el parámetro END

Veamos como leer un archivo secuencial en el cual no sé la cantidad de renglones que tiene.

```
...
      OPEN(23, FILE='datos.dat')
      I=1
10    READ(23,END=99) A(I),B(I)
      I=I+1
      GOTO 10
99    N= I - 1
...
```

Archivos con datos binarios

La menor cantidad de parámetros para leer o escribir es si lo hacemos en binario. Es decir, los datos no están en ASCII, no son por lo tanto accesibles con un editor común. Con la ventaja de que ocupan mucho menos lugar y son más rápidos al momento de leer o escribir.

READ(UNIT= ***número entero)*** Lista de variables

WRITE(UNIT= ***número entero)*** Lista de variables

Do implicito en lectura/escritura

Se permite el DO implícito en la escritura/lectura de esta forma:

```
read(22,*) (a(i), i=1,100)
write(*,*) ((C(i,j), i=1,10, j=1,10)
```

Lectura y escritura de archivos de acceso directo

Se debe primero indicar en la sentencia OPEN()

OPEN(22, file='3C299,fits', ACCESS='DIRECT',RECL=2880)

WRITE(22, REC=99) x,y

READ(22,REC=3) A, B

SENTENCIAS ESPECIALES:

REWIND permite rebobinar el archivo, empiezo a leer por el principio

REWIND(UNIT= ENTERO, ERR=ENTERO, IOSTAT=VARIABLE ENTERA)

BACKSPACE Vuelve un renglón para atrás

BACKSPACE(UNIT= ENTERO, ERR=ENTERO, IOSTAT=VARIABLE ENTERA)

Formatos
Sentencia Format()

Se usa así:

```
read(*,100) x,y,z
```

```
100   format(“serie de órdenes”)
```

Estas ordenes pueden códigos que actúan sobre el número o donde este se posiciona

Códigos para datos:

- Para enteros se usa el “In” donde n es la cantidad de dígitos
- Para reales se us el ”Fn.d” donde Nos la cantidad total de dígitos y d la cantidad de decimales.
- Para la notación científica se usa “En.d” pero con el cuidado que cuando se escribe el número terminara de la forma
- E+xx o E-xx y por lo tanto del “n”, debe descontar 4 lugares
- Para los caracteres se usa el formato A, en la forma “An”, donde n ahora es la cantidad de caracteres.

Códigos de posicionamiento:

Si bien hay muchos veremos sólo el “X” que indica la cantidad de lugar libre, por ejemplo espacios entre números

Por ejemplo:

```
      write(*,99) b1,b2

99 format(f8.3,3x,f6.2)
```