
fortdepend Documentation

Release 2.2.0

Peter Hill, David Dickinson

Jul 05, 2023

CONTENTS:

1	Installation	3
2	Basic usage	5
3	Advanced usage	7
4	API reference	9
5	Limitations	13
6	Indices and tables	15
	Python Module Index	17
	Index	19

`fortdepend` is a python package to automatically generate Fortran dependencies.

Given a set of files, `fortdepend` automatically constructs the dependency graph for the programs and files and can write a dependency file suitable for Makefiles. `fortdepend` now uses `pcpp`, a preprocessor written in Python, so it can determine which modules will actually be used when you compile.

You can even use `fortdepend` to draw the graph of the module dependencies (requires `graphviz`)!

Original script by D. Dickinson

INSTALLATION

You can install fortdepend with pip:

```
pip3 install --user fortdepend
```


BASIC USAGE

The quickest way to run `fortdepend` is like so:

```
fortdepend -o Makefile.dep
```

This will look for all files ending in `.f90` or `.F90` in the current directory and write the output to `Makefile.dep`. The output will be something like this:

```
test : \
    moduleA.o \
    moduleB.o \
    moduleC.o \
    moduleD.o \
    programTest.o

moduleA.o :

moduleB.o : \
    moduleA.o

moduleC.o : \
    moduleA.o \
    moduleB.o

moduleD.o : \
    moduleC.o
```

You could then get a basic makefile working by putting the following in `Makefile`:

```
.f90.o:
    gfortran -c $<

test:
    gfortran $^ -o $@

include Makefile.dep
```

And `make test` will magically build everything in the correct order!

ADVANCED USAGE

You can specify preprocessor macros with `-D` and search paths with `-I`:

```
fortdepend -DMACRO=42 -Isome/include/dir -o Makefile.dep
```

will replace instances of `MACRO` with `42` according to the usual C99 preprocessor rules. This can be used to conditionally use some modules or change which module is used at compile time.

Full command line arguments:

Generate Fortran dependencies

```
usage: fortdepend [-h] [-f FILES [FILES ...]] [-D NAME[=DESCRIPTION]
                  [NAME[=DESCRIPTION] ...]] [-I dir] [-b BUILD] [-o OUTPUT]
                  [-g] [-v] [-w] [-c] [-e EXCLUDE_FILES [EXCLUDE_FILES ...]]
                  [-i IGNORE_MODULES [IGNORE_MODULES ...]] [-s] [-n]
                  [--version]
```

3.1 Named Arguments

-f, --files	Files to process
-D	Preprocessor define statements
-I	Add dir to the preprocessor search path
-b, --build	Build Directory (prepended to all files in output) Default: ""
-o, --output	Output file
-g, --graph	Make a graph of the project Default: False
-v, --verbose	explain what is done Default: False
-w, --overwrite	Overwrite output file without warning Default: False

-c, --colour	Print in colour
	Default: False
-e, --exclude-files	Files to exclude
-i, --ignore-modules	Modules to ignore
-s, --skip-programs	Don't include programs in the output file
	Default: False
-n, --no-preprocessor	Don't use the preprocessor
	Default: False
--version	show program's version number and exit

Here's a slightly more advanced example of how to use `fortdepend` in your makefiles:

```
# Script to generate the dependencies
MAKEDEPEND=/path/to/fortdepend

# $(DEP_FILE) is a .dep file generated by fortdepend
DEP_FILE = my_project.dep

# Preprocessor flags
CPPFLAGS = -DFEATURE -Iinclude/dir

# Source files to compile
OBJECTS = mod_file1.f90 \
         mod_file2.f90

# Make sure everything depends on the .dep file
all: $(actual_executable) $(DEP_FILE)

# Make dependencies
.PHONY: depend
depend: $(DEP_FILE)

# The .dep file depends on the source files, so it automatically gets updated
# when you change your source
$(DEP_FILE): $(OBJECTS)
    @echo "Making dependencies!"
    cd $(SRCPATH) && $(MAKEDEPEND) -w -o /path/to/$(DEP_FILE) $(CPPFLAGS) -f $(OBJECTS)

include $(DEP_FILE)
```

This will automatically rebuild the dependency file if any of the source files change. You might not like to do this if you have a lot of files and need to preprocess them!

