OpenFOAM directory organization

OpenFOAM directory organization

Prerequisites

You know how to *use* OpenFOAM, and how to find hints on usage in `$FOAM_TUTORIALS`

Learning outcomes

- Browse through the OpenFOAM installation in different ways
- Understand the organization of directories in OpenFOAM
- Understand the difference between applications and libraries, and their relation to programming
- Get a first understanding of the compilation process in OpenFOAM, and how it is related to the environment

OpenFOAM directory organization

We will first browse the directories graphically in Linux.
Below we do the same thing in a terminal window. Later we will use Doxygen.

You can use the Linux command `tree` to examine the source code directory organization, located in the installation directory `$WM_PROJECT_DIR`:

```
tree -d -L 1 $WM_PROJECT_DIR
```

yielding (version dependent):

```
$WM_PROJECT_DIR
|-- applications
|-- bin
|-- build
|-- doc
|-- etc
|-- modules
|-- platforms
|-- src
|-- tutorials
`-- wmake
```

In `$WM_PROJECT_DIR` you can also find release notes etc.
There is also an `Allwmake` script, which compiles all of OpenFOAM. We will have a look at this later.

OpenFOAM directory organization

Now we will investigate the directories in a sort of logical order:

- `etc`: The OpenFOAM environment and global settings, needed for everything.
- `tutorials`: Usage examples. Assumed to be already known.
- `applications`: Source code of solvers and utilities.
- `src`: Source code of libraries, used by the applications.
- `wmake`: Command and instructions for compilation of applications and libraries. Followed by a look at `Allwmake` and the `Make` directories.
- `build`: Intermediate compilation files, for applications and libraries.
- `platforms`: Final binaries for applications and libraries.
- `bin`: Executable bash scripts.
- `doc`: Doxygen source files and coding style instructions
- `modules`: Additional modules (not in slides)

This is followed by:

- Browsing with Doxygen.
- The `.git` directory.

The etc directory

The most important content of the `etc` directory is the files used to **set up the OpenFOAM environment**, such as:

```
bashrc
config.sh/settings
config.sh/aliases
```

The `bashrc` file is the one you source to set up the OpenFOAM environment. It sources the other files shown above.

The global `controlDict` gives OpenFOAM some global settings (not at case-level, as the `system/controlDict` files).
It is useful for debugging. We will have a look at this later.
It also sets some DimensionedConstants, and defines DimensionSets for the SI units (or USCS, if the unitSet is switched to that).
Read more about the global `controlDict` in the Programmers Guide.

The `thermoData` directory contains CHEMKIN data.

There are some templates for dictionaries and code.

The tutorials directory

The `tutorials` directory (`$FOAM_TUTORIALS`) contains example cases for each solver.

You should already know all about this, so we move on.

The applications directory

```
tree -d -L 1 $WM_PROJECT_DIR/applications #or $FOAM_APP
```

yields (version dependent):

```
$WM_PROJECT_DIR/applications
|-- solvers
|-- test
`-- utilities
```

Here is a short description of the `applications` directory contents:

- `solvers`: source code for the solvers
- `test`: source code that test and show example of the usage of some of the OpenFOAM libraries and classes (we will use some later)
- `utilities`: source code for the utilities

There is also an `Allwmake` script, which compiles all the contents of `solvers` and `utilities`. The contents of the `test` directory is not compiled by default. We will have a look at it later.

It should be noted in particular that the compilation of the source code in the `applications` directory yields executables!

The src directory

```
tree -d -L 1 $WM_PROJECT_DIR/src #or $FOAM_SRC
```

The `src` directory contains source code for all libraries, organized in sub-directories

The most relevant are:

- `finiteVolume`. This library provides all the classes needed for the finiteVolume discretization, such as the `fvMesh` class, `finiteVolume` discretization operators (divergence, laplacian, gradient, and fvc/fvm), and boundary conditions (`fields/fvPatchFields`). In `cfdTools/general/include/` you also find the very important file `fvCFD.H`, which is included in most applications.
- `OpenFOAM`. This *core* library includes the definitions of the containers used for the operations, the field definitions, the declaration of the mesh and of all the mesh features such as zones and sets
- `TurbulenceModels` which contains the source code of turbulence models

There is also an `Allwmake` script, which compiles all the libraries.

It should be noted in particular that the compilation of the source code in the `src` directory yields *shared object* (*.so) files, that are NOT executable!

The wmake directory

OpenFOAM uses a special make command: `wmake`.

If you are not familiar with make commands, read more here:

`https://en.wikipedia.org/wiki/Make_(software)`

`wmake` understands the file structure in OpenFOAM and has some default compiler directives that are set in the `wmake` directory. There is also a command, `wclean`, that cleans up (some of) the output from the `wmake` command.

If you added a new compiler name in the `bashrc` file, you should also tell `wmake` how to interpret that name. In `wmake/rules` you find the default settings for the available compilers.

You can also find some scripts that are useful when organizing your files for compilation, or for cleaning up.

Compilation using Allwmake and wmake, and the Make directories

The `Allwmake` scripts are simply calling other `Allwmake` scripts that eventually use `wmake` to compile individual applications and libraries.

The `wmake` command must be executed in directories where there is a `Make` directory. The `Make` directory contains specific compilation instructions for that application or library.

It is convenient to understand the library organization by searching for `Make` directories in `$FOAM_SRC`, since there is one library for each `Make` directory:

```
find $FOAM_SRC -name Make
```

The same can be done for solvers and utilities, although that organization is more obvious since there is one `Make` directory for each solver and utility:

```
find $FOAM_SOLVERS -name Make
find $FOAM_UTILITIES -name Make
```

The build directory

The compilation process for both applications and libraries is in two steps. First the source code is compiled without linking libraries. This yields intermediate object files (`*.o`). In the second step those object files are linked with the pre-compiled libraries. This yields the final compiled binaries.

