Fortran Package Manager

Vincent Magnin (MdC)

https://github.com/vmagnin

Mastodon https://floss.social/@vmagnin

Journée Fortran - groupe Calcul CNRS
13 juin 2025, Observatoire de Paris

FortranCon 2025 : 4 et 5 novembre 2025 sur Zoom

- https://events.fortrancon.org/event/1/
- *Keynote :* John Reid, membre actif du comité Fortran de l'ISO/IEC JTC1/SC22/WG5 (animateur du WG5 de 1999 à 2017).
- Dates limites :
  - Inscription (gratuite) : 15 octobre 2025.
  - Premier appel à résumés : 1er août 2025 (décisions avant 30/08).
  - Second appel à résumés : 1er sept. 2025 (décisions avant 30/09).
- Deux formats :
  - présentations : 25 minutes + 5 minutes de discussion ;
  - communications : 15 minutes + 5 minutes de discussion.
- FortranCon 2020 et 2021 :
  https://www.youtube.com/channel/UCmrUfKC3D2ZMHLRyQANeP_Q

Plan

Fortran-lang, la communauté qu'on attendait

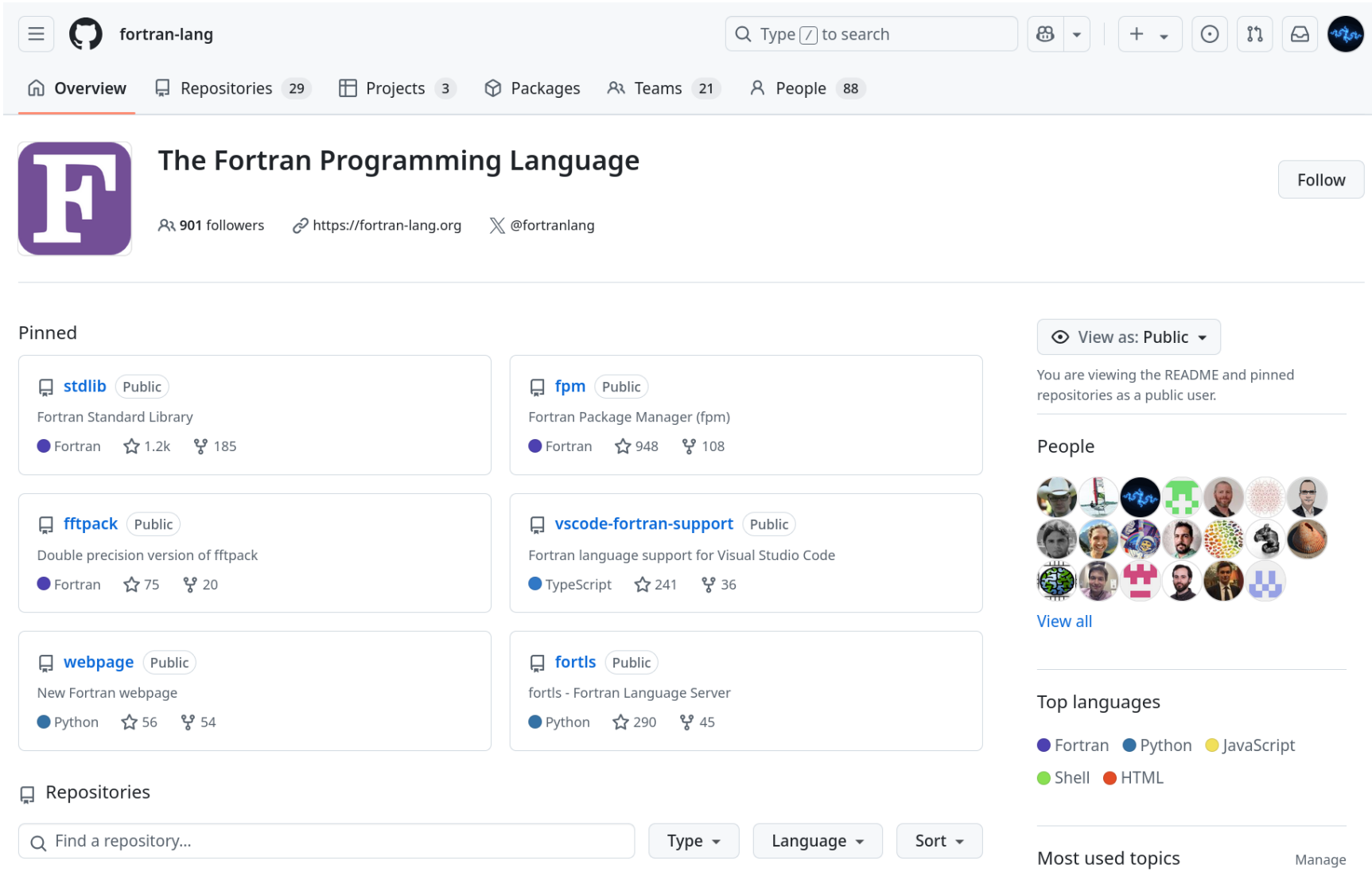
Un diagnostique impitoyable en 2019 :

- Fortran à la traîne par rapport à d'autres langages plus récents.
- Une galaxie éclatée :
  - utilisateurs loin du comité de normalisation.
  - Difficile de distribuer des projets Fortran (systèmes de construction multiples, pas de dépôt central).
  - Pas de librairie standard comme en C ou Python.
  - Pas de site web centralisé.
- Pas de compilateur pour expérimenter de nouvelles fonctionnalités.

