Fortran 2018 …and Beyond

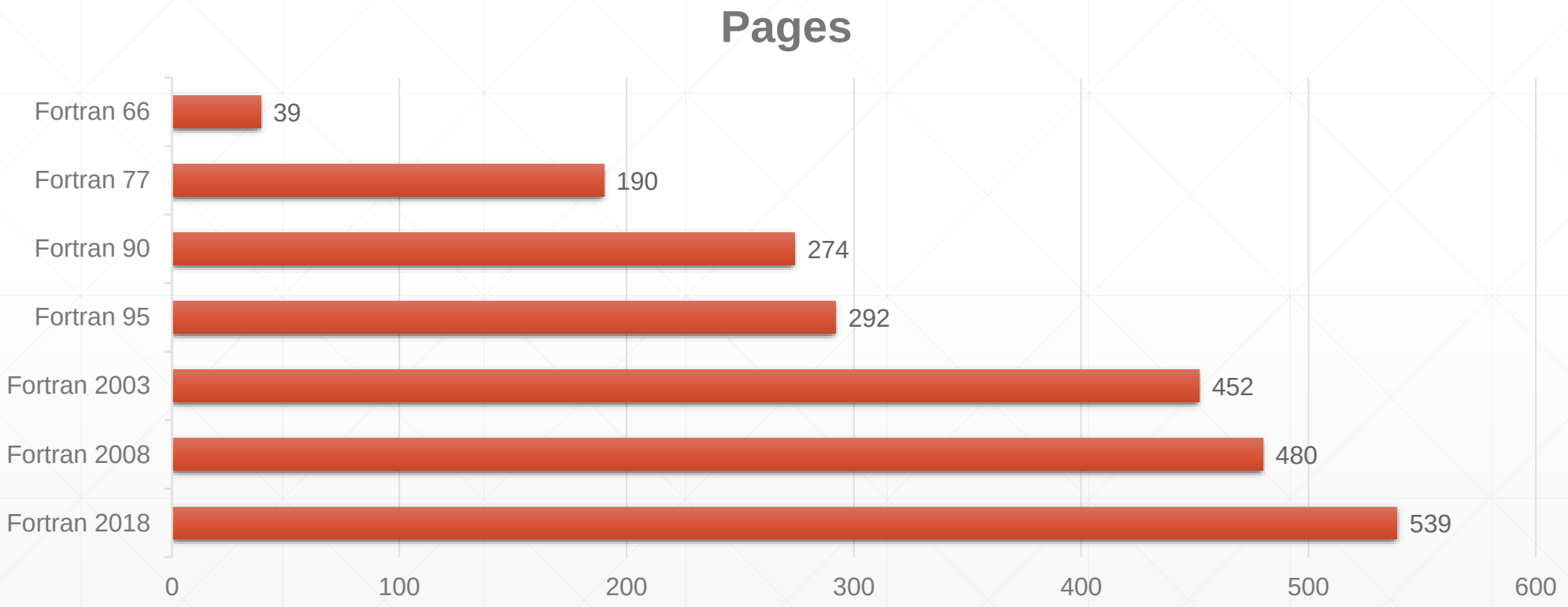
Steve Lionel, Convenor, ISO/IEC JTC1/SC22/WG5 Fortran Standards Committee
https://stevelionel.com/drfortran

July 2020

How is a new Fortran standard made?

- International Fortran committee is ISO/IEC JTC1/SC22/WG5
- Experts from individual countries make up National Bodies
- WG5 determines general content of the standard
- Development of features is done by the US National Body (INCITS PL22.3, informally “J3”)
- All WG5 members vote to approve a “Final Committee Draft”
- Fortran 2018 was published November 2018
- Next revision working title “Fortran 202X”

The Fortran Standard Through the Years



Fortran 2018

TS18508 “Additional Parallel Features in Fortran”

- Technical Specification (TS) published in 2015
- Major new coarray features
  - Teams
  - Events
  - Failed Images

