

Fortran Package Manager

Brad Richardson
Ondrej Certik
Milan Curcic
FortranCon2020

Outline

- What problems does a package manager solve?
- How does fpm solve them?
- Live demo
- Future development
- Questions

What problems does a package manager solve?

- How do I manage my external dependencies?
- How do I build my project?
- How do I test my project?
- How do I create a new project?
- How do I find available libraries?

How does fpm solve these problems?

Managing Dependencies

- Specify the name of the dependency
- Where to find it
- And what version you need

```
[dependencies]
library1 = { git = "https://github.com/someone/library1.git", tag = "v1.2.3" }
package2 = { git = "https://github.com/other/package2.git", rev = "a12bc3" }
```

How does fpm solve these problems?

Building

- `fpm build`
- Fetches any dependencies
- Scans your sources
- Builds them in the proper order

How does fpm solve these problems?

Testing

- `fpm test`
- A program in `test/main.f90` is compiled and run
- Other test programs can be specified in `fpm.toml`

How does fpm solve these problems?

New Projects

- `fpm new new_project_name [--with-executable] [--with-test]`
- Creates new project with
 - basic `fpm.toml`
 - a module with a single subroutine that prints “Hello, new_project_name!”
 - (optionally) a program that just calls the provided subroutine
 - (optionally) a test that just prints “Put some tests in here!”

How does fpm solve these problems?

Finding Libraries

- COMING SOON
- ``fpm search for_something``
- Check the registry for packages with matching names and/or descriptions

Demo Time

Future Development

- Detailed Specification
- Re-write in Fortran
- Centralized Registry

Learn more at:
<https://github.com/fortran-lang/fpm>

Questions

Email: everythingfunctional@protonmail.com
Github: [everythingfunctional](https://github.com/everythingfunctional)
Twitter: [@everythingfunct](https://twitter.com/everythingfunct)

Demo Backup Slides

Finished in 0.04s

Finished in 0.04s 158x42

```
[darter:~/fpm_demo] fpm new fortrancon2020 --with-executable --with-test
Initialized empty Git repository in /home/brad/fpm_demo/fortrancon2020/.git/
[darter:~/fpm_demo] cd fortrancon2020
[darter:~/fpm_demo/fortrancon2020] fpm run
# gfortran (for build/gfortran_debug/fortrancon2020/fortrancon2020.o build/gfortran_debug/fortrancon2020/fortrancon2020.mod)
# ar (for build/gfortran_debug/fortrancon2020/libfortrancon2020.a)
ar: creating build/gfortran_debug/fortrancon2020/libfortrancon2020.a
# gfortran (for build/gfortran_debug/app/main.o)
# gfortran (for build/gfortran_debug/app/fortrancon2020)
# gfortran (for build/gfortran_debug/test/main.o)
# gfortran (for build/gfortran_debug/test/runTests)
Hello, fortrancon2020!
[darter:~/fpm_demo/fortrancon2020] _
```

(master)

(master)



Finished in 0.04s

Finished in 0.04s 78x42

```
rtran_debug/fortrancon2020/fortrancon2020.mod)
# ar (for build/gfortran_debug/fortrancon2020/libfortrancon2020.a)
ar: creating build/gfortran_debug/fortrancon2020/libfortrancon2020.a
# gfortran (for build/gfortran_debug/app/main.o)
# gfortran (for build/gfortran_debug/app/fortrancon2020)
# gfortran (for build/gfortran_debug/test/main.o)
# gfortran (for build/gfortran_debug/test/runTests)
Hello, fortrancon2020!
```

[darter:~/fpm_demo/fortrancon2020] atom .

(master

[darter:~/fpm_demo/fortrancon2020] _

(master)