Un diagnostique efficient :

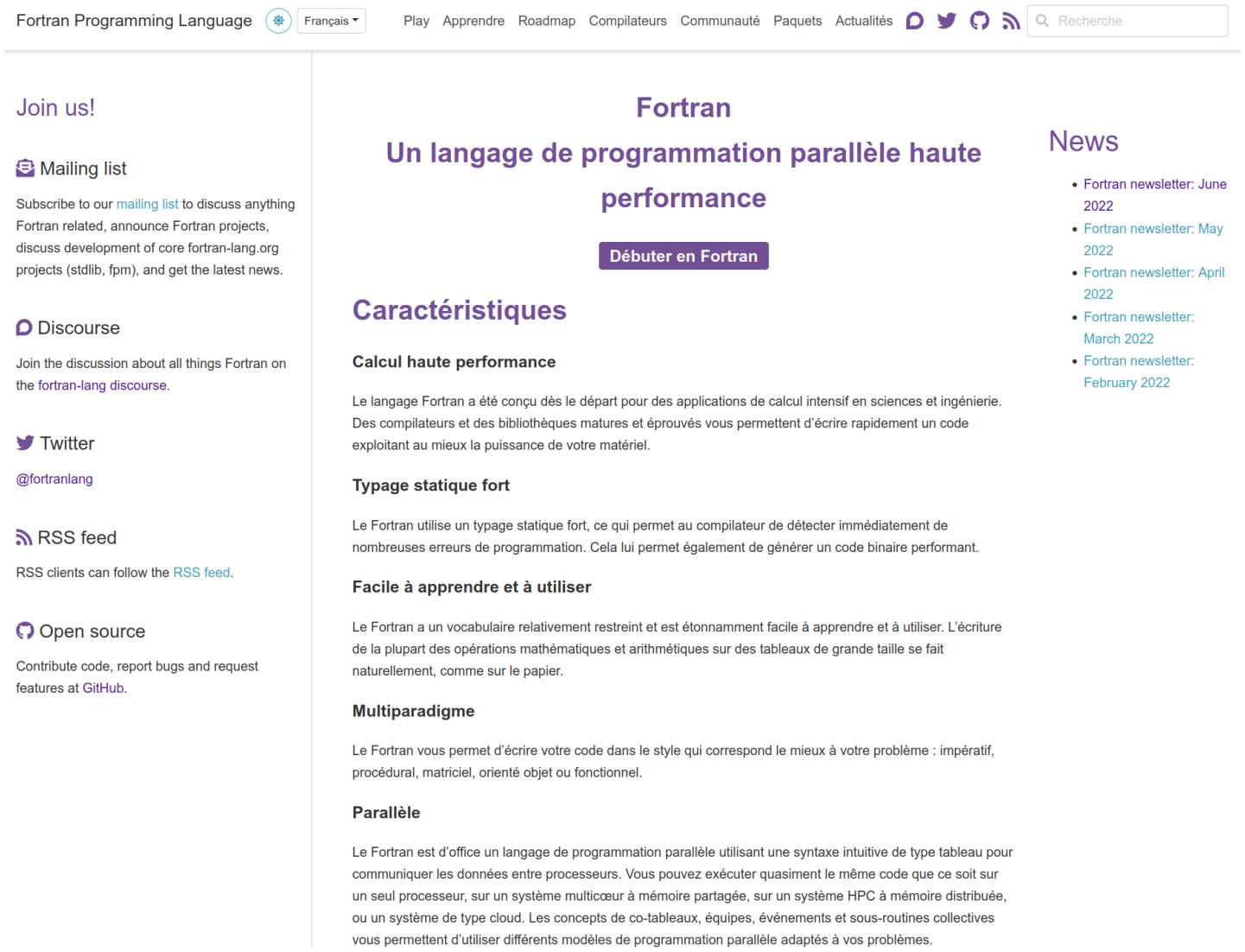
- création de la communauté Fortran-lang fin 2019, en particulier par Ondřej Čertík et Milan Curcic.
- De l'étudiant au membre du comité de normalisation.
- **Un projet pour remédier à chaque point.**

Centralisée sur GitHub



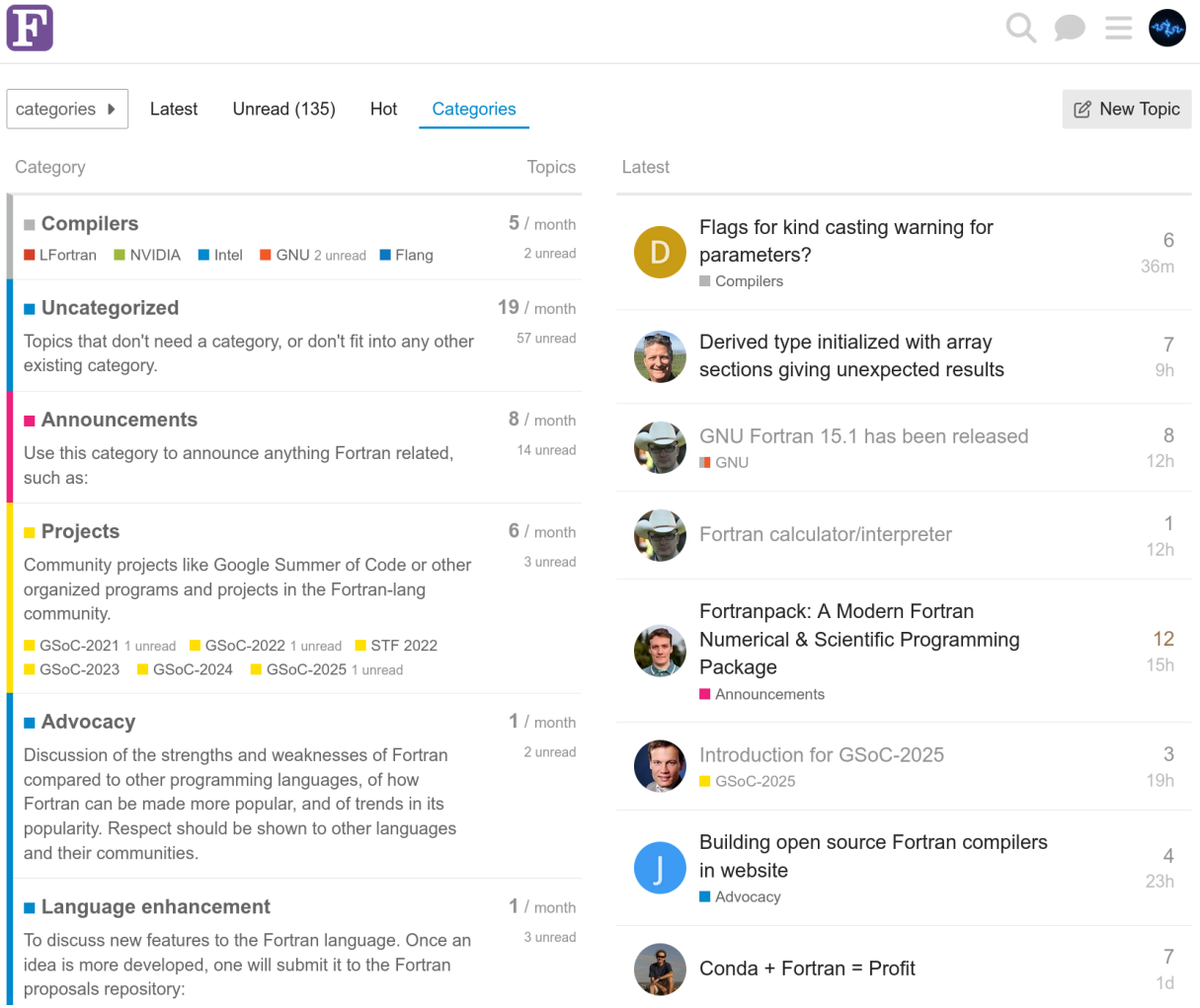
- https://github.com/fortran-lang

Fortran-lang.org : la porte d'entrée du langage



- https://fortran-lang.org/
- Partiellement traduit en français.

