

Cross-compilation de programme Qt4 de Linux vers Windows

par Michaël Todorovic ([Autres articles](#)) ([Blog](#))

Date de publication : 28/04/2007

Dernière mise à jour : 28/04/2007

Ce tutoriel vous permettra de compiler des exécutables Qt4 windows sous linux afin de tester la portabilité de votre code.

I - Introduction.....	3
II - Préparation de l'environnement de cross-compilation.....	4
II-1 - Installation des paquets requis.....	4
II-2 - Test de l'environnement.....	4
II-3 - Installation de Qt 4.2.3 pour Windows.....	5
II-4 - Configuration de Qt pour Windows dans l'environnement de cross-compilation.....	6
III - Conclusion.....	10
III-1 - Conclusion.....	10
III-2 - Références.....	10

I - Introduction

Vous développez une application avec Qt4 sous Linux et vous devez également faire en sorte que l'application tourne sous Windows. Soit vous avez deux machines : une sous Linux et l'autre sous Windows et dans ce cas, vous n'avez pas de problème "logistique" pour tester. Cependant si vous n'avez qu'une machine avec Linux et Windows, vous devez redémarrer sous Windows pour tester votre application ce qui est loin d'être pratique. Vous pouvez également virtualiser Windows. Dans ce tutoriel, nous proposons de compiler un exécutable Windows à partir de Linux. La technique utilisée est la cross-compilation. Cette technique permet de générer un exécutable qui ne tournera pas sur la machine hôte mais sur une machine cible avec un autre environnement d'exploitation. Pour la cross-compilation, on parle d'environnement d'exploitation et non de système d'exploitation. C'est une notion plus large dans le sens où on peut avoir le même type de système d'exploitation sur deux machines mais que ces machines soient différentes par leur processeur (32 bits ou 64 bits par exemple).

II - Préparation de l'environnement de cross-compilation

II-1 - Installation des paquets requis

Vous devez commencer par installer l'environnement de cross-compilation.

```
apt-get install wine mingw32 mingw32-binutils mingw32-runtime libqt4-core libqt4-gui libqt4-qt3support  
libqt4-sql libqt4-debug-dev libqt4-dev
```

II-2 - Test de l'environnement


Nous allons commencer par tester le compilateur. Nous allons donc compiler ce fichier source. Ce code va ouvrir une fenêtre ayant pour titre "Hello World" et pour texte "Cette fenêtre prouve que le cross-compilateur est fonctionnel !"

testCrossCompilation.cpp

```
#include <windows.h>  
  
int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)  
{  
    MessageBox(NULL, "Cette fenêtre prouve que le cross-compilateur est fonctionnel !", "Hello World",  
    MB_OK);  
    return 0;  
}
```

Voici à entrer dans une console pour compiler le fichier source

```
i586-mingw32msvc-g++ -o testCrossCompilation.exe testCrossCompilation.cpp
```

 *Il se peut que vous ayez le message "**testCrossCompilation.cpp:7:6: attention : pas de retour chariot à la fin du fichier**". Dans ce cas, ajoutez simplement une ligne à la fin du fichier.*

