Getting started with OpenFOAM

Eric Paterson

Senior Scientist, Applied Research Laboratory
Professor of Mechanical Engineering
The Pennsylvania State University
University Park, PA 16802 USA

PENNSTATE
1855

Acknowledgements

These slides are based upon training slides from previous workshops.

- Håkan Nilsson, Department of Applied Mechanics, Chalmers University of Technology, OpenFOAM Workshop Training 2009
- Gianluca Montenegro, Department of Energy, Politecnico di Milano, OpenFOAM Workshop Training 2008

Learning outcomes

You will learn ...

- to boot the USB-Stick with Kubuntu and test the OpenFOAM-1.5-dev installation
- to understand the difference between different OpenFOAM versions
- the principles of installing OpenFOAM
- how to compile all of OpenFOAM
- how to understand and modify the OpenFOAM environment
- the OpenFOAM directory organization
- OpenFOAM user directory organization
- OpenfOAM user development compilation procedure
- some useful Linux commands and concepts

USB-Stick for OFW5

A bootable USB-Stick has been prepared for the 5th OpenFOAM Workshop. You can get instructions on how to use it from:

- Paper copy in Workshop packet
- http://web.student.chalmers.se/groups/ofw5/Instr.htm
- http://web.student.chalmers.se/groups/ofw5/Instr.pdf
- Link to the instructions from the Desktop (this assumes you have already figured it out!)

What's on the USB-stick?

- Workshop material (program, abstracts, presentations and training material) which can be updated automatically via the Internet.
- Fully operational Linux operating system based on Kubuntu (Ubuntu 10.04 + KDE), and lots of useful CAE software including OpenFOAM (including pyFoam, cgnstools funkySetFields, groovyBC and simpleFunctionObjects), Open CASCADE, calculix, paraview with Takuya's OpenFOAM reader, enGrid, netgen, gmsh, blender, freecad, qcad, elmer, gerris, xfoil, bladedesigner, mittel, skv, octave, yacas and many more.
- The workshop material is located in the OFW5 folder which resides in the USB-stick's root-directory and can be accessed without booting the stick. If you boot from the USB-stick, you will find a link labeled 5th Workshop in the Desktop Folder which takes you there - "cd /cdrom/OFW5" for those who dislike GUIs.

How to boot it?

- On "standard" laptops and PCs, things should be straight forward.
    1. Simply plug the USB-stick in and boot your machine.
    2. You will be asked for the language you prefer before you get to the kubuntu splash screen.
    3. Select "Try Kubuntu without installing".
    4. Booting your machine from the USB-stick will not change your normal operating system in any way.
    5. Later, if you like, you have the option of installing Kubuntu and the included software on your computer permanently.
- If the machine boots from harddisk and not into Kubuntu, you will have to change the boot order in the BIOS.
    1. You can enter the BIOS by pressing a specific key during start-up (usually Del, F1, F2, F10, F12 - watch out for instructions on the screen during the boot procedure).
    2. In the BIOS, make sure that the USB device is booted in preference to anything else, or choose a one-time boot from the USB device.

How to boot in a VirtualBox?

This is for people who would like to run the USB-stick within their normal operating system (Linux, Mac OS X, Windows).

- Here, we assume that the VirtualBox-software is installed on your computer and that you are familiar with the command line (the "Terminal Window") or at least know where to find it.
- You will also need to have administrative rights on your machine, ie. you are allowed to do "sudo somecommand".
- The general procedure is as follows:
    1. Plug in the USB-stick
    2. Find out which device the USB-stick is
    3. Create a raw disk file using "VBoxManage internalcommands createrawvmdk". NOTE: this command is machine specific. Please see the detailed notes for Linux, Mac, or Windows.
    4. Create a new virtual machine in VirtualBox and attach the raw disk file that you just created
    5. Boot the virtual machine. It should boot from the USB-stick

Run the icoFoam/cavity tutorial to test the installation

Type the following commands to test the installation by running the icoFoam/cavity tutorial:

```
mkdir $WM_PROJECT_USER_DIR/run
cp -r $FOAM_TUTORIALS/icoFoam/cavity $FOAM_RUN
run
cd cavity
blockMesh
icoFoam
```

You should get lots of text and numbers in your terminal window – we will examine this more later.

