Seven easy steps to install ***Torch*** on a *Linux* or *Unix* machine (using *xmake*)

Ronan Collobert

August 11, 2004

1 What do I need ?

You must have:

- A *Linux* or *Unix* machine (a motherboard, an hard disk, a screen,a keyboard and a processor could be interesting).
- A `C++` compiler. You can take for example the *GNU compiler*[1], but other compilers should work.
- *Python*[2]. This is a script language needed for *xmake*.

2 Download the library an unpack archives

The library is available in one big archive. Just go in the download[3] section of the ***Torch***[4] website, and take the *Unix/Linux* archive. Unpack it: a `Torch3` directory should appear, with some directories inside (the sources of the core will be in the `Torch3/core` directory...). The library is divided into several parts: the *core*, which is the foundation of the library (it should be stable...) and some packages developped by any user.

3 *xmake* setup

The first time you install ***Torch***, you will find in the root directory `Torch3` a python script called `xmake`. Move it into a directory which is available in your path variable environment. By default, *xmake* supposes that *python* is in `/usr/bin/python`. If it is not your case, change the first line of `xmake`. Test if *xmake* is working with the command `xmake help` which gives a little help on *xmake*.

4 Set your compilation options

The library needs a file named `<os>.cfg` to compile, where `<os>` is the name of you operating system, given by the command `xmake os`. This file must be in the `Torch3` directory. There are some examples of this file in the `Torch3/config` directory.

[1] `http://www.gnu.org/software/gcc`
[2] `http://www.python.org`
[3] `http://www.torch.ch/downloads.php`
[4] `http://www.torch.ch`

Therefore, check the name of your OS with `xmake os` and copy an example of `*.cfg` from the directory `Torch3/config` to `Torch3/`. For example, if you have a *SunOs* system, and you're using the CC workshop compiler, copy the file `Torch3/config/CC.cfg` in the file `Torch3/SunOS.cfg`.
After you've copied this file, edit it. It should look like the following (if you are using the *GNU compiler* file example).

```
# Don't touch this first line
[torch]

# Need more verbosity ?
# Uncomment this line if you want to see the full compiling command
# called by xmake, for instance
#verbose = 1

# Packages you want to use
# For example if you want to use gradient machines and distributions, put
# "packages = gradients distributions"
# Don't include packages which contain main programs (such as "examples")
# Don't include "core"
packages =

# Magik key if you want several libraries
# for the same platform
# (It's useful if you're using two different compilers: a different file
# for dependencies and for binaries will be generated for each magic_key)
#magic_key =

# Compiler, linker and archiver
compiler = g++
linker = g++
#archiver = g++ -shared -o
archiver = ar -rus

# Your librairies
# (for example "-lm", but not needed on most systems...)
libraries =

# Your includes
# (for example -I/usr/local/special)
includes =

# optimize mode
# Comment one of these lines... "opt" is for optimized code,
# and "dbg" for debug code (if you plan to use a debugger)
debug = opt
# debug mode
#debug = dbg

# Comment one of these lines... if you take "double" (and comment "float")
# the "real" variables will be "double". Otherwise "float".
# double version
```

```
#floating = double
# float version
floating = float

# Check now the flags for your compiler.
# -DUSE_DOUBLE is used to define USE_DOUBLE in the code (for *_double flags).
# -DDEBUG is used to define DEBUG in the code (for dbg_* flags)

# Debug double mode
dbg_double = -g -Wall -DUSE_DOUBLE -DDEBUG

# Debug float mode
dbg_float = -g -Wall -DDEBUG

# Optimized double mode
opt_double = -Wall -O2 -ffast-math -DUSE_DOUBLE

# Optimized float mode
opt_float = -Wall -O2 -ffast-math
```

5 Compile the library

That's easy. Go in the `Torch3` directory and do `xmake`. If everything goes ok, several directories should be created: `objs` in which you'll find all the object files, and `libs` in which you'll find the library.
You should'nt have any warning during the compilation... otherwise, send me an email, if it's related to the code of the library.
The following commands are available for *xmake*:

- `xmake help` : gives a little help on *xmake*.
- `xmake os` : gives the name of your operating system.
- `xmake` : redo the dependencies if necessary, and compile the whole library.
- `xmake clean` : remove the dependency files, the objects and the library for the current system.
- `xmake distclean` : remove the dependency files, the objects and the library for all the systems.
- `xmake depend` : force the (re-)creation of the dependency files. This is needed *only* in the case you add a include file (something like `#include "MyNewClass.h"`) in an *existing* code file. Note that *xmake* autodetects new files: you do not need to perform this command after the creation of a new file.

6 Compile your program

Create a directory in `Torch3/`. Go in this directory, edit your program. Then, if the name of you program is `foo.cc`, just do `xmake foo` or `xmake foo.cc`. A subdirectory will appear (with a name corresponding to your compilation flags and your operating system) with the program `foo`...

7 Notes on *xmake*

- If you want to compile a whole directory of main program, just do `xmake *.cc`.
- If you set the options in the `.cfg` file to the "optimized" mode, and that you want to compile in debug mode, do `xmake -dbg`.
- In a similar way, if the debug mode is the default, you can compile optimized code with `xmake -opt`.
- To force the compilation of main programs in optimized or debug mode, you can also do something like `xmake -opt *.cc` or `xmake -dbg *.cc`.
- You can compile the library from anywhere in `Torch3` (in any subdirectory) by just a `xmake` command.