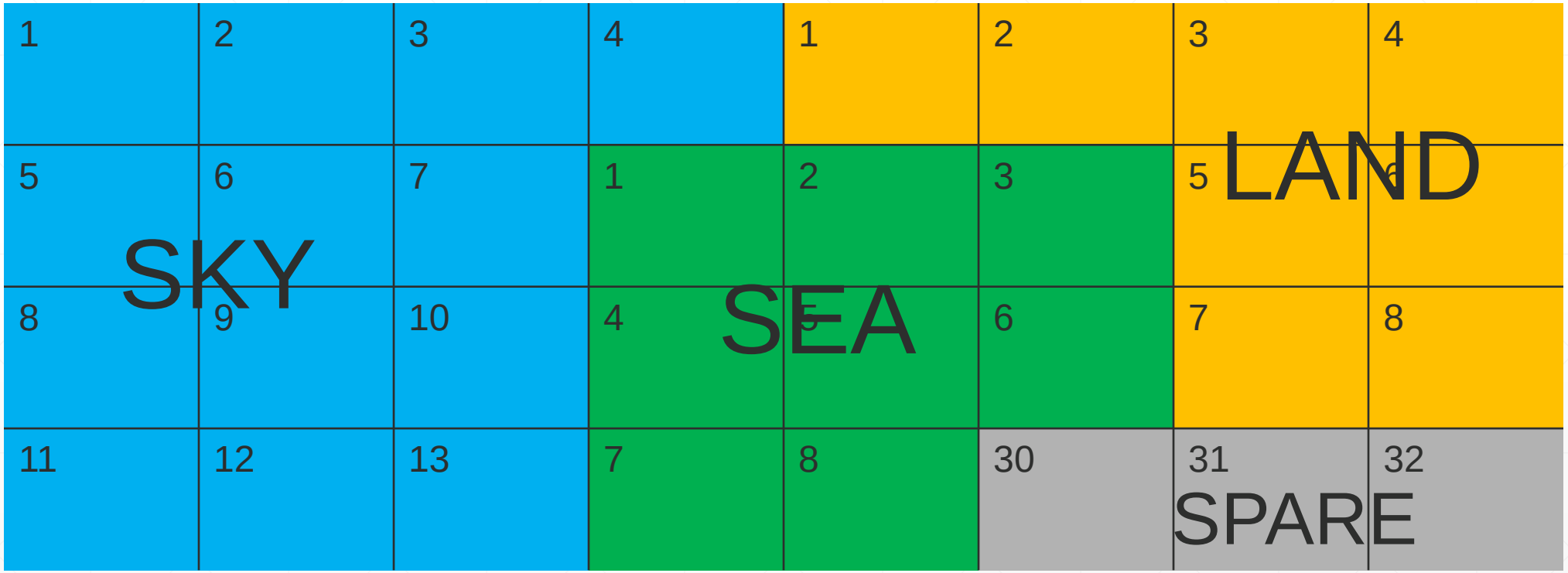
Teams

In Fortran 2008, program images were uniformly numbered starting at 1

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32

Teams

Teams allow splitting images into groups



More coarray features

- Can detect that an image has “failed” on any operation
- Image selector now has optional STAT=, TEAM=, TEAM_NUMBER specifiers
- SYNC TEAM synchronizes with the current team, an ancestor team or a child team
- CRITICAL construct enhanced to optionally get status
- Events make it easier to synchronize activity between images
  - EVENT_POST, EVENT_WAIT
- Collectives perform an operation across all images of a team
- More atomic subroutines (add, and, or, xor, fetch_add, fetch_and, fetch_or, fetch_xor, cas

TS29113 “Further Interoperability of Fortran with C”

- Expands on C interoperability features first appearing in Fortran 2003
- Technical Specification published in 2012
- Designed in cooperation with the MPI Forum for Fortran interfaces to MPI

TS29113 continued

- In Fortran 2003 and 2008, these kinds of dummy arguments were not interoperable:
  - Assumed-shape arrays
  - Assumed-size arrays
  - Character length other than 1
  - Allocatable or pointer variables
- In Fortran 2018, all of these are now interoperable when a “C Descriptor” is passed