Project - ~/fpm_demo/fortrancon2020 - Atom

File Edit View Selection Find Packages Help Debugger Nuclide

Project

fpm.toml

x

fortrancon2020

app

main.f90

build

src

fortrancon2020.f90

test

main.f90

.gitignore

fpm.toml

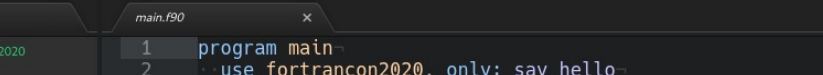
README.md

```
1 name = "fortrancon2020"
2 version = "0.1.0"
3 license = "license"
4 author = "Jane Doe"
5 maintainer = "jane.doe@example.com"
6 copyright = "2020 Jane Doe"
7
```

```

Finished in 0.04s 78x42
rtran_debug/fortrancon2020/fortrancon2020.mod)
# ar (for build/gfortran_debug/fortrancon2020/libfortrancon2020.a)
ar: creating build/gfortran_debug/fortrancon2020/libfortrancon2020.a
# gfortran (for build/gfortran_debug/app/main.o)
# gfortran (for build/gfortran_debug/app/fortrancon2020)
# gfortran (for build/gfortran_debug/test/main.o)
# gfortran (for build/gfortran_debug/test/runTests)
Hello, fortrancon2020!
[darter:~/fpm_demo/fortrancon2020] atom .
)
[darter:~/fpm_demo/fortrancon2020]

```



The screenshot shows the Nuclide IDE interface. The top menu bar includes File, Edit, View, Selection, Find, Packages, Help, Debugger, and Nuclide. The left sidebar shows a project named 'fortrancon2020' with a file explorer. The file explorer shows a folder 'app' containing 'main.f90'. The 'main.f90' file is selected. The editor window displays the code for 'main.f90'.

```
1 program main
2   use fortrancon2020, only: say_hello
3
4   implicit none
5
6   call say_hello
7 end program main
8
```


Finished in 0.04s

Finished in 0.04s 78x42

```
rtran_debug/fortrancon2020/fortrancon2020.mod)
# ar (for build/gfortran_debug/fortrancon2020/libfortrancon2020.a)
ar: creating build/gfortran_debug/fortrancon2020/libfortrancon2020.a
# gfortran (for build/gfortran_debug/app/main.o)
# gfortran (for build/gfortran_debug/app/fortrancon2020)
# gfortran (for build/gfortran_debug/test/main.o)
# gfortran (for build/gfortran_debug/test/runTests)
Hello, fortrancon2020!
```

[darter:~/fpm_demo/fortrancon2020] atom .

(master

[darter:~/fpm_demo/fortrancon2020] _

(master)

Project - ~/fpm_demo/fortrancon2020 - Atom

File Edit View Selection Find Packages Help Debugger Nuclide

Project

fortrancon2020.f90

x

fortrancon2020
 app
 main.f90
 build
 src
 fortrancon2020.f90
 test
 main.f90
 .gitignore
 fpm.toml
 README.md

```
1 module fortrancon2020
2   implicit none
3   private
4
5   public :: say_hello
6 contains
7   subroutine say_hello
8     print *, "Hello, fortrancon2020!"
9   end subroutine say_hello
10 end module fortrancon2020
11
```

Finished in 0.00s

Finished in 0.00s 78x42

```
rtran_debug/fortrancon2020/fortrancon2020.mod)
# ar (for build/gfortran_debug/fortrancon2020/libfortrancon2020.a)
ar: creating build/gfortran_debug/fortrancon2020/libfortrancon2020.a
# gfortran (for build/gfortran_debug/app/main.o)
# gfortran (for build/gfortran_debug/app/fortrancon2020)
# gfortran (for build/gfortran_debug/test/main.o)
# gfortran (for build/gfortran_debug/test/runTests)
Hello, fortrancon2020!
[darter:~/fpm_demo/fortrancon2020] atom .
```

(master

```
)
[darter:~/fpm_demo/fortrancon2020] fpm test
Put some tests in here!
```

(master)

```
[darter:~/fpm_demo/fortrancon2020] _
```

(master)

Project - ~/fpm_demo/fortrancon2020 - Atom

File Edit View Selection Find Packages Help Debugger Nuclide

Project

main.f90

x

fortrancon2020

app

main.f90

build

src

fortrancon2020.f90

test

main.f90

.gitignore

fpm.toml

README.md

```
1 program main
2   implicit none
3
4   print *, "Put some tests in here!"
5 end program main
6
```

Finished in 0.00s

```
Finished in 0.00s 78x42
rtran_debug/fortrancon2020/fortrancon2020.mod)
# ar (for build/gfortran_debug/fortrancon2020/libfortrancon2020.a)
ar: creating build/gfortran_debug/fortrancon2020/libfortrancon2020.a
# gfortran (for build/gfortran_debug/app/main.o)
# gfortran (for build/gfortran_debug/app/fortrancon2020)
# gfortran (for build/gfortran_debug/test/main.o)
# gfortran (for build/gfortran_debug/test/runTests)
Hello, fortrancon2020!
[darter:~/fpm_demo/fortrancon2020] atom .
)
[darter:~/fpm_demo/fortrancon2020] fpm test
Put some tests in here!
[darter:~/fpm_demo/fortrancon2020] cp ../addDependency/fpm.toml .
[darter:~/fpm_demo/fortrancon2020] cp ../addDependency/fortrancon2020.f90 src
[darter:~/fpm_demo/fortrancon2020] cp ../addDependency/main.f90 app
[darter:~/fpm_demo/fortrancon2020] _
```

