FORTRAN - FILE INPUT OUTPUT

Fortran allows you to read data from, and write data into files.

In the last chapter, you have seen how to read data from, and write data to the terminal. In this chapter you will study file input and output functionalities provided by Fortran.

You can read and write to one or more files. The OPEN, WRITE, READ and CLOSE statements allow you to achieve this.

Opening and Closing Files

Before using a file you must open the file. The **open** command is used to open files for reading or writing. The simplest form of the command is:

```
open (unit = number, file = "name").
```

However, the open statement may have a general form:

```
open (list-of-specifiers)
```

The following table describes the most commonly used specifiers:

Specifier	Description
[UNIT=] u	The unit number u could be any number in the range 9-99 and it indicates the file, you may choose any number but every open file in the program must have a unique number
IOSTAT= ios	It is the I/O status identifier and should be an integer variable. If the open statement is successful then the ios value returned is zero else a non-zero value.
ERR = err	It is a label to which the control jumps in case of any error.
FILE = fname	File name, a character string.
STATUS = sta	It shows the prior status of the file. A character string and can have one of the three values NEW, OLD or SCRATCH. A scratch file is created and deleted when closed or the program ends.
ACCESS = acc	It is the file access mode. Can have either of the two values, SEQUENTIAL or DIRECT. The default is SEQUENTIAL.
FORM= frm	It gives the formatting status of the file. Can have either of the two values FORMATTED or UNFORMATTED. The default is UNFORMATTED
RECL = rl	It specifies the length of each record in a direct access file.

After the file has been opened, it is accessed by read and write statements. Once done, it should be closed using the **close** statement.

The close statement has the following syntax:

```
close ([UNIT=]u[,IOSTAT=ios,ERR=err,STATUS=sta])
```

Please note that the parameters in brackets are optional.

Example

This example demonstrates opening a new file for writing some data into the file.

```
program outputdata
implicit none

   real, dimension(100) :: x, y
   real, dimension(100) :: p, q
   integer :: i

   ! data
   do i=1,100
      x(i) = i * 0.1
      y(i) = sin(x(i)) * (1-cos(x(i)/3.0))
   end do

   ! output data into a file
   open(1, file='data1.dat', status='new')
   do i=1,100
      write(1,*) x(i), y(i)
   end do

   close(1)

end program outputdata
```

When the above code is compiled and executed, it creates the file data1.dat and writes the x and y array values into it. And then closes the file.

Reading from and Writing into the File

The read and write statements respectively are used for reading from and writing into a file respectively.

They have the following syntax:

```
read ([UNIT=]u, [FMT=]fmt, IOSTAT=ios, ERR=err, END=s)
write([UNIT=]u, [FMT=]fmt, IOSTAT=ios, ERR=err, END=s)
```

Most of the specifiers have already been discussed in the above table.

The END=s specifier is a statement label where the program jumps, when it reaches end-of-file.

Example

This example demonstrates reading from and writing into a file.

In this program we read from the file, we created in the last example, data1.dat, and display it on screen.

```
program outputdata
implicit none

   real, dimension(100) :: x, y
   real, dimension(100) :: p, q
   integer :: i

   ! data
   do i=1,100
      x(i) = i * 0.1
      y(i) = sin(x(i)) * (1-cos(x(i)/3.0))
   end do

   ! output data into a file
   open(1, file='data1.dat', status='new')
   do i=1,100
      write(1,*) x(i), y(i)
   end do
```

```
   close(1)

   ! opening the file for reading
   open (2, file='data1.dat', status='old')

   do i=1,100
      read(2,*) p(i), q(i)
   end do

   close(2)

   do i=1,100
      write(*,*) p(i), q(i)
   end do

end program outputdata
```

When the above code is compiled and executed, it produces the following result:

```
0.100000001  5.54589933E-05
0.200000003  4.41325130E-04
0.300000012  1.47636665E-03
0.400000006  3.45637114E-03
0.500000000  6.64328877E-03
0.600000024  1.12552457E-02
0.699999988  1.74576249E-02
0.800000012  2.53552198E-02
0.900000036  3.49861123E-02
1.00000000   4.63171229E-02
1.10000002   5.92407547E-02
1.20000005   7.35742599E-02
1.30000007   8.90605897E-02
1.39999998   0.105371222
1.50000000   0.122110792
1.60000002   0.138823599
1.70000005   0.155002072
1.80000007   0.170096487
1.89999998   0.183526158
2.00000000   0.194692180
2.10000014   0.202990443
2.20000005   0.207826138
2.29999995   0.208628103
2.40000010   0.204863414
2.50000000   0.196052119
2.60000014   0.181780845
2.70000005   0.161716297
2.79999995   0.135617107
2.90000010   0.103344671
3.00000000   6.48725405E-02
3.10000014   2.02930309E-02
3.20000005  -3.01767997E-02
3.29999995  -8.61928314E-02
3.40000010  -0.147283033
3.50000000  -0.212848678
3.60000014  -0.282169819
3.70000005  -0.354410470
3.79999995  -0.428629100
3.90000010  -0.503789663
4.00000000  -0.578774154
4.09999990  -0.652400017
4.20000029  -0.723436713
4.30000019  -0.790623367
4.40000010  -0.852691114
4.50000000  -0.908382416
4.59999990  -0.956472993
4.70000029  -0.995793998
4.80000019  -1.02525222
4.90000010  -1.04385209
5.00000000  -1.05071592
```

```
5.09999990  -1.04510069
5.20000029  -1.02641726
5.30000019  -0.994243503
5.40000010  -0.948338211
5.50000000  -0.888650239
5.59999990  -0.815326691
5.70000029  -0.728716135
5.80000019  -0.629372001
5.90000010  -0.518047631
6.00000000  -0.395693362
6.09999990  -0.263447165
6.20000029  -0.122622721
6.30000019   2.53026206E-02
6.40000010   0.178709000
6.50000000   0.335851669
6.59999990   0.494883657
6.70000029   0.653881252
6.80000019   0.810866773
6.90000010   0.963840425
7.00000000   1.11080539
7.09999990   1.24979746
7.20000029   1.37891412
7.30000019   1.49633956
7.40000010   1.60037732
7.50000000   1.68947268
7.59999990   1.76223695
7.70000029   1.81747139
7.80000019   1.85418403
7.90000010   1.87160957
8.00000000   1.86922085
8.10000038   1.84674001
8.19999981   1.80414569
8.30000019   1.74167395
8.40000057   1.65982044
8.50000000   1.55933595
8.60000038   1.44121361
8.69999981   1.30668485
8.80000019   1.15719533
8.90000057   0.994394958
9.00000000   0.820112705
9.10000038   0.636327863
9.19999981   0.445154816
9.30000019   0.248800844
9.40000057   4.95488606E-02
9.50000000  -0.150278628
9.60000038  -0.348357052
9.69999981  -0.542378068
9.80000019  -0.730095863
9.90000057  -0.909344316
10.0000000  -1.07807255
```