“Defensive programming with Elmer/Ice” or “Contributing code to Elmer(/Ice) with a clear conscience”

Talk outline:

- What is defensive programming?
- Some specific examples of defensive programming and code legibility in Elmer/Ice.
- A bit more about version control and git, depending on levels of interest and on how time is going…

What is defensive programming?

“Defensive programming is a form of **defensive** design intended to ensure the continuing function of a piece of software under unforeseen circumstances. The idea can be viewed as reducing or eliminating the prospect of Finagle's law having effect.”
(Finagle’s law is a kind of corollary to Murphy’s law: “*Anything that can go wrong, will—at the worst possible moment.*”)
https://en.wikipedia.org/wiki/Defensive_programming

“Defensive programming defends against the currently impossible.”
http://c2.com/cgi/wiki?DefensiveProgramming

“Impossible things become possible when new people join the team.”
“Humans make anything possible when it comes to errors.”
http://c2.com/cgi/wiki?DefensiveProgramming

"*The major difference between a thing that might go wrong and a thing that cannot possibly go wrong is that when a thing that cannot possibly go wrong goes wrong it usually turns out to be impossible to get at and repair.*"

Douglas Adams

"It may hide bugs instead of making them visible, if misapplied."
http://c2.com/cgi/wiki?DefensiveProgramming

"Defensive Programming is NOT about swallowing errors or hiding bugs. It's about deciding on the trade-off between robustness (keep running if there is a problem you can deal with) and correctness (never return inaccurate results)."

"*The whole point of defensive programming is guarding against errors you don't expect*."
Steve McConnell, Code Complete http://cc2e.com/

What is Fail-fast programming?

"*Fail-fast systems are usually designed to stop normal operation rather than attempt to continue a possibly flawed process*."

https://en.wikipedia.org/wiki/Fail-fast

"*Hiding errors lets bugs breed. Blowing up the application in your face forces you to fix the real problem*."

http://johannesbrodwall.com/2013/09/25/offensive-programming/

A few more links…

Some info about defensive compiler flags:

https://source.ggy.bris.ac.uk/wiki/Debugging#Defensive_Programming

http://faculty.washington.edu/rjl/uwamath583s11/sphinx/notes/html/gfortran_flags.html

For some opinionated discussions on defensive programming and related subjects:

http://johannesbrodwall.com/2013/09/25/offensive-programming/

http://danielroop.com/blog/2009/10/15/why-defensive-programming-is-rubbish/

Now for some Elmer/Ice examples of defensive programming…

Check your retrieved Elmer variables and act accordingly

This example is from:
MyElmerClone/elmerice/Solvers/IDSSolver.F90

Failing to successfully retrieve a variable that the solver needs is probably going to be fatal:

```
WaterPressure => VariableGet( Model % Mesh % Variables, TRIM(Solver % Variable % Name) // ' Pressure' )
IF (ASSOCIATED(WaterPressure)) THEN
   Wpress => WaterPressure % Values
   WpPerm  => WaterPressure % Perm
ELSE
   WRITE(Message,'(A)') TRIM(Solver % Variable % Name) // ' WaterPressure not associated'
   CALL FATAL( SolverName, Message)
END IF
```

“Fail-fast systems are usually designed to stop normal operation rather than attempt to continue a possibly flawed process.”

Check your retrieved sif parameters and act accordingly

This example is from:
MyElmerClone/elmerice/UserFunctions/USF_Sliding.F90

You might decide failure to retrieve a parameter is fatal:

```
C = GetConstReal( BC, 'Budd Friction Coefficient', GotIt )
IF (.NOT. GotIt) THEN
   CALL FATAL(USF_name, 'Need a Friction Coefficient for the Budd sliding law')
END IF
```

Or maybe you can resort to a default value:

```
Zab_offset = GetConstReal( BC, 'Budd Zab Offset', GotIt )
IF (.NOT. GotIt) THEN
   Zab_offset = 0.0_dp
END IF
```

“It’s about deciding on the trade-off between robustness (keep running if there is a problem you can deal with) and correctness (never return inaccurate results).”

Careful of zero array bound when using permutations

This example is from:
MyElmerClone/elmerice/UserFunctions/USF_Zs.F90

Here CYCLE is used to avoid array bounds errors:

```
DO i = 1, Model % NumberOfNodes
   IF (ZsPerm(i)==0) CYCLE
   IF (dim==2) THEN
      Zs0(ZsPerm(i)) = Model % Nodes % y (i)
   ELSE
      Zs0(ZsPerm(i)) = Model % Nodes % z (i)
   END IF
END DO
```

If bounds checking is not switched on at compile time this kind of bug can result in memory errors which may manifest in different ways. If you are lucky, a seg fault.

