Debugging OpenFOAM implementations with GDB

(Acknowledgements to Dr. Fabian Peng-Kärrholm)

- It is impossible to do bug-free programming (trust me!), so you should always verify your implementations.
- When you run into problems, such as code crash, or mysterious behaviour, you also need some debugging approach.
- There are many debugging approaches, and we will discuss three alternatives here:
  - `Info` statements in the code
  - Build-in `DebugSwitch` option in OpenFOAM (similar to the above - you will see)
  - Debugging using the Gnu debugger, GDB (`http://www.gnu.org/software/gdb/`)
- We will now go through these alternatives...

Debugging using Info statements

- The simplest way to do debugging is to write out intermediate results to the screen, and check that those results are correct.
- In OpenFOAM this is done using `Info` statements.
- This kind of debugging does not allow any control of the running of the code while debugging. It will just print out additional information.
- `Info` debugging requires that new lines are inserted in the source code, and that the source code must be re-compiled whenever a new `Info` statement has been added.
- When the code has been debugged, all the `Info` statements must be deleted, or commented, so that you don't get all that information when you are no longer debugging.
- OpenFOAM provides an alternative to removing all the `Info` statements, so that these `Info` statements can be activated again later.
- This brings us to the next level of debugging in OpenFOAM...

Debugging using OpenFOAM DebugSwitches

- In `$WM_PROJECT_DIR/etc/controlDict` (global `controlDict` dictionary), you can find a list of `DebugSwitches`:

```
DebugSwitches
{
    APIdiffCoefFunc     0;
    Ar                  0;
    BICCG               0;
    ...
}
```

- Each class thus has its own `DebugSwitch`.
- `DebugSwitches` set to zero will produce no debug information.
- Different levels of debugging can be chosen by setting a `DebugSwitch` to `1`, `2`, `3` ...
- You are not able to modify the `DebugSwitches` in our pre-installed OpenFOAM, since you do not have write-access to that file. In OpenFOAM-1.4.1 it was possible to have a personal version of this file, but I haven't found a way to do this with OpenFOAM-1.5. (Help me!)
- On your own computer, with your own installation of OpenFOAM, you will however have write-access to this file.

What is a DebugSwitch?

- Let's have a look at the `lduMatrix` `DebugSwitch`, which is set to `1`.
- The `lduMatrix` class is implemented in `$FOAM_SRC/OpenFOAM/matrices/lduMatrix/lduMatrix`
- Looking inside `lduMatrix.C`, we see a line:
  ```
  defineTypeNameAndDebug(lduMatrix, 1);
  ```
  This line defines the `DebugSwitch` name `lduMatrix`, and sets its default value to `1`.
- In `lduMatrixTests.C` you can find a member function `print()`, where all its contents are within an if-statement:
  ```
  if (debug)
  {
  ...
  }
  ```
- Boolean `debug` corresponds to the `lduMatrix` `DebugSwitch`, and it is `true` if the `DebugSwitch` is *greater than* `0`. The default value, both in the class definition, and in the global `controlDict` is `1`, so the contents of the if-statement will be executed.

Result of DebugSwitch lduMatrix 1; (or 2;)

- We see that the class will print out the member data `solverName_`, `fieldName_`, `initialResidual_`, `finalResidual_`, and `noIterations_` (unless the solution is singular). In other words, this is most of the information we usually get in the log-file:

```
DILUPBiCG:  Solving for Ux, Initial residual = 1, Final residual = 2.96338e-06, No Iterations 8
DILUPBiCG:  Solving for Uy, Initial residual = 0, Final residual = 0, No Iterations 0
DICPCG:  Solving for p, Initial residual = 1, Final residual = 7.55402e-07, No Iterations 35
```

- In the same file we can see that a `lduMatrix DebugSwitch` value >=2 will give some extra information. In this case:

```
   Normalisation factor = 0.004
DILUPBiCG:  Iteration 0 residual = 1
DILUPBiCG:  Iteration 1 residual = 0.153298
DILUPBiCG:  Iteration 2 residual = 0.0375508
DILUPBiCG:  Iteration 3 residual = 0.00820823
DILUPBiCG:  Iteration 4 residual = 0.00174405
DILUPBiCG:  Iteration 5 residual = 0.000357418
DILUPBiCG:  Iteration 6 residual = 7.4212e-05
DILUPBiCG:  Iteration 7 residual = 1.51806e-05
DILUPBiCG:  Iteration 8 residual = 2.96338e-06
```

- In summary, the `DebugSwithes` only control different levels of `Info`-statements. No re-compilation is needed when switching the level, but if new `Info`-statements are included, re-compilation is needed. You can use this feature in your own development.
- This still offers no control of the running of the code while debugging...