Post-process by typing:

```
paraFoam
```

You can also type

```
foamInstallationTest
```

and search the output for possible problems

Which version is on the USB-Stick?

- We are using 1.5-dev
- This is the current version released by the OpenFOAM-extend project
- The OpenFOAM-extend project is an *open-source development* version
- 1.5-dev contains important libraries, solvers, and utilities which greatly extend the core OpenFOAM library

PENN STATE

Where to get OpenFOAM, and a note on different versions

1. **OpenFOAM-1.6** is distributed by OpenCFD, at `www.openfoam.org`
2. **OpenFOAM-1.6.x** is a patched version of OpenFOAM-1.6 which is obtained from a git repository.
3. **OpenFOAM-1.5-dev** is an extended version, distributed via svn from sourceforge.net, and can be found at `http://openfoam-extend.wiki.sourceforge.net/`
4. **User contributions** at `http://openfoam-extend.wiki.sourceforge.net/`, with corresponding descriptions at `http://openfoamwiki.net`

PENN STATE

Some more details on OpenFOAM-1.6

- Download and installation instructions available at www.openfoam.org
- Source code
- Single and double precision binaries for 32 and 64 bits, meaning that ideally you don't have to compile the code, but in practice it is always best to compile it on your own machine.
- Third-Party source code and binaries
- **Never updated with bug fixes!**

PENN STATE

Some more details on OpenFOAM-1.6.x

- Same as `OpenFOAM-1.6`, but with bug fixes!
- Check out source code using the Git version control system (instructions at `www.openfoam.org`)
- No binaries distributed - you have to compile everything!
- Use the same Third-Party products as `OpenFOAM-1.6`
- Update your installation using Git, and re-compile to have the latest bug-fixed version.

Some more details on OpenFOAM-1.5-dev

- OpenFOAM-1.5-dev is maintained by a small, but growing, group of OpenFOAM developers
- Hosted at SourceForge
  (http://openfoam-extend.wiki.sourceforge.net/)
- Exact location:
  http://openfoam-extend.svn.sourceforge.net/viewvc/openfoam-extend/trunk/Core/OpenFOAM-1.5-dev/
- Has all the features in `OpenFOAM-1.5.x`, but also many extensions.
- Check out source code using the Svn version control system (instructions at the link above)
  Basically:
  svn checkout <theLinkAboveButWith'svnroot'InsteadOf'viewvc'>
- No binaries distributed - you have to compile everything!
- Use the same Third-Party products as `OpenFOAM-1.5`
- Update your installation using Svn, and re-compile to have the latest version. Update by going to the `OpenFOAM-1.5-dev` directory, and type:
  svn update
- This is the version used in this training.

Some more details on user contributions

- The OpenFOAM-extend project at SourceForge:
  `http://openfoam-extend.wiki.sourceforge.net/` was developed by
  OpenFOAM users to allow user contributions.
- See:
  `http://openfoam-extend.svn.sourceforge.net/viewvc/openfoam-extend/trunk/`
- The main contribution is `OpenFOAM-1.5-dev`, click on `Core`
- Many plug-in libraries and applications can be found, click on `Breeder-1.5`
- The OpenFOAM Wiki, `http://openfoamwiki.net/`, is the place for
  descriptions of the code in the OpenFOAM-extend project.
- OpenFOAM Working Groups share files and information at OpenFOAM-extend
  and in the OpenFOAM Wiki, see:
  `http://openfoam-extend.svn.sourceforge.net/viewvc/openfoam-extend/trunk/Breeder_1.5/OSIG/`
  `http://openfoamwiki.net/index.php/Main_Special_Interest_Groups`

PENN STATE

OpenFOAM installation

OpenFOAM can be installed for a single user (local installation), or for many users (network installation):

1. **Local installation**: This is the default, and most common way of installing OpenFOAM. The installation will be located in `$HOME/OpenFOAM/OpenFOAM-1.5-dev`. Benefits: Each user will 'own' their own installation and may update it any time. Drawbacks: Requires extra disk space if there are several users with their own installations, and all users must know how to install OpenFOAM and the Third-Party products.