The advantage of using a CASE DEFAULT clause (or an ELSE clause in an IF statement)

This example is from:

MyElmerClone/elmerice/UserFunctions/USF_Contact.F90

Here Elmer needs to know which sliding law to use in the case of grounded nodes when simulating a marine ice sheet.

```
! grounded node
SELECT CASE(Sl_law)
CASE ('weertman')
   Bdrag = Sliding_weertman(Model, nodenumber, y)
CASE ('budd')
   Bdrag = Sliding_Budd(Model, nodenumber, y)
CASE ('coulomb')
   Bdrag = Friction_Coulomb(Model, nodenumber, y)
CASE DEFAULT
   WRITE(Message, '(A,A)') 'Sliding law not recognised ',Sl_law
   CALL FATAL( USF_Name, Message)
END SELECT
```

What happens if the sliding law is spelled wrong, or the user adds a new sliding law only to the USF_Sliding.F90 code?

Code duplication/redundancy

- Will all users always know to apply changes to duplicate code as well as original code?

Finally the most important thing: who can we blame for these heinous crimes of coding?

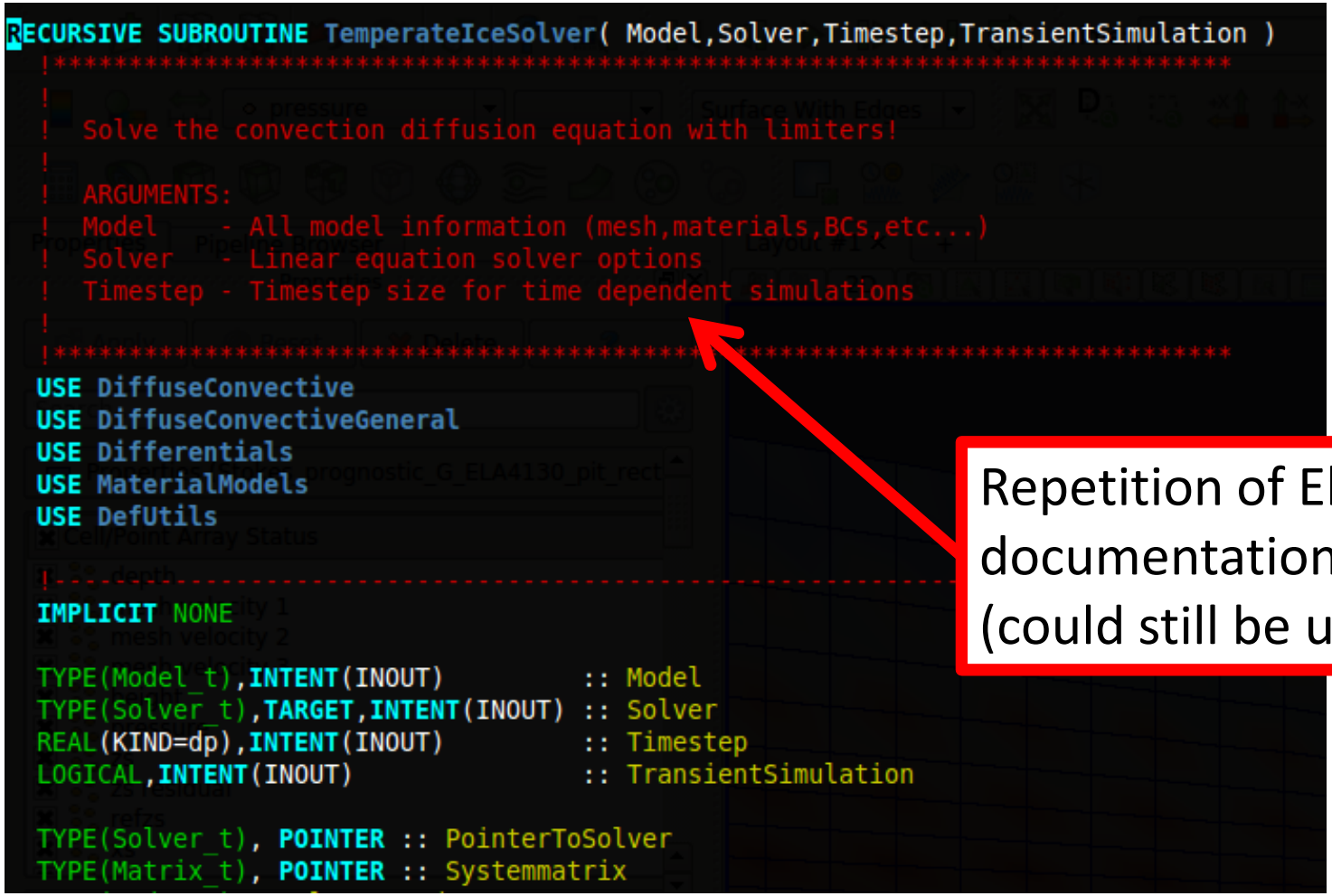
Crime	Culprit
Trying to use variables that are not associated	
Code duplication	
Failing to check sif parameter retrieval	
Array out of bounds	
Omitting important DEFAULT clause	
Failing to run tests before pushing to Elmer GitHub repository	

Code legibility: comments

- Try to write self-describing code.
- This is not always possible/practical, hence the need for comments.
- Keep comments concise.