The `build` directory contains intermediate files during compilation. These are files only used by the compiler. The only thing that happens if the `build` directory is removed is that the compiler has to repeat the compilation again if the compilation process is started over again. If they are left in place, the compiler will skip the steps that have already been done. We will therefore not investigate the `build` directory further.

After the second step, the final binaries are created. They are saved in the `platforms` directory...

The platforms directory (1/3)

The `platforms` directory contains the *binaries* of the applications and libraries after their compilation. They are organized according to the Linux environment used during compilation, more specifically the `$WM_OPTIONS` environment variable.

For example, if `$WM_OPTIONS` has the value `linux64GccDPInt32Opt` it means that the binaries were compiled for Linux, 64-bit, by the `gcc` compiler, for double precision floats, and 32 bit integers. That is set in `$WM_PROJECT_DIR/etc/config.sh/settings`:

```
export WM_OPTIONS=$WM_ARCH$WM_COMPILER$WM_PRECISION_OPTION$WM_LABEL_OPTION$WM_COMPILE_OPTION
```

Some of those are set in `$WM_PROJECT_DIR/etc/bashrc`:

```
export WM_COMPILER=Gcc
export WM_PRECISION_OPTION=DP
export WM_COMPILE_OPTION=Opt
```

while two of them are set in `$WM_PROJECT_DIR/etc/config.sh/settings`:

```
export WM_ARCH=$(uname -s)
export WM_LABEL_OPTION=Int$WM_LABEL_SIZE
```

where the last one is set (again) in `$WM_PROJECT_DIR/etc/bashrc`:

```
export WM_LABEL_SIZE=32
```

The platforms directory (2/3)

We see that `WM_OPTIONS` is set using environment variables that are set in `etc/bashrc`. That file also includes two options to manipulate those settings without changing the file itself:

```
# Load shell functions
# ~~~~~~~~~~~~~~~~~~~
. $WM_PROJECT_DIR/etc/config.sh/functions

# Override definitions via prefs, with 'other' first so the sys-admin
# can provide base values independent of WM_PROJECT_SITE
_foamEtc -mode=o  prefs.sh
_foamEtc -mode=ug prefs.sh

# Evaluate command-line parameters and record settings for later.
# These can be used to set/unset values, specify additional files etc.
export FOAM_SETTINGS="$@"
_foamEval $@
```

The first option is to add the setting of an enviroment variable when sourcing the `bashrc` file, such as for the `Debug` version:

```
. $HOME/OpenFOAM/OpenFOAM-plus/etc/bashrc WM_COMPILE_OPTION=Debug
```

This option uses the `_foamEval()` function `*=*` in `etc/config.sh/functions`.

The other option follows...

The platforms directory (3/3)

The other option is to copy the file `etc/config.sh/example/prefs.sh` to `etc`, and set the environment variables in that file.

Have a look in that file and see that the `prefs.sh` file must be found by `foamEtcFile`. You can list the appropriate paths by:

```
foamEtcFile -list
```

yielding (if your user name is `oscfd` and if you have installed OpenFOAM in your home directory):

```
/home/oscfd/.OpenFOAM/plus
/home/oscfd/.OpenFOAM
/home/oscfd/OpenFOAM/site/plus
/home/oscfd/OpenFOAM/site
/home/oscfd/OpenFOAM/OpenFOAM-plus/etc
```

Those are searched in order, and you can therefore do the modifications at different levels (for only you, at a site, or for all users).

The use of the `$WM_OPTIONS` level in the directory structure makes it possible to have the same source code compiled in as many ways as desired.

The bin directory

The `bin` directory contains *shell scripts* that you can execute in a terminal window.

You already know about `paraFoam` and `foamLog`

We will later use `foamNew`

Find written descriptions in the files, and use your bash knowledge to read the scripts.

The doc directory

The `doc` directory contains source code for the Doxygen documentation of OpenFOAM. It needs to be compiled to generate all the html pages, by (do not do this now!):

```
./Allwmake doc #If you are in $WM_PROJECT_DIR
```

or:

```
./Allwmake #If you are in $WM_PROJECT_DIR/doc
```

For now, have a look at:
`http://openfoam.com/documentation/cpp-guide/html/`

The `doc` directory previously also contained the User Guide and Programmer's Guide, which are now moved to `http://openfoam.com/`

There is as well a file named `codingStyleGuide.org`. It can be opened with `emacs`, which gives an additional Org drop-down menu. The visibility of the sections in the document can be toggled by pressing the TAB key while standing on each section header.
Read the contents to learn the coding style of OpenFOAM, in particular if you intend to contribute code.
Learn more about org at `https://orgmode.org/worg/org-tutorials/org4beginners.html`

Browse the source code using Doxygen

- Go to `http://openfoam.com/documentation/cpp-guide/html/`
- Click on `OpenFOAM API` and then in the left menu:
  `Files/File List/applications/solvers/incompressible/icoFoam/icoFoam.C`
- You see a nice representation of a description of the solver.
- Click on 'Original source code icoFoam.C' to show the contents of the file (does not work at the time of writing, but you can do the corresponding at `openfoam.org`, under `Resources/C++ Source Guide`). Note that you get information about some parts of the code while hovering the mouse pointer over them. Click to go to the corresponding files and figure out what the code does.

The .git directory

This slide is only valid if you have installed the git version.

The `.git` directory is hidden and is only shown with a flag:

```
ls -a
```

This is not OpenFOAM, but belongs to the `git` version control system, see:

```
https://git-scm.com/
```

This gives you the possibility to control the modifications of the source code, compare different versions of the code, and revert specific modifications if needed. Try:

```
foam
git status
```

We will have a look at git later.