#### Fortran Standard Library

Jeremie Vandenplas

Bálint Aradi, Izaak Beekman, Ondrej Certik, Milan Curcic, Pierre de Buyl, Juan Fiol, Michael Hirsch, Ivan Pribec, Nathaniel Shaffer

July 2, 2020

#### Fortran Standard

- Published by the International Organization for Standardization (ISO)
- Limited set of intrinsic procedures
- Possibility to add new intrinsic procedures and modules
  - After standardization and implementation in compilers
- No Standard Library
  - Several attempts in the past (e.g. available on GitHub)

Consequence: we all reinvent the wheel continuously!

#### Aim

## Develop and provide

a community driven and agreed-upon de facto

standard library

for Modern Fortran

## Fortran Standard Library - stdlib

- One of the four pillars of fortran-lang
- MIT License
- Aims to collaborate with the Fortran Standard

#### Committee

- Links:
  - GitHub: https://github.com/fortran-lang/stdlib
  - API docs: https://stdlib.fortran-lang.org

# General scope

Similar to SciPy or to the default built-in Matlab scientific environment

Algorithms

Three topics

- Merging, searching, sorting, ...
- Mathematics
  - Linear algebra, sparse matrices, special functions, fast Fourier transform, random numbers, statistics, ordinary differential equations, numerical integration, optimization, ...
- Utilities
  - Containers, strings, files, OS/environment integration, unit testing, assertions, logging, ...

#### Current state of stdlib

#### Since December 2019 on GitHub:

- Issues / ideas / comments
  - 47 contributors
  - > 110 GitHub issues
- Source codes
  - 16 contributors
  - > 100 Pull Requests

### Currently discussed

#### Several discussions on:

- Assertion
- Logging
- OS integration
- Random numbers

- Sparse matrices
- Special functions
- Strings
- ...

# Currently implemented in stdlib

Module	Description	# procedures
ascii	String manipulations	16
error	Catching and handling errors	2
io	Input/output helper and convenience	3
kinds	Kind definition	-
linalg	Linear algebra	3
optval	Fallback value for optional arguments	1
quadrature	Numerical integration	4
stats	Descriptive statistics	5
system	OS utilities	1

### Currently implemented in stdlib

Support of any rank (up to 15) and any kind (integer, real, complex)

- When appropriate
- Meta-programming
  - *fypp* = Python powered preprocessor
- E.g., for the function 'mean': 600 auto-generated functions

### Example - optval + fypp

```
#:set KINDS_TYPES = REAL_KINDS_TYPES + INT_KINDS_TYPES + &
     & CMPLX_KINDS_TYPES + [('I1', 'logical')]
2
3
    . . .
   #: for k1, t1 in KINDS_TYPES
    pure elemental function optval_\{t1[0]\}$$\{k1\}$(x, default)&
5
        result(y)
6
        ${t1}$, intent(in), optional :: x
        ${t1}$, intent(in) :: default
8
        \{t1\}$ :: y
10
        . . .
   end function optval_\{t1[0]\}$${k1}$
11
   #:endfor
12
```

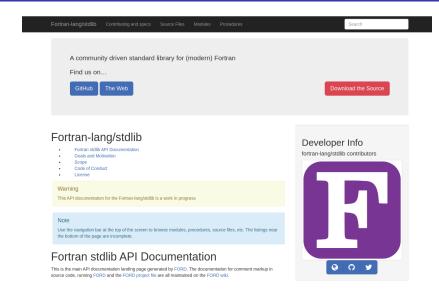
## Example - optval

```
. . .
       stdlib_experimental_optval, only: optval
  use
3
   . . .
   real function root(x, n)
   real, intent(in) :: x
   integer, intent(in), optional :: n
     root = x**(1.0/optval(n, 2))
   end function
   . . .
```

#### **Examples**

```
use stdlib_experimental_io, only: loadtxt, savetxt
    use stdlib_experimental_linalg, only: diag
3
    use stdlib_experimental_stats, only: moment
5
    real, allocatable :: A(:,:)
6
    call loadtxt('example.dat', A)
7
8
    . . .
    print*, diag(A)
9
10
    . . .
    call savetxt('moment.dat',&
11
      moment(A, order = 3, dim = 1, mask = (A > 5.))
12
13
```

# API docs (https://stdlib.fortran-lang.org)



# API docs (https://stdlib.fortran-lang.org)

#### Automatically generated by FORD

- Source codes
- Markdown specs for all procedures
  - Description
  - Syntax
  - Arguments
  - Output / Return value
  - Example(s)

How to contribute to stdlib?

# Any contribution is welcome!

#### How to contribute to stdlib?

#### Through GitHub

- Issues
  - Proposition of ideas, issues, comments
- Pull Requests
  - To contribute to the source code and specs
  - Might be based on existing contributors' code (License!)

#### Code of Conduct

• Please check it first!

# Contributing to the source code?

#### Workflow (See the complete description on GitHub)

- Proposition of an idea
- Proposition of the API
- Oiscussion of the specs
- Pull request of an implementation in the experimental namespace + associated unit tests
- Stable release of procedures in the experimental namespace (still to be clarified)

# Thank you to all contributors!

Bálint Aradi	Martien Diehl	
Izaak Beekman	Juan Fiol	Nathaniel Shaffer
Neil Carlson	J. Henneberg	Pedro Costa
Ondrej Certik	Michael Hirsch	Jeremie Vandenplas
Milan Curcic	Ivan Pribec	Ashwin Vishnu
Pierre de Buyl	Yuichiro Sakamoto	

Also to all contributors who opened and commented issues/PR on GitHub!

# Thank you!