On discute Fortran sur le Discourse

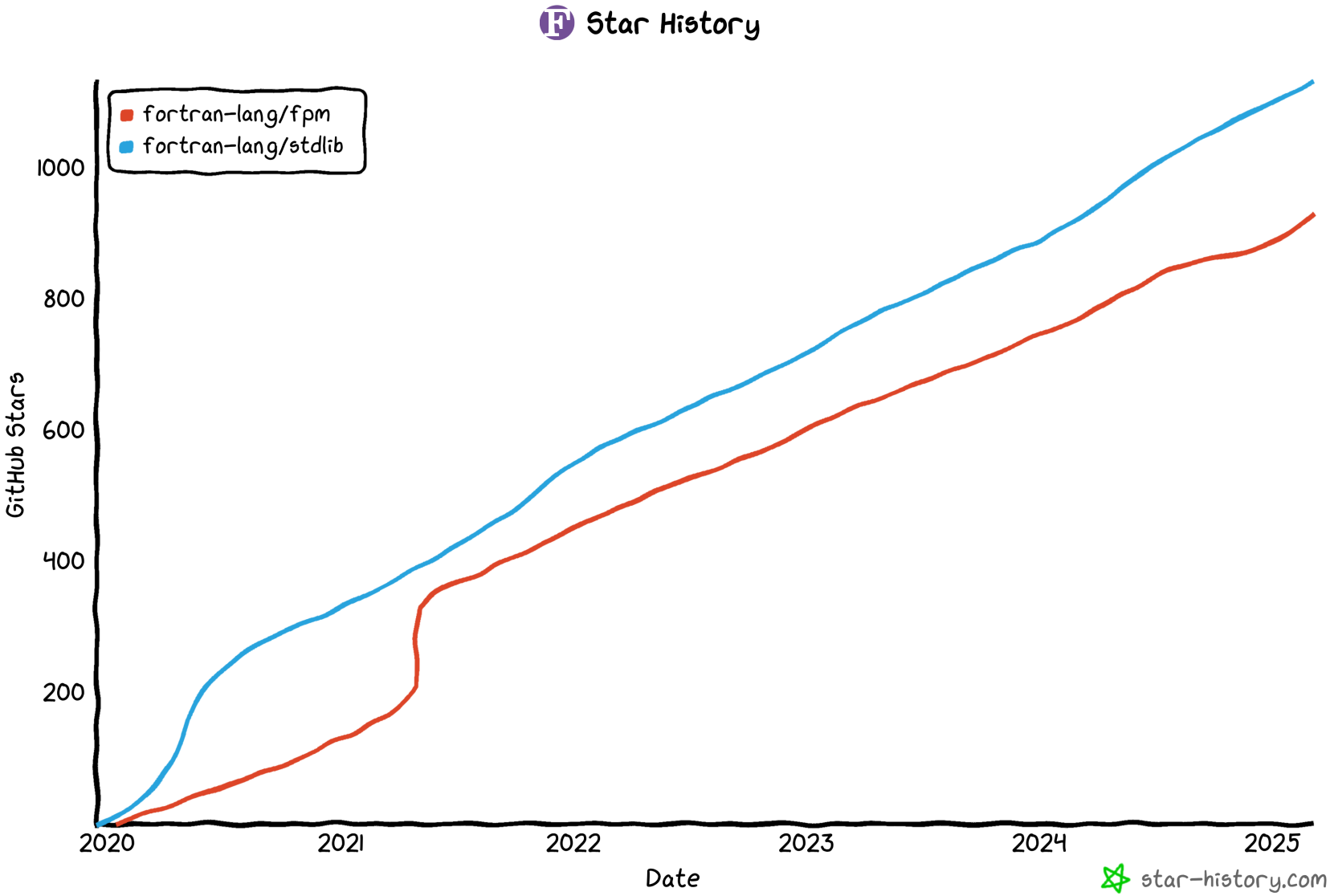


- https://fortran-lang.discourse.group/
- Lancé le 5 mai 2020.
- 1853 utilisateurs inscrits (le 29 avril 2025).

Mais on lève aussi des fonds !

- *Google Summer Of Code* (GSoC) : Fortran-lang et LFortran (étudiants financés durant 10 semaines).
  https://summerofcode.withgoogle.com/
- Le *Sovereign Tech Agency* (ministère fédéral de l'Économie et du Climat allemand) : https://www.sovereign.tech/
- L'organisation américaine *NumFOCUS* : https://numfocus.org/

fpm et stdlib, des outils de plus en plus reconnus



On avait besoin d'un outil moderne...

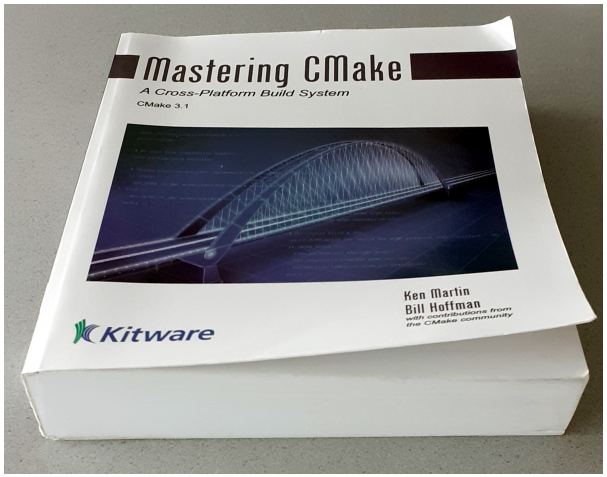
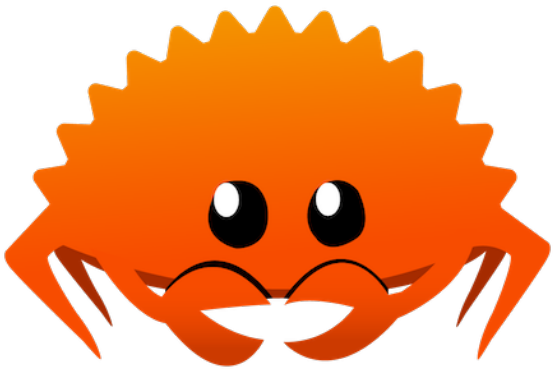


Figure – 700 pages...

