IO Enhancements

Jules Kouatchou

NASA GSFC Code 606 (ASTG)
Greenbelt, Maryland 20771

Jules.Kouatchou@nasa.gov

October 25, 2018

Agenda

IO Enhancements

- Stream IO
- Asynchronous I/O

Steam I/O - 1

Stream access is a new method for allowing fine-grained, random positioning within a file for read/write operations.

- Complements pre-existing DIRECT and SEQUENTIAL access
- Advantages:
  - Random access (as with DIRECT)
  - Arbitrary record lengths (as with SEQUENTIAL)
- Disadvantages:
  - Presumably poorer performance than both DIRECT and SEQUENTIAL
  - Lack of record separators increases risk of inability to read file under small changes.
  - Index for positioning within file might be less natural than those for DIRECT.

Steam I/O - 2

```
OPEN(unit, ACCESS = STREAM)
! both formatted and unformatted files

READ(unit, POS=n) x,y,z
! File starts at position POS=1
! Position is specified in file storage units -
! usually bytes

INQUIRE(unit, POS=currentPosition , ...)
! Formatted I/O must use POS obtained from INQUIRE()
! (or POS=1)
```

Examples

Check the files:

- *writeUstream.F90*
- *readUstream.F90*

Asynchronous I/O - 1

Potential performance enhancement allowing some I/O operations to be performed in parallel with other computations.

- To open a file for asynchronous operations, the new optional keyword **ASYNCHRONOUS** is used.
- An asynchronous read/write operation is initiated with the same keyword.
- An optional keyword, **ID**, can be used to return a handle for later use in identifying specific pending operations

Asynchronous I/O - 2

```
OPEN(10, ..., ASYNCHRONOUS=yes)
WRITE(10,..., id=id, ASYNCHRONOUS=yes) A
CALL do_something(....) ! Not involving A here
WAIT(10, id=id) ! Blocks here until A has been written
CALL do_something(...) ! OK to use A here
```

Asynchronous I/O - 3

If the asynchronous file access is performed in a procedure other than the one called for OPEN, the data involved has to be declared with asynchronous attribute

```
OPEN(10, ..., asynchronous=yes)
CALL async_write(10, A, id)
CALL do_something_else_here()
WAIT(10, id=id)
  ...
SUBROUTINE async_write(iu, data, id)
  INTEGER, INTENT(IN) :: iu
  INTEGER, INTENT(IN), DIMENSION(:), ASYNCHRONOUS :: data
  INTEGER, INTENT(OUT) :: id
  ...
  WRITE(iu, id=id, asynchronous=yes) data
  ...
END SUBROUTINE async_write
```

Asynchronous I/O - 4

An alternative for calling **WAIT** is to periodically call **INQUIRE** to check the status of the operation and in the meantime keep on doing something else

```
LOGICAL :: status
...
OPEN(10, ..., asynchronous=yes)
WRITE(10,..., id=id, asynchronous=yes) A
DO WHILE (!status)
   CALL do_something(....) ! Not involving A
   INQUIRE (10, id=id, pending=status)
END DO
```

Exercises

Edit the file *exampleAsyncIO.F90* to:

- Write the "regular version" (without using asynchronize I/O) of the routine *doingAsyncIO*. You can call the new routine *notDoingAsyncIO*.
- Time the four calls to *validWrite*, *invalidWrite*, *doingAsyncIO*, and *notDoingAsyncIO*.