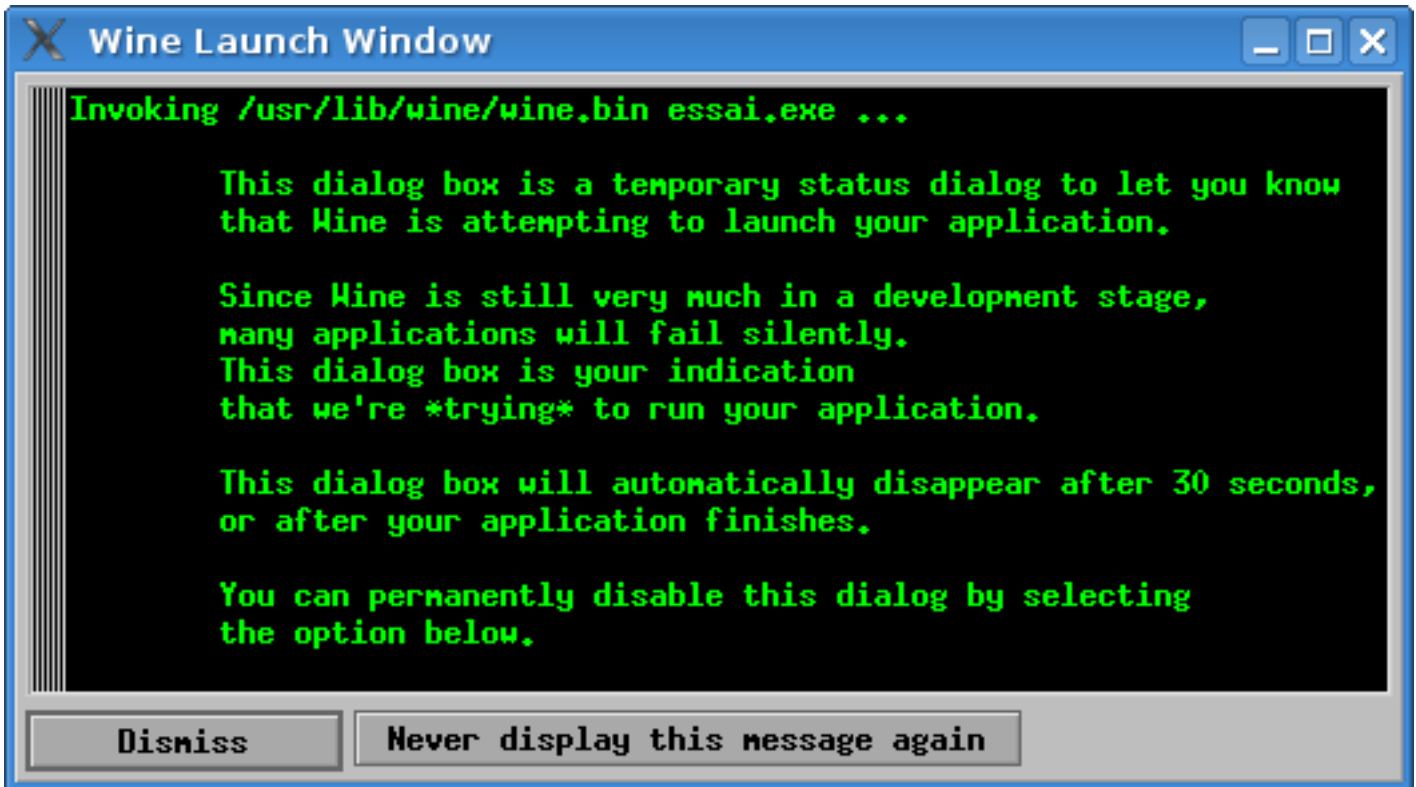
Pour que l'exécutable windows généré fonctionne, il lui faut mingwm10.dll. C'est le fichier contenant les appels système nécessaires à l'exécution du programme. mingw32 fournit ce fichier. Vous devez le décompresser dans le répertoire où est stocké l'exécutable windows

```
gunzip -c /usr/share/doc/mingw32-runtime/mingwm10.dll.gz > mingwm10.dll
```

Maintenant, vous pouvez tester le programme

```
wine essai.exe
```

Si vous lancez wine pour la première fois, vous aurez probablement cette fenêtre que vous pouvez faire disparaître définitivement en cliquant sur "Never display this message again"



Vous aurez ensuite votre fenêtre qui s'affichera prouvant ainsi que votre environnement de cross-compilation est correctement installé



Maintenant que l'environnement de base est en place, il faut installer Qt.

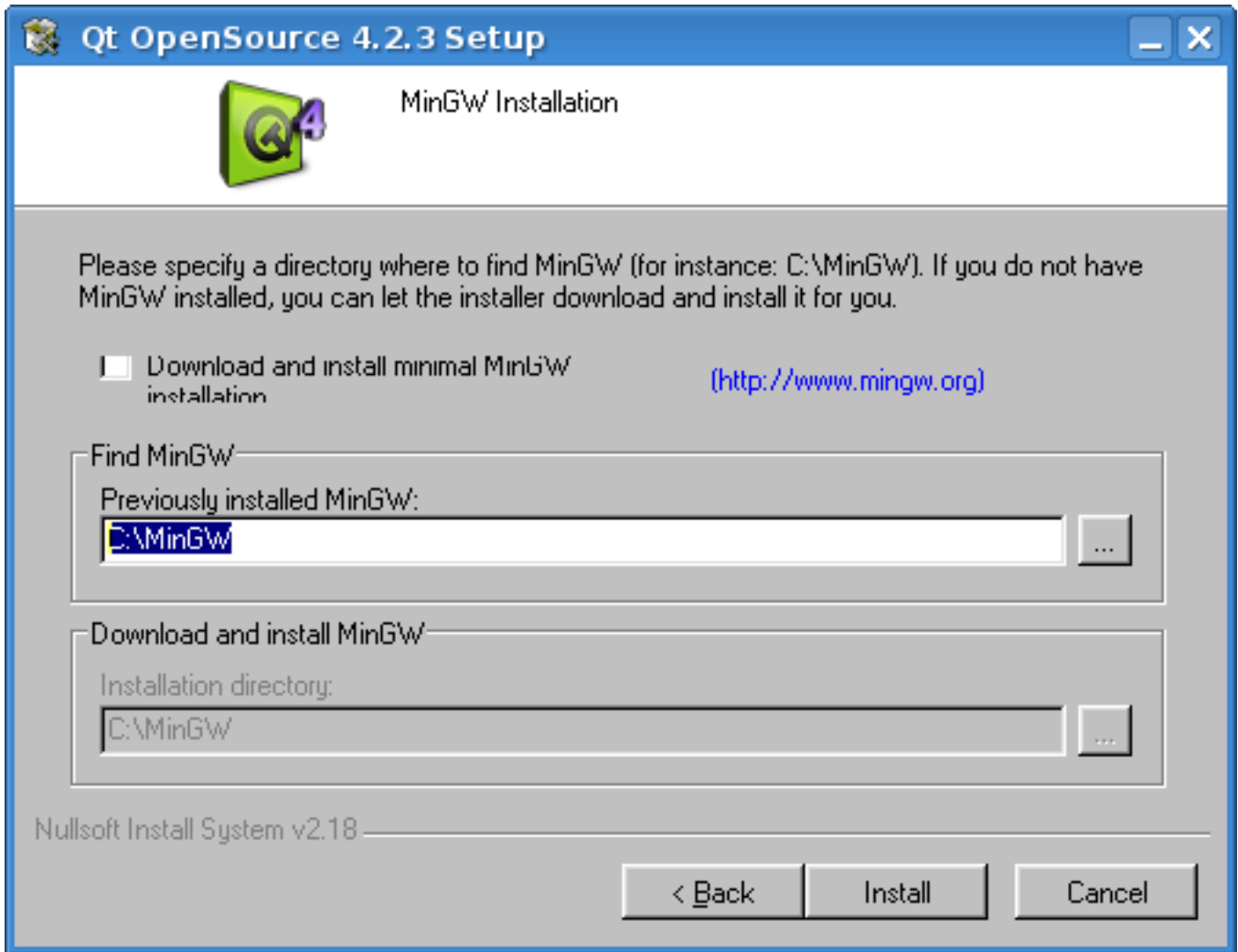
II-3 - Installation de Qt 4.2.3 pour Windows

Qt 4.2.3 est utilisé dans ce tutoriel. Commencez par récupérer la version windows de Qt [ici](#).

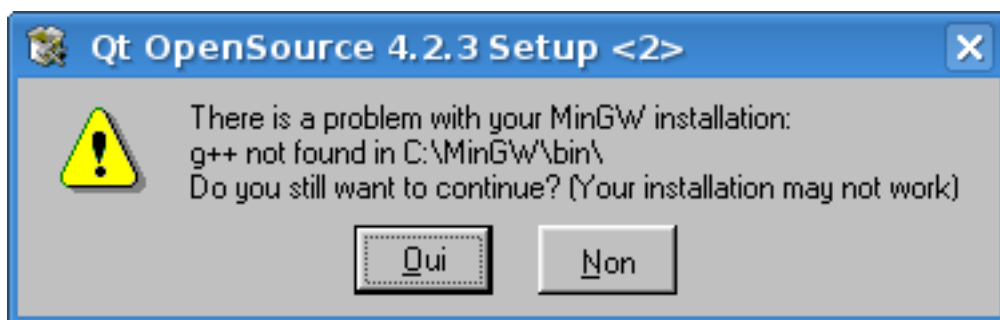
Ensuite, installez Qt grâce à wine

```
wine qt-win-opensource-4.2.3-mingw.exe
```

Validez toutes les étapes jusqu'à arriver à cette étape



Cliquez sur "Install". Vous aurez alors un message vous prévenant que votre installation de mingw est incorrecte. Effectivement, mingw n'est pas installé sous wine donc l'installation n'est pas correcte. Cependant, il est installé sous Linux donc cliquez sur "Oui" malgré l'avertissement.