API REFERENCE

4.1 fortdepend.fort_depend module

```
class fortdepend.fort_depend.FortranProject(name=None, exclude_files=None, files=None,  
                                           ignore_modules=None, macros=None,  
                                           cpp_includes=None, use_preprocessor=True,  
                                           verbose=False)
```

Bases: object

Read a set of Fortran source files and produce a set of *FortranFile*, *FortranModule* and the dependencies between them

Example

This is the main class for interacting with a Fortran project. The minimal “useful” thing is:

```
>>> import fortdepend  
>>> my_project = fortdepend.FortranProject()  
>>> my_project.write_depends()
```

This will read all the .f90 and .F90 files in the current directory and write the dependencies to “makefile.dep”

Parameters

- **name** (*str*) – Name of the project (default: name of current directory)
- **exclude_files** (*list of str*) – List of files to exclude
- **files** (*list of str*) – List of files to include (default: all in current directory)
- **ignore_modules** (*list of str*) – List of module names to ignore_mod (default: iso_c_binding and iso_fortran_env)
- **macros** (*dict, list or str*) – Preprocessor macro definitions
- **cpp_includes** (*list of str*) – List of directories to add to preprocessor search path
- **use_preprocessor** (*bool*) – Use the preprocessor (default: True)
- **verbose** (*bool*) – Print more messages (default: False)

get_all_used_files(*module_name*)

Get the complete set of files that a module requires, either directly or indirectly

Parameters

module_name (*str*) – A module name

Returns

list of filenames (str)

get_all_used_modules(*module_name*)

Get the complete set of modules that *module_name* requires, either directly or indirectly

Parameters

module_name (*str*) – A module name

Returns

list of module names (str)

get_depends_by_file(*verbose=False*)

Return the set of which files each file directly depends on

Parameters

verbose – Print progress messages

Returns

dict of [FortranFile](#) and a list of [FortranFile](#)

get_depends_by_module(*verbose=False*)

Return the set of which modules each module directly depends on

Parameters

verbose – Print progress messages

Returns

dict of [FortranModule](#) and a list of [FortranModule](#)

get_modules()

Return a dict of all the modules found in the project

Works by iterating over the list of [FortranFile](#) and merging their dicts of [FortranModule](#)

Returns

dict of module name (str) and [FortranModule](#) objects

get_source(*extensions=None*)

Return a list of filenames ending with any of extensions

Parameters

extensions – List of file extensions (defaults to [“.f90”, “.F90”])

make_graph(*filename=None, format='svg', view=True*)

Draw a graph of the project using graphviz

Parameters

- **filename** (*str*) – Name of the output file
- **format** (*str*) – Image format (default: ‘svg’)
- **view** (*bool*) – Immediately display the graph [True]

remove_ignored_modules(*ignore_modules=None*)

Remove the modules in iterable *ignore_modules* from all dependencies

Parameters

ignore_modules (*iterable of str*) – module names to ignore (default: `iso_c_binding` and `iso_fortran_env`)

write_depends(*filename*='makefile.dep', *overwrite*=False, *build*='', *skip_programs*=False)

Write the dependencies to file

Parameters

- **filename** (*str*) – Name of the output file
- **overwrite** (*bool*) – Overwrite existing dependency file [False]
- **build** (*str*) – Directory to prepend to filenames
- **skip_programs** (*bool*) – Don't write dependencies for programs

4.2 fortdepend.graph module

class fortdepend.graph.**Graph**(*tree*, *filename*=None, *format*='svg', *view*=True)

Bases: object

Draw a graph of the project using graphviz

Parameters

- **filename** (*str*) – Name of the output file
- **format** (*str*) – Image format
- **view** (*bool*) – Immediately display the graph [True]

draw()

Render the graph to an image

4.3 fortdepend.preprocessor module

class fortdepend.preprocessor.**FortranPreprocessor**

Bases: Preprocessor

parse_to_string(*text*, *source*)

4.4 fortdepend.smartopen module

fortdepend.smartopen.**smart_open**(*filename*, *mode*='Ur')