(master)

(master)

(master)

(master)

(master)

Project - ~/fpm_demo/fortrancon2020 - Atom

File Edit View Selection Find Packages Help Debugger Nuclide

Project	fpm.toml
fortrancon2020	1 name = "fortrancon2020"
app	2 version = "0.1.0"
main.f90	3 license = "license"
build	4 author = "Jane Doe"
src	5 maintainer = "jane.doe@example.com"
fortrancon2020.f90	6 copyright = "2020 Jane Doe"
test	7
main.f90	8 [dependencies]
.gitignore	9 iso_varying_string = { git = "https://gitlab.com/everythingfunction"
fpm.toml	10
README.md	

Finished in 0.00s

Finished in 0.00s 78x42

```
rtran_debug/fortrancon2020/fortrancon2020.mod)
# ar (for build/gfortran_debug/fortrancon2020/libfortrancon2020.a)
ar: creating build/gfortran_debug/fortrancon2020/libfortrancon2020.a
# gfortran (for build/gfortran_debug/app/main.o)
# gfortran (for build/gfortran_debug/app/fortrancon2020)
# gfortran (for build/gfortran_debug/test/main.o)
# gfortran (for build/gfortran_debug/test/runTests)
Hello, fortrancon2020!
[darter:~/fpm_demo/fortrancon2020] atom .
)
[darter:~/fpm_demo/fortrancon2020] fpm test
Put some tests in here!
[darter:~/fpm_demo/fortrancon2020] cp ../addDependency/fpm.toml .
[darter:~/fpm_demo/fortrancon2020] cp ../addDependency/fortrancon2020.f90 src
[darter:~/fpm_demo/fortrancon2020] cp ../addDependency/main.f90 app
[darter:~/fpm_demo/fortrancon2020] _
```

(master

(master)

(master)

(master)

(master)

Project - ~/fpm_demo/fortrancon2020 - Atom

File Edit View Selection Find Packages Help Debugger Nuclide

Project

fortrancon2020.f90

x

fortrancon2020

app

main.f90

build

src

fortrancon2020.f90

test

main.f90

.gitignore

fpm.toml

README.md

```
1 module fortrancon2020
2   use iso_varying_string, only: varying_string, assignment(=)
3
4   implicit none
5   private
6
7   public :: create_greeting, say_hello
8 contains
9   subroutine say_hello
10    print *, "Hello, fortrancon2020!"
11  end subroutine say_hello
12
13  function create_greeting(name) result(greeting)
14    character(len=*), intent(in) :: name
15    type(varying_string) :: greeting
16
17    greeting = "Hello, " // name // "!"
18  end function create_greeting
19 end module fortrancon2020
20
```

Finished in 0.03s

Finished in 0.03s 78x42

Put some tests in here!

```
[darter:~/fpm_demo/fortrancon2020] cp ../addDependency/fpm.toml . (master)
[darter:~/fpm_demo/fortrancon2020] cp ../addDependency/fortrancon2020.f90 src
[darter:~/fpm_demo/fortrancon2020] cp ../addDependency/main.f90 app (master)
[darter:~/fpm_demo/fortrancon2020] fpm run (master)
```

Cloning into 'build/dependencies/iso_varying_string'...

remote: Enumerating objects: 59, done.

remote: Counting objects: 100% (59/59), done.

remote: Compressing objects: 100% (47/47), done.

remote: Total 445 (delta 34), reused 15 (delta 10), pack-reused 386

Receiving objects: 100% (445/445), 90.70 KiB | 1.04 MiB/s, done.

Resolving deltas: 100% (299/299), done.

Note: switching to 'v1.0.0'.

