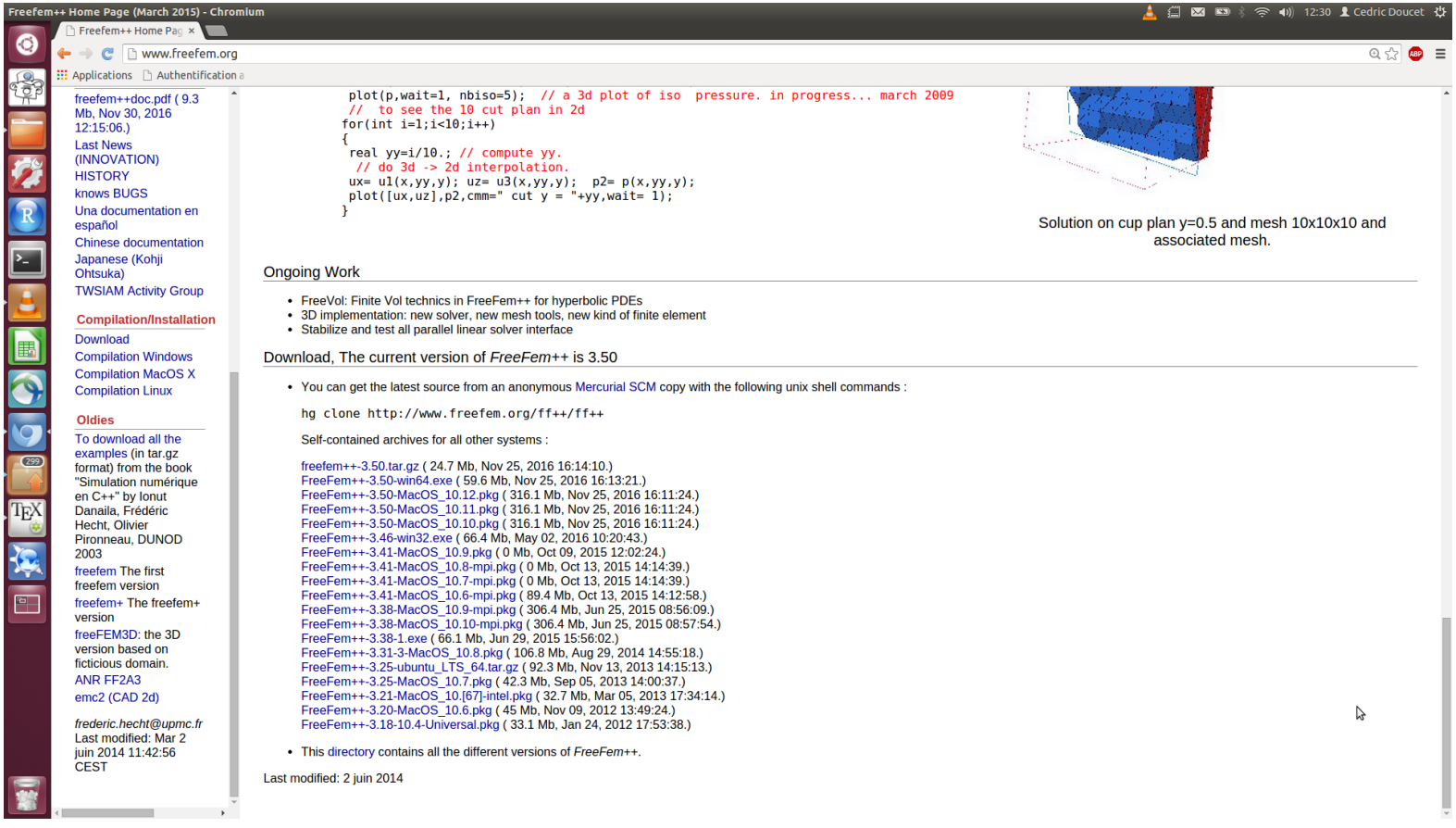
CMake support in FreeFem++

Cédric Doucet

Inria Paris

December 8, 2016

Current installation of FreeFem++



Current installation of FreeFem++

Installation from binaries/packages:

- very easy
- Mac, Ubuntu and Windows

Installation from sources:

- building with minimal dependencies
- automatic installation of many scientific modules
- platform-dependent (Linux, Mac, Windows)

Maintenance and evolution:

- development of Autoconf/configure is almost stopped
- manual porting on different platforms
- difficult to customize (31 pages configuration script)

Could we do better?