2. **Network installation**: This installation is suitable when a group of people is supposed to use OpenFOAM, and when not everyone want to learn how to install OpenFOAM. All users will use exactly the same installation. Benefits: A single installation for each version of OpenFOAM, maintained by your IT-staff. Drawbacks: You have to be nice to your IT-staff so that they quickly install new versions and keep all versions updated.

Once the installation is there, it is just a matter of setting an environment variable in a file to point at the installation directory. The user will notice no difference (just a tiny one, which we will discuss later).

Local installation (OpenFOAM-1.5, 32 bit, double precision)

The OpenFOAM-1.5 installation procedure is here described using a number of Linux commands (**don't do it now!**). This is based on the descriptions at www.openfoam.org.

```
mkdir $HOME/OpenFOAM
cd $HOME/OpenFOAM
wget http://dfn.dl.sourceforge.net/sourceforge/foam/OpenFOAM-1.5.General.gtgz
wget http://dfn.dl.sourceforge.net/sourceforge/foam/OpenFOAM-1.5.linuxGccDPOpt.gtgz
wget http://dfn.dl.sourceforge.net/sourceforge/foam/ThirdParty.General.gtgz
wget http://dfn.dl.sourceforge.net/sourceforge/foam/ThirdParty.linuxGcc.gtgz
tar xzf *; rm *.gtgz
. $HOME/OpenFOAM/OpenFOAM-1.5/etc/bashrc
```

Done!

Well, you might have problems with some ThirdParty products such as Paraview, which requires exactly the correct version of Qt, or gcc, which requires mpfr. Those are not actually OpenFOAM, so we will not discuss those issues here. Consult www.openfoam.org, the OpenFOAM Forum, or your IT-staff.

The links shown above can be found by looking at the preferences of the files you download at www.openfoam.org.

The final line sets up the OpenFOAM environment, which will be discussed later.

Network installation (don't do it now!)

Make sure that all the users are able to reach the directory where OpenFOAM will be installed (for instance /OpenFOAM). Change the first two lines in the previous slide to:

```
mkdir /OpenFOAM
cd /OpenFOAM
```

wget and un-tar all the files as in the previous slide.

Make the /OpenFOAM/OpenFOAM-1.5/etc/bashrc file point at the network installation by changing the line saying foamInstall=$HOME/$WM_PROJECT to foamInstall=/$WM_PROJECT

Instead of the last line in the previous slide, the user must source the installed bashrc file:

```
. /OpenFOAM/OpenFOAM-1.5/etc/bashrc
```

We will have a deeper look at the effect of the sourcing of the bashrc file later, but among other things, it will define some environment variables (such as WM_PROJECT, WM_PROJECT_INST_DIR, and WM_PROJECT_DIR) that will be used in the coming slides.

PENN STATE

The tiny difference between local and network installations

There is a global control file, `$WM_PROJECT_DIR/etc/controlDict`, that each user might want to modify to some personal settings.

When `OpenFOAM 1.5-dev` starts, it will be looking at the following locations for a valid `controlDict` file, in that specific order. As soon as a match is found, this is the `controlDict` file that will be used.

