How to use Fordocu

Current version: Johan Meijdam, Bas van 't Hof and Lennart Ros, VORtech, 2010-04-01

1 Contents

This document starts with a description how to install Fordocu and Perl, so it can be used (Section 2 and 3).

The next step is testing the installation and learning how to use Fordocu. This is done in section 4: "First use". Finally, the Fordocu command-line options are described in more detail in section 5 "Detailed use". The last section tells something about how to read the Fordocu error messages.

2 Overview of the installation

To install Fordocu follow the following steps, which are explained in the following sections:

1. Install Perl, version 5.003 or higher
   This step may be skipped if Perl is already available
2. Install the Perl-package XML::Simple
   This step may be skipped if XML::Simple is already available
3. Install the application SVN
   This step may be skipped
   - *if the source files you want to document are* **not** *managed using Subversion;*

     Fordocu will function without the SVN application, which is only used to create version history information for files managed with SVN. If your files are not under version management with SVN, you're not missing anything when you have no SVN.
   - *if SVN is already available (which you may test by typing '***svn --version'** *on the command line;*

     Software maintenance with Subversion on the Linux platform is usually done with the SVN application (or a GUI which communicates with it). On the Linux platform, SVN is therefore almost always available or unnecessary.

     The Windows platform, on the other hand, usually has a graphical equivalent of the SVN application. On Windows, the installation of SVN is therefore sometimes necessary.
4. Install Fordocu itself
5. Set environment variables.

3 Steps in the installation

3.1 Perl installation

Perl version 5.003 or higher is required. If a version of Perl is installed on your computer, you can determine its version by typing

perl --version

on the command line.

If there is no (suitable) Perl-installation on your computer, you can do the following:

- On Windows, download ActivePerl from www.perl.com . Click on 'download' and download 'ActivePerl'. Just click on “Activestate Binaries” for your Operating System and download the right Windows Installer. This should download a file named “ActivePerl-*.msi” .
- On Linux, install the distribution's Perl.

Check whether perl is on the search path by trying to determine the perl version as described above.

3.2 Installation of the package XML::Simple

The package XML::Simple is included in ActivePerl, so if you have installed ActivePerl (Section 3.1), you do not have to install XML::Simple.

A simple way to check whether installation of XML::Simple is necessary is to skip this step and to proceed with the installation of Fordocu. Installation of XML::Simple is necessary if you get the error message

Cant locate XML::Simple (@INC contains …)

The package XML::Simple is installed in one of the following ways (sometimes, security settings make some of the given options impossible).

After installation on Linux, check whether the directory containing the new package can be found on the path specified by environment variable $PERL5LIB.

Method 1

The easiest, Windows-specific way is installing XML::Simple using the Perl Package Manager. The PPM started using the Windows Start menu or from the command line, using:

ppm

Method 2

Type

cpan -i XML::Simple

on the command line (either on Linux or on Windows).

If the **cpan**-installation does not work on linux enter the XML-directory. Type the following commands on the command line:

perl Makefile.PL PREFIX=<*target_directory*>
make
make install

On Linux, check whether the directory containing the new package can be found on the path specified by environment variable $PERL5LIB.

3.3 Installation of the application SVN

In Section 2, it was explained that the SVN application probably does not have to be installed if you are using a Linux platform: either you already use it, or you don't need it (because you don't manage your source code with Subversion).

On the windows platform, sometimes a graphical client is used for SVN, e.g. Tortoise. These clients can sometimes not be used directly by Fordocu to get version information. It then is needed to instal the slik version of SVN, which can be downloaded from http://www.sliksvn.com/en/download/ . Choose the 32-bits or 64-bits version, depending on which system you would like to use Fordocu. SVN is installed by clicking on the downloaded file. This should be **Slik-Subversion-*-x64.msi** (on 64-bit Windows platforms), or **Slik-Subversion-*-win32.msi** (on 32 bit Windows platforms). This will not affect your SVN repositories and the graphical client can still be used.

3.4 Fordocu installation

Fordocu can be installed by unpacking its zip-file in a directory.

3.5 Environment variables

Fordocu uses some environment variables. To set these, change directory to the Fordocu directory and run the setup_fordocu script appropriate to your shell/operating system:

•type "`setup_fordocu`" on Microsoft Windows to start setup_fordocu.bat;

•or type "`. setup_fordocu.sh`" (including the leading dot and space characters) on bash, ksh and related shells;

•or type "`source setup_fordocu.csh`" on tcsh, csh and related shells.

Alternately, it is also possible to start these scripts from another subdirecory, but in that case the Fordocu subdirectory should be added as a command-line argument.

Important: these scripts set the environment variables only for the command window you are working in. To set these globally, it is possible to set the enviroment variables through My Computer / Properties / Advanced / Environment Variables on Windows or in .bashrc or .cshrc on Linux.

