Quantum GIS(QGIS)の翻訳

by　HaruoTakeuchi@Kernel

ソースコードから QGIS をステップバイステップでビルドする方法

更新日付:20101105　最近の変更日付：20101028

訳註　Apt は Debian のパッケージイングツール（The advanced packaging tool)である。

訳註　依存性は、オブジェクトがオブジェクトに依存して動く意味

訳註　使用するメモリーが大きいための対策かスピードアップの為,

ここでは、コンパイル時間を稼ぐためのようである。

訳註：MrSID はラスターデータ圧縮フォーマット ERDAS 社(現 ESRI 社に統合）が使っている。MrSID は画像用圧縮ソフト ECW もウエイブレット画像圧縮フォーマット Ubuntu はリナックスの最新バージョン

訳註：Visual　Studio はマイクロソフトのグラフィックスパッケージ

訳註：　MinGW は WinAPI のためのフリーコンパイラーで GCC を利用できる。MinGW と MSYS を使えば日本語化がある程度容易である。

訳註：frameworks は Flex や Air のアプリを作るためのアプリ

１．イントロダクション（始めに）

本**手引書**は以下に記述されているソフトウエア Quantum GIS（量子化 GIS）の初めて書かれたインストールに関する**手引書**である。
この**手引書**中のソフトウエアとハードウエアの記述は大部分は商標登録されており、したがって法的な求めに従って記述しているものである。 Quantum GIS は GNU の General Public License に則っている。さらなる情報が必要ならば Quantum GIS のホームページ

http：//www.qgis.org をチェックされたい。

この**手引書**にあらわれている内容が詳細に及ぶが、それは、編集者のもっとも確かな責任に基づいて検証されている。しかしながら、内容に関する過誤は免れないので、あらゆるデータは何らかの信頼性や保障を意味するものではない。
編集者と出版人は過誤やその結論に関して、いかなる責任も信頼も負うものではない
あなたが何らかのエラーを報告することはいつでも歓迎するところである。

この**手引書**は Quantum GIS の一部としての「ユーザーマニアルとインストールガイド」

であり、HTML や PDF のフォーマットで　http://www.qgis.org からダウンロードすることができる。

最新のバージョンに関しては、wiki　にて入手可能であるから、次を参照されたい。

http://www.qgis.org/wiki/Installation_Guide

この**手引書**の翻訳は Quantum GIS　プロジェクトの　http://www.qgis.org　中にある

文書「量子化 GIS プロジェクト」領域で利用可能である。

　また更に情報がほしい場合には

http://wiki.qgis.org/qgiswiki/DocumentationWritersCorner

この URL にあるので、参照されたい。

現在進捗中の情報に関してはメーリングリストがあるので　http://qgis.org　にアクセスしてメーリングリストに参加して、プロジェクトに協力されることをお願いする。

/!\　**文書作成者への注意**：**手引書**書はビルド手続きの中心的な場所に置き、決して削除されないように、お願いします。

/!\　**文書作成者への注意**：この**手引書**書は doc/INSTALL.t2t -　から作られている。もし、あなたがこの**手引書**書を編集したい場合は source ディレクトリーの root にある INSTALL 文書（生成されたものでなく）そのものを編集していただきたい。

２．概要

QGIS は、大部分のプロジェクト（例　KDE4.0）と同様にソースからビルドする場合に CMake (http://www.cmake.org)を使っている。

以下は。ビルドに必要とされる依存性をまとめたものである：
必要なビルドツール：

- CMake >= 2.6.0
- Flex
- Bison

必要なビルドの deps（依存性：dependencies）

- Qt >= 4.4.0
- Proj >= 4.4.x
- GEOS >= 3.0
- Sqlite3 >= 3.0.0
- GDAL/OGR >= 1.4.x
- Qwt >= 5.0

訳註：＞はより大、＞＝は以上、他も同様に解釈

オプションの依存性:

- GRASS のプラグイン - GRASS >= 6.0.0 (ライブラリーは Linux32ビットで例外なくコンパイルされた)
- 地理座標系は-GSL >= 1.8
- postgis サポートと SPIT プラグインについて- PostgreSQL >= 8.0.x

- Gps プラグインについて- expat >= 1.95 と gpsbabel
- マップサーバーエクスポートと PyQGIS について - Python >= 2.3 (2.5+ 推奨)
- Python サポートについて- SIP >= 4.8, PyQt >= Qt バージョンは合致が必要
- Qgis のマップサーバーは - FastCGI を使っている。

３．GNU/Linux へビルドする

3.1. Qt4.x を使った QGIS のビルディング

必要条件：Distro から派生した Debiann 版の Ubuntu

これらの注意事項は distros から派生した Devian 版やほかの版違いの Ubantu ではパッケージ名に若干の変更が必要であろう。

これらの注意書きは QGIS をソースコードからビルドしたいと思う人のためのものであるが、大きな目的の一つは、ここで、バイナリパッケージを使って＊あらゆる＊依存性をどのように解決することができるかを示すことである。

ソースから単に QGIS の核となる部分をビルドするだけのことを述べているのではない。

この方法を著者が採用しているわけは、システムパッケージを管理する業務を残しているが、これは、QGIS をコーディングする我々自身に関する問題であるためである。

この**手引書**は、新規にインストールしようとしているものと仮定しているし、いわゆるクリーンなシステムに導入するするものとしている。したがって、これらの **手引書**がこれらのシステムに習熟している人の場合、詳しすぎるようであり、またそれが煩わしいようであれば、読み飛ばしてもいいだろう。

/!\ **注意:** QGIS を開発する計画がない場合、QGIS をコンパイルしてインストールする多分

もっとも簡単な選択は"Building Debian packages" （Devian パッケージをビルドする）の

章を参照することである。

3.2. Apt を準備する

Qgis のパッケージは Ubantu のコンポーネントである"universe"で利用できるビルドに依拠している。これは。デフォールトで起動するようになっていないので、これをまず起動するようにしなければならない:

１．まず/etc/apt/sources.list file. を編集して、　２．"deb"で始まる全行のコメントを外すことである。もちろん(K)Ubuntu　の 'edgy' もしくはその上位のものが、あらゆる依存性に適合して動いている必要がある。
では、ローカルマシンのソースデータベースを更新しましょう：

```
sudo apt-get update
```