RECURSIVE SUBROUTINE TemperateIceSolver(Model,Solver,Timestep,TransientSimulation)
Solve the convection diffusion equation with limiters!
ARGUMENTS:
TYPE(Model_t) :: Model,
INPUT: All model information (mesh,materials,BCs,etc...)
TYPE(Solver_t) :: Solver
INPUT: Linear equation solver options
REAL(KIND=dp) :: Timestep
INPUT: Timestep size for time dependent simulations
USE DiffuseConvective
USE DiffuseConvectiveGeneral
USE Differentials
USE MaterialModels
USE Adaptive
USE DefUtils
IMPLICIT NONE
External variables
TYPE(Model_t) :: Model
TYPE(Solver_t), TARGET :: Solver
LOGICAL :: TransientSimulation
REAL(KIND=dp) :: Timestep
Local variables
TYPE(Solver_t), POINTER :: PointerToSolver
Repetition
Not needed if using INTENT

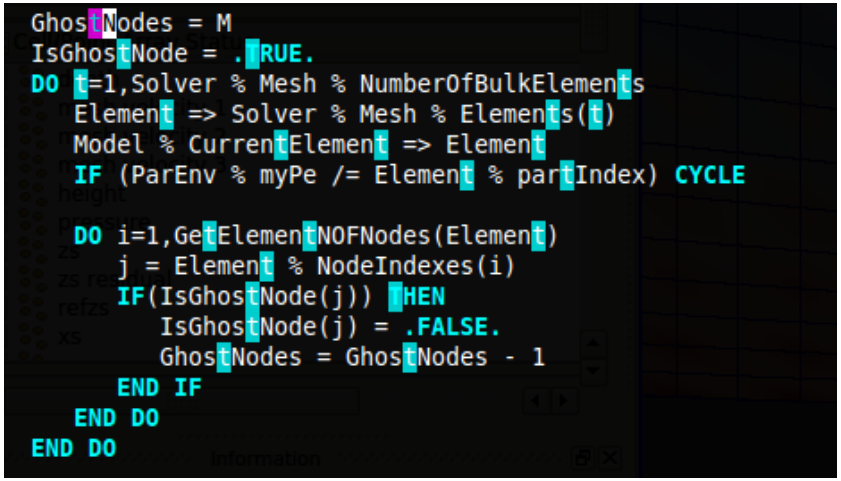
The same code with some fewer comments. Easier to read maybe?



Repetition of Elmer documentation (could still be useful)

Code navigability: single letter variable names (e.g. i -> ii, x -> xx).

Code navigability: single letter variable names (e.g. i -> ii, x -> xx).



```
GhostNodes = M
IsGhostNode = .TRUE.
DO t=1,Solver % Mesh % NumberOfBulkElements
   Element => Solver % Mesh % Elements(t)
   Model % CurrentElement => Element
   IF (ParEnv % myPe /= Element % partIndex) CYCLE

   DO i=1,GetElementNOFNodes(Element)
      j = Element % NodeIndexes(i)
      IF(IsGhostNode(j)) THEN
         IsGhostNode(j) = .FALSE.
         GhostNodes = GhostNodes - 1
      END IF
   END DO
END DO
```

Code navigability: single letter variable names (e.g. i -> ii, x -> xx).

```
GhostNodes = M
IsGhostNode = .TRUE.
DO t=1,Solver % Mesh % NumberOfBulkElements
   Element => Solver % Mesh % Elements(t)
   Model % CurrentElement => Element
   IF (ParEnv % myPe /= Element % partIndex) CYCLE

   DO i=1,GetElementNOFNodes(Element)
      j = Element % NodeIndexes(i)
      IF(IsGhostNode(j)) THEN
         IsGhostNode(j) = .FALSE.
         GhostNodes = GhostNodes - 1
      END IF
   END DO
END DO
```

```
GhostNodes = M
IsGhostNode = .TRUE.
DO tt=1,Solver % Mesh % NumberOfBulkElements
   Element => Solver % Mesh % Elements(tt)
   Model % CurrentElement => Element
   IF (ParEnv % myPe /= Element % partIndex) CYCLE

   DO ii=1,GetElementNOFNodes(Element)
      jj = Element % NodeIndexes(ii)
      IF(IsGhostNode(jj)) THEN
         IsGhostNode(jj) = .FALSE.
         GhostNodes = GhostNodes - 1
      END IF
   END DO
END DO
```

A couple more points about readable code…

- Code legibility: variable names (too long vs not informative.
- Code legibility: indents (emacs tab)

What is version control?