3.6 Setting an application for opening source files

A useful browser setting when using Fordocu is to allow yourself to open the source files directly from the documentation. In Linux and Windows Vista this can be done by associating your editor or Programming Environment to files with the file name extensions *.for and *.f90. Such a setting is created for the Firefox browser in the following steps:

1. Open the file *mimeTypes.rdf* in an editor.
   This file is located:
   - On Windows Vista:
     in *"%APPDATA%\Mozilla\Firefox\Profiles*.default\mimeTypes.rdf"*.
   - *On Linux:*
     in "~/.mozilla/firefox/.default/mimeTypes.rdf"*.
2. Find the following line in the file:
   <RDF:Seq RDF:about="urn:mimetypes:root">
   and replace it with the contents of the file *mimeTypes.xtra,* which is found in the directory of the Fordocu-delivery.
3. Start up Firefox (first close it if it is open already).
4. Select Tools/Edit → Options/Preferences → Content/FileTypes in Firefox.
   There should be an item '*Fortran Files*' among the file types for which you can specify an

application.
5. Select your editor/programming environment.

After you have successfully done these steps of installation, you should be able to run Fordocu by starting the script.

4 First use

We will try to use Fordocu on the file test.f90 in the subdirectory "test" in the Fordocu installation.

This section assumes you have a command window open.

4.1 Step 1: set the environment variables

In case the environment variables aren't set globally, ensure the appropriate `setup_fordocu` script is run (see section "Environment variables" above).

4.2 Step 2: use Fordocu on the example source

1. Change directory to the directory containg the files you want to be documented by Fordocu. In this case we'll change to the test directory, containing the file test.f90.
2. Start the appropriate `example` script. This produces a file called `comments.xml` in the directory and a subdirectory called `html`. This `html` directory contains the documentation, which can be read in a web browser by opening file `index.html`.

5 Detailed use

When documenting multiple source files, it is important to combine them all in a single invocation of Fordocu. It is possible to use wildcards like the asterisk to indicate multiple files, but Fordocu doesn't recurse into subdirectories. Whether filenames are case sensitive is platform dependent (on Windows they aren't, on Linux they are).

5.1 Custom settings

A convenient way to start the program is using a script file (`.bat` file for Windows, `.sh` for Linux). In the examples directory some sample script files can be found.

The file is created in the following steps:

- Copy the most appropriate example script file from the examples directory to the Fortran source directory.
- Open the script file in an editor.
- Change the settings in the script file. A description of the command-line arguments that can be used can be found in the secion "Command-line arguments" below.
- On linux, make the script executable; type
  `chmod +x <scriptname>`
  on the command line (replace `<scriptname>` with the name of the script).

5.2 Command-line arguments

It is possible to change Fordocu's behaviour by setting its command-line arguments.

Source type

The first category indicates whether input files are fixed format or free format. These arguments are **`-fixed`** and **`-free`**. There command-line arguments are always processed in the order given. For instance, it is possible to process a free format file free.f90 as well as a fixed format file fixed.f77 in a single Fordocu invocation:

```
fordocu.pl -free *.f90 -fixed *.f77
```

Include files

To find include files, an include path can be added with the **`-I <path>`** option. This path can contain multiple subdirectories, which should be colon-separated. For instance:

```
-I some_directory:other_directory:other_directory/other_directorys_subdirectory
```

Comments type

To specify how comments are formatted, two categories of arguments are available.

The first indicates what source lines are assumed to be comments. Normally, only lines indicated with a double exclamation mark "`!!`" are assumed to be comment lines. When using command-line argument **`-1`**, lines containing a single exclamation mark will be used for documentation as well. Another restriction can be made using command-line argument **`-eol`**: only end-of-line comments are accepted in case of variable declarations.

The second command formatting arguments are `-cp`, `-cs` and `-ch`. When **`-cp`** is given, comments are used exactly as specified in the source code: it is assumed they are "preformatted". This is the default option, so `-cp` can be omitted. Using **`-ch`**, it is assumed the comments contain HTML mark-

up. Finally, "smart" comments are available through option **-cs**. These "smart comments" provide the following features:

1.Verbatim mode, the first line indicated with !!>, the last line with !!<. Comments between these marks are considered pre formatted (like -cp).

2.List mode. List items in comments are indicated with an arbitrary number of spaces, followed by a minus sign "-".

3.Mark-up. Words that should appear in italics, should be put between underscores (e.g. _word_); In a similar fashion, words that should appear between asterisks (e.g. *word*).

HTML output

Option **-nir** hides the listing of all routines being represented by an interface.

Improving HTML performance

When option **-nic** is used, routines in interface do not appear as "being called" by this interface in call trees. This improves the performance of the call trees in the documentation, but will remove actual links.

When using **-hide <name>**, the size of the call tree is reduced, because the routine indicated by <name> won't be shown in the call tree. Using this argument improves the performance of the call trees and reduces the number of links. It is mainly aimed at routines that are called in each routine, like trace functions. It is possible to specify the -hide option multiple times, for instance:

```
fordocu.pl -hide tracemsg -hide siconc -fixed *.f -free *.f90
```

GraphViz output

Fordocu can generate GraphViz input files (.dot) in the documentation directory containing a graphical representation of the the call trees generated for the HTML documentation. The generated .dot files are written to the documentation directory. To enable this, the **-g** flag must be used. To modify the behaviour, "flags" can be added immediately after the -g. Flag "c" groups all nodes by container, e.g. subroutines and functions in the same module will be grouped together. Flag "j" tries to join arrows that join the same end point. This will reduce the number of arrows.