のコマンドを打ち込む。

3.3. ビルドの依存性をインストールする。

Distribution 配布版	install command for packages パッケージに関するコマンドのインストール
hardy ハーデイ	apt-get install bison cmake fcgi-dev flex grass-dev libexpat1-dev libgdal1-dev libgeos-dev libgsl0-dev libpq-dev libqt4-core libqt4-dev libqt4-gui libqt4-sql libsqlite3-dev proj pyqt4-dev-tools python python-dev python-qt4 python-qt4-dev python-sip4 python-sip4-dev sip4

intrepid イントレピッド	apt-get install bison cmake flex grass-dev libexpat1-dev libfcgi-dev libgdal1-dev libgeos-dev libgsl0-dev libpq-dev libqt4-core libqt4-dev libqt4-gui libqt4-sql libqwt5-qt4-dev libsqlite3-dev proj pyqt4-dev-tools python python-dev python-qt4 python-qt4-dev python-sip4 python-sip4-dev sip4
jaunty ジョンテイ	apt-get install bison cmake flex grass-dev libexpat1-dev libfcgi-dev libgdal1-dev libgeos-dev libgsl0-dev libpq-dev libqt4-core libqt4-dev libqt4-gui libqt4-sql libqwt5-qt4-dev libsqlite3-dev proj pyqt4-dev-tools python python-dev python-qt4 python-qt4-dev python-sip4 python-sip4-dev sip4
karmic カルミック	apt-get install bison cmake flex grass-dev libexpat1-dev libfcgi-dev libgdal1-dev libgeos-dev libgsl0-dev libpq-dev libqt4-core libqt4-dev libqt4-gui libqt4-sql libqwt5-qt4-dev libsqlite3-dev proj pyqt4-dev-tools python python-dev python-qt4 python-qt4-dev python-sip4 python-sip4-dev sip4
lenny レニ -	apt-get install bison cmake flex grass-dev libexpat1-dev libfcgi-dev libgdal1-dev libgeos-dev libgsl0-dev libpq-dev libqt4-dev libqwt5-qt4-dev libsqlite3-dev pkg-config proj pyqt4-dev-tools python python-dev python-qt4

	python-qt4-dev python-sip4-dev sip4
lucid ルシッド	apt-get install bison cmake flex grass-dev libexpat1-dev libfcgi-dev libgdal1-dev libgeos-dev libgsl0-dev libpq-dev libproj-dev libqt4-dev libqwt5-qt4-dev libspatialite-dev libsqlite3-dev pkg-config pyqt4-dev-tools python python-dev python-qt4 python-qt4-dev python-sip python-sip-dev
maverick メーベリック	apt-get install bison cmake flex grass-dev libexpat1-dev libfcgi-dev libgdal1-dev libgeos-dev libgsl0-dev libpq-dev libproj-dev libqt4-dev libqtwebkit-dev libqwt5-qt4-dev libspatialite-dev libsqlite3-dev pkg-config pyqt4-dev-tools python python-dev python-qt4 python-qt4-dev python-sip python-sip-dev
sid シッド	apt-get install bison cmake flex grass-dev libexpat1-dev libfcgi-dev libgdal1-dev libgeos-dev libgsl0-dev libpq-dev libproj-dev libqt4-dev libqwt5-qt4-dev libspatialite-dev libsqlite3-dev pkg-config pyqt4-dev-tools python python-dev python-qt4 python-qt4-dev python-sip python-sip-dev
squeeze スクイーズ	apt-get install bison cmake flex grass-dev libexpat1-dev libfcgi-dev libgdal1-dev libgeos-dev libgsl0-dev libpq-dev libproj-dev libqt4-dev libqwt5-qt4-dev libspatialite-dev libsqlite3-dev pkg-config pyqt4-dev-tools python python-dev python-qt4 python-qt4-dev python-sip python-sip-dev

(debian/フォルダー中の各コントロールファイルから抽出される）

/!\ **重要:** もし、この**手引書**に従ってシステムにインストールしようとしている人がすでにQt3の開発ツールをインストールしている場合には、Qt3ツールとQt4ツールとの間で競合が起こることがある、たとえば、qmake が、Qt4ではなくQt3をポイントしているとしよう、そうすればQt4版のUbuntuはQt3版のパッケージが指示されたと思い、相互に隣で動き続けることになるためである。

この意味は例として仮に両者をインストールしていると結局3つのqmakeを動かしていることになる。

例：

```
/usr/bin/qmake -> /etc/alternatives/qmake

/usr/bin/qmake-qt3

/usr/bin/qmake-qt4
```

同様のことがほかのQtのバイナリーコードに関しても言えることであり、いわゆるカノニカル（正統的な）qmakeは　apt の代用品によって管理されている。したがって、QGISのビルドを開始する前にQt4をデフォルトとして作っておく必要がある。　将来Qt3にデフォルトとして戻るためには、ここで述べている処理過程を使うことができる。

この問題を修正するために、この　apt　の代替品を使うことができる。だから、Qt4版のアプリケーションはあらゆる場合について使用できる：

```
sudo update-alternatives --config qmake

sudo update-alternatives --config uic
```

```
sudo update-alternatives --config designer

sudo update-alternatives --config assistant

sudo update-alternatives --config qtconfig

sudo update-alternatives --config moc

sudo update-alternatives --config lupdate

sudo update-alternatives --config lrelease

sudo update-alternatives --config linguist
```

この単純なコマンドラインをダイアログの中で使えば、上記コマンドが各々起動した後では、各コマンドは対応するアプリケーションで、Qt4版を選択することになる。

/!\ **注意:** python 言語でビルドするには SIP 4.5 以上 と PyQt4 4.1 以上が必要である。いくつかの安定した GNU/Linux 版（例えば Debian または SuSE など）では SIP 4.5よりした。PyQt4は 4.1.より下のみが提供されている。