1. `~/.OpenFOAM/1.5-dev/controlDict` (user file: version dependent)
2. `~/.OpenFOAM/controlDict` (user file: version independent)
3. `$WM_PROJECT_INST_DIR/site/1.5-dev/controlDict` (site file: version dependent, where `site` is simply the string 'site')
4. `$WM_PROJECT_INST_DIR/site/controlDict` (site file: version independent, where `site` is simply the string 'site')
5. `$WM_PROJECT_DIR/etc/controlDict` (default installation file: version dependent)

Advanced information: Have a look at: `Foam::dotFoam()` in the file `$WM_PROJECT_DIR/src/OSspecific/Unix/Unix.C`

In other words, do the following to get a personal global controlDict file:

```
mkdir -p ~/.OpenFOAM/1.5-dev
cp $WM_PROJECT_DIR/etc/controlDict ~/.OpenFOAM/1.5-dev
```

Compile all of OpenFOAM yourself

It is usually good to compile all of OpenFOAM yourself. The three main reasons are:

1. `OpenFOAM-1.6` is not updated with bug fixes, so it is not really useful
2. `OpenFOAM-1.6.x` is only distributed as source code
3. `OpenFOAM-1.5-dev` is only distributed as source code

Once the OpenFOAM source code is there, and all ThirdParty products are up and running, you simply do (**don't do it now!**):

```
. $HOME/OpenFOAM/OpenFOAM-1.5-dev/etc/bashrc
cd $HOME/OpenFOAM/OpenFOAM-1.5-dev
./Allwmake
```

It is also possible to re-compile parts of OpenFOAM. Simply find an occurance of `Allwmake`, and run it the same way as above.

This is similar for all versions, `OpenFOAM-1.5`, `OpenFOAM-1.5.x`, and `OpenFOAM-1.5-dev`.

How to get OpenFOAM-1.5-dev

To conclude the section of different versions, here are the commands to download and compile `OpenFOAM-1.5-dev`, if you already have a working ThirdParty installation (**don't do it now!**):

```
cd $HOME/OpenFOAM
svn co https://openfoam-extend.svn.sourceforge.net/svnroot/\
openfoam-extend/trunk/Core/OpenFOAM-1.5-dev
cd OpenFOAM-1.5-dev
. $HOME/OpenFOAM/OpenFOAM-1.5-dev/etc/bashrc
./Allwmake
```

(note that the '\' at the end of the second line means that the text on the third line should be put right after the '/', without spaces)

Update the installation by:

```
cd $HOME/OpenFOAM/OpenFOAM-1.5-dev
svn update
./Allwmake
```

Read more at: http://openfoam-extend.wiki.sourceforge.net/

Sourcing bashrc

(In the following we assume that `OpenFOAM-1.5-dev` is installed in `$HOME/OpenFOAM`, and that you are using `bash`. **Don't do the following**, it has already been done for you!)

The OpenFOAM environment is set in
`$HOME/OpenFOAM/OpenFOAM-1.5-dev/etc/bashrc`

The usual way to source this file is to add a line in `$HOME/.bashrc`, saying:

`. $HOME/OpenFOAM/OpenFOAM-1.5-dev/etc/bashrc`

When you open a new Konsol, the `$HOME/.bashrc` file will be sourced, which in turn will source the OpenFOAM `bashrc` and set up the OpenFOAM environment.

Environment variables

Sourcing the OpenFOAM `bashrc` file loads all the needed environmental variables

For example, the ones we have used so far:

- `$WM_PROJECT` is just the string 'OpenFOAM'
- `$WM_PROJECT_INST_DIR` is the `OpenFOAM` directory with all the installed versions
- `$WM_PROJECT_DIR` is the directory of the currently used version
- `$WM_PROJECT_USER_DIR` is the user directory, where developments or cases can be located (you don't have to use this directory).

Learn to use the environment variables to be less dependent on which version you are using!

You can find all enviroment variables by typing

```
env
```

or, for instance

```
env | grep WM
env | grep FOAM
```

to see only those which contain the string 'WM' or 'FOAM'

Common user modifications in bashrc

- The most common is to modify the installation directory, as we discussed earlier:

  `foamInstall=/$WM_PROJECT`

  which will then set environment variable `WM_PROJECT_INST_DIR`.

- If the ThirdParty directory is not in the default location, modify:

  `export WM_THIRD_PARTY_DIR=$WM_PROJECT_INST_DIR/ThirdParty`

- To add a new compiler option, modify `WM_COMPILER` - this variable is used when setting up the compiler paths, and also for compilation specific directory names.