CMake

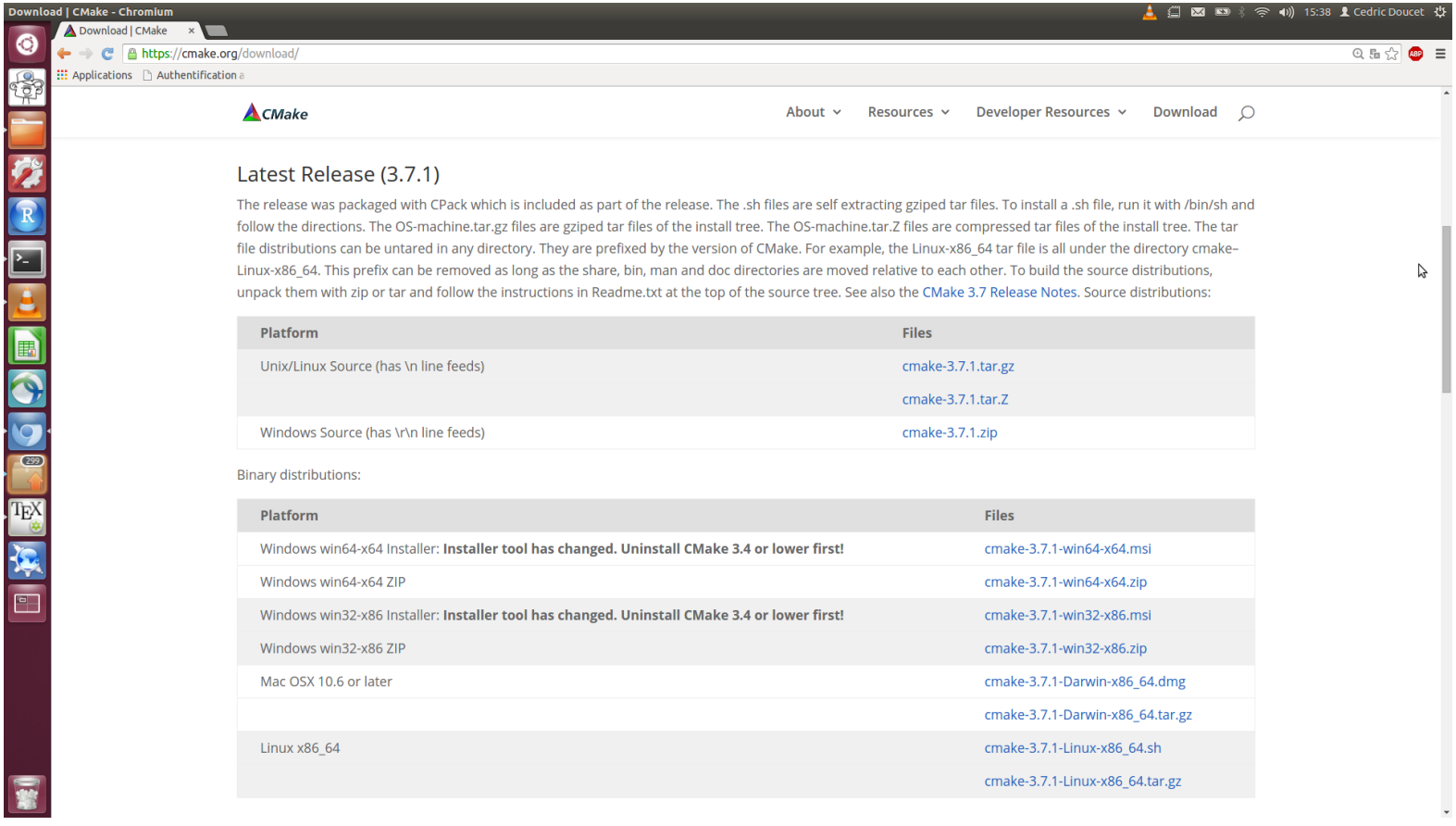
CMake: **C**ross-platform **Make**

- cross-platform
- based on Make
- free and open-source
- customizable
- Kitware product
- stable (15 years)
- well established
- active support

Other Kitware products: *ITK*, *Paraview*, *VTK*

Other young but promising tools: *Scons*, *Waf*, ...