Python 言語のビルドを含むサポートを行うには、これらのパッケージをソースコードからビルドするなり、インストールする必要がある。

3.4. ccache(オプション）のセットアップ

コンパイルの時間をスピードアップするためには ccache(c キャッシュ)を勿論セットアップしなければならない。

```
cd /usr/local/bin
```

```
sudo ln -s /usr/bin/ccache gcc

sudo ln -s /usr/bin/ccache g++
```

3.5. 開発環境を準備する

開発を便利に進めるために、すべての開発作業を　$HOME/dev/<language>　このディレクトリーで行う。このケースでは C++の開発作業のための開発環境を次のように作ることになる：

```
mkdir -p ${HOME}/dev/cpp

cd ${HOME}/dev/cpp
```

このディレクトリーパスは以下のあらゆる説明に関して共通に仮定されている。

3.6. QGIS のソースコードをチェックアウトする

ソースコードをチェックアウトするには2つの方法がある。リポジトリーにある QGIS のソースコードの編集権限がない場合には匿名ユーザーとしての方法を使う、もしくは、ソースコードの許諾された改変権がある場合には開発者としてのチェックアウトを実施できる。

１．匿名ユーザーとしてのチェックアウトの方法

```
cd ${HOME}/dev/cpp

svn co https://svn.osgeo.org/qgis/trunk/qgis qgis
```

２．開発者としてのチェックアウト方法

```
cd ${HOME}/dev/cpp

svn co --username <yourusername> https://svn.osgeo.org/qgis/trunk/qgis qgis
```

まず最初のソースチェックアウトのためには、qgis.org の認証を受け入れる旨の応答をしなければならない。　恒久的に（permanently)認証を受けるには「P」を押し下げて返答することである。

```
Error validating server certificate for 'https://svn.qgis.org:443':

 - The certificate is not issued by a trusted authority. Use the

   fingerprint to validate the certificate manually!  Certificate

   information:

 - Hostname: svn.qgis.org

 - Valid: from Apr  1 00:30:47 2006 GMT until Mar 21 00:30:47 2008 GMT

 - Issuer: Developer Team, Quantum GIS, Anchorage, Alaska, US

 - Fingerprint:

   2f:cd:f1:5a:c7:64:da:2b:d1:34:a5:20:c6:15:67:28:33:ea:7a:9b (R)eject,

   accept (t)emporarily or accept (p)ermanently?
```

以上のサーバー応答の内容を紹介すれば

```
Error validating server certificate for 'https://svn.qgis.org:443':
```

'https://svn.qgis.org:443':の URL とポート番号に関してエラー評価サーバーの認証

- The certificate is not issued by a trusted authority. Use the

fingerprint to validate the certificate manually!　Certificate

information:

- 認証は何らかの機関によって行われるものではない。確かに人間の手で認証されている Fingerprint を使用する。

- Hostname: svn.qgis.org

ホスト名をこたえる。

- Valid: from Apr　1 00:30:47 2006 GMT until Mar 21 00:30:47 2008 GMT

認証開始日時

- Issuer: Developer Team, Quantum GIS, Anchorage, Alaska, US

認証ユーザー　米国　アラスカ州　アンカレッジ市　Quantum　GIS 開発者チーム

- Fingerprint:

割り当てられた認証指紋

2f:cd:f1:5a:c7:64:da:2b:d1:34:a5:20:c6:15:67:28:33:ea:7a:9b (R)eject,

accept (t)emporarily or accept (p)ermanently?

R は拒否、T は一時的に受け入れ　P は恒久的に受け入れ

3.7. コンパイルの開始

著者の場合には、開発版の QGIS を~/apps ディレクトリーにいれてコンパイルしているが、これは　/user の下にあるであろう Ubuntu のパッケージと競合しないようにしたものである。この例として使った方法は Qgis のバイナリーパッケージを使うことができるが、これは、自分のシステムを開発版と一緒においてあるものである。著者も同様である言いたい。

```
mkdir -p ${HOME}/apps
```

ここで、ビルドディレクトリーを作って、ccmake を動かすことにする：

```
cd qgis

mkdir build

cd build

ccmake ..
```

Ccmake を動かすと（注意... ccmake が必要）メニューがあらわれて、そこでビルドのさまざまな局面を構成してやることができる。

もし、ルート権限がなかったり、または既存の QGIS　インストール（たとえばパッケージマネージャーによって作られた）を上書きしたくない場合には、CMAKE_BUILD_PREFIX（Cmake　ビルドのプレフィックス）を書き込み権限あるどこかに設定する必要がある（著者がいつも使っているのは　/home/timlinux/apps　である)。

ここで、「ｃ」　押せば構成を選択でき　「e」を押せばあらわれるいかなるエラーメッセージも無視される。また 「g」を押せば make files を生成できる。

注意すべきことは　「ｇ」オプションが利用できるようになる前に「ｃ」を何回かオス必

要がある。　「g」の生成が完了した後で、「ｑ」を押して ccmake　のインタラクティブ（対話的な）な状態から抜け出せる。
ここでビルドを使って次のコマンドを入力する。

```
make

make install
```

この処理は若干の時間がかかるがそれは、インストールするプラットフォーム（PC）に依存する。

3.8.Debian パッケージをビルドする。

個人的なインストールを行う代わりに前の段階で debian パッケージを勿論作っている。

これは、qgis のルートディレクトリーで行われているから、そこで、debian ディレクトリーを見つけることができる。

まず最初に debian のパッケージングツールを一緒にインストールする必要がある。

```
apt-get install build-essential
```

まず、入手したバージョンに関するチェンジログエントリーを1つ作る必要がある。

Lucid の場合には：

```
dch -l ~lucid　--force-distribution --distribution lucid "lucid build"
```

QGIS パッケージは次のようにして作られる。

```
dpkg-buildpackage -us -uc -b
```

/!\　**注意:** 仮に　dpkg-buildpackage　のコマンドがビルド依存性が見つからないと行って

来たら、apt-get を使って、それらをインストールしてからもう一度コマンドを実行すればよい。

/!\ **注意:** もし、libqgis1-dev がインストール済みであれば、最初の dpkg -r libqgis1-dev を使えるので、それをわざわざ削除する必要がない。