- Choose 32 or 64 bits (both are possible on 64 bit architectures) by setting `WM_ARCH_OPTION`. Appears in directory names.

- Choose single or double precision by setting `WM_PRECISION_OPTION`. Appears in directory names.

- Choose optimal (`Opt`), debug (`Debug`), or profiling (`Prof`) compilation by setting `WM_COMPILE_OPTION`. Appears in directory names.

- Choose message passing interface by setting `WM_MPLIB` (default OPENMPI).

- Enable halt on floating-point exception by setting `FOAM_SIGFPE`.

These are used to set up compiler options etc., and some other files are sourced...

Other files that are sourced

There are mainly three other files that are sourced from `bashrc`:

```
$WM_PROJECT_DIR/etc/settings.sh
$WM_PROJECT_DIR/etc/aliases.sh
$WM_PROJECT_DIR/etc/apps/paraview3/bashrc
```

- In `settings.sh`, you can specify how your choice of `WM_COMPILER` and `WM_MPLIB` should be interpreted, or if the system compiler should be used.
- In `aliases.sh`, some useful aliases are set (see next slide)
- In `apps/paraview3/bashrc`, the environment for `paraview` is set.

Useful aliases

An alias is an abbreviation of a one-line command. These are defined in the `OpenFOAM-1.5-dev` environment:

```
alias wm64='export WM_ARCH_OPTION=64; . $WM_PROJECT_DIR/etc/bashrc'
alias wm32='export WM_ARCH_OPTION=32; . $WM_PROJECT_DIR/etc/bashrc'
alias wmSP='export WM_PRECISION_OPTION=SP; . $WM_PROJECT_DIR/etc/bashrc'
alias wmDP='export WM_PRECISION_OPTION=DP; . $WM_PROJECT_DIR/etc/bashrc'
alias wmSchedON='export WM_SCHEDULER=$WM_PROJECT_DIR/wmake/wmakeScheduler'
alias wmSchedOFF='unset WM_SCHEDULER'
alias src='cd $FOAM_SRC'
alias lib='cd $FOAM_LIB'
alias run='cd $FOAM_RUN'
alias foam='cd $WM_PROJECT_DIR'
alias foamsrc='cd $FOAM_SRC/$WM_PROJECT'
alias foamfv='cd $FOAM_SRC/finiteVolume'
alias app='cd $FOAM_APP'
alias util='cd $FOAM_UTILITIES'
alias sol='cd $FOAM_SOLVERS'
alias tut='cd $FOAM_TUTORIALS'
```

For instance, if you type

```
src
```

you will actually do

```
cd $FOAM_SRC
```

OpenFOAM directory organization

We will use the Linux command `tree` to examine the directory structure:

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

yielding:

```
$WM_PROJECT_DIR
|-- applications
|-- bin
|-- doc
|-- etc
|-- lib
|-- src
|-- tutorials
`-- wmake
```

In `WM_PROJECT_DIR` you can also find `ReleaseNotes` etc., but most importantly:

```
Allwmake
```

which compiles all of OpenFOAM, as discussed earlier.

The applications directory

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

yields:

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

Here is a short description of the applications directory contents:

- `bin` contains the binaries generated when compiling the applications
- `solvers` contains source code for the distributed solvers
- `test` contains source code that test and show example of the usage of some of the OpenFOAM classes
- `utilities` contains source code for the distributed utilities

There is also an `Allwmake` script, which will compile all the contents of `solvers` and `utilities`

PENN STATE

The src directory

This directory contains the source code for all the libraries

It is divided in different subdirectories each of them can contain several libraries

The most relevant are:

- `finiteVolume`. This library provides all the classes needed for the finiteVolume discretization, such as fvMesh, divergence, laplacian, gradient discretization operators, matrix solvers, and boundary conditions.
- `OpenFOAM`. This 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 several turbulence models
- `engine` declaration of classes for engine simulation
- `dynamicMesh` for moving meshes algorithms

The bin, doc, etc, lib, and tutorials directories

The `bin` directory contains *shell scripts*, such as `paraFoam`, `foamNew`, `foamLog` ...