C Descriptor

- A C descriptor includes:
  - Attribute code (POINTER, ALLOCATABLE, OTHER)
  - Data type
  - Base address
  - Element length
  - Rank
  - Bounds and extents

C Descriptors

- Fortran creates and passes C descriptors to routines declared as BIND(C)
- C code can operate on C descriptors with CFI_xxx functions
- C code can create C descriptors and pass to Fortran
  - Fortran procedure must have BIND(C) attribute
- Descriptor layout, constants, functions declared in ISO_Fortran_binding.h

```
#include "ISO_Fortran_binding.h"
#include <memory.h>
#include <stdio.h>

extern "C" void greetings(CFI_cdesc_t * descr);

int main()
{
        int status;
        CFI_CDESC_T(0) cdesc;

        // Create our own local descriptor for an allocatable string
        status = CFI_establish((CFI_cdesc_t *)&cdesc, NULL,
                               CFI_attribute_allocatable,
                               CFI_type_char, 1, 0, NULL);
        //Allocate the string to length 7
        status = CFI_allocate((CFI_cdesc_t *)&cdesc, NULL, NULL, 7);
        // Copy in 'Hello, '
        memcpy(cdesc.base_addr, "Hello, ", 7);
        // Call Fortran to append to the string and print it
        greetings((CFI_cdesc_t *)&cdesc);
        printf("Length is now %zd\n", cdesc.elem_len);
        status = CFI_deallocate((CFI_cdesc_t *)&cdesc);
}
```

```
subroutine greetings (string) bind(C)
    implicit none
    character(:), allocatable :: string

    string = string // 'Zurich!'
    print *, string
    end subroutine greetings
```

```
Hello, Zurich!
Length is now 14
```

Assumed Type

- Syntax is `TYPE(*)`
- Unlimited polymorphic - has no declared type
- May be used only for dummy arguments
- Like C `void`
- Limited use in Fortran code

Allocatable Dummy Arguments

- `ALLOCATABLE, INTENT(OUT)` dummy arguments get deallocated on entry to a Fortran procedure
- In Fortran 2018, a `BIND(C)` procedure can now have such an argument
- Fortran processor is required to do the deallocation on the call

More Interoperability Changes

- A Fortran procedure with a `CONTIGUOUS` dummy argument must be able to handle a C descriptor for a non-contiguous array
- Interoperable procedures may now have `OPTIONAL` dummy arguments
- `ASYNCHRONOUS` attribute extended to data access other than input/output

Assumed Rank

- Syntax is `DIMENSION(..)`
- May be used only for dummy arguments
- New `SELECT RANK` construct for use in Fortran code
  - `RANK(n)`
  - `RANK(*)` for assumed-size array
  - `RANK DEFAULT`

IEEE Floating Point Changes

- Many small changes to support IEEE 60559:2011
- “Denormal” is now “Subnormal”
- `IEEE_GET_MODES` and `IEEE_SET_MODES` gets and stores all the floating-point modes
- More rounding modes; separate modes for base 2 and base 10
- `IEEE_INT`, `IEEE_REAL` rounded conversion functions
- Quiet and Signaling comparison functions

IMPLICIT NONE enhancement

- `IMPLICIT NONE (EXTERNAL)` requires explicit interface or `EXTERNAL` declaration
- `IMPLICIT NONE (TYPE)` same as previous `IMPLICIT NONE`
- You can combine these:
  `IMPLICIT NONE (EXTERNAL, TYPE)`

More Changes

- Restrictions on constant expressions relaxed
  - `integer :: b = bit_size(b)`
  - `integer :: iota(10) = [ ( i, i = 1, size(iota,1) ) ]`
- `D0.d`, `E0.d`, `ES0.d`, `EN0.d`, `G0.d` and `Ew.dE0` edit descriptors
- `G0.d` may be used for integer or character values
- `STOP` and `ERROR STOP` can have any scalar expression
  - Optional `QUIET=` can disable messages
- `MOVE_ALLOC` has optional `STAT=` and `ERRMSG=`

More Changes continued

- `DO CONCURRENT` can now have locality specifiers
- `ERRMSG=` added to `GET_COMMAND`, `GET_COMMAND_ARGUMENT`, `GET_ENVIRONMENT_VARIABLE`
- `OUT_OF_RANGE` intrinsic tests conversion
- `REDUCE` intrinsic applies user function to an array
- `COSHAPE` intrinsic like `SHAPE` for coarrays
- `RANDOM_INIT` specifies behavior of random number generator
  - Is sequence repeatable?
  - Does each image have its own sequence?