Pseudocode output

Pseudo code can be marked in the source code with adding **@p** to comments, like in:

```
!! @p This comment line is considered pseudo code
```

It is possible to collect these comments per file and write them to a text file using the **-pseudo** command-line option.

Source output

A special feature of Fordocu is that it is able to generate sources from the comments used in the documentation. This only affects the declaration parts of modules, routines, types, interfaces and variables: "the headers". This is why this is called "header generation". This can be enabled with command-line option **-h**. It is possible to add "flags" immediately after this -h. With flag "**!**", only these "headers" are generated, but no HTML documentation. When adding the flag "**s**", individual program units (modules and routines) will get their own files. Flag "**$**" forces all comments on variable declarations to be end-of-line comments, otherwise, these comments will be places before the declaration, like the other comments. Flag "**k**" forces all variable declarations to be of type "real(kind=8)", "character(len=80)" or "character(len=*)", while flag "**a**" forces them to be formatted like "real*8", "character*80" or "character*(*)".

Flag **"U"** will force the reserved words in the declarations to upper case, they will be lower case otherwise. When flag **"v"** is given, an attempt will be made to align variable declarations vertically. Flag **"d"** restricts the use of the `dimension` statement to combinations with either the `pointer` or the `allocatable` statements. To suppress generating comments indicated with a double exclamation mark, the **"1"** flag can be used. When flag **"m"** is specified, `module, program, subroutine` and `function` statements do not cause indentation. All flags (except `"s"` and `"a"`) can be combined, like in: `-hvU$ad1m`

Output directories

It is possible to change the output directories: flag **-O <dir>** changes the documentation directory (default name: `html`); flag **-H <dir>** changes the header file directory (default: `header`); flag **-P <dir>** changes the directory for pseudo code files (the default being `pseudocode`).

Keywords

You may specify keywords in the source file using comments which start with an '@', for example

```
!   @author : Joost van den Vondel
```

Such a line must be specified in the first explanation of a subroutine, program or module. A special entry for this keyword is made in the 'header'-tab for the subroutine.

Keep old output files

At the start of a new run, the (old) output generated by a previous run is deleted to make sure no files that are not needed are kept. It is possible to keep the old files by using flag **-keep.** In this case some files will be overwritten, but files which are not generated anymore will still be available. This flag must be specified before the specification of the files which need to be documented. Note that only files in the given output directories are deleted, so if a previous run had another ouptut directory, these file will not be deleted. The file `comments.xml` is not deleted, because it can be edited by the user to improve the documentation (see section 5.3)

1 Example 1

To document a system which uses .for and .FOR files on Linux, which has case-sensitive file names, one must include both the .for and the .FOR file extensions. All files are fixed format; due to the `-1` flag, it accepts normal comments. This leads to:

```
fordocu.pl -fixed -1 *.for *.FOR
```

2 Example 2

It is possible to use a whole bunch of options. `-1` flag to accept normal comments. Include files can be obtained from the `main` directory. Calls to module `message` and everything in it do not show in the resulting call tree. Also fixed and free files can be used both.

```
fordocu.pl -1
           -I main
           -hide message
           -fixed
             main/*.f
             main2/*.f
           -free
             main/*.f90
             main2*.f90
```

5.3 comments.xml

Variable comments found by the documentation tool are stored in a file `comments.xml`, a "dictionary" with comments. This text file is sorted alphabetically. In case multiple comment texts are found for a specific item, it will appear multiple times. In case no comments are found in the source code, this file is used as an additional source for comments. In case of multiple comments, it uses the first one specified. The comments with `comments.xml` as source are indicated in the colour grey in the HTML documentation or with !!? in generated source code.

Since `comments.xml` is an editable text file and the first comments are used, it is possible to influence which comments are taken from this file by changing the order of the comments.

The first time Fordocu runs, there is no comments.xml file yet, so there are no comments that can be used yet - the file is built and comments are added, but they can only be used after they are encountered. To initialize `comments.xml`, it is a good idea to run Fordocu twice the first time: the first run to build a `comments.xml` file so comments from this file can be used during the second run.

5.4 stats.txt

During the generation of the documentation an ASCII file `stats.txt` is generated. This file tells how many times an interface, subroutine or function is called to by another interface of subroutine. The file can be found in `html/stats.txt`.

6 Troubleshooting

When a serious error occurs, the program aborts with an error message. When the program was designed, it was decided aborting was better than continuing with faulty data (if that would be possible). In many cases an error message points to an error in the input source code, rather than an error in Fordocu. Therefore, it is important to know how to read these error messages.

An example error message:

```
Redefinition of var abcdef in subroutine ghij at
/home/someone/tools/fordocu/stmts.pm line 1245, <IN> line 47.
```

This message means that a variable called abcdef is defined twice in subroutine ghij, the second time at line 47 in the source file.

In addition, extra information is included so the error can be traced back to the Fordocu sources. In this case, the error message was generated in file stmts.pm, at line 1245.