"*Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later*."

https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

"***version control,*** *also known as* ***revision control*** *or* ***source control,*** *is the management of changes to documents, computer programs … and other collections of information*."

https://en.wikipedia.org/wiki/Version_control

Why use version control?

- **Traceability**. Retrieve any previous version of your code.
- **Traceability**. View logged developer comments and or version differences.
- **Collaboration**. Colleagues can work on the same files at the same time (probably through branching and merging).
- **Collaboration**. Your repository is probably accessible through the internet (if submitting to GitHub by default) meaning you can share development with anyone with an internet connection.
- **Backup**. As a side effect, you have your code both locally and in a remote repository (and probably on your colleagues herd drives too, depending on how branching is managed in your project), ensuring protection against failure of one location.

Why not use version control?

- You have to learn how to use it.

If that puts you off, and since there is too much whitespace on this slide, read the top answer to this question:

http://stackoverflow.com/questions/1408450/why-should-i-use-version-control

Git

SVN

- Distributed version control
- Powerful and flexible
- Available offline
- Non-intuitive commands
- Lots to learn

- Centralised version control
- Easy to learn

Git

Github

- Distributed version control system

- Web host for git repositories
- It is apparently possible to use SVN to access github repositories, but I have not tried this…

“Why would you? It is like opting for a hangover without getting drunk in the first place” – Thomas Zwinger

Branch structure

Directory structure

The Elmer/Ice branch contains all of the Elmer code, including the glaciological functions and solvers.

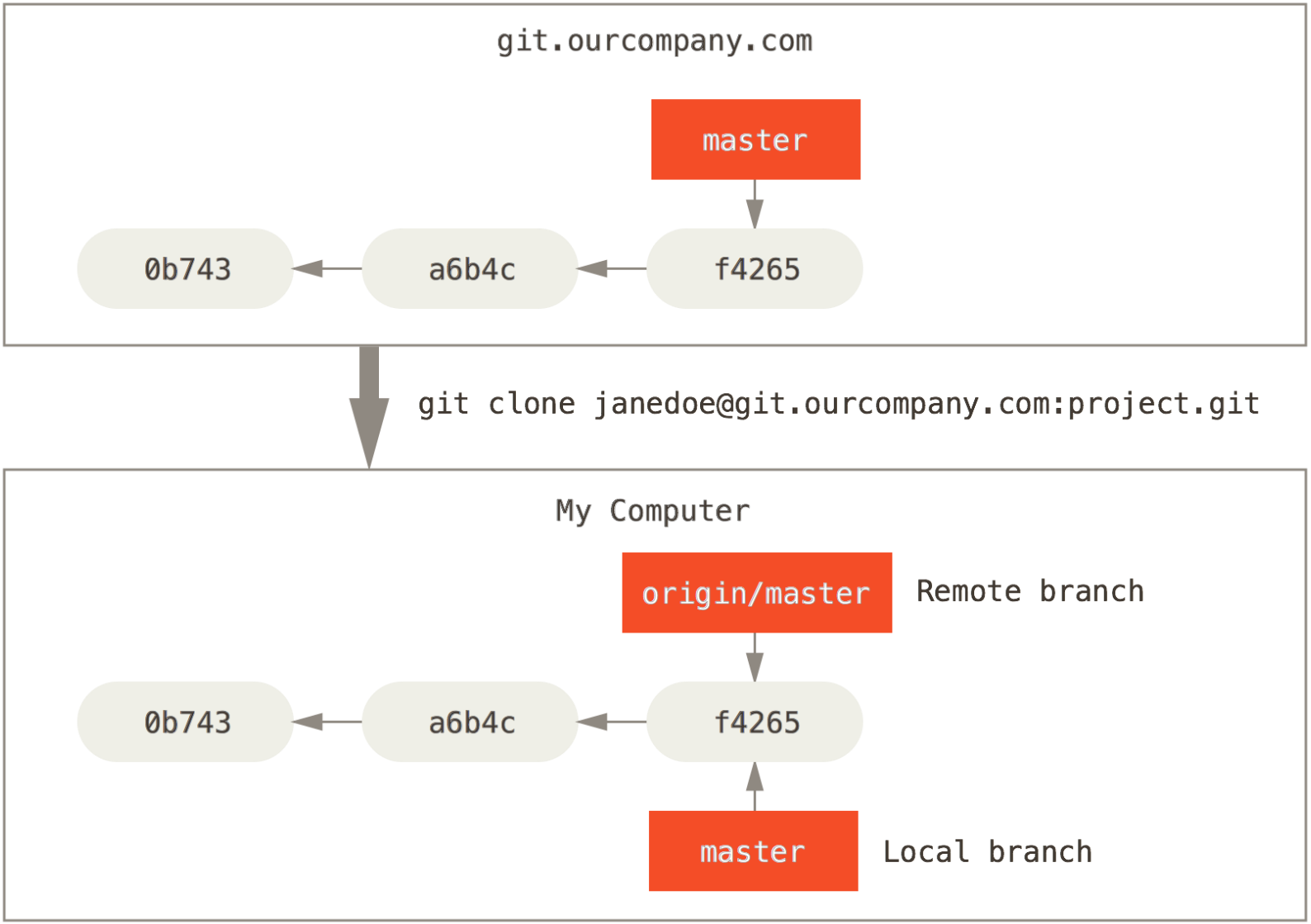
It is called Elmer/Ice because it is intended to be used and developed by the Elmer/Ice community.

