Getting Up and Running

Version 6.2.1

January 19, 2017

There are two ways of installing the Encore compiler. The first (§1 "Playing Around with Encore") uses a virtual box image and it is easy getting the compiler up and running. The disadvantage of this approach is that benchmarks are meaningless. The second approach (§2 "Build Encore from source") is a proper installation of all the dependencies (Haskell, llvm, etc).

1 Playing Around with Encore

1. Download the compiler:

   ```
   git clone git@github.com:parapluu/encore.git
   ```

2. Install VirtualBox and Vagrant
3. Install the Encore compiler

   ```
   cd encore
   make vagrant
   ```

   This installs the Encore compiler in a Virtual Machine (VM). From now on, we are going to refer to your local machine as `localhost$` and to the VM as `vm$`.

   This process may take about 15 to 20 min (internet required).

4. After installing Encore in a VM, you can connect to the VM by:

   ```
   localhost$ vagrant ssh
   ```

5. From the VM, you can compile and run programs:

   ```
   vm$ encorec example.enc
   ```

6. To exit the VM:

   ```
   vm$ exit
   ```

7. If you are not going to work in Encore for a couple of days, you should halt or suspend the VM:

   ```
   localhost$ vagrant halt
   ```

   or

   ```
   localhost$ vagrant suspend
   ```

   `halt` stops the VM as in turning off the computer while `suspend` saves the current state of the VM and allows you to get a faster boot up next time you run `vagrant up`.

8. To connect to the VM:

   ```
   localhost$ vagrant up && vagrant ssh
   ```

You can work on your `localhost` machine with your favorite editor and use the terminal with the `vm` only for compilation purposes.

Please take into account that your local folder is shared with the VM, so any files you remove from the VM will be removed in your localhost as well.

Furthermore, if you compile a program inside the `VM`, this won't run in your local machine since the compiled version is not cross-platform.

2 Build Encore from source

Make sure that you have `doxygen` (for documentation), `premake4`, an up-to-date `clang` and `ghc` in your path.

2.1 Dependencies

Make sure you have installed:

1. `doxygen v1.8.6`
2. `clang`: Apple LLVM version 7.0.0 (clang-700.0.72) Target: x86_64-apple-darwin14.5.0 Thread model: posix
3. `g++ 4.8`
4. `ghc` The Glorious Glasgow Haskell Compilation System, version 7.10.2
5. `premake4` (Premake Build Script Generator) 4.3

Installing dependencies on Mac OS X

Installing homebrew

Go to http://brew.sh/, the instructions there work nicely. Make sure that your normal user is an admin (that you can use `sudo`). You should not need `sudo` to use `brew` in the future.

Installing `doxygen`

Run: `brew update; brew install doxygen`

Installing `clang`

Run: `brew update; brew install llvm`

Installing ghc

You need version `7.10.2`.

If you have an older version of `ghc` installed with `homebrew`: get rid of it by saying:

```
brew uninstall haskell-platform; brew uninstall ghc
```

If you have an older version of `ghc` installed downloaded from the haskell webpage: you need to remove it. Here is a http://www.haskell.org/pipermail/haskell-cafe/2011-March/090170.html.

Warning: we did not test this, and even if we did, every computer is configured differently. In the future: please use homebrew for every installation where a formula is available. It allows you to uninstall stuff easily once you don't need it any more.

Then install the newest version:

```
brew update && brew install cabal && cabal install cabal-
install && brew install ghc
```

Installing dependencies on Linux

It's only tested on Ubuntu 14.04 and hopefully it works on other distributions based on Ubuntu or Debian.

```
sudo add-apt-repository -y ppa:hvr/ghc

sudo apt-get update

sudo apt-get install -y clang lldb-3.5 g++ make premake4 zlib1g-
dev \                         ghc-7.10.2 cabal-install-
1.22 racket doxygen

cabal update && cabal install cabal-install

ln -s /usr/bin/cabal-1.22 /usr/bin/cabal
ln -s /usr/bin/lldb-3.5 /usr/bin/lldb

export PATH=/opt/ghc/7.10.2/bin:$PATH
cabal update
```

2.2 Compiling and installing Encore

```
cd encore
make
make test
```

2.3 Adding `encorec` to the path

We recommend that you add the `release` directory to your `PATH` environment variable. This will allow you to invoke the compiler:

```
$ encorec my_file.enc
```

in any directory. To do this, add this line to your `~/.bashrc` file, inserting the proper path for `<SOME_DIR>`:

```
export PATH="<SOME_DIR>/encore/release:${PATH}"
```

2.4 Compiling and Running Encore Programs

Now you can compile a program by using

```
$ encorec my_file.enc
```

This will produce an executable that you can run as usual:

```
./my_file
```

Alternatively, you can use the .enc-file as a script by adding:

```
#! /usr/bin/env encorec -run
```

as its FIRST line. After you made the file executable:

```
$ chmod u+x my_file.enc
```

you can execute it:

```
$ ./my_file.enc
```

This will compile the file, run it and remove the executable.

You can find some example programs in the programs directory `encore/programs`.

3 Emacs Support

3.1 encore-mode

Some extra support for emacs from Elias Castegren; add the following to ∼./emacs or ∼/.emacs.d/init.el

```
(add-to-list 'load-path "PATH/TO/encore/emacs")
(require 'encore-mode)
```

where `PATH/TO` is the path to the folder containing the `encore` folder.

If you also want better automatic indentation, you can additionally add:

```
(add-to-list "PATH/TO/encore/emacs/dtrt-indent-20140325.1330/")
(require 'dtrt-indent)
(dtrt-indent-mode 1)
```

You can add compilation support by adding the lines:

```
(add-hook 'encore-mode-hook (lambda () (global-set-key (kbd "C-
c C-c") 'compile)))
(add-hook 'encore-mode-hook (lambda () (global-set-key (kbd "C-
c C-n") 'next-error)))
(add-hook 'encore-mode-hook (lambda () (global-set-key (kbd "C-
c C-p") 'previous-error)))
```

This allows you to compile an `encore` source file by pressing `C-c C-c` in emacs. If there are errors, you can jump to them by pressing `C-c C-n` (to jump to the next error) or `C-c C-p` (to jump to the previous error.

3.2 yas-minor-mode

If you are a user of `yas-minor-mode`, you can find snippet files in `PATH/TO/encore/emacs/yas`. You can use them by executing:

```
cd ∼/.emacs.d/snippets/ # (the snippets directory may differ on your
system)
ln -s PATH/TO/encore/emacs/yas/encore-mode .
```