File I/O

Carlos Cruz
Jules Kouatchou
Bruce Van Aartsen

NASA GSFC Code 606 (ASTG)
Greenbelt Maryland 20771

October24, 2018

Simple PRINT

A simple form of Output in Fortran is the PRINT statement:

PRINT *format-reference, I/O list*

- The simplest ***format-reference*** is the * (asterisk), which directs unformatted data to the standard output channel (the terminal screen)
- The ***I/O list*** must be a comma-separated list of variables or strings

For example:

```
character(len=*), parameter :: myString = “Fortran is”
real :: x = 1
logical :: myLogic = .FALSE.
PRINT *, "X = ", x
PRINT *, myString, myLogic, “U N !”
```

Output:

```
X =      1.00000000
Fortran is F U N !
```

Simple READ

A simple form of Input in Fortran is the READ statement:

READ *format-reference, I/O list*

Similar to PRINT, the ***format-reference*** * (asterisk), accepts unformatted data from the standard input channel (the terminal screen), and the ***I/O list*** must be a comma-separated list of variables or strings

For example:

```
integer :: A, B, sum
PRINT *,'Please type in 2 numbers'
READ *, A, B
sum = A + B
PRINT *, A,' + ', B,' = ', sum
```

- After printing the instruction, this code will wait for you to enter 2 numbers, separated by a comma:

```
$ Please type in 2 numbers
$ 3,4
$ 3 + 4 = 7
```

Opening Files

Most I/O is done by accessing data files. The OPEN command is required to access files for reading or writing:

```
OPEN (UNIT = number, FILE = "name", IOSTAT = "ios", &
      & STATUS = "status", ERR = "err")
```

Specifier	Description
number	Normally in the range 9 – 99; each file's unit # must be unique
name	Complete file name; a character variable or expression
ios	INTEGER file-opening status, 0 if success, non-zero if failure
status	"NEW", "OLD"-must exist, "REPLACE"-create or replace existing file
err	Label to which code jumps in case of an error

Reading and Writing Files

After opening the file, we can READ from or WRITE to the file, using the following syntax:

```
READ ([UNIT=]number, [FMT=]fmt, [ADVANCE=adv,] [IOSTAT=ios,] &
      & [ERR=err,] [END=end])
WRITE ([UNIT=]number, [FMT=]fmt, [ADVANCE=adv,] [IOSTAT=ios,] [ERR=err])
```

The optional specifier **ADVANCE="no"** means the reading or writing should continue on the same record (or line), starting from the position where the last READ or WRITE stopped.

The READ statement also permits the important option "**END=**", with the equals followed by a label number to which to program branches when the End of File is reached.

For instance:

```
WRITE (unit=10, fmt='(1X, A9, F8.3)', iostat=ios) clouds, windMps
```

or

```
WRITE (10, '(1X, A9, F8.3)', iostat=ios) clouds, windMps
```

I/O Format

The READ and WRITE commands also require a format specification:

FMT='*fmt*'

- The * (asterisk) can be used for list-directed output (1 item per record/line).
- There are numerous format descriptors, but these are most commonly used:

Descriptor	Meaning
X	One space
/	New line
A	One character string
Iw	Integer with w digits (w characters wide, including sign)
Fw.d	Fixed point, w digits wide, d decimal places
Ew.d	Floating point with Exponent, w digits, d decimal places, E11.4 --> sx.xxxxEsxx

I/O Format (cont'd)

Repetitions

- An **A, I, F, E** format descriptor may be preceded by a number showing how many times it should be used:

'(I6, I6, I6, I6)' ➔ '(4I6)'

- Format groups can also be repeated, using parenthesis:

'(2X, F8.3, I2, F8.3, I2)' ➔ '(2X, 2(F8.3, I2))'

Closing Files

After you've finished reading or writing your files, it is good practice to close the associated file, and free up the unit number.

The syntax is very simple:

`CLOSE (UNIT=`*`number`*`)`

`*Note:` This is not essential, since files are automatically closed when the program ends

Unformatted I/O

So far we've only shown I/O in human-readable (ASCII) format. Fortran also allows reading and writing native computer format (binary data). This is very useful on large datasets, due to:

1. Reduced file size
2. Faster read/write times

The general way to *read or write* unformatted data is as follows:

```
real :: variable
OPEN(UNIT = 10, FORM = 'unformatted', FILE = filename)
WRITE(10) variable
CLOSE(UNIT = 10)
```

Notes:

- The default FORM is 'formatted' ASCII text, so the FORM keyword is only useful when opening files for binary I/O.
- Binary files are stored slightly differently on different computer platforms, so you may not be able to open a file written on another system
- The storage efficiency of binary files is greatly affected by the manner of writing – storing 1000 elements individually takes much more space than storing one 1000-element array

Example

```
program Kelvin2Fahrenheit

    ! Read in file of Kelvin temps from a list of unknown length.
    ! Output the list in both Kelvin and Fahrenheit, with headings

    implicit none
    real :: tempK, tempF

    open (unit=10, file="KelvinList.txt", err=200, status="old")
    open (unit=20, file="FahrenheitList.txt", err=200, status="replace")

    ! Write a heading for the output file
    WRITE (unit=20, fmt='(2(A10,3X))') "Kelvin","Fahrenheit"

    do
       READ (unit=10, fmt='(F7.2)', END=100) tempK
       tempF = (tempK – 273.15) * 9 / 5 + 32
       !*************************************************************
       !Complete the statement below to write tempK and tempF columns below the heading
       !*************************************************************
       WRITE
    end doll
100 continue

    close(10)
    close(20)
    stop

200 PRINT *, "Error opening file"

end program Kelvin2Fahrenheit
```

Example Solution

```
program Kelvin2Fahrenheit

    ! Read in list of Kelvin temps of unknown length.
    ! Output the list in both Kelvin and Fahrenheit, with headings

    implicit none
    real :: tempK, tempF

    open (unit=10, file="KelvinList.txt", err=200, status="old")
    open (unit=20, file="FahrenheitList.txt", err=200, status="replace")

    ! Write a heading for the output file
    WRITE (unit=20, fmt='(2(A10,3X))') "Kelvin","Fahrenheit"

    do
       READ (unit=10, fmt='(F7.2)', END=100) tempK
       tempF = (tempK – 273.15) * 9 / 5 + 32
       WRITE (unit=20, fmt='(2(F10.2))') tempK, tempF
    end do
100 continue

    close(10)
    close(20)
    stop

200 PRINT *, "Error opening file"

end program Kelvin2Fahrenheit
```