- Pas évident de construire et distribuer un projet Fortran...
- Outils multiples (CMake, autotools, scripts, Meson...)
  - souvent peu intuitifs,
  - pas toujours multi-plateforme,
  - ne gérant pas toujours bien Fortran.
- N'encourage pas l'utilisation de bibliothèques...
- dispersées sur de nombreux sites.

Comme dans Rust !



- Un site central :
  https://www.rust-lang.org/fr
- Cargo, le couteau suisse :
  - `cargo new projet`
  - Configuration simple : `cargo.toml`
  - `cargo run`
  - Un catalogue de bibliothèques en ligne :
    https://crates.io/

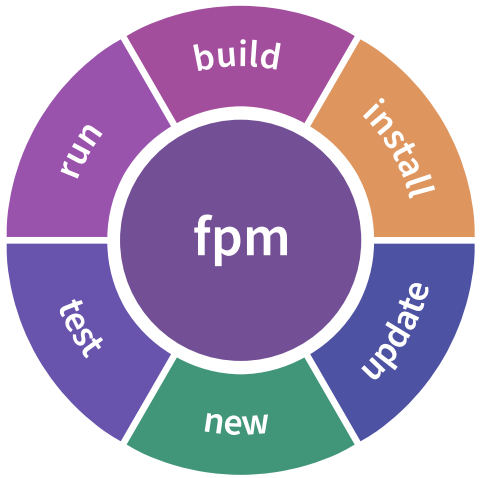
```
~$ ./demo
```

Fortran se dérouille : le Fortran Package Manager fpm

- Inspiré de cargo.
- Essentiellement écrit en Fortran 2018 (49000 lignes) !
- Multi-plateforme.
- Supporte la plupart des compilateurs Fortran.
- Version 0.12 sortie le 18 mai 2025.
- Simple, intuitif, courbe d'apprentissage douce.
- Documentation multilingue :
  https://fpm.fortran-lang.org/fr/
- Licence MIT.
- https://github.com/fortran-lang/fpm

fpm s'occupe de tout

- Scanne les sources (module/submodules), découvre les exécutables et les tests,
- détermine l'ordre dans lequel compiler l'ensemble.
- Construction incrémentale.
- Fonctionne avec les codes Fortran mais aussi C.
- Télécharge les dépendances fpm via `git clone` !
- Remarque : fpm est lui-même un projet fpm (avec cinq dépendances).



On peut installer fpm en un clin d'oeil n'importe où

Pas besoin d'être administrateur !

https://anaconda.org/conda-forge/fpm

```
❯ conda create -n fpm fpm
❯ conda activate fpm
```

https://packages.msys2.org/base/mingw-w64-fpm

```
❯ pacman -S mingw-w64-x86_64-fpm
```

```
❯ spack install fpm +openmp
❯ spack load fpm +openmp
```

https://formulae.brew.sh/formula/fpm

```
❯ brew install fpm
```

Unix-like

```
❯ git clone https://github.com/fortran-lang/fpm
❯ cd fpm && ./install.sh
```

Macports

```
❯ sudo port install fpm
```

On peut même le compiler à partir d'un fichier *standalone*

```
~$ ./demo
```

Sous-commandes de fpm

~$ ./demo

```
$ fpm
Fortran Package Manager:

USAGE: fpm [ SUBCOMMAND [SUBCOMMAND_OPTIONS]
↪  ]|[--list|--help|--version]
       where SUBCOMMAND is commonly new|build|run|test

 subcommand may be one of

  build     Compile the package placing results in the "build"
  ↪  directory
  help      Display help
  list      Display this list of subcommand descriptions
  new       Create a new Fortran package directory with sample files
  run       Run the local package application programs
  test      Run the test programs
  update    Update and manage project dependencies
  install   Install project
  clean     Delete the build
  publish   Publish package to the registry

 Enter "fpm --list" for a brief list of subcommand options. Enter
 "fpm --help" or "fpm SUBCOMMAND --help" for detailed descriptions.
```

A l'aide !

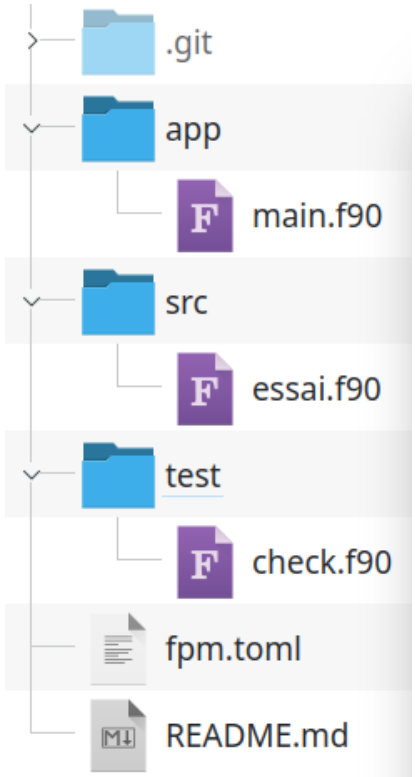
- `$ fpm`
- `$ fpm help`
- `$ fpm new --help`
- `$ fpm manual`
- Documentation en ligne : https://fpm.fortran-lang.org/fr/
- Le Discourse : https://fortran-lang.discourse.group/

```
~$ ./demo
```

Créer un projet avec fpm, c'est simple

```
~$ ./demo
```

```
$ fpm new essai
$ cd essai
$ fpm build
 + mkdir -p build/dependencies
essai.f90                                done.
libessai.a                               done.
main.f90                                 done.
essai                                    done.
[100%] Project compiled successfully.
$ fpm test
check.f90                                done.
check                                    done.
[100%] Project compiled successfully.
 Put some tests in here!
$ fpm run
Project is up to date
 Hello, essai
```



Sa configuration est dans son manifeste fpm.toml

- A droite, manifeste par défaut obtenu avec :
  `$ fpm new mon_projet`

- Pour obtenir un manifeste hyperdocumenté :
  `$ fpm new mon_projet --full`

```
~$ ./demo
```

```
name = "mon_projet"
version = "1.0.0"
license = "GNU GPL"
author = "vous"
maintainer = "votre courriel"
copyright = "(c) 2024, vous"

[build]
auto-executables = true
auto-tests = true
auto-examples = true
module-naming = false

[install]
library = false

[fortran]
implicit-typing = false
implicit-external = false
source-form = "free"
```

Tester un projet fpm en 30 secondes

Un `git clone`,
un `cd`
et un `fpm test` !

