HANDS-ON: BUILDING OWN SOLVERS & LIBRARIES

Joris Degroote
OUTLINE

– Basics about compiling, executables and libraries
– Make your own solver, based on existing solver
– Make your own library and use it in existing solver
COMPILING, EXECUTABLES AND LIBRARIES
COMPILING, EXECUTABLES AND LIBRARIES

Compiling = source files (.C) → object files (.o)
          human readable          machine readable

Libraries = collections of object files which cannot be executed directly
          → Can be used by multiple executables
COMPILING, EXECUTABLES AND LIBRARIES

Compiling = source files (.C) → object files (.o)
             human readable                    machine readable

       ↓

Linking = combining object files (.o), static libraries (.a) or shared libraries (.so) to create executable (binary, no extension on Linux)
COMPILING, EXECUTABLES AND LIBRARIES

Static linking = include (part of) static library (.a) in executable
→ Large executable

Dynamic linking = create link to shared library (.so) in executable, so functions can be found
→ Small executable
COMPILING, EXECUTABLES AND LIBRARIES

- **Source Code** (.c, .cpp, .h) → **Preprocessing** → **Compilation** → **Assemble** → **Linking** → **Executable Machine Code** (.exe)

  - **Step 1**: Preprocessor (cpp)
  - **Step 2**: Compiler (gcc, g++)
  - **Step 3**: Assembler (as)
  - **Step 4**: Linker (ld)
Example
COMPILING, EXECUTABLES AND LIBRARIES

./src/Piece.h  King.h  Queen.h  Tower.h  ...  Chess.C
            King.C  Queen.C  Tower.C

↓ Compile

King.o  Queen.o  Tower.o  ...  Chess.o

↓ Link

./bin/chess
COMPILING, EXECUTABLES AND LIBRARIES

./src/Piece.h  King.h  ...  Chess.C  ...  Screen.C  ...
  King.C
↓ Compile

  King.o  ...  Chess.o  ...  Screen.o  ...
↓ Link

./bin/chess
./lib/libgraphics.so
OpenFOAM uses dynamic linking with shared libraries

\texttt{wmake} = compile all required source code and link as executable (binary)
→ Typically depends on several libraries

\texttt{wmake libso} = compile all required source code and package as shared library
→ Can depend on other libraries
MAKE YOUR OWN SOLVER
MAKE YOUR OWN SOLVER

1. Interactive job
2. Copy existing solver
3. Change name
4. Change settings
5. Compile
6. Test run
MAKE YOUR OWN SOLVER

qsub -I -l walltime=00:59:59

module load OpenFOAM/4.1-intel-2017a

module list

source $FOAM_BASH
MAKE YOUR OWN SOLVER

```bash
echo $FOAM_APPBIN
→ Location of official binaries

echo $FOAM_USER_APPBIN
→ Location of own binaries
```
MAKE YOUR OWN SOLVER

OpenFOAM structure

```
cd $WM_PROJECT_DIR

src/ → source code of libraries
applications/solvers → source code of solvers
applications/utilities → source code of utilities
platforms/ → binaries and libraries
```
MAKE YOUR OWN SOLVER

Create same structure in own directory

```bash
mkdir -p $WM_PROJECT_USER_DIR
cd $WM_PROJECT_USER_DIR
```

(typically $VSC_HOME/OpenFOAM/username-version)

- run/ → simulation cases and results
- src/ → source code of own libraries
- applications/solvers → source code of own solvers
- applications/utilities → source code of own utilities
- platforms/ → binaries and libraries
MAKE YOUR OWN SOLVER

cd $WM_PROJECT_DIR/applications/solvers

cd incompressible/

cp -r icoFoam $WM_PROJECT_USER_DIR/applications/solvers/myFoam

cd $WM_PROJECT_USER_DIR/applications/solvers/myFoam
MAKE YOUR OWN SOLVER

mv icoFoam.C myFoam.C

Edit “myFoam.C”

    Info<< "Bye bye from myFoam\nEnd\n" << endl;
MAKE YOUR OWN SOLVER

cd Make
ls

files
options
MAKE YOUR OWN SOLVER

Edit “files”

myFoam.C

EXE = $(FOAM_USER_APPBIN)/myFoam
MAKE YOUR OWN SOLVER

Edit “options”

EXE_INC = \n   -I$(LIB_SRC)/finiteVolume/lnInclude \n   -I$(LIB_SRC)/meshTools/lnInclude
→ Headers to be included when compiling executable

EXE_LIBS = \n   -lfiniteVolume \n   -lmeshTools
→ Libraries to be included when linking executable
MAKE YOUR OWN SOLVER

cd ..

wmake

ls $FOAM_USER_APPBIN
MAKE YOUR OWN SOLVER

cd $FOAM_RUN

cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity/cavity myCavity

cd myCavity

blockMesh

myFoam
MAKE YOUR OWN LIBRARY
MAKE YOUR OWN LIBRARY

1. Interactive job
2. Copy part of existing library
3. Change name
4. Change settings
5. Compile
6. Test run
MAK E YOUR OWN LIBRARY

```
echo $FOAM_LIBBIN
→ Locations of official libraries

echo $FOAM_USER_LIBBIN
→ Locations of own libraries
```
MAKE YOUR OWN LIBRARY

cd $WM_PROJECT_DIR/src

cd functionObjects/utilities

cp -r writeDictionary $WM_PROJECT_USER_DIR/src/myWriteDictionary

cd $WM_PROJECT_USER_DIR/src/myWriteDictionary
MAKE YOUR OWN LIBRARY

mv writeDictionary.C myWriteDictionary.C
mv writeDictionary.H myWriteDictionary.H

Edit both files and replace “writeDictionary” by “myWriteDictionary”

Edit “myWriteDictionary.C”

Foam::functionObjects::myWriteDictionary::~myWriteDictionary()
{
    Info<< "Bye bye from myWriteDictionary" << endl;
}
MAKE YOUR OWN LIBRARY

cp -r $WM_PROJECT_DIR/src/functionObjects/utilities/Make .
cd Make
ls

files
options
MAKE YOUR OWN LIBRARY

Edit “files”

myWriteDictionary.C

LIB = $(FOAM_USER_LIBBIN)/libmyWriteDictionary
MAKE YOUR OWN LIBRARY

Edit “options”

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

LIB_LIBS = \n   -lfiniteVolume
MAKE YOUR OWN LIBRARY

cd ..

wmake libso

ls $FOAM_USER_LIBBIN
MAKE YOUR OWN LIBRARY

cd $FOAM_RUN/myCavity

Edit “system/controlDict”

functions
{
    writeDictionary1
    {
        type myWriteDictionary;

        libs ("libmyWriteDictionary.so");

        dictNames (controlDict);
    }
}
MAKE YOUR OWN LIBRARY

myFoam

Check output

...  
Bye bye from myFoam
End

Bye bye from myWriteDictionary
TIPS

Use variables for paths, do not hard code them

Use binaries and libraries only on cluster that has been used for compiling

Use ldd to check dependency on shared libraries

Study C++ (Stroustrup, …)
REFERENCES

https://www.youtube.com/playlist?list=PLqxhrJj6bcnY9R0IgzeF6xDh5L9bbeK3BL
Joris Degroote
Associate professor

DEPARTMENT OF FLOW, HEAT AND COMBUSTION MECHANICS

E   joris.degroote@ugent.be
T   +32 9 264 95 22

www.ugent.be