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using `-c` with the switch command. Example:

```
git switch -c <new-branch-name>
```

Or undo this operation with:

```
git switch -
```

Turn off this advice by setting config variable `advice.detachedHead` to false

HEAD is now at b5877b9 Merge branch 'MigrateToFpm' into 'master'

```
# gfortran (for build/gfortran_debug/iso_varying_string/iso_varying_string.o b
build/gfortran_debug/iso_varying_string/iso_varying_string.mod)
```

```
# ar (for build/gfortran_debug/iso_varying_string/libiso_varying_string.a)
```

```
ar: creating build/gfortran_debug/iso_varying_string/libiso_varying_string.a
```

```
# gfortran (for build/gfortran_debug/fortrancon2020/fortrancon2020.o build/gfo
rtran_debug/fortrancon2020/fortrancon2020.mod)
```

```
# ar (for build/gfortran_debug/fortrancon2020/libfortrancon2020.a)
```

```
# gfortran (for build/gfortran_debug/app/main.o)
```

```
# gfortran (for build/gfortran_debug/app/fortrancon2020)
```

```
# gfortran (for build/gfortran_debug/test/runTests)
```

Hello, FortranCon 2020!

[darter:~/fpm_demo/fortrancon2020]

Project - ~/fpm_demo/fortrancon2020 - Atom

File Edit View Selection Find Packages Help Debugger Nuclide

Project

main.f90

x

fortrancon2020

app

main.f90

build

src

fortrancon2020.f90

test

main.f90

.gitignore

fpm.toml

README.md

1

program main

2

use fortrancon2020, only: create_greeting

3

use iso_varying_string, only: put_line

4

5

implicit none

6

7

call put_line(create_greeting("FortranCon 2020"))

8

end program main

9

Finished in 0.03s

```
Finished in 0.03s 78x42
[darter:~/fpm_demo/fortrancon2020] cp ../addDependency/main.f90 app (master)
[darter:~/fpm_demo/fortrancon2020] fpm run (master)
Cloning into 'build/dependencies/iso_varying_string'...
remote: Enumerating objects: 59, done.
remote: Counting objects: 100% (59/59), done.
remote: Compressing objects: 100% (47/47), done.
remote: Total 445 (delta 34), reused 15 (delta 10), pack-reused 386
Receiving objects: 100% (445/445), 90.70 KiB | 1.04 MiB/s, done.
Resolving deltas: 100% (299/299), done.
Note: switching to 'v1.0.0'.
```

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using `-c` with the switch command. Example:

```
git switch -c <new-branch-name>
```

Or undo this operation with:

```
git switch -
```

Turn off this advice by setting config variable `advice.detachedHead` to false

```
HEAD is now at b5877b9 Merge branch 'MigrateToFpm' into 'master'
# gfortran (for build/gfortran_debug/iso_varying_string/iso_varying_string.o b
uild/gfortran_debug/iso_varying_string/iso_varying_string.mod)
# ar (for build/gfortran_debug/iso_varying_string/libiso_varying_string.a)
ar: creating build/gfortran_debug/iso_varying_string/libiso_varying_string.a
# gfortran (for build/gfortran_debug/fortrancon2020/fortrancon2020.o build/gfo
rtan_debug/fortrancon2020/fortrancon2020.mod)
# ar (for build/gfortran_debug/fortrancon2020/libfortrancon2020.a)
# gfortran (for build/gfortran_debug/app/main.o)
# gfortran (for build/gfortran_debug/app/fortrancon2020)
# gfortran (for build/gfortran_debug/test/runTests)
Hello, FortranCon 2020!
```

```
[darter:~/fpm_demo/fortrancon2020] cp ../addTest/fpm.toml . (master)
```

```
[darter:~/fpm_demo/fortrancon2020] cp ../addTest/greeting_test.f90 test
```

```
[darter:~/fpm_demo/fortrancon2020] cp ../addTest/main.f90 test (master)
```

```
[darter:~/fpm_demo/fortrancon2020]
```

Project - ~/fpm_demo/fortrancon2020 - Atom

File Edit View Selection Find Packages Help Debugger Nuclide

Project fpm.toml x

```
1 name = "fortrancon2020"
2 version = "0.1.0"
3 license = "license"
4 author = "Jane Doe"
5 maintainer = "jane.doe@example.com"
6 copyright = "2020 Jane Doe"
7
8 [dependencies]
9 iso_varying_string = { git = "https://gitlab.com/everythingfunction
10
11 [dev-dependencies]
12 vegetables = { git = "https://gitlab.com/everythingfunctional/veget
13
```

fortrancon2020

- app
 - main.f90
- build
 - fortrancon2020.f90
- src
 - test
 - greeting_test.f90
 - main.f90
 - .gitignore
 - fpm.toml
 - README.md