Essayons avec ForImage *(A Fortran library for processing and editing PNM images and managing colors)* :
https://github.com/gha3mi/forimage/

```
~$ ./demo
```

On définit les dépendances dans le fpm.toml

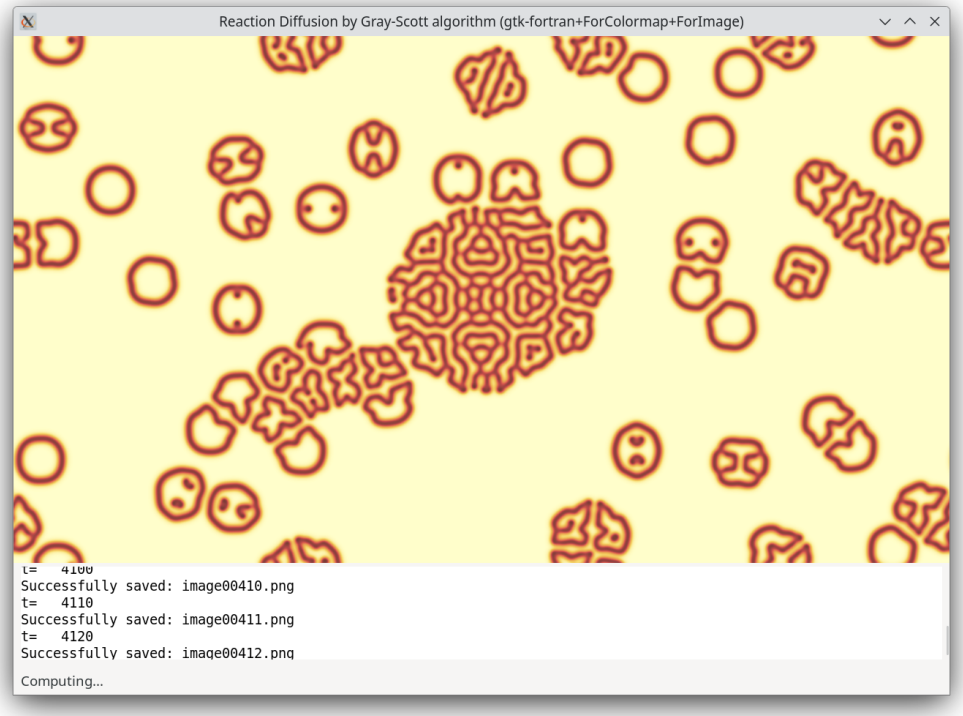
- Essayons ForColormap *(A Fortran library for colormaps)* : https://github.com/vmagnin/forcolormap
- Elle possède une dépendance : ForImage.
- Qu'on peut déclarer sur GitHub ou en local.
- https://fpm.fortran-lang.org/tutorial/dependencies.html

```
~$ ./demo
```

Dépendances installées dans le système

- Essayons l'exemple reaction_diffusion du dépôt `https://github.com/vmagnin/gtk-fortran-extra`
- Deux dépendances : ForColormap (+ForImage) et gtk-fortran (+GTK, bibliothèque en C dans le système).

```
~$ ./demo
```



Dépendances : link et external-modules

- link : attention à l'ordre.
- `external-modules` : les modules `.mod` nécessaires.

```
[build]
link = ["plplot", "plplotfortran"]
external-modules = ["plplot"]
```

Commande de compilation :

```
$  fpm run --flag '$(pkg-config --cflags --libs plplot
↪  plplot-fortran)'
```

Installer une bibliothèque dans le système

- fpm install

```
~$ ./demo
```

```
[library]
#type = "monolithic"   # Default (dependencies included)
#type = "static"
type = "shared"

[install]
library = true
```

Dépendances : les *metapackages*

- Gérés intrinsèquement par fpm : stdlib, OpenMP, MPI, MINPACK, HDF5, NetCDF, BLAS
- fpm ne gère pas encore leurs versions (une étoile * à la place).
- https://fpm.fortran-lang.org/spec/metapackages.html

```
[dependencies]
stdlib = "*"
minpack = "*"
hdf5 = "*"
```

- Exemple : calcul de π avec OpenMP
- https://fpm.fortran-lang.org/how-to/compute-pi-openmp.html

```
~$ ./demo
```

- Exemple avec stdlib

```
~$ ./demo
```

On peut régler finement les dépendances

- Ne concernant que les tests,
- ou que certains exécutables du projet.

```
[dev-dependencies]
test-drive.git = "https://github.com/fortran-lang/test-drive"
test-drive.tag = "v0.4.0"

[[executable]]
name = "demo"
[executable.dependencies]
M_CLI2.git = "https://github.com/urbanjost/M_CLI2"
```

Configuration d'un préprocesseur

- Préprocesseur C cpp, Fortran fypp...
- Par exemple, dans stdlib :

```
[preprocess]
[preprocess.cpp]
suffixes = [".F90", ".f90"]
macros = ["MAXRANK=7"]
```

https://fpm.fortran-lang.org/fr/spec/manifest.html?highlight=preprocess#installation-configuration

Commandes diverses

- `$ fpm update`
- `$ fpm run --runner gdb`
- `$ fpm run --runner "time -v"`
- `$ fpm run --runner ldd`