パッケージは（すなわち1段上の）親ディレクトリーに作られる。それらを dpkg を使ってインストールすればよい。
例えば：

```
sudo debi
```

3.9. QGIS を起動する

ここで、QGIS を起動することができる。

```
$HOME/apps/bin/qgis
```

仮に、すべての作業が正しく行われていれば QGIS アプリケーションが起動して画面上に現れる。

3.10. 実際の例 :ECW と MrSID フォーマットを扱える QGIS と GRASS をソースからビルドする。

以下の手続きは Ubuntu 8.04, 8.10と9.04 32ビット版で実験されている。もし、異なる版のソフトウエア（gdal, grass, qgis）を使う場合は必要な改変を以下のコードに実施する必要がある。 このガイドは gdal grass qgis の以前の版を一つもインストールしたことがないということを前提としている。

3.10.1. ステップ１ ：インストールベースのパッケージ

まず最初にソースコードをダウンロードしそれをコンパイルするのに必要なパッケージをインストールしなければならない。

```
sudo apt-get install build-essential g++ subversion
```

3.10.2. ステップ２　ecw ライブラリーをコンパイルしてインストールする。

ERDAS のサイトである http://www.erdas.com/ を訪れ以下のリンクをたどって、

""products --> ECW JPEG2000 Codec SDK --> downloads""

次の""Image Compression SDK Source Code 3.3"" （これには登録を行いライセンスを取得する必要がある）をダウンロードする必要がある。 また適切な場所にあるアーカイブを解凍する必要もある。（このガイドの過程を守っていればすべてのダウンロードされたソースコードはユーザーの home に置かれているはずである）

その後、新たに作ったフォルダーへ移動することになる。

```
cd /libecwj2-3.3
```

コードを標準的なコマンドでコンパイルする

```
./configure
```

そして

```
make
```

更に

```
sudo make install
```

leave the folder

```
cd ..
```

3.10.3. ステップ３：MrSID のバイナリーをダウンロードする

LIZARDTECH 社のウエッブサイト http://www.lizardtech.com/ へ行って、以下のリンクをたどり "'download --> Developer SDKs"'' 次の"''GeoExpress SDK for Linux (x86) - gcc 4.1 32-bit"'' (登録を行いライセンスを受ける必要がある） をダウンロードする。

ダウンロードファイルを解凍して次の名前によく似ているフォルダー名のディレクトリーが生成されている。"Geo_DSDK-7.0.0.2167"

3.10.4. ステップ４：gdal ライブラリーのコンパイルとインストール

最新の gdal ソースコードをダウンロードする。

```
svn checkout https://svn.osgeo.org/gdal/trunk/gdal gdal
```

次に MrSID のバイナリーフォルダーから２，３のファイルを gdal ソースコードのあるフォルダーへコピーする。

(この時に"USERNAME" を実際のアカウントの自身のユーザーネームに置き換える）

```
cp /home/USERNAME/Geo_DSDK-7.0.0.2167/include/*.*
/home/USERNAME/gdal/frmts/mrsid/
```

Gdal ソースコードのフォルダーへ移って

```
cd /gdal
```

configure コマンドを２， ３の特定のパラメーターを使って起動する

```
./configure --without-grass --with-mrsid=../Geo_DSDK-7.0.0.2167 --without-jp2mrsid
```

コンフィグレーション処理の最終過程で以下のような文言があらわれる

```
...

GRASS support:                  no

...

...

...

ECW support:                    yes

MrSID support                   yes

...
```

これで、コンパイルが正常に終了したことがわかる。

```
make
```

次に

```
sudo make install
```

もっとも最近のライブラリーに必要なリンクを作ることによって処理過程を終える。

```
sudo ldconfig
```

この時点で、仮に gdal が MrSID と ECW とをつかって正しくコンパイルされているかチェックできるが、それは以下のコマンドの一つ（もしくは両方）のコマンドを使うことによって実行される

```
gdalinfo --formats | grep 'ECW'

gdalinfo --formats | grep 'SID'
```

フォルダーを移る。

```
cd ..
```

3.10.5. ステップ５：GRASS のコンパイルとインストール

GRASS のソースコードをダウンロードしコンパイルする前に２，３のほかのライブラリやプログラムをインストールする必要がある。ここでは apt を通じてこれを行う。

```
sudo apt-get install flex bison libreadline5-dev libncurses5-dev lesstif2-dev debhelper
dpatch libtiff4-dev \
tcl8.4-dev tk8.4-dev fftw-dev xlibmesa-gl-dev libfreetype6-dev autoconf2.13
autotools-dev \
libgdal1-dev proj libjpeg62-dev libpng12-dev libpq-dev unixodbc-dev doxygen fakeroot
cmake \
python-dev python-qt4-common python-qt4-dev python-sip4 python2.5-dev sip4
libglew1.5-dev libxmu6 \
libqt4-dev libgsl0-dev python-qt4 swig python-wxversion python-wxgtk2.8 libwxgtk2.8-0
libwxbase2.8-0 tcl8.4-dev \
tk8.4-dev tk8.4 libfftw3-dev libfftw3-3
```

この時点で GRASS のソースコードを入手しよう。Svn を通じてダウンロードを希望すればそれもできるが、ほしければ、ソースコードアーカイブから最新版をダウンロードして来ることもできる。

例えば GRASS 6.4rc4　は次でダウンロード可能である。

http://grass.itc.it/grass64/source/grass-6.4.0RC4.tar.gz

アーカイブを解凍して新たに作成した新しいフォルダーに移していくつかのパラメーターを使って configure を起動する。

```
CFLAGS="-fexceptions" ./configure --with-tcltk-includes=/usr/include/tcl8.4
--with-proj-share=/usr/share/proj --with-gdal=/usr/local/bin/gdal-config \
--with-python=/usr/bin/python2.5-config
```

追加する　gcc のオプション　-fexceptions　は GRASS ライブラリーの例外処理を機能させるために必要なものである。　実際、現在のところ、QGIS が GRASS ライブラリーを参照しているときに膨大なエラーが起こることを回避する唯一の方法であることがわかっている。勿論、http://trac.osgeo.org/grass/ticket/869　を参照するべきである。