Finished in 0.03s

```
Finished in 0.03s 78x42
[darter:~/fpm_demo/fortrancon2020] cp ../addDependency/main.f90 app (master)
[darter:~/fpm_demo/fortrancon2020] fpm run (master)
Cloning into 'build/dependencies/iso_varying_string'...
remote: Enumerating objects: 59, done.
remote: Counting objects: 100% (59/59), done.
remote: Compressing objects: 100% (47/47), done.
remote: Total 445 (delta 34), reused 15 (delta 10), pack-reused 386
Receiving objects: 100% (445/445), 90.70 KiB | 1.04 MiB/s, done.
Resolving deltas: 100% (299/299), done.
Note: switching to 'v1.0.0'.
```

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using `-c` with the switch command. Example:

```
git switch -c <new-branch-name>
```

Or undo this operation with:

```
git switch -
```

Turn off this advice by setting config variable `advice.detachedHead` to false

```
HEAD is now at b5877b9 Merge branch 'MigrateToFpm' into 'master'
# gfortran (for build/gfortran_debug/iso_varying_string/iso_varying_string.o b
uild/gfortran_debug/iso_varying_string/iso_varying_string.mod)
# ar (for build/gfortran_debug/iso_varying_string/libiso_varying_string.a)
ar: creating build/gfortran_debug/iso_varying_string/libiso_varying_string.a
# gfortran (for build/gfortran_debug/fortrancon2020/fortrancon2020.o build/gfo
rtan_debug/fortrancon2020/fortrancon2020.mod)
# ar (for build/gfortran_debug/fortrancon2020/libfortrancon2020.a)
# gfortran (for build/gfortran_debug/app/main.o)
# gfortran (for build/gfortran_debug/app/fortrancon2020)
# gfortran (for build/gfortran_debug/test/runTests)
Hello, FortranCon 2020!
```

```
[darter:~/fpm_demo/fortrancon2020] cp ../addTest/fpm.toml . (master)
```

```
[darter:~/fpm_demo/fortrancon2020] cp ../addTest/greeting_test.f90 test
```

```
[darter:~/fpm_demo/fortrancon2020] cp ../addTest/main.f90 test (master)
```

```
[darter:~/fpm_demo/fortrancon2020]
```

Project - ~/fpm_demo/fortrancon2020 - Atom

File Edit View Selection Find Packages Help Debugger Nuclide

```
Project main.f90 x
▼ fortrancon2020
  ▼ app
    main.f90
  ▼ build
  ▼ src
    fortrancon2020.f90
  ▼ test
    greeting_test.f90
    main.f90
  .gitignore
  fpm.toml
  README.md

1 program test
2 use greeting_test, only: &
3 greeting_greeting => test_greeting
4 use Vegetables_m, only: TestItem_t, testThat, runTests
5
6 implicit none
7
8 call run()
9 contains
10 subroutine run()
11 type(TestItem_t) :: tests
12 type(TestItem_t) :: individual_tests(1)
13
14 individual_tests(1) = greeting_greeting()
15 tests = testThat(individual_tests)
16
17 call runTests(tests)
18 end subroutine run
19 end program test
20
```


Finished in 0.03s

```
Finished in 0.03s 78x42
[darter:~/fpm_demo/fortrancon2020] cp ../addDependency/main.f90 app (master)
[darter:~/fpm_demo/fortrancon2020] fpm run (master)
Cloning into 'build/dependencies/iso_varying_string'...
remote: Enumerating objects: 59, done.
remote: Counting objects: 100% (59/59), done.
remote: Compressing objects: 100% (47/47), done.
remote: Total 445 (delta 34), reused 15 (delta 10), pack-reused 386
Receiving objects: 100% (445/445), 90.70 KiB | 1.04 MiB/s, done.
Resolving deltas: 100% (299/299), done.
Note: switching to 'v1.0.0'.
```

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using `-c` with the switch command. Example:

```
git switch -c <new-branch-name>
```