Un mécanisme pour les greffons *(plugins)*

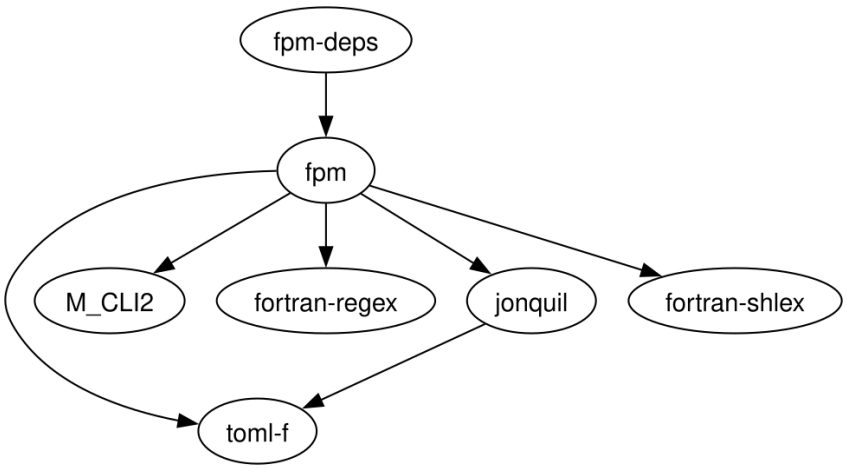
- Si une sous-commande `<NAME>` n'est pas reconnue, fpm cherche dans le système une commande `fpm-<NAME>` et l'appelle.
- Exemples :
  - https://github.com/urbanjost/fpm-gdb
  - man-pages pour les routines intrinsèques Fortran : `fpm man` (https://github.com/urbanjost/M_intrinsics)

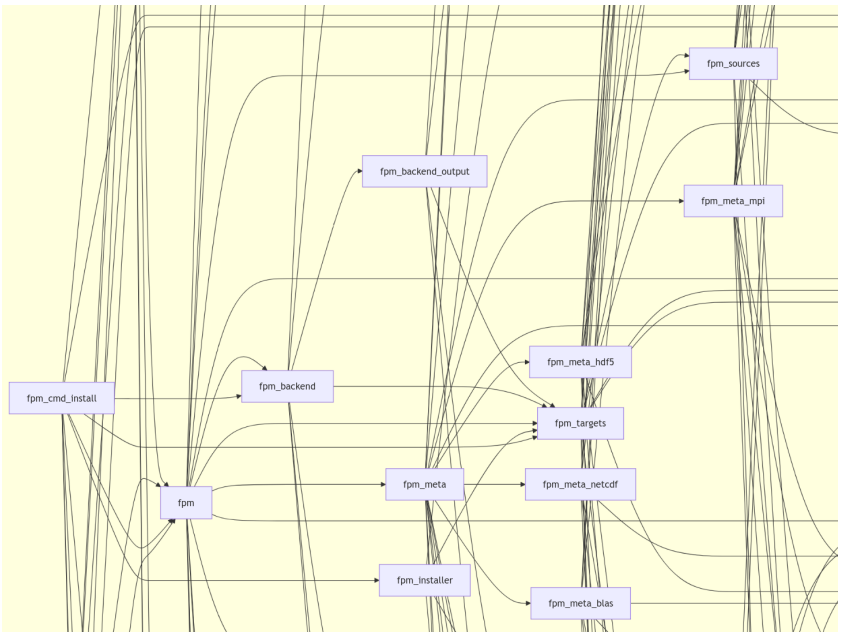
```
$ git clone git@github.com:urbanjost/M_intrinsics.git
$ cd M_intrinsics/
$ fpm build --profile release
$ fpm install
$ fpm man sinh | more
*set_mode* unknown key name auto_response_file

sinh(3fortran)
↪  sinh(3fortran)

NAME
  SINH(3) - [MATHEMATICS:TRIGONOMETRIC] Hyperbolic sine function
...
```

`fpm install` installe `fpm-man` dans `.local/bin/`. Il sera reconnu comme un greffon.

Des greffons pour visualiser les dépendances

- fpm-deps pour générer des Graphviz :
  https://github.com/ivan-pi/fpm-deps
- fpm-modules pour générer des graphes Mermaid :
  https://github.com/davidpfister/fpm-modules





Packager son projet Fortran pour fpm

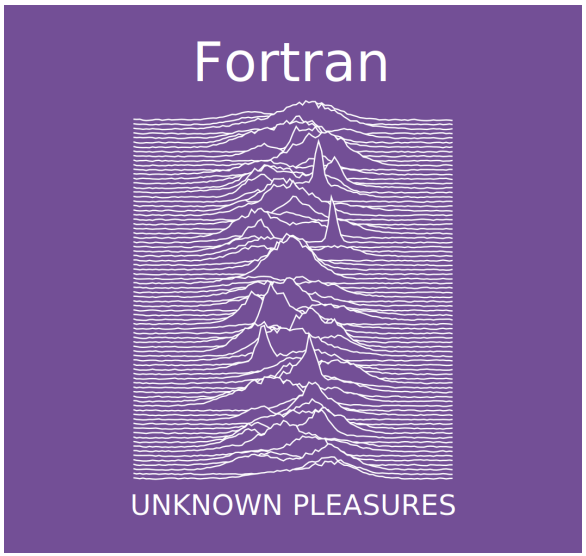
- Nouveau projet :
  https://github.com/fortran-lang/fpm/blob/main/PACKAGING.md
  - Options de `fpm new` :
    `--lib --app --test --example ...`
- Transformer un projet existant en projet fpm :
  - Créer les sous-répertoires, le manifeste fpm.toml, etc.
    `$ fpm new --backfill mon_projet`
  - Migrer les fichiers dans app/, `src/`, `test/` (avec `git mv` si le projet utilise git)
  - Adapter le `fpm.toml`, en particulier les dépendances.
  - Pour l'instant pas de constructions optionnelles : mais on peut créer plusieurs manifestes et utiliser des liens symboliques.

Ça n'est pas fini, le développement continue

- Projets déjà créés ou portés sous fpm : https://github.com/Beliavsky/Fortran-packages-list

- Un *fpm registry* en développement : https://registry-phi.vercel.app/
- `fpm publish`
- `fpm search` (https://github.com/urbanjost/fpm-search), en développement

Conclusion : le plaisir retrouvé !



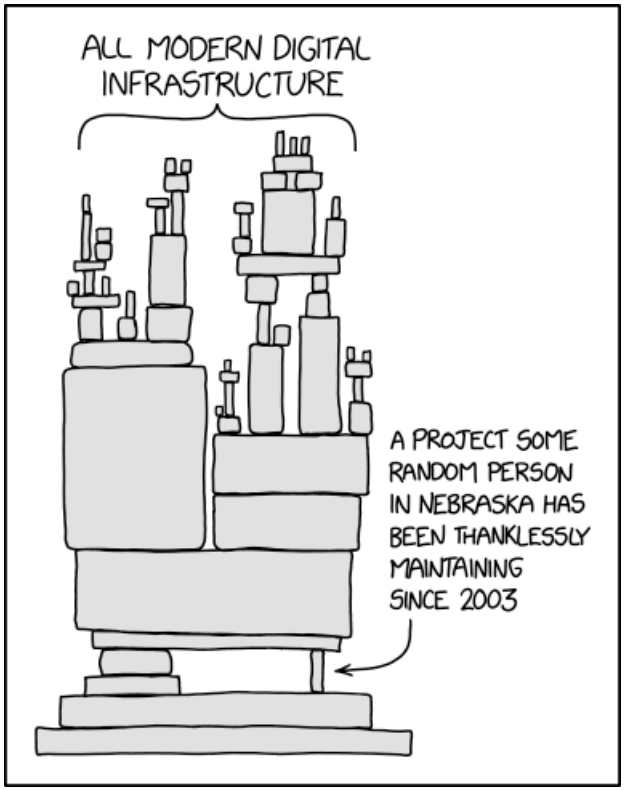
- Un système de construction et gestionnaire de paquets simple et intuitif !
- Bien documenté.
- Extensible par greffons.
- Écrit en Fortran (contribuer sur GitHub),
- par une communauté prête à vous aider sur le Fortran Discourse.

ANNEXES

Publications parlant de fpm

- Brad Richardson, Ondřej Čertík, Milan Curcic, « Fortran Package Manager », FortranCon2020, Zurich (Switzerland), https://youtu.be/gO76uWUtKvE.
- Milan Curcic, Ondřej Čertík, Laurence Kedward, Vincent Magnin, Ivan Pribec, Brad Richardson, et Jeremie Vandenplas. « Toward a Thriving Open Source Fortran Community », FortranCon2020, Zurich (Switzerland), https://youtu.be/JUHS-JFvs90.
- Laurence Kedward, Balint Aradi, Ondrej Certık, Milan Curcic, Sebastian Ehlert, Philipp Engel, Rohit Goswami, et al. « The State of Fortran 2021 ». FortranCon2021, Zurich (Switzerland), https://youtu.be/EgkQdqJIJU0.
- Sebastian Ehlert, Ondřej Čertík, Milan Curcic, Jakub Jelínek, Laurence Kedward, et al., « Fortran Package Manager : Toward a rich ecosystem of Fortran packages ». PackagingCon 2021, nov. 2021, Online, United States. ⟨hal-03519886⟩
- Laurence J. Kedward, Balint Aradi, Ondrej Certik, Milan Curcic, Sebastian Ehlert, Philipp Engel, Rohit Goswami, et al. « The State of Fortran ». Computing in Science & Engineering 24, n° 2 (1 mars 2022) : 63-72. https://doi.org/10.1109/MCSE.2022.3159862.
- Vincent Magnin, Jose Alves, Antoine Arnoud, Arjen Markus, Michele Esposito Marzino, « Fortran… et puis quoi encore ? », *Bulletin 1024,* Société Informatique de France, n° 22, p. 143-161, novembre 2023, https://doi.org/10.48556/SIF.1024.22.143, licence CC BY-NC-ND 4.0. Traduction officielle en anglais “Fortran... ok, and what’s next ?” : https://hal.science/hal-04448657 ou https://arxiv.org/abs/2402.07520

Enfer et damnation



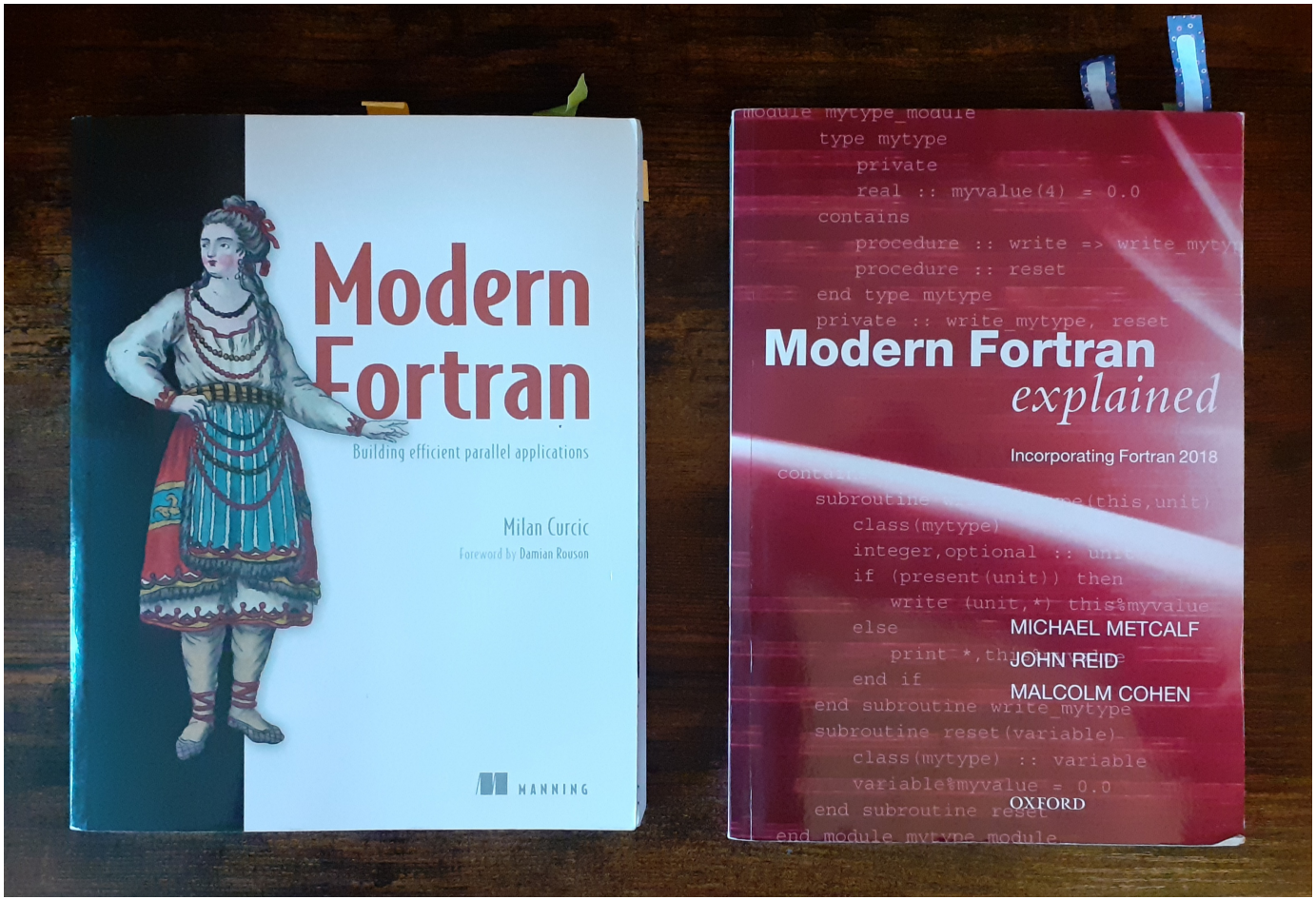
- Être raisonnable avec le nombre de dépendances.
- Gérer les dépendances circulaires.
- Si les dépendances sont fpm, fpm charge les codes sources. Pas de risque de perdre l'accès, sauf si on a fait un `fpm clean --all` !