Installation of CMake



+ installed by default on many systems
+ available as a package (apt-get, brew, conda, port, yum, ...)

Why is CMake better?

Reason 1: it is the same... or almost

Current installation from sources:

```
./configure --prefix=/home --enable-download=true
make
make check
make install
```

CMake installation from sources:

```
cmake -D CMAKE_INSTALL_PREFIX=/home -D ENABLE_DOWNLOAD=true .
make
make check
make install
```

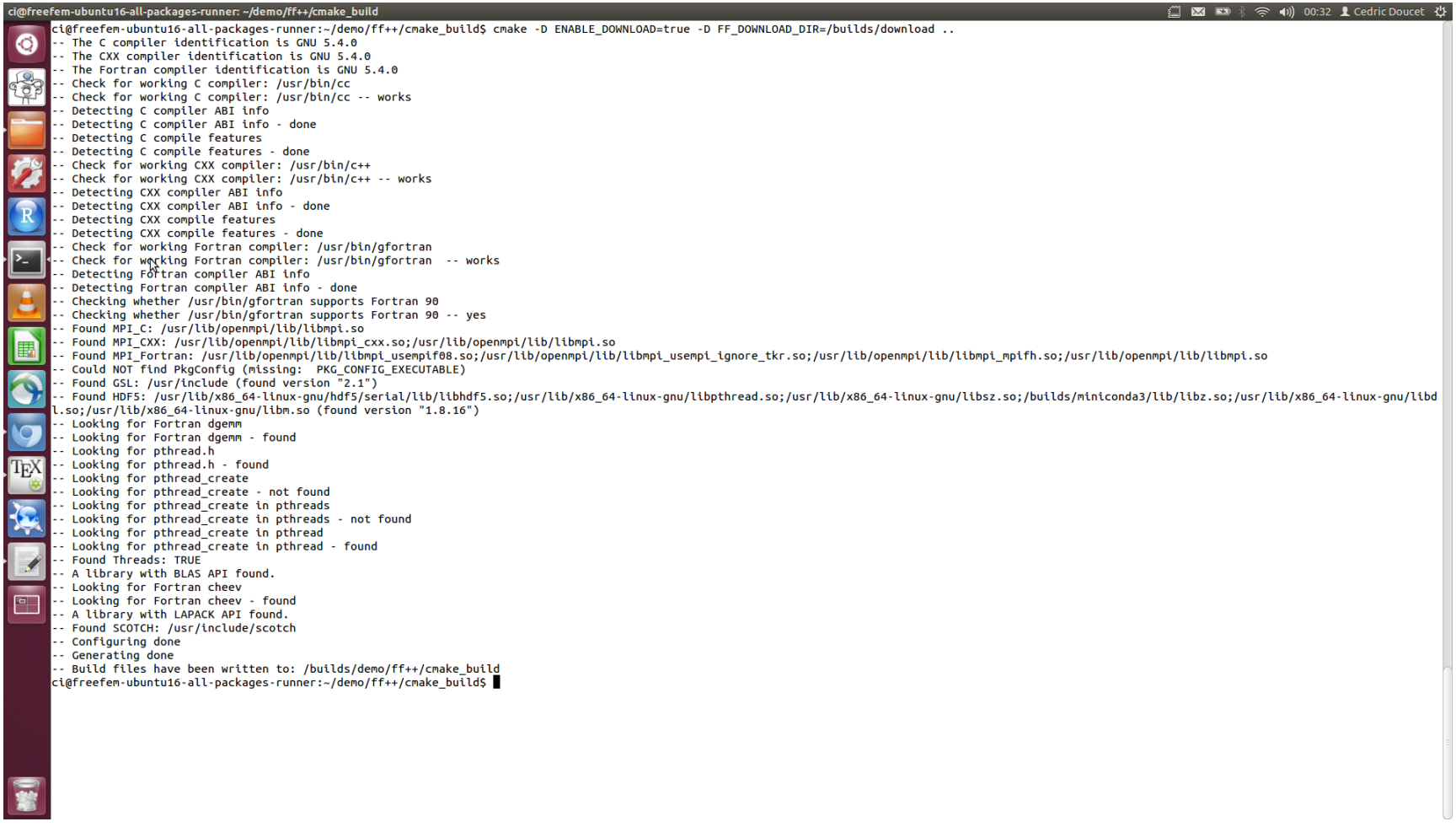


Figure: Output of `cmake`

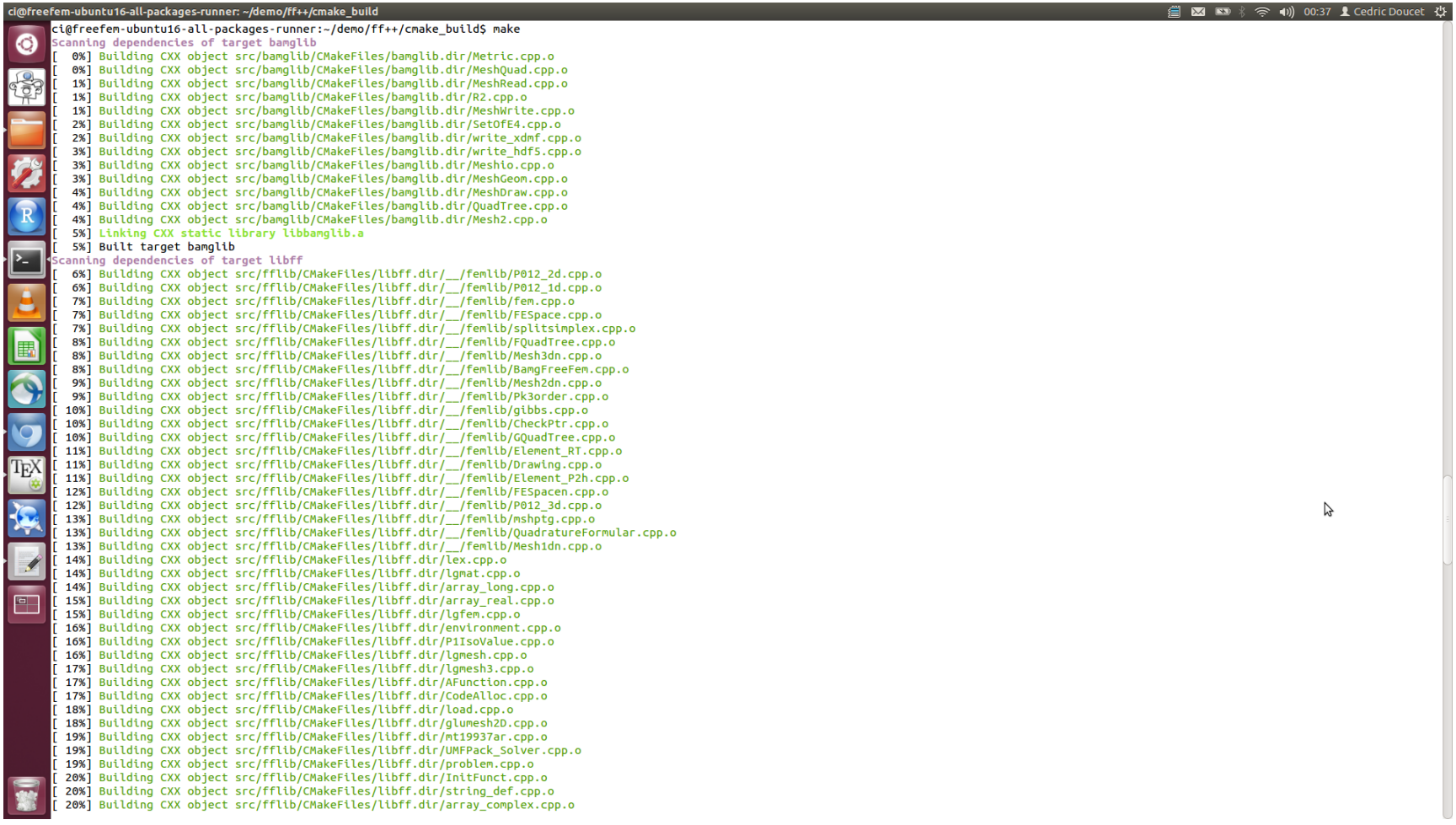


Figure: Output of `make`

Reason 2: generators

```
cmake -G "Visual Studio 15 2017 Win64" ..
```

Command-Line tools:

- Borland
- JOM
- MinGW
- MSYS
- Ninja
- NMake
- Unix
- Watcom

IDEs:

- Codeblocks
- CodeLite
- Eclipse CDT4
- Kate
- KDevelop3
- SublimeText 2
- Visual Studio
- XCode

Note: some IDEs can parse CMake scripts (e.g. NetBeans)