Or undo this operation with:

```
git switch -
```

Turn off this advice by setting config variable `advice.detachedHead` to false

```
HEAD is now at b5877b9 Merge branch 'MigrateToFpm' into 'master'
# gfortran (for build/gfortran_debug/iso_varying_string/iso_varying_string.o b
build/gfortran_debug/iso_varying_string/iso_varying_string.mod)
# ar (for build/gfortran_debug/iso_varying_string/libiso_varying_string.a)
ar: creating build/gfortran_debug/iso_varying_string/libiso_varying_string.a
# gfortran (for build/gfortran_debug/fortrancon2020/fortrancon2020.o build/gfo
rtran_debug/fortrancon2020/fortrancon2020.mod)
# ar (for build/gfortran_debug/fortrancon2020/libfortrancon2020.a)
# gfortran (for build/gfortran_debug/app/main.o)
# gfortran (for build/gfortran_debug/app/fortrancon2020)
# gfortran (for build/gfortran_debug/test/runTests)
Hello, FortranCon 2020!
```

```
[darter:~/fpm_demo/fortrancon2020] cp ../addTest/fpm.toml . (master)
[darter:~/fpm_demo/fortrancon2020] cp ../addTest/greeting_test.f90 test
[darter:~/fpm_demo/fortrancon2020] cp ../addTest/main.f90 test (master)
[darter:~/fpm_demo/fortrancon2020]
```

Project - ~/fpm_demo/fortrancon2020 - Atom

File Edit View Selection Find Packages Help Debugger Nuclide

```
Project: greeting_test.f90 x
└─ forttrancon2020
  └─ app
    └─ main.f90
  └─ build
    └─ src
      └─ forttrancon2020.f90
    └─ test
      └─ greeting_test.f90
      └─ main.f90
      └─ .gitignore
      └─ fpm.toml
      └─ README.md

1 module greeting_test
2   use forttrancon2020, only: create_greeting
3   use Vegetables_m, only: Result_t, TestItem_t, assertEquals, des
4
5   implicit none
6   private
7
8   public :: test_greeting
9 contains
10  function test_greeting() result(tests)
11    type(TestItem_t) :: tests
12
13    type(TestItem_t) :: individual_tests(1)
14
15    individual_tests = it("says hello", check_greeting)
16    tests = describe("create_greeting", individual_tests)
17  end function test_greeting
18
19  function check_greeting() result(result_)
20    type(Result_t) :: result_
21
22    result_ = assertEquals("Hello, FortranCon 2020!", create_gree
23  end function check_greeting
24 end module greeting_test
25
```


Finished in 0.12s

Finished in 0.12s 78x42

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using `-c` with the switch command. Example:

```
git switch -c <new-branch-name>
```

Or undo this operation with:

```
git switch -
```

Turn off this advice by setting config variable `advice.detachedHead` to false

HEAD is now at c2df5c8 Merge branch 'MigrateToFpm' into 'master'

```
# gfortran (for build/gfortran_debug/strff/strff.o build/gfortran_debug/strff/strff.mod)
```

```
# ar (for build/gfortran_debug/strff/libstrff.a)
```

```
ar: creating build/gfortran_debug/strff/libstrff.a
```

```
# gfortran (for build/gfortran_debug/vegetables/vegetables_m.o build/gfortran_debug/vegetables/vegetables_m.mod)
```

```
# ar (for build/gfortran_debug/vegetables/libvegetables.a)
```

```
ar: creating build/gfortran_debug/vegetables/libvegetables.a
```

```
# gfortran (for build/gfortran_debug/test/greeting_test.o build/gfortran_debug/test/greeting_test.mod)
```

```
# gfortran (for build/gfortran_debug/test/main.o)
```

```
# gfortran (for build/gfortran_debug/test/runTests)
```

Running Tests

Test that

```
  create_greeting
    says hello
```

A total of 1 test cases

All Passed

Took 3.6e-5 seconds

A total of 1 test cases containing a total of 1 assertions

Finished in 0.12s 78x42

Project - ~/fpm_demo/fortrancon2020 - Atom

File Edit View Selection Find Packages Help Debugger Nuclide

Project

greeting_test.f90

x

```
1 module greeting_test
2   use fortrancon2020, only: create_greeting
3   use Vegetables_m, only: Result_t, TestItem_t, assertEquals, de
4
5   implicit none
6   private
7
8   public :: test_greeting
9 contains
10  function test_greeting() result(tests)
11    type(TestItem_t) :: tests
12
13    type(TestItem_t) :: individual_tests(1)
14
15    individual_tests = it("says hello", check_greeting)
16    tests = describe("create_greeting", individual_tests)
17  end function test_greeting
18
19  function check_greeting() result(result_)
20    type(Result_t) :: result_
21
22    result_ = assertEquals("Hello, FortranCon 2020!", create_gre
23  end function check_greeting
24 end module greeting_test
25
```