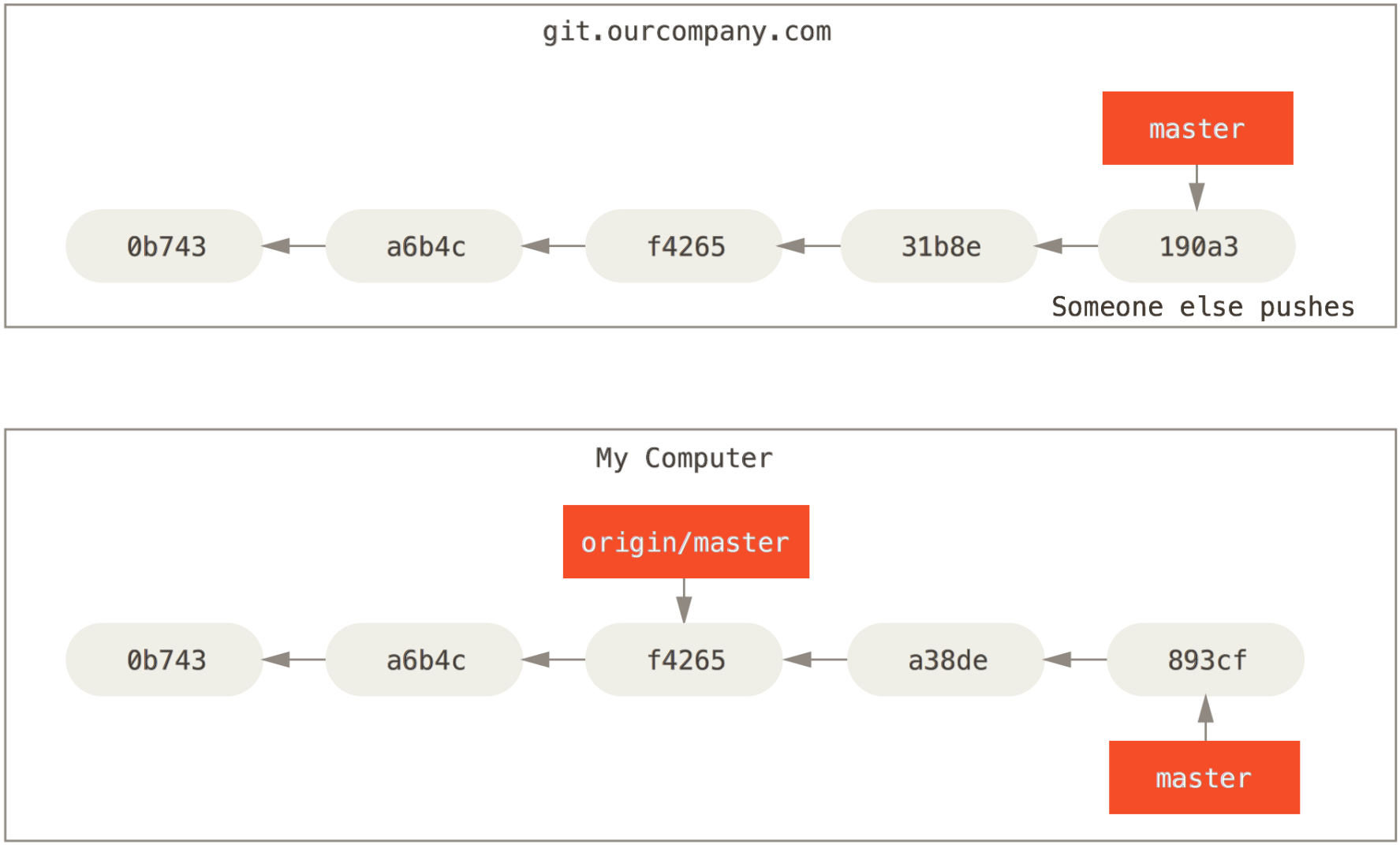
The elmerice subdirectory is present in all the branches (in fact all the code is available in each branch).

