



# Build facili con CMake

Il sistema di build -definitivo-

Carlo Nicolini

November 21, 2012



- 1 Introduzione
- 2 Struttura di base di un CMakeLists.txt
- 3 Supporto a librerie esterne
- 4 Creazione pacchi con CPack



- 1 Introduzione
- 2 Struttura di base di un CMakeLists.txt
- 3 Supporto a librerie esterne
- 4 Creazione pacchi con CPack



# Motivazioni



- Split fra sviluppatore e maintainer
- Portabilità su molti O.S.
- Semplicità delle build
- Migliore rapporto con cliente/comunità di sviluppatori
- Unit testing



# Il passato



## Sistema di build

- Come fare una build?
- Come differenziare Debug e Release
- Flag al compilatore?
- Come linkare? Shared o static?



Compilazione manuale (anni 60): `gcc -c mylibrary.cpp`  
`mylibrary.h -o mylibrary.o`

Sistemi di build automatici:

- *Scons poco usato, cross-platform, necessita competenze di programmatore*
- *Autotools molto usato, sintassi complicatissima*
- *Jam buggy, dipendenze fatte a mano*
- *Waf lasciam perdere*
- *eccetera...*



Un sistema di build moderno gestisce building, testing e packaging tutto insieme.

- Cross platform (veramente).
- Dipendenze soddisfatte sempre (veramente).
- Un linguaggio di scripting che da libertà (vera).
  - `#define` a compile-time.
  - Creazione menu Gnome, aggiunta icone e creazione setup personalizzati.
  - Build differenziate per multi-architettura.
- **Out-of-source** builds



- Sistema di *meta-make*
- Genera Makefile o progetti (detti *generatori*) Kdevelop3, Eclipse, XCode, makefiles (Unix, NMake, Borland, Watcom, MinGW, MSYS, Cygwin), Code::Blocks etc
- Progetti multipli-eseguibili multipli





# CMake tree e primi passi



Supponiamo questo tree:

- **src**

- `myapp.cpp`
- `myapp.h`
- `CMakeLists.txt`

- **build**

- `CMakeLists.txt`

Si fa la build con:

- `cmake .`
- `make`



- 1 Introduzione
- 2 **Struttura di base di un CMakeLists.txt**
- 3 Supporto a librerie esterne
- 4 Creazione pacchi con CPack



# HelloWorld eseguibile



#Creiamo il nome del progetto

**PROJECT( helloworld )**

# Impostiamo la variabile hello\_SRCS a contenere la hello.cpp

**SET( hello\_SRCS hello.cpp )**

# Crea l'eseguibile di nome hello dal file contenuto nella variabile  
hello\_SRCS

**ADD\_EXECUTABLE( hello \${hello\_SRCS} )**

- Tutte le variabili sono **stringhe**
- Le variabili si dereferenziano bash-style **\${NOMEVARIABILE}**
- Le variabili si impostano con **SET**



# Creazione libreria statica .a



#Creiamo il nome del progetto

**PROJECT( mylibrary )**

# Impostiamo la variabile mylibrary\_SRCS a contenere tutti i file che definiscono la libreria

**SET( mylibrary\_SRCS Foo.cpp Bar.cpp Qux.cpp)**

# Crea una libreria STATICA (di default in CMake) a partire dai sorgenti

# in Linux con gcc genera un file *libmyLibrary.a*

**ADD\_LIBRARY( myLibrary \${mylibrary\_SRCS} )**

# Oppure crea una libreria SHARED (o DINAMICA in Windows) a partire dai sorgenti

# in Linux con gcc genera un file *libmyLibrary.so*

**ADD\_LIBRARY( myLibrary SHARED \${mylibrary\_SRCS} )**



# Linguaggio - Variabili



- Non serve dichiararle (stringa vuota se non esistono)
- Atipizzate
- SET crea e modifica variabili
- SET si affianca a LIST
- SEPARATE\_ARGUMENTS spezza argomenti separati da spazio in una LIST
- In Cmake 2.4: globali (name clashing problems) In Cmake 2.6: scoped
- CMake corrente 2.9



# Linguaggio - Variabili



- Non serve dichiararle (stringa vuota se non esistono)
- Atipizzate
- SET crea e modifica variabili
- SET si affianca a LIST
- SEPARATE\_ARGUMENTS spezza argomenti separati da spazio in una LIST
- In Cmake 2.4: globali (name clashing problems) In Cmake 2.6: scoped
- CMake corrente 2.9



# Flessibilità - Sintassi



- Costrutto condizionale IF

```
IF ( expression )  
...  
ELSE ( expression )  
...  
ENDIF ( expression )
```
- Costrutto FOREACH ( comodo per liste)

```
FOREACH ( loopvariable )  
...  
ENDFOREACH ( loopvariable )
```
- Ciclo WHILE

```
WHILE ( condition )  
...  
ENDWHILE ( condition )
```



# Esempio di foreach e wildcards su files

```
file(GLOB mytestfiles "test*.cpp")
foreach(testfile ${mytestfiles})
    message(STATUS "This is a test file ${testfile}")
endforeach(testfile ${mytestfiles })
```





