Meet and Greet With Enzo

We're going to take some time to get Enzo compiled and running on your laptop or on Triton. Once we've done that, we're going to take a stroll through the code base to orient ourselves, and we might even run a test problem or two.

We're going to use yt for some of our analysis of Enzo simulations; if you are running on Triton, you do not need to install yt (or HDF5) but if you want to run on your laptop, I recommend using the yt installation script. There are separate installation scripts for Linux, for OSX 10.5 and OSX 10.6.

Intermission: On Triton

On Triton, we have constructed an environment. Type this to get that environment set up for you:

```
source /projects/lca-group/workshop_env.sh
```

If you're having trouble with anything, check that you have done this.

Getting Enzo

Enzo is an open source code with a community of developers and users. We've recently moved to the Mercurial distributed version control system. You can find information about Mercurial all over the web, but here are a couple places to start looking:

- http://mercurial.selenic.com/
- http://www.hginit.com/
- http://hgbook.red-bean.com/read/

Also, please feel free to email the Enzo Users' List with any questions you might have. We could talk a lot about Mercurial here, but I'll save it and just give you the one takeaway message:

When you check out a Mercurial repository, you not only get the entire development history, but you also become a fully-fledged developer in your own right. You can commit changes to your local repository, you can share those changes with others, and most importantly, you can trivially submit them upstream to become part of the larger Enzo community.

To get the code, you should head over to the brand new Enzo project page on Google Code, which is located at http://code.google.com/p/enzo/ . It's still pretty bare bones, as we're still moving over from the current website, but you can browse the source and download copies. The older website is where the documentation currently lives, and it's located at http://lca.ucsd.edu/projects/enzo .

To get the code, you have a couple options. I'd recommend that you follow the standard checkout instructions::

```
hg clone https://enzo.googlecode.com/hg/ ./enzo-2.0-beta2
```

This will copy the entire Mercurial repository to your local machine. But, it requires that you have Mercurial,

and it also creates a slightly larger source tree. Alternately, on Triton I have provided a copy of the latest source tree::

```
/projects/lca-group/local-dev/enzo/52weeks-of-code-5de37fa5be37.tar.bz2
```

which you can copy and untar and use.

Intermission: Enzo Requirements

Enzo only has one dependency: HDF5. HDF5 is one of the easiest libraries to install and use, and it's also one of the most value-added libraries. It provides a self-describing data format, so that you don't have to have any guess work about the binary format, the way data records are dumped into a file, or anything like that. To install it, you can either use the yt install script, or you can download and run something like::

```
tar xvfz hdf5-1.8.4.tar.gz
cd hdf5-1.8.4
./configure --prefix=$HOME/local-tree
make && make install
```

Alternately, if you use the yt installation script, it will place HDF5 in a local tree as well.

Compiling Enzo

Enter into your Enzo top-level directory and type:

```
./configure
```

Now you have initialized the Enzo repository. We now go into the Enzo main source tree:

```
cd src/enzo
```

To get Enzo to compile, you have to tell it which makefile to use. To do that, we type `make machine-something`. On Triton, you can do `make machine-triton-intel` to get the latest makefile. On OS X, Enzo assumes you have `gfortran` installed (check http://hpc.sf.net/) and that HDF5 is in `/usr/local/`, and you have to type `make machine-darwin` to get the Makefile.

If you are running somewhere else, you will have to make a new Makefile. It's not too bad, and ask me or the Enzo mailing list if you have any trouble. Just copy one of the existing ones and then modify it to have the location of your compilers and so on. You can check `Make.mach.darwin` (corresponding to `make machine-darwin`) and `Make.mach.linux-gnu` (corresponding to `make machine-linux-gnu`) for some examples.

Once you've selected your makefile, do `make help-config` and `make show-config` to see what you can do and what is done currently. Enzo has as few compile-time options as possible, but these are most or all of them.

Once you're satisfied ... type `make` and see if it goes. If it didn't, there's probably an issue with paths or modules. Try googling any error messages first, and then see me or someone else that has had some success for help.

When it has finished compiling -- and with 785 C/C++ files plus 140 F/F90 files it will take a while -- you should have a happy little `enzo.exe` in the current directory. However, we still need to generate an `inits.exe` to use to generate cosmological initial conditions. So now, switch into the `src/inits` directory:

```
cd ../inits
```

When you run `make` now, it will create `inits.exe`, which you can copy around the same as you would `enzo.exe`.

Running a Problem

If you go up two levels in the directory structure, you will be in the root directory of the repository. We previously went into the `src/enzo` directory. Now, go into `run/`:

```
cd run
```

Do an `ls` to see what is in there -- it's all separated into directories, each of which contains a single problem and parameter file. This is where all the example files, along with sample plotting scripts and notes about all of them. There are 100 parameter files and 128 plotting scripts, and we are constantly documenting and adding more with more plotting scripts.

You can run one of these by going into a directory, copying `enzo.exe` to that directory, and then running it like so:

```
./enzo.exe -d some_parameter_file.enzo
```

There's a lot here to play with, and most have detailed notes, so please explore and find something you're interested in. Some will be better suited to running with a larger number of processors than just one, in which case you'll have to launch `enzo.exe` either in batch or on multiple processors. Please be courteous on Triton -- do not launch long-lived or heavy processes on the head node. Launch them on the batch nodes!

Not all of the example initialization files will have corresponding notes and plot scripts -- we're still working on that!

Looking At The Code

So now that we've seen how to compile, where the example files are and we've maybe even gotten our feet wet, let's head back to the source directory. If we do:

```
ls *.h *.C *.src *.src90
```

in `src/enzo`, we'll see all the files that make up Enzo. There are a couple files that I would particularly like to draw your attention to:

- `enzo.C` -- this is where the code is initialized
- `EvolveLevel.C` -- this is the main loop of the code; this gets called on every level, on every timestep, and it's where physics modules get called, communication occurs, and so on and so forth.
- `Grid.h` -- this is the header file for the primary class, the `grid` class. The `grid` is the fundamental unit in Enzo, and this is where it all lives.

- `InitializeNew.C` -- this is where new problems get initialized.
- `Grid_*.C` -- All of the grid methods live in one of these files.

Let's explore `EvolveLevel.C` and see if we can figure out how the chemistry in Enzo gets called.