Reason 3: expressive high-level language

CMakeLists.txt:

```
CMAKE_MINIMUM_REQUIRED(VERSION 2.8)
PROJECT(HELLO C)
FILE(GLOB SOURCE_FILES *.c)
ADD_EXECUTABLE(hello ${SOURCE_FILES})
```

Makefile:

```
CC=gcc
EXEC=hello
SRC= $(wildcard *.c)
OBJ= $(SRC:.c=.o)

all: $(EXEC)

hello: $(OBJ)
$(CC) -o $@ $^ $(LDFLAGS)

main.o: hello.h

%.o: %.c
$(CC) -o $@ -c $< $(CFLAGS)
```

configure.ac:

```
ff_uname=‘uname‘
case $ff_uname in
    CYGWIN*|MINGW*|MSYS_NT*)
ff_suffix_dylib="dll"
ff_suffix_dylib_a="dll.a";;
     Darwin)
ff_suffix_dylib="dylib"
ff_suffix_dylib_a="dylib";;
    *)
        ff_suffix_dylib="so";
        ff_suffix_dylib_a="so";
esac
```

Reason 4: packaging is made easy

```
cpack .
```

Linux:

- Debian
- Fedora
- Red Hat
- Ubuntu

Mac:

- Bundle
- DMG

Windows:

- Cygwin
- NSIS
- WIX

Note: cross-compilation is possible

```
CMAKE_MINIMUM_REQUIRED(VERSION 2.6)

PROJECT(FreeFem)

ADD_EXECUTABLE(ff++ freefem.cxx)
INSTALL(TARGETS ff++ DESTINATION freefem++/bin)

SET(CPACK_GENERATOR "DEB")
SET(CPACK_DEBIAN_PACKAGE_MAINTAINER "Cedric Doucet")

INCLUDE(CPack)

# To use this:
# make package
# sudo dpkg -i ff++-0.1.1-Linux.deb

# This will result in the file:
#/usr/freefem++/bin/ff++
```

Note: it seems possible to let CMake list dependencies

Reason 5: Customisable detection of modules

FindScotch.cmake:

```
SET(SCOTCH_FOUND False)
FIND_PATH(SCOTCH_INCLUDES NAMES scotch.h
                          PATHS /usr/include
                          PATH_SUFFIXES scotch)
FIND_LIBRARY(SCOTCH_LIBRARIES NAMES scotch
                              PATHS /usr/lib)
IF(SCOTCH_INCLUDES AND SCOTCH_LIBRARIES)
  SET(SCOTCH_FOUND True)
ENDIF(SCOTCH_INCLUDES AND SCOTCH_LIBRARIES)
```

Usage:

```
FIND_PACKAGE(Scotch)
IF(SCOTCH_FOUND)
  ADD_LIBRARY(scotch SHARED scotch.cpp)
  INCLUDE_DIRECTORIES(/usr/include/scotch)
ENDIF()
```

Note: *CMake provides such files for BLAS, LAPACK, MPI, ...*

Reason 5: Customisable installation of modules

InstallIPOPT.cmake:

```
SET(LIBNAME IPOPT)
SET(URL http://www.coin-or.org/download/source/Ipopt/Ipopt-3.12.4.tgz)
SET(URL_MD5 12a8ecaff8dd90025ddea6c65b49cb03)
SET(CONFIGURE_COMMAND ../src/configure --prefix=<INSTALL_DIR>
                                       CXX=${CMAKE_CXX_COMPILER})
SET(BUILD_COMMAND make)
SET(INSTALL_COMMAND make install)
```

Usage:

```
INSTALL_PACKAGE(IPOPT)
ADD_LIBRARY(ff-Ipopt SHARED ff-Ipopt.cpp)
ADD_DEPENDENCIES(ff-Ipopt ipopt)
INCLUDE_DIRECTORIES(${IPOPT_INCLUDES})
TARGET_LINK_LIBRARIES(${IPOPT_LIBRARIES})
```

Note: *targets and variables are automatically generated*

When will CMake support be available?

CMake support is still in progress...

Core building without any download	Done
Automatic installation process	Done
Core building with downloads	Done
Test building	Done
Recipe files for external libraries	Doing
Find files for external libraries	Doing
Porting	Doing
Continuous integration	Doing
Automatic testing	Todo
Packaging	Todo
Documentation	Todo

but will be available soon!