Open stdin or stdout using a contextmanager

From: <http://stackoverflow.com/a/29824059/2043465>

Parameters

- **filename** (*str*) – name of file to open. Can be '-' for stdin/stdout
- **mode** (*str*) – usual mode string for open()

4.5 fortdepend.units module

class fortdepend.units.**FortranFile**(*filename=None, macros=None, readfile=True, cpp_includes=None, use_preprocessor=True*)

Bases: object

The modules and dependencies of a Fortran source file

Parameters

- **filename** (*str*) – Source file
- **macros** (*iterable*) – Dict of preprocessor macros to be expanded
- **readfile** (*bool*) – Read and process the file [True]
- **cpp_includes** (*list of str*) – List of directories to add to preprocessor search path
- **use_preprocessor** (*bool*) – Preprocess the source file [True]

get_modules(*contents, macros=None*)

Return all the modules or programs that are in the file

Parameters

- **contents** (*str*) – Contents of the source file

get_uses()

Return a sorted list of the modules this file USEs

class fortdepend.units.**FortranModule**(*unit_type, name, source_file=None, text=None, macros=None*)

Bases: object

A Fortran Module or Program

Parameters

- **unit_type** (*str*) – ‘module’ or ‘program’
- **name** (*str*) – Name of the module/program
- **source_file** (*str*) – Name of the file containing the module/program
- **text** (*tuple*) – Tuple containing source_file contents, and start and end lines of the module
- **macros** (*dict*) – Any defined macros

get_uses(*contents, macros=None*)

Return which modules are used in the file after expanding macros

Parameters

- **contents** (*str*) – Contents of the source file
- **macros** (*dict*) – Dict of preprocessor macros to be expanded

LIMITATIONS

- `fortdepend` requires Python 3.
- `fortdepend` works by looking for matching pairs of `program <name>/end program <name>` and `module <name>/end module <name>`, and so will not work on Fortran 77-style files that just use `end` without the appropriate label.

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

f

`fortdepend.fort_depend`, 9
`fortdepend.graph`, [11](#)
`fortdepend.preprocessor`, [11](#)
`fortdepend.smartopen`, [11](#)
`fortdepend.units`, [12](#)

INDEX

D

`draw()` (*fortdepend.graph.Graph* method), 11

F

`fortdepend.fort_depend`
module, 9

`fortdepend.graph`
module, 11

`fortdepend.preprocessor`
module, 11

`fortdepend.smartopen`
module, 11

`fortdepend.units`
module, 12

`FortranFile` (class in *fortdepend.units*), 12

`FortranModule` (class in *fortdepend.units*), 12

`FortranPreprocessor` (class in *fortdepend.preprocessor*), 11

`FortranProject` (class in *fortdepend.fort_depend*), 9

G

`get_all_used_files()` (*fortdepend.fort_depend.FortranProject* method), 9

`get_all_used_modules()` (*fortdepend.fort_depend.FortranProject* method), 10

`get_depends_by_file()` (*fortdepend.fort_depend.FortranProject* method), 10

`get_depends_by_module()` (*fortdepend.fort_depend.FortranProject* method), 10

`get_modules()` (*fortdepend.fort_depend.FortranProject* method), 10

`get_modules()` (*fortdepend.units.FortranFile* method), 12

`get_source()` (*fortdepend.fort_depend.FortranProject* method), 10

`get_uses()` (*fortdepend.units.FortranFile* method), 12

`get_uses()` (*fortdepend.units.FortranModule* method), 12

`Graph` (class in *fortdepend.graph*), 11

M

`make_graph()` (*fortdepend.fort_depend.FortranProject* method), 10

module

`fortdepend.fort_depend`, 9

`fortdepend.graph`, 11

`fortdepend.preprocessor`, 11

`fortdepend.smartopen`, 11

`fortdepend.units`, 12

P

`parse_to_string()` (*fortdepend.preprocessor.FortranPreprocessor* method), 11

R

`remove_ignored_modules()` (*fortdepend.fort_depend.FortranProject* method), 10

S

`smart_open()` (in module *fortdepend.smartopen*), 11

W

`write_depends()` (*fortdepend.fort_depend.FortranProject* method), 10