# switch condizionali di sistema



```
IF ( MSVC )  
ENDIF ( MSVC )
```

```
IF (WIN32)  
ENDIF (WIN32)
```

```
IF ( UNIX )  
ENDIF (UNIX)
```

```
IF (APPLE)  
ENDIF (APPLE)
```

- Tutte le variabili sono scoped nel singolo CMakeLists.txt
- Non esiste un costrutto **switch**



# Espressioni regolari



Complicate ma possibili

```
STRING( REGEX MATCH ... )
```

```
STRING( REGEX MATCHALL ... )
```

```
STRING( REGEX REPLACE ... )
```



How to convert a semicolon separated list to a whitespace separated string?

```
set(foo abc.c abc.b abc.a)
```

```
foreach(arg ${foo})  
    set(bar "${bar} ${arg}")  
endforeach(arg ${foo})
```

```
message("foo: ${foo}")  
message("bar: ${bar}")
```



# Compatibilità con versioni precedenti

- Molto importante impostare la compatibilità con le versioni precedenti, cambi di sintassi e bug-fix
- Mantenere sempre CMake all'ultima versione (attuale 2.8.10)
- Variabile di sistema apposita

```
CMAKE_MINIMUM_REQUIRED(VERSION 2.6.0 FATAL_ERROR)
```



# Cmake cache



- CMake salva le variabili non variate in un file CMakeCache.txt
- Veloce su Unix, lento su Windows (MSVC)
- Utile ripulire la cache



# Gestione di Debug e Release



- Una variabile definisce il tipo di build
- `SET(CMAKE_BUILD_TYPE XXX)`
  - Debug
  - Release
  - RelWithDebInfo
  - MinSizeRel
  - Profile
- oppure da linea di comando:  
`cmake -DCMAKE_BUILD_TYPE=Debug .`
- Debug → gdb+valgrind, grosse dimensioni
- Si rilascia il pacchetto sempre in Release
- Profile utile quando accoppiata con gprof/KCacheGrind



# Versionamento compile-time



- Utilizzare il modulo UseSubversion  
`http://bit.ly/WjwiRP` (ripulire il repo prima)
- Vengono definite alcune variabili utili:
  - `SUBVERSION_REPO_REVISION`  
`SUBVERSION_REPO_LAST_CHANGED_DATE`  
etcetera
  - Passabili al compilatore come flags:  
`ADD_DEFINITIONS(-DREV_NUMBER=`  
`"${SUBVERSION_REPO_REVISION}")`
- Nel codice C/C++ quindi:  

```
int revision=REV_NUMBER;  
printf("La versione corrente è %d\n", revision);
```



# Generare la documentazione



Aggiungere un target doc ed usare FindDoxygen.cmake

```
# Aggiunge un target al Makefile per generare
# la doc usando Doxygen
find_package(Doxygen)
if(DOXYGEN_FOUND)
  configure_file(${CMAKE_CURRENT_SOURCE_DIR}/Doxyfile
    ${CMAKE_CURRENT_BINARY_DIR}/Doxyfile @ONLY)
  add_custom_target(doc
    ${DOXYGEN_EXECUTABLE} ${CMAKE_CURRENT_BINARY_DIR}/Doxyfile
    WORKING_DIRECTORY ${CMAKE_CURRENT_BINARY_DIR}
    COMMENT "Generating API documentation with Doxygen"
    VERBATIM )
endif(DOXYGEN_FOUND)
```

1

---

<sup>1</sup><http://bit.ly/9eel8b>





- 1 Introduzione
- 2 Struttura di base di un CMakeLists.txt
- 3 Supporto a librerie esterne
- 4 Creazione pacchi con CPack



# Highlighting

Hey! Look here! here!



If it's very very important...

... you can "alert" in an "alertblock"

Ewww, nasty, heh?



- 1 Introduzione
- 2 Struttura di base di un CMakeLists.txt
- 3 Supporto a librerie esterne
- 4 Creazione pacchi con CPack



CPack è il progetto cugino di CMake con cui è strettamente integrato e permette di creare pacchi:

- Linux generici ( .sh, .tgz )
- Distro-based ( .deb, .rpm, )
- OSX (creazione .app o dischi .dmg )
- Windows (creazione setup.exe grazie a NSIS installer e 7Zip )
- Pacchi architettura-specifici
- Cross compilazione!



Mailing list ufficiale degli utenti di CMake

<http://www.cmake.org/mailman/listinfo/cmake>



Sito di domande e risposte, frequentato da molti utenti di CMake

[www.stackoverflow.com](http://www.stackoverflow.com)



CMake tutorial online

[http:](http://www.cmake.org/cmake/help/cmake_tutorial.html)

[//www.cmake.org/cmake/help/cmake\\_tutorial.html](http://www.cmake.org/cmake/help/cmake_tutorial.html)



Il libro ufficiale, insostituibile (in biblioteca di ingegneria a Mesiano )

<http://www.kitware.com/products/books/CMakeBook.html>



Slides fatte con il tema beamer offerto da KDE



# Domande ?

Carlo Nicolini

[nicolini.carlo@gmail.com](mailto:nicolini.carlo@gmail.com)