Debugging OpenFOAM implementations with GDB

- Now it is time for some real debugging with the Gnu debugger...
- GDB can be used for
  - Programs written in C, C++, Ada, Pascal etc
  - Running and stopping at specific positions in the code
  - Examining variables at run-time
  - Changing your program at run-time
- Bad news: The complete code needs to be re-compiled with a debug flag. This will produce ∼1Gb extra of OpenFOAM binaries.
- Good news: I have done it for you here at Chalmers.
- More bad news: The code will run much slower in debug mode, in particular when running under GDB. The reason is that nothing can be optimized, and that there is much more information to keep track of.
- Best news: GDB can help you implement a bug-free code, which can be run in an optimized version.

Compiling OpenFOAM in Debug mode

- In `$WM_PROJECT_DIR/etc/bashrc` you find an environment variable `WM_COMPILE_OPTION` that can be set to `Debug`. That is what you need to do if you want to compile using the debug flag, or use the `Debug` version.
- What you need to do right now is to open a new terminal window with the following lines in your `~/.bashrc` file (it seems to be necessary for GDB that these are set there. It is not sufficient to set them in the terminal window, at least not in `tcsh`)

```
export WM_COMPILE_OPTION=Debug
. /chalmers/sw/unsup/OpenFOAM/OpenFOAM-1.5.x/etc/bashrc
```

  Make sure that you use the `Debug` mode by typing:
  `which icoFoam`
  which should point at a path containing `linuxGccDPDebug`.
  Now you can compile or run all or parts of OpenFOAM in `Debug` mode.
- Note that you may consider not compiling `ThirdPartyProducts` in `Debug` mode, and simply make sure that the `Opt` version of those are used also for the `Debug` mode.
- Now it is time to learn the basics of GDB...

Debugging the icoFoam solver

- Let's practice GDB on the `icoFoam` solver.
- The objective is to find out what a part of the code does.
- In `icoFoam.C` there is a line saying:
  `adjustPhi(phi, U, p);`
  What does this line do?
- Make sure that you have an `icoFoam/cavity` case, and that you have run `blockMesh` on it.
- Start `icoFoam` under GDB in the `cavity` case directory:
  `gdb icoFoam`
- Set a break point at the line in `icoFoam.C` where `adjustPhi(phi, U, p);` is used:
  `b icoFoam.C:77`
- Start the execution by typing:
  `run` (this will take some time...)
- The exectution will stop at the breakpoint saying:

```
Breakpoint 1, main (argc=1448, argv=0x3d1) at icoFoam.C:77
77              adjustPhi(phi, U, p);
```

Line-by-line checking of adjustPhi

- There are two ways of stepping in the file:
  `n` (next) will step to the next line in the current file, but will not go into functions and included files.
  `s` (step) will go to the next line, also in functions and included files.
  Both can be followed by a number specifying how many times to repeat the command.
- Step to the next line by typing 's' (allowing GDB to go inside the `adjustPhi` function).
  We are now at line `42` in `cfdTools/general/adjustPhi/adjustPhi.C`
- Type '`where`' to see which file you are debugging, and which file called it.
- Type '`list cfdTools/general/adjustPhi/adjustPhi.C:30,50`' to see source lines `30-50` of `adjustPhi.C`. Line `42` is the first line of the `adjustPhi` function.
- Type '`n 3`' to avoid going into the evaluation of the boolean, instead we will see that we are at line `47` in `adjustPhi.C`, so the if-statement is evaluated.
- Type '`p massIn`' and then '`p fixedMassOut`'. Note that line `47` has not yet been evaluated, so `fixedMassOut` can have any value!!
- Type '`run`' to restart the degugging from the beginning if needed. Type '`quit`' to quit.
- Stepping and listing the code will show every single step that will be taken. You will understand that `adjustPhi` ensures global continuity.

Learn more on GDB

- See `gdbfoam.pdf`, by Fabian Peng-Kärrholm, on the course homepage.
- See `http://www.gnu.org/software/gdb`
- There are some interfaces to GDB:
  - `ddd`
  - `emacs`