したがって通常は（しばらくの間）次のコマンドを動かす。

```
make
```

And

```
sudo make install
```

これでディレクトリーフォルダーを移動する

```
cd ..
```

これで、GRASS（もちろん新しい wxpyhton のインターフェースを持っているもの）がコンパイルされインストールされたことになる。

```
grass64 -wxpython
```

3.10.6. ステップ６：QGIS のコンパイルとインストール

ここで、QGIS のソースコードを異なるソースから得る必要があるので、たとえば、svn もしくは、次の　http://www.qgis.org/download/sources.html　　サイトからソースコードアーカイブを探して、利用できるものをダウンロードすることによって得られるであろう。

例えば、QGIS　1.1.0　のソースコードは、ここ

http://download.osgeo.org/qgis/src/qgis_1.1.0.tar.gz

で得られる。
このアーカイブを解凍して、新しく作ったフォルダーへ移り

```
cd /qgis_1.1.0
```

それから、ccmake　を走らせる

```
ccmake .
```

ここで、"GRASS_PREFIX"　パラメーターを手動で構成するのに必要なオプションリストがあらわれた時に"c"を押し下げればよい。

スクロールを下げて"GRASS_PREFIX" が現れたらリターンキーを押して、次のように手動でセットすればよい。

```
/usr/local/grass-6.4.0RC4
```

再びリターンキーをもう一度押し下げ、

"c" キーをもう一度押し下げ、更に　"Press [g] to generate and exit" これがあらわれたら

"g" キーを押し下げて、生成して、出ることになる。

そして、いつものように（若干時間がかかるが）次のコマンドを起動する

```
make
```

更に　インストールを実行する。

```
sudo make install
```

処理の最後に当たって、QGIS と GRASS とが MrSID と ECW のラスターフォーマットのサポートができるようになっている。

QGIS を走らせるためには次のコマンドを実行する

```
qgis
```

４．Windows でビルドする。

4.1. Microsoft の Visual Studio をつかってビルドを行う。

この節では Windows 上で Visual Studio を使って QGIS をどのようにしてビルドするかを記述するものである。これは勿論バイナリー QGIS パッケージが現在つくれているものである(以前のバージョンでは MinGW をつかって作られていた)。

この節では QGIS をビルドするとき使われるようになる Visual Studio に必要なセットアップを記述している。

4.1．1．Visual C++ のエクスプレス版（Express Edition）

この無料（ただのビールのような）のエクスプレス版のインストーラーは以下のサイトで入手できる。

http://download.microsoft.com/download/d/c/3/dc3439e7-5533-4f4c-9ba0-8577685b6e7e/vcsetup.exe

ここにアクセスしてダウンロードする。オプションの製品は必ずしも必要がない。　この処理の中で、Windows の Vusial Studio 2008 の SDK 群は勿論ダウンロード出来るし、インストールもできる。

勿論、Microsoft Windows Server® 2003 R2 Platform SDK (api のセットアップのために）が必要である。

http://download.microsoft.com/download/f/a/d/fad9efde-8627-4e7a-8812-c351ba099151/PSDK-x86.exe

ただ Microsoft Windows の CoreSDK／ビルド環境（x8632－Bit)が必要なだけである。

訳註 :Win2003が必要との意味ではなくサーバー版を作る意味で必要のように解釈している。

4.1.2. そのほかのツールや依存性について

以下のパッケージをダウンロードし、インストールする。

ツール	ウエッブサイト
CMake	http://www.cmake.org/files/v2.8/cmake-2.8.2-win32-x86.exe
Flex	http://gnuwin32.sourceforge.net/downlinks/flex.php
Bison	http://gnuwin32.sourceforge.net/downlinks/bison.php
SVN	http://sourceforge.net/projects/win32svn/files/1.6.13/Setup-Subversion-1.6.13.msi/download
OSGeo4W	http://download.osgeo.org/osgeo4w/osgeo4w-setup.exe

OSGeo4W は単に出来合いの QGIS の現在のリリースパッケージを提供するだけでなく、高いビルドのトランクになっているので、それをビルドするときに必要とされる依存性の大部分のものを提供するものである。

QGIS ビルドに関して、以下のパッケージを OSGeo4W から（もっとも進化したインストールを選んで）作成している。

- expat
- fcgi
- gdal17
- grass
- gsl-devel
- iconv
- pyqt4
- qt4-devel
- qwt5-devel-qt4
- sip

上記のパッケージが依存するパッケージを選択することになる。

追加的に QGIS はもちろん　インクルードファイル（include file）**unistd.h** が必要であり、これは、通常は Windows には存在しない。これは、GnuWin32\ include　として　Flex/Bison と一緒に提供される。また、これは C++ インストールの VC\ include ディレクトリーにコピーしてやる必要がある。
この**手引書**書の以前の版では上記のすべての依存性に関してどのようにビルドするかという内容をすべて述べている。

4.1.3. CMake を使った Visual Studio のセットアップ

コマンドプロンプトをはじめるにあたって、一定の環境下で VC++と OSGeo4W の両者の変数は以下のバッチファイル（上記のパッケージがデフォールトの場所にインストールされていると仮定して）を生成する。

```
@echo off

path  %SYSTEMROOT%\ system32;%SYSTEMROOT%;%SYSTEMROOT%\ System3
2\ Wbem;%PROGRAMFILES%\ CMake
2.8\ bin;%PROGRAMFILES%\ subversion\ bin;%PROGRAMFILES%\ GnuWin32\ bin
set PYTHONPATH=

set VS90COMNTOOLS=%PROGRAMFILES%\ Microsoft Visual Studio
9.0\ Common7\ Tools\
call "%PROGRAMFILES%\ Microsoft Visual Studio 9.0\ VC\ vcvarsall.bat" x86

set INCLUDE=%INCLUDE%;%PROGRAMFILES%\ Microsoft Platform SDK for
Windows Server 2003 R2\ include
set LIB=%LIB%;%PROGRAMFILES%\ Microsoft Platform SDK for Windows Server
2003 R2\ lib

set OSGEO4W_ROOT=C:\ OSGeo4W
```

```
call "%OSGEO4W_ROOT%\ bin\ o4w_env.bat"



@set GRASS_PREFIX=c:/OSGeo4W/apps/grass/grass-6.4.0

@set

INCLUDE=%INCLUDE%;%OSGEO4W_ROOT%\ apps\ gdal-17\ include;%OSGEO4W

_ROOT%\ include

@set

LIB=%LIB%;%OSGEO4W_ROOT%\ apps\ gdal-17\ lib;%OSGEO4W_ROOT%\ lib



@cmd
```