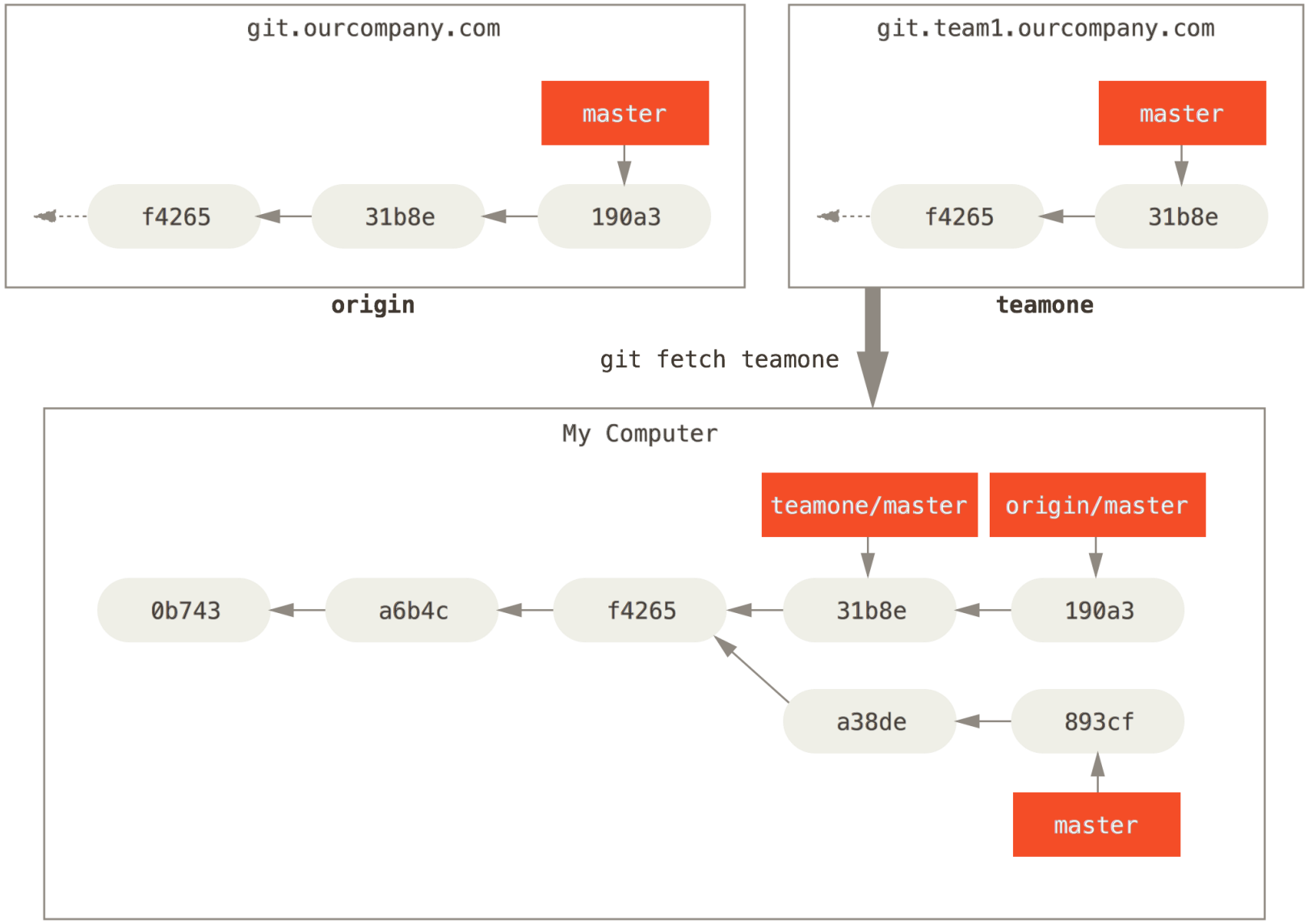
It is called elmerice because it contains the glaciological user functions and solvers.

For contributors: if you use Elmer primarily for glaciology, and contribute code (either by pushing directly or via pull requests), you should use the Elmer/Ice branch, even if your changes are not in the elmerice subdirectory.