L'installation se continue normalement. A la fin de l'installation, ce n'est pas la peine de lancer les démonstrations Qt et de regarder le fichier README donc décochez ces cases.

II-4 - Configuration de Qt pour Windows dans l'environnement de cross-compilation

Maintenant que Qt est installé, il faut le configurer. Commencez par copier **en root** les fichiers Qt dans un répertoire externe à votre répertoire personnel pour que tous les utilisateurs sur votre ordinateur puissent en profiter.

```
mkdir -p /usr/local/qt4-win32
cp -rf /home/mik/.wine/drive_c/Qt/4.2.3/* /usr/local/qt4-win32
cp -rf /usr/share/qt4/mkspecs/win32-g++/ /usr/share/qt4/mkspecs/win32-crossCompil-g++
```

La dernière ligne copie le profil Qt pour la compilation windows. win32-crossCompil-g++ est le nom du profil que l'on utilisera pour la compilation. La configuration ne s'arrête pas là : il faut éditer le fichier de configuration du profil /usr/share/qt4/mkspecs/win32-crossCompil-g++/qmake.conf. Prenez votre éditeur habituel avec les droits root afin de pouvoir sauvegarder les modifications. Voici les lignes du fichier que vous devez modifier.

```
QMAKE_CXX                = i586-mingw32msvc-g++

QMAKE_INCDIR             = /usr/i586-mingw32msvc/include/
QMAKE_INCDIR_QT          = /usr/local/qt4-win32/include
QMAKE_LIBDIR_QT          = /usr/local/qt4-win32/lib

QMAKE_LINK               = i586-mingw32msvc-g++

# ne pas oublier le -mwindows ici
QMAKE_LFLAGS              = -mthreads -Wl,-enable-stdcall-fixup -Wl,-enable-auto-import -Wl,-enable-
runtime-pseudo-reloc -mwindows

#isEqual(MINGW_IN_SHELL, 1) {
QMAKE_DIR_SEP             = /
QMAKE_COPY               = cp
QMAKE_COPY_DIR           = cp -r
QMAKE_MOVE               = mv
QMAKE_DEL_FILE           = rm -f
QMAKE_MKDIR              = mkdir -p
QMAKE_DEL_DIR            = rm -rf
#} else {
#    QMAKE_COPY             = cp
#    QMAKE_COPY_DIR        = cp -r
#    QMAKE_MOVE            = mv
#    QMAKE_DEL_FILE        = rm -f
#    QMAKE_MKDIR           = mkdir -p
#    QMAKE_DEL_DIR         = rm -rf
#}

# Enlever les .exe et rajouter -qt4
QMAKE_MOC                = $$[QT_INSTALL_BINS]${DIR_SEPARATOR}moc-qt4
QMAKE_UIC                = $$[QT_INSTALL_BINS]${DIR_SEPARATOR}uic-qt4

QMAKE_RC                 = i586-mingw32msvc-windres
QMAKE_STRIP              = i586-mingw32msvc-strip
```

Votre environnement de cross-compilation est désormais configuré correctement. Maintenant, testons son bon fonctionnement. Prenons le projet [testCrossCompilation](#). Décompressez l'archive et allez dans le répertoire

```
tar xvj testCrossCompilation.tar.bz2
cd testCrossCompilation
```

Maintenant, générez le Makefile avec le profil win32-crossCompil-g++ et compilez le projet.

```
qmake-qt4 -spec win32-crossCompil-g++ testCrossCompilation.pro
make
```


Ceci devrait s'afficher


```
mik@debian-mik:/tmp/testCrossCompilation$ make
make -f Makefile.Release
```

```
make[1]: entrant dans le répertoire « /tmp/testCrossCompilation »
/usr/bin/uic-qt4 src/Form.ui -o build/ui_Form.h
i586-mingw32msvc-g++ -c -O2 -Wall -mthreads -DUNICODE -DQT_LARGEFILE_SUPPORT
-DQT_NO_DEBUG -DQT_GUI_LIB -DQT_CORE_LIB -I"/usr/local/qt4-win32/include/QtCore"
-I"/usr/local/qt4-win32/include/QtCore" -I"/usr/local/qt4-win32/include/QtGui"
-I"/usr/local/qt4-win32/include/QtGui" -I"/usr/local/qt4-win32/include" -I"."
-I"src" -I"build" -I"build" -I"/usr/i586-mingw32msvc/include"
-I"/usr/share/qt4/mkspecs/win32-crossCompil-g++" -o build/Form.o src/Form.cpp
i586-mingw32msvc-g++ -c -O2 -Wall -mthreads -DUNICODE -DQT_LARGEFILE_SUPPORT
-DQT_NO_DEBUG -DQT_GUI_LIB -DQT_CORE_LIB -I"/usr/local/qt4-win32/include/QtCore"
-I"/usr/local/qt4-win32/include/QtCore" -I"/usr/local/qt4-win32/include/QtGui"
-I"/usr/local/qt4-win32/include/QtGui" -I"/usr/local/qt4-win32/include" -I"."
-I"src" -I"build" -I"build" -I"/usr/i586-mingw32msvc/include"
-I"/usr/share/qt4/mkspecs/win32-crossCompil-g++" -D__GNUC__ -DWIN32 src/Form.h
-o build/moc_Form.cpp
i586-mingw32msvc-g++ -c -O2 -Wall -mthreads -DUNICODE -DQT_LARGEFILE_SUPPORT
-DQT_NO_DEBUG -DQT_GUI_LIB -DQT_CORE_LIB -I"/usr/local/qt4-win32/include/QtCore"
-I"/usr/local/qt4-win32/include/QtCore" -I"/usr/local/qt4-win32/include/QtGui"
-I"/usr/local/qt4-win32/include/QtGui" -I"/usr/local/qt4-win32/include" -I"."
-I"src" -I"build" -I"build" -I"/usr/i586-mingw32msvc/include"
-I"/usr/share/qt4/mkspecs/win32-crossCompil-g++" -o build/moc_Form.o
build/moc_Form.cpp
i586-mingw32msvc-g++ -mthreads -Wl,-enable-stdcall-fixup
-Wl,-enable-auto-import -Wl,-enable-runtime-pseudo-reloc -mwindows -Wl,-s
-o "release/testCrossCompilation.exe" build/Form.o build/main.o build/moc_Form.o
-L"/usr/local/qt4-win32/lib" -lQtGui4 -lQtCore4
make[1]: quittant le répertoire « /tmp/testCrossCompilation »
```

Pour que votre programme soit redistribuable facilement sous windows sans avoir besoin d'installer Qt4 et mingw, copiez les bibliothèques dynamiques dans le répertoire où est stocké l'exécutable.

```
cd release
cp /usr/local/qt4-win32/bin/QtCore4.dll /usr/local/qt4-win32/bin/QtGui4.dll .
gunzip -c /usr/share/doc/mingw32-runtime/mingwm10.dll.gz > mingwm10.dll
```

 **Comment savoir qu'il faut copier QtCore et QtGui ou d'autres modules ? C'est simple.**
Lors de la création de votre projet, vous allez utiliser plusieurs modules. Les plus courants sont les deux citées précédemment. Si vous utilisez QtNetwork, vous devrez copier QtNetwork4.dll. Si vous utilisez QtSvg vous devrez copier QtSvg4.dll et ainsi de suite

 **Si vous avez un message du type "err:module:import_dll Library QtCore4.dll (which is needed by L"Z:\\tmp\\testCrossCompilation\\release\\QtGui4.dll") not found" dans la console wine, c'est que vous avez oublié de copier une dll**

Ensuite lancez le programme. Il devrait vous afficher cette fenêtre






Le programme compilé est redistribuable sous windows sans craintes. Si vous voulez simplifier la vie à vos utilisateurs, distribuez directement le contenu intégral du répertoire release qui contient tout ce qu'il faut pour faire tourner le programme sans devoir installer Qt sur windows.

III - Conclusion

III-1 - Conclusion

Vous pouvez désormais compiler des exécutables windows sous linux et les lancer directement sous linux grâce à wine. La plupart des programmes Qt4 pour windows devraient marcher avec wine, cependant ceux utilisant QtOpenGL auront plus de difficultés à tourner étant donné que ce n'est pas la spécialité de wine. Pour cela, tournez vous vers winex/cedega.

III-2 - Références

-  **Discussion sur les forums**
-  **Cross-compilation vers un environnement Win32**
-  **Cross compiling Qt/Win Apps on Linux**