バッチファイルをスタートしてコマンドプロンプト上で svn から qgis-trunk のディレクトリーに移して QGIS ソースコードのチェックアウトを実行する：

```
svn co https://svn.osgeo.org/qgis/trunk/qgis qgis-trunk
```

‘build’ ディレクトリーをどこかで作らなければならない。この場所に、すべてのビルドの出力が生成されることになる。

ここで、cmake-gui を走らせるのは、ソースコードのある場所であり box,は QGIS ディレクトリーの最上段にブラウズされる。

バイナリーをビルドする場所で：作成した‘build’のディレクトリーに box をブラウズしてや

る。

Configure をヒットして構成作業をスタートさせ、また Visual Studio 9 2008と（もともとある）ネイティブコンパイラーを使う、最後に Finish をクリックする。

構成作業がさらなる何等の質問もなく完了すれば、Generate をクリックできることになる。

cmake-gui(cmake のグラフィカルユーザーインターフェース）を閉じてコマンドプロンプトを継続し、vcexpress を開始する。その際次のファイルを使う

File / Open / Project/Solutions

次に プロジェクトディレクトリーにある qgis-x.y.z.sln ファイルを開く。
多分ソリューションの構成をデバッグから変更したくなるものである、これ が RelWithDebInfo　であり、デバッグ情報（Debug Info）と共にリリースされるものである。

INSTALL　project　をビルドすることによって、QGIS をインストールするが、デフォルトに従えば、これは、C:￥Program Files￥qgis<version>　(cmake-gui の中の CMAKE_INSTALL_PREFIX の変数を変更することによって可能である)。

勿論必要があれば QGIS のインストールディレクトリーにすべての依存性を DLL に追加するか、もしくは PATH の各々のディレクトリーに加えるかすることができる。

4.1.4. パッケージング

このパッケージングの処理は現在文章になっていないが、次のコマンドを見てもらいたい。

ms-windows/osgeo4w/package.cmd

4.2. MinGW を使ってビルドする

注意: この節は今では"official"（正式の）パッケージをビルドするのに Visual C++を使う今では時代遅れかもしれない。

注意: 自分自身ですべての依存性をビルドする詳しい説明のためには Marco Pasetti のウエッブサイトを訪れたほうがいい。それは次の通りである。

http://www.webalice.it/marco.pasetti/qgis+grass/BuildFromSource.html

事前にビルドしたライブラリーを読むことがもっともシンプルにアプローチする方法である。

4.2.1. MSYS

MSYS は unix 流のビルド環境を windows の下で提供するものである。それにはこれを入手すること：

http://download.osgeo.org/qgis/win32/msys.zip

そして c:¥msys　へ解凍してやることである。Msys 環境を当方で作成した事前に準備したものを使う代わりに、を自分自身で準備したい向きはこの文章のどこかで詳しい説明が提供されている。

4.2.2. Qt

事前にコンパイルされた exe 版のオープンソースの Qt をダウンロードしインストールする。

（ダウンロードしたものを含めて mingw のインストールはここから得られる。）

http://qt.nokia.com/downloads/

ここで、インストーラーが MinGW に関して聞いてきたら、それをダウンロードしたりインストールする必要がなく、単にインストーラーに c:\ msys\ mingw を指摘してやるだけでよい。

Qt のインストレーションが完成したら：

C:\ Qt\ 4.7.0\ bin\ qtvars.bat　を編集して以下のラインを追加すること：

```
set PATH=%PATH%;C:\ msys\ local\ bin;c:\ msys\ local\ lib

set PATH=%PATH%;"C:\ Program Files\ Subversion\ bin"
```

ここで著者が指摘したいことは、windows システムのプレファレンス（preferences）にある Path 変数環境に C:\ Qt\ 4.7.0\ bin\ を追加することである。

仮にいくらかデバッグする計画がある場合は Qt のデバッグ版をコンパイルする必要があるだろう。すなわち次の対象である。

C:\ Qt\ 4.7.0\ bin\ qtvars.bat compile_debug

注意: Qt4.7のデバッグ版をコンパイルするときにある問題がある。それは script が次のメッセージでおわっていることである。 "mingw32-make: *** No rule to make target `debug'. Stop.". したがって、デバッグ版をコンパイルする場合は src ディレクトリーから出て以下のコマンドを実行することである。

```
c:\ Qt\ 4.7.0 make
```

4.2.3. Flex と Bison

Flex を http://sourceforge.net/project/showfiles.php?group_id=23617&package_id=16424

から入手し（zip bin で)それを c:\ msys\ mingw\ bin に解凍してやることである。

4.2.4. Python 関連の物(オプション)

QGIS に組み込んだ Python の組み込み関数を使いたいという場合はこの節を読むといい。

この組み込み関数をコンパイルすることができるようにするためには、SIP と PyQt4とをソースからコンパイルする必要がある。しかも、このインストラーがいくつかの開発ファイルの中には、それが必要であるにも関わらずふくまれていないのである。

4.2.4.1. Windows のインストーラーを使って、Python をダウンロードしてインストールする。

（それをどのフォルダーへインストールしようとも問題ではない）

http://python.org/download/

4.2.4.2. SIP と PyQt4 のダウンロードとソース

http://www.riverbankcomputing.com/software/sip/download

上記のダウンロードフォルダーにある zip ファイルを一時的なデイレクトリーに解凍して

現在の自分のインストールした Qt のバージョンと適合するものを確認して取得するべきである。

4.2.4.3. SIP をコンパイルする

```
c:\ Qt\ 4.7.0\ bin\ qtvars.bat

python configure.py -p win32-g++

make

make install
```

4.2.4.4. PyQt をコンパイルする

```
c:\ Qt\ 4.7.0\ bin\ qtvars.bat

python configure.py

make

make install
```

4.2.4.5. python に関する最終注意

/!\ この解凍した SIP と PyQt4のソースについては、インストールが成功裏に終わった時には、それ以上もはや必要がないので、ディレクトリーごと削除することができる。