A bit more about local and remote branches




git.ourcompany.com
master
0b743
a6b4c
f4265
31b8e
190a3
Someone else pushes
My Computer
origin/master
0b743
a6b4c
f4265
a38de
893cf
master

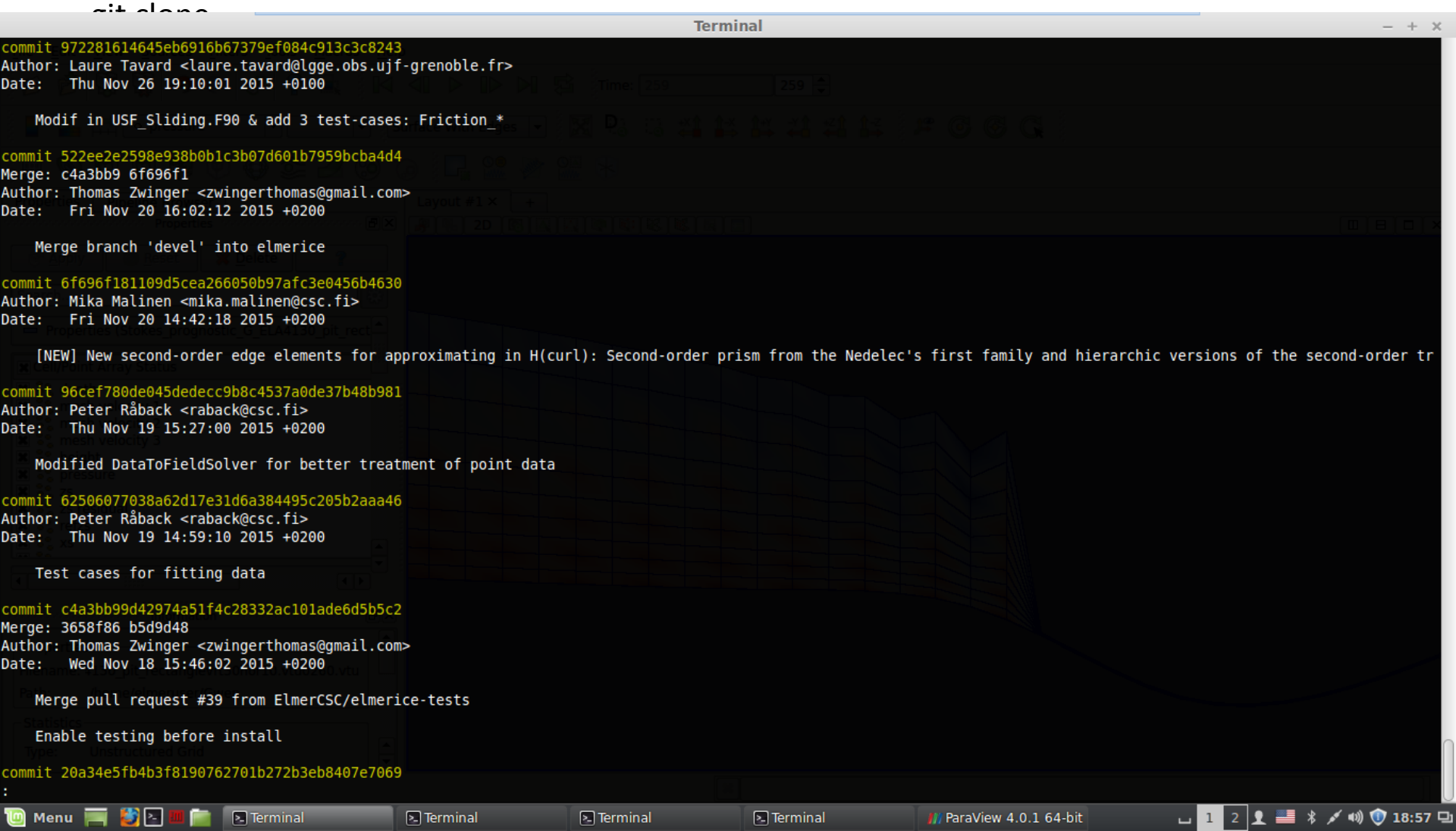

git.ourcompany.com
master
f4265
31b8e
190a3
origin
git.team1.ourcompany.com
master
f4265
31b8e
teamone
git fetch teamone
My Computer
teamone/master
origin/master
0b743
a6b4c
f4265
31b8e
190a3
a38de
893cf
master

Some git commands I use quite often (well, towards the bottom are some less common commands I guess)

git clone
Git fetch
git pull
git add
git add -u
git commit
git push
git checkout -- <filename>
git checkout <branchname>
git diff
git branch
git merge
git log
git log --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit

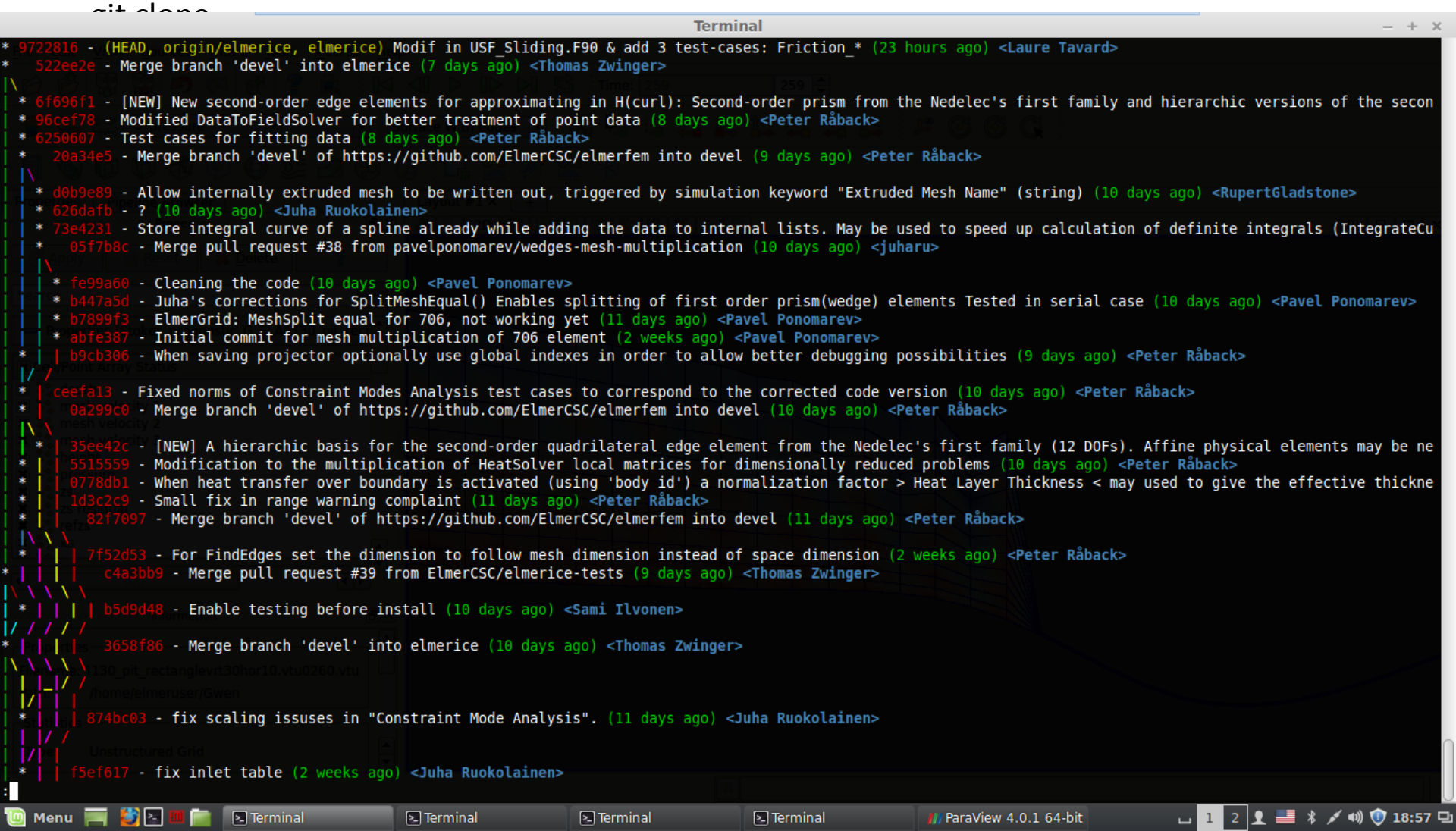
Some git commands I use quite often (well, towards the bottom are some less common commands I guess)

git clone



Some git commands I use quite often (well, towards the bottom are some less common commands I guess)

git clone



Git resources

Git online book (really good for starting learning concepts and syntax):
https://git-scm.com/book/en/v1/

Elmer online repository at GitHub:
https://github.com/ElmerCSC/elmerfem

A good place to find answers for your git questions:
http://stackoverflow.com/questions/tagged/git

Git cheat sheet:
https://training.github.com/kit/downloads/github-git-cheat-sheet.pdf

I find key sharing makes for smooth GitHub access from Linux:
https://help.github.com/categories/ssh/

Compilation with Cmake

Edit

Table of Contents

Download

Elmer/Ice can be retrieved through the whole Elmer package via GitHub. From the shell of a system that has git installed, the command

```
git clone git://www.github.com/ElmerCSC/elmerfem -b elmerice elmerice
```

creates a local copy of the repository, directly linking the local elmerice branch to the one of the repository on GitHub (this seems to be necessary in less recent git-versions). Please, bear in mind that there are different branches in this repository. The Elmer development branch, `devel`, usually does not contain the latest Elmer/Ice developments. In order to be sure to have an Elmer/Ice development that doesn't interfere with the main Elmer branch development, we created a branch named `elmerice`. In order to check, which branch one is in, one can give the command

```
git status
```

Or, for a more concise view of your current branch (starred) and other local branches you can run