More Changes continued

- Arguments to `SIGN` intrinsic may have different kinds
- Non-polymorphic pointer arguments to `EXTENDS_TYPE_OF` and `SAME_TYPE_OF` need not be defined
- Kind of DO variable in implied DO may now be specified in array constructors and `DATA` statements
  - `[ (a(i,i), integer(long) :: i=1,n) ]`
- All procedures are `RECURSIVE` by default. New `NON_RECURSIVE` keyword
- Hexadecimal input and output with EX edit descriptor
  - Example: -15.625 with `EX14.4E3` might give `-0X1.F400P+003`

Deleted and Obsolescent Features

- Deleted
  - Arithmetic `IF` statement
  - Non-block `DO` construct
- Obsolescent
  - `COMMON` and `EQUIVALENCE`
  - Labelled DO statements
  - Specific names for standard intrinsics (for example `DACOS`)
  - `FORALL`

Fortran 202X

Future Revisions

- Next revision is informally called Fortran 202X
- Goal is to have it published no later than 2023
- Six-month survey of users 2017-2018
  - Results in WG5 document N2147
- After that, Fortran 202Y

Fortran 202X Features

(nn-nnn refers to papers at https://j3-fortran.org/)

- Add optional argument to C_F_POINTER to specify lower bounds (19-238r1)
- Longer source lines and statement length (19-138r1)
  - Require reporting of ignored characters after line length limit, if any (19-149r1)
- Trigonometric functions in degrees (SIND, COSD, etc.) (19-203r1)
- Trigonometric functions scaled by π (SINPI, COSPI, etc.) (19-204r1)
- SELECTED_LOGICAL_KIND intrinsic (19-147r1)
- LOGICALnn constants in ISO_FORTRAN_ENV (19-139r1)

Fortran 202X Features Continued

- SPLIT function splits strings into tokens based on separators (19-254r1)
- C_F_STRPOINTER and F_C_STRING for help with C strings (19-197r3)
- AT format specifier for trimming strings (19-137r2)
- Format control over leading zeros for reals (19-156r1)
- Allow arrays of derived type with coarray components (19-250r1)
- Put with notify for coarrays (19-259r1)
- Automatically allocate deferred-length character in internal WRITE and IOMSG/ERRMSG (19-252r2)

Fortran 202X Features Continued

- Reduction specifier in DO CONCURRENT (19-255r2)
- Allow BOZ constants in more places (19-256r2)
- SIMPLE procedures are PURE with more restrictions (19-201r1)
- TYPEOF, CLASSOF intrinsics to help with generic programming (19-142r1)
- Rank-agnostic allocation and pointer assignment (20-120r1)
- BOUNDS() and RANK() specifiers for DIMENSION attribute (19-202r2)

Approved F202X Features not yet finished

- Rank-agnostic array notation (20-125)
- Protected components (20-106)
- Typed enumerators (19-249r1)
- Conditional expressions
- Short-circuit logical operators (18-239)
- Variant of INTENT that applies to a pointer target (18-144r1)

More Information

- WG5 web site https://wg5-fortran.org
  - Documents > N2161 The New Features of Fortran 2018
  - Fortran Standards > Fortran 2018
- J3 (PL22.3) web site https://j3-fortran.org
  - Repository for papers related to technical content of the standard
  - 18-007r1 is the committee reference for Fortran 2018
- Doctor Fortran blog https://stevelionel.com/drfortran
- Ideas for future revisions: https://github.com/j3-fortran/fortran_proposals