4.2.5. 副次的なバージョン（Subversion）

QGIS のソースをリポジトリ（ソースを置いてあるディレクトリー）からチェックアウトするためには

サブバージョンクライアント（Subversion client）が必要となる。

http://www.sliksvn.com/pub/Slik-Subversion-1.6.13-win32.msi

4.2.6. CMake

CMake は Quantum GIS で使われているビルドシステムで、それをここからダウンロードできる。

http://www.cmake.org/files/v2.8/cmake-2.8.2-win32-x86.exe

4.2.7. QGIS

cmd.exe をウインドーズで開始して（Start -> Run -> cmd.exe ）開発用のディレクトリーを作成し、そのディレクトリーに移って次のコマンドを動かす

```
md c:\ dev\ cpp

cd c:\ dev\ cpp
```

SVN からソースをチェックアウトする：

Svn trunk に関しては：

```
svn co https://svn.osgeo.org/qgis/trunk/qgis
```

Svn 1.5 ブランチ（branch）に関しては次の通り

```
svn co https://svn.osgeo.org/qgis/branches/Release-1_5_0 qgis1.5.0
```

4.2.8. コンパイル

背景を理解するためにこの文章の文末にある CMake を使ったビルディングの生成を読むべきである。ウインドウズで cmd.exe をスタートし(Start -> Run -> cmd.exe) 、もし、それ

をすでに持っている場合には、コンパイラーと自分たちの MSYS 環境とにパスを追加することである：

```
c:\ Qt\ 4.7.0\ bin\ qtvars.bat
```

簡単のために、c:\ Qt\ 4.7.0\ bin\ のパスを自分のシステムパスに加える必要がある、それには、単にシステムプロパティで　qtvars.bat　とタイプしてやるだけでよい、勿論、これは Cmd コンソールを開いた上でのはなしである。そこで build ディレクトリーを作りそこを現在のディレクトリーにするわけである。

```
cd c:\ dev\ cpp\ qgis

md build

cd build
```

4.2.9. 構成する（設定する）

```
cmakesetup ..
```

注意： 上記の'..' は必ず含めなければならない。

ここで、'Configure' （設定）ボタンをクリックする。生成するためのソフトとして何を選択するか問われたら 'MinGW Makefiles' を答えなければならない。

ここで、Win2K 上で MinGW Makefiles に若干の問題がある、もしこの PC をプラットフォームとしてコンパイルを実行する場合、'MSYS Makefiles'　を必ず生成用として使わなければならないことである。

すべての依存性はもし、パスが正しく設定されていれば、自動的に選択される。ただ一つ自分で変更しなければならないことは。インストール先(CMAKE_INSTALL_PREFIX) を変更するかあるいは同時に 'Debug'をセットすることである。

NSIS パッケージスクリプトとの両立を図るために、著者が進めるのはインストールプリフィックスをデフォールトの c:\ program files\ のままにしておくことである。

構成、設定作業（configuration）が終了したときに 'OK'をクリックすればセットアップユーテリティを出ることができる。

4.2.10. コンパイルとインストール

```
make make install
```

4.2.11. Qgis をインストールしたディレクトリー (CMAKE_INSTALL_PREFIX)から実行する

ために qgis.exe のバイナリーをインストールしたディレクトリーにすべての.dll をコピーしたかを確認する必要がある、もしそれがまだであれば、QGIS は開始したときにライブラリーがないといって文句をいうはずである。

qgis を実行するときに自分のパスが c:\ msys\ local\ bin と c:\ msys\ local\ lib directories のライブラリが含んでいれば結果、DLL はその場所から使われる。

4.2.12. インストールパッケージを作成する: (オプション)

NSIS を (http://nsis.sourceforge.net/Main_Page)からダウンロードしてインストールする。

今や、ここでウインドーズ explorer を使って QGIS ソースがあるツリーの win_build　ディレクトリーに入る。そこにある README file を読み、その説明に従う。次に qgis.nsi を右クリックして、オプションから'Compile NSIS Script'　（NSIS スクリプトをコンパイルする）を選択する。

4.3. Quantum GIS をコンパイルするための MSYS 環境

4.3.1. 初期設定

4.3.1.1. MSYS(ユニックス風のダイアログボックス)

これは UNIX やウインドーズの世界からたくさんの道具となるソフト（utilities）が提供されている環境である。
ここからダウンロードできる。

http://puzzle.dl.sourceforge.net/sourceforge/mingw/MSYS-1.0.11-2004.04.30-1.exe

これを c:\ msys にインストールする。

すべてのものが、コンパイルされこのディレクトリー（resp. とそのサブディレクトリー）にインストールされる。

4.3.1.2. MinGW

ここからダウンロードできる。

http://puzzle.dl.sourceforge.net/sourceforge/mingw/MinGW-5.1.3.exe

これを、c:\ msys\ mingw のディレクトリーにインストールする。

ダウンロードが大変な場合には g++と mingw-make の成分だけをインストールすればよい。

4.3.1.3. Flex と Bison

Flex と Bison はパーサーを生成するツールである。それらには GRASS やもちろん QGIS のコンパイルにも必要とされているものである。
以下のパッケージをダウンロードする。