The `doc` directory contains the documentation of OpenFOAM:

- Programmers and User Guide
- Doxygen generated documentation in html format

Usage:

```
acroread $WM_PROJECT_DIR/doc/Guides-a4/UserGuide.pdf
acroread $WM_PROJECT_DIR/doc/Guides-a4/ProgrammersGuide.pdf
mozilla file://$WM_PROJECT_DIR/doc/Doxygen/html/index.html
```

The `etc` directory contains environment set-up files, global OpenFOAM instructions, and default `thermoData`.

The `lib` directory contains the *binaries* of the dynamic libaries.

The `tutorials` directory contains example cases for each solver.

The wmake directory

OpenFOAM uses a special make command: `wmake`.

`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.

User directory organization

Some of the OpenFOAM environment is set up for a specific user directory organization, in $WM_PROJECT_USER_DIR.

In a clean installation of OpenFOAM you find there two directories.
`tree -L 1 -d $WM_PROJECT_USER_DIR` yields

```
$WM_PROJECT_USER_DIR
|-- applications
`-- lib
```

In `applications`, it is recommended to put user developed applications in the same structure as in $WM_PROJECT_DIR/applications

In $WM_PROJECT_DIR/applications/bin, the binaries of the user developed applications will be located

In `lib`, the binaries of the user developed libraries will be located

It is recommended to create two more directories:

```
$WM_PROJECT_USER_DIR/run
$WM_PROJECT_USER_DIR/src
```

Place user developed library source code in `src` directory, with the same directory structure as in $FOAM_SRC, and case files in the `run` directory.

PENN STATE

User development compilation procedure

We have already discussed how to compile the installation using the `Allrun` script. Now we will discuss how to compile our own developments.

In the previous slides we learned where to find the source code for applications and libraries.

Now we will learn the basic procedure how to compile any of those if they have been updated, or to copy one of them and compile it to implement a new application or library.

You can locate the main directory of applications or libraries by looking for `Make` directories

There is a specific `Make` directory for each application

The libraries are however grouped together as larger libraries

We will now have a look at the principles of compilation of applications and libraries

But first a recommendation

Do not modify anything in the installation, except for updates!

You can do everything you need to do with your own copies, and then you don't risk to mess things up.

Another recommendation is to keep the same directory structure in your copies, as the original code, so that you only have to keep track of one directory structure.

Compilation of user developed applications

- Find the directory of the application you want to modify and compile
- Copy it to your working directory
- Re-name the directory name and file names (not necessary, but nice)
- Modify `Make/files` to your new names, and change `FOAM_APPBIN` to `FOAM_USER_APPBIN`
- Type `wclean` and `wmake`

Example of how to copy and compile the `icoFoam` solver as `myIcoFoam`:

```
cd $FOAM_APP
cp -r --parents solvers/incompressible/icoFoam \
                $WM_PROJECT_USER_DIR/applications
cd $WM_PROJECT_USER_DIR/applications/solvers/incompressible/
mv icoFoam myIcoFoam
cd myIcoFoam
mv icoFoam.C myIcoFoam.C
sed -i s/icoFoam/myIcoFoam/g Make/files
sed -i s/FOAM_APPBIN/FOAM_USER_APPBIN/g Make/files
wclean
wmake
```

Compilation of user developed libaries (1/2)

Usually you would like to do a small modification to an existing library

Find the directory of the library you want to modify and copy it to your working directory

Find the `Make` directory that is used when compiling the original library and copy it to your working directory

Remove all lines, in `Make/files`, that don't correspond to the piece of the library you are compiling

Add a line in `Make/options` saying `-I` and the path to the `lnInclude` file that was located next to the original `Make` directory. This step is necessary since `wmake` implicitly searches for include files in the directory where the compilation is started, and now we have moved the compilation procedure elsewhere, and must thus explicitly point at the `lnInclude` directory.

Re-name folders, files and entries in `Make/files`, just as when compiling applications.

Re-name the class name in the files so that it can be distinguished from the original one.

Type `wmake libso`

Compilation of user developed libaries (2/2)

Here is an example of how to copy and compile the `kEpsilon` turbulence model:

```
foam
cp -r --parents src/turbulenceModels/RAS/incompressible/\
{kEpsilon,Make} $WM_PROJECT_USER_DIR
cd $WM_PROJECT_USER_DIR/src/turbulenceModels/RAS/incompressible
wclean
rm -rf Make/linux*
mv kEpsilon mykEpsilon
cd mykEpsilon
mv kEpsilon.C mykEpsilon.C
mv kEpsilon.H mykEpsilon.H
sed -i s/kEpsilon/mykEpsilon/g mykEpsilon.*
cd ..
sed -i s/els/\
"els \\\ \n-I\$(LIB_SRC)\/turbulenceModels\/RAS\/incompressible\/lnInclude"\
/g Make/options
echo "mykEpsilon/mykEpsilon.C" > Make/files
echo "LIB = \$(FOAM_USER_LIBBIN)/libmyIncompressibleRASModels" >> Make/files
wmake libso
```

We will discuss later how to *use* new libraries.

The lnInclude directory

In order to make it easier for the compiler to find the include-files, they are linked to from `lnInclude`. This linking is done when running `wmake libso`.

The compiler searches for the included header files in the following order

1. Explicit paths set in `Make/options`
2. A local lnInclude directory, i.e. in the directory where `wmake` is run
3. The local directory, i.e. the directory where `wmake` is run
4. The `$WM_PROJECT DIR/src/OpenFOAM/lnInclude` directory;
5. The `$FOAM_SRC/OSspecific/Unix/lnInclude` directory

The Make directory

The `Make` directory contains instructions on how to compile the code. The original instructions are arranged in two files:

- files
- options

We will discuss these in the coming slides.

After compilation there will also be one or several directories, containing compilation information derived in the compilation procedure as well as the object files:

- linuxGccDPOpt
- linux64GccDPOpt
- linuxGccDPDebug
- linuxGccDPProf

There will be one such directory for each kind of compilation that has been made.

The Make/files file

The `Make/files` file consists of a list of relative paths and names of the files to be compiled

The location and name of the final binary is specified by `EXE = <path>/<name>` for applications, and `LIB = <path>/<name>` for libraries.

OpenFOAM offers two recommended choices for the path of application and library binaries, respectively:

1. `$FOAM_APPBIN` for standard release *applications*
2. `$FOAM_USER_APPBIN` for user developed *applications*

1. `$FOAM_LIBBIN` for standard release *libraries*
2. `$FOAM_USER_LIBBIN` for user developed *libraries*

The Make/options file

The `Make/options` file contains the full directory paths to include files and libraries

```
EXE_INC = \
    -I$(LIB_SRC)/finiteVolume/lnInclude
```

The directory names are preceeded by the `-I` flag and the syntax uses the `\` to continue the `EXE_INC` across several lines, with no `\` after the final entry

```
EXE_LIBS = \
    -lfiniteVolume \
    -llduSolvers
```

The `libOpenFOAM.so` library is implicitly used, and the libraries are implicitly searched for in `$FOAM_LIBBIN`. Other paths can be added using the `-L` flag in the `EXE_LIBS` section in the `Make/options` file.

Cleaning up after compilation: wclean and rmdepall

wmake creates some files and folders, as we have seen.

One file that has not been mentioned yet is a `*.dep` file, which contains a list of files that the compilation depends on.

The `*.dep` file and the `Make/$WM_OPTIONS` directory can be removed by typing:

wclean

This is a way to make sure that re-complation will take place next time wmake is run.

If the local lnInclude directory should also be deleted, type:

wclean lib

rmdepall removes all dependency .dep files recursively down the directory tree from the point at which it is executed. It is useful when updating OpenFOAM libraries