fpm est lui-même un projet fpm ! La preuve

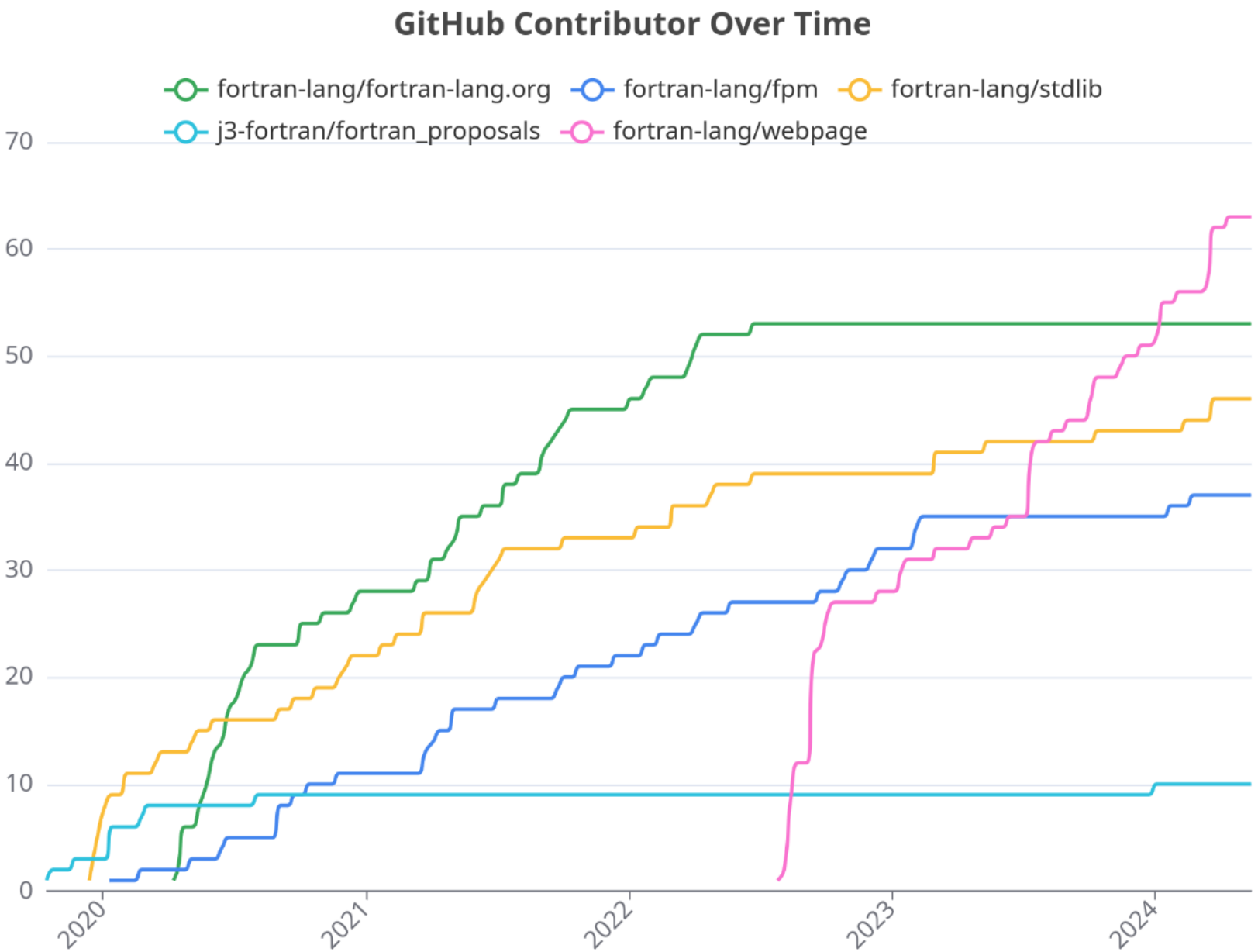
```
[dependencies]
toml-f.git = "https://github.com/toml-f/toml-f"
toml-f.rev = "d7b892b1d074b7cfc5d75c3e0eb36ebc1f7958c1"
M_CLI2.git = "https://github.com/urbanjost/M_CLI2.git"
M_CLI2.rev = "7264878cdb1baff7323cc48596d829ccfe7751b8"
fortran-regex.git  = "https://github.com/perazz/fortran-regex"
fortran-regex.tag  = "1.1.2"
jonquil.git = "https://github.com/toml-f/jonquil"
jonquil.rev = "4fbd4cf34d577c0fd25e32667ee9e41bf231ece8"
fortran-shlex.git = "https://github.com/perazz/fortran-shlex"
fortran-shlex.tag = "1.0.1"
```

Fortran. . . et puis quoi encore ?

- Rejoignez la communauté Fortran-lang.org.
- Essayez ses outils : fpm, stdlib, etc.
- Si vous avez besoin de vous mettre à niveau en Fortran (moderne !) :



Une communauté dynamique



Quelques liens supplémentaires

- https://github.com/Beliavsky/Fortran-packages-list
- https://github.com/Beliavsky/Fortran-code-on-GitHub
- https://github.com/Beliavsky/Fortran-Tools
- ...