http://gnuwin32.sourceforge.net/downlinks/flex-bin-zip.php

http://gnuwin32.sourceforge.net/downlinks/bison-bin-zip.php

http://gnuwin32.sourceforge.net/downlinks/bison-dep-zip.php

これを c:\ msys\ local に解凍する。

4.3.2. 依存性をインストールする

4.3.2.1. 開始準備

Paul Kelly は事前にコンパイルした GRASS ライブラリーのパッケージを準備するという大きな仕事した。

- zlib-1.2.3
- libpng-1.2.16-noconfig
- xdr-4.0-mingw2

- freetype-2.3.4
- fftw-2.1.5
- PDCurses-3.1
- proj-4.5.0
- gdal-1.4.1

これらは、ここからダウンロードすることが可能である。

http://www.stjohnspoint.co.uk/grass/wingrass-extralibs.tar.gz

さらに彼はそれをどうコンパイルするかという注意書きを残してくれた。それがつぎにある興味あるライブラリーである。

http://www.stjohnspoint.co.uk/grass/README.extralibs

この全パッケージを c:\ msys\ local へ解凍しておくことである。

4.3.2.2. グラス（GRASS：フリーの GIS）

訳者注：（The Geographical Resource Analysis Support System；U.S. Army Corp. of Engineers, 1993）は USACERL　U.S. Army Construction Engineering Research Laboratories　GRASS は無償で公開されている GIS で当初ラスター GIS であったが現在はベクターも扱っている。Shape があつかえることでもわかる。また次の講義は GRASS の日本語の講義で http://www.sci.osaka-cu.ac.jp/~masumoto/vuniv2000/はよくできている）

GRASS のソースは CVS から入手できる、すなわち週刊スナップ（weekly snapshot）を見ればいい。

http://grass.itc.it/devel/cvs.php

MSYS のコンソール画面で自分ですでに解凍し、チェックしたソース (たとえば c:\ msys\ local\ src\ grass-6.3.cvs)のあるディレクトリーへ行き

これらのコマンドを実行する。

```
export PATH="/usr/local/bin:/usr/local/lib:$PATH"

./configure --prefix=/usr/local --bindir=/usr/local --with-includes=/usr/local/include

--with-libs=/usr/local/lib --with-cxx --without-jpeg \

--without-tiff --with-postgres=yes --with-postgres-includes=/local/pgsql/include

--with-pgsql-libs=/local/pgsql/lib --with-opengl=windows --with-fftw \

--with-freetype --with-freetype-includes=/mingw/include/freetype2 --without-x

--without-tcltk --enable-x11=no --enable-shared=yes \

--with-proj-share=/usr/local/share/proj

make

make install
```

これは、c:\ msys\ local\ grass-6.3.cvs のディレクトリーにインストールされているべきである。

ところで、以下のページは便利であろう。

- http://grass.gdf-hannover.de/wiki/WinGRASS_Current_Status
- http://geni.ath.cx/grass.html

4.3.2.3. ジオス（GEOS:GNU ライセンスの幾何学計算エンジン）

ソースをダウンロードする。

http://geos.refractions.net/geos-2.2.3.tar.bz2

これを、たとえば c:\ msys\ local\ src　に解凍する。

これをコンパイルするために、ソースにパッチを当てなければならない。ファイルの中の source/headers/timeval.h　の13行を次から

```
#ifdef _WIN32
```

次へと変更することである。

```
#if defined(_WIN32) && defined(_MSC_VER)
```

ここで MSYS のコンソールはソースディレクトリーにはいって実行できる。

```
./configure --prefix=/usr/local

make

make install
```

4.3.2.4. SQL ライト（SQLITE：小型で local で使用可能な RDBMS）

自分で事前にコンパイルした DLL を使うことができるので、ソースからコンパイルする必要がない。:
このアーカイブをダウンロードする：

http://www.sqlite.org/sqlitedll-3_3_17.zip

そして、sqlite3.dll を c:\ msys\ local\ lib へコピーする。

それから、このアーカイブをダウンロードする。

http://www.sqlite.org/sqlite-source-3_3_17.zip

そして、sqlite3.h を c:\ msys\ local\ include へコピーする。

4.3.2.5. ジーエスエル（GSL：GNU Scientific Library: グニュー科学ライブラリー)

ソースをダウンロードする：

ftp://ftp.gnu.org/gnu/gsl/gsl-1.9.tar.gz

これを、c:\ msys\ local\ src へ解凍する。

MSYS コンソールからソースディレクトリーにはいって次を実行する

```
./configure

make

make install
```

4.3.2.6. エックスパット（EXPAT）

訳註：C 言語で作られた XML パーサーで James Clark 氏の作、特徴は高速、認証を行わない。イベント方式の API、アプリの組み込みなどが容易。

ソースをダウンロードする。：

http://dfn.dl.sourceforge.net/sourceforge/expat/expat-2.0.0.tar.gz

これを c:\ msys\ local\ src　へ解凍する。

MSYS のコンソールからソースディレクトリーに入って次のコマンドを実行する。

```
./configure

make

make install
```

4.3.2.7. ポストグレ（POSTGRES）

事前にコンパイルされたライブラリーを使っていこう。ダウンロードのために
以下のリンクを使用する。

http://wwwmaster.postgresql.org/download/mirrors-ftp?file=%2Fbinary%2Fv8.2.4%2Fwin32

%2Fpostgresql-8.2.4-1-binaries-no-installer.zip

アーカイブから pgsql の内容を c:\ msys\ local　へコピーする。

4.3.3. クリーンアップ整理

これで、MSYS 環境の準備が整った。ここで　c:\ msys\ local\ src にあるすべてのものを削除することができる。　それは非常に膨大な容量であり、また以後必要がないためである。

5. マック OSX(１０）: frameworks と Cmake を使ってビルドを実行する

このやり方において、著者が、どこでもむしろ可能な限り frameworks を使うことによって可能な限りソースからくるビルディングの依存性を避けていくようにした。

ここの、基本となるシステムは Mac OS X 10.4 (Tiger、虎)　であり、単一構造のビルドを

使うことにする。また、ここには、Mac OS X 10.5 (Leopard 豹）や 10.6 (Snow Leopard 雪豹)などにビルドする際の若干の注意書きをしている。

一般的な端末(terminal)の使い方に関する一般的な注意は：著者が端末（terminal)のフォルダーに "cd" と言った時は"cd" (引用符なしで単純に cd とタイプしてその後ろにスペースを入れる）そしていわゆるフォルダーのパスをタイプしてそれからリターンキーを入力することである。

このことを知らないでこれをする単純な方法は"cd"の後ろにフォルダーをドラックすることである（ウィンドウズのタイトルバーにあるアイコンを使うかウインドウの中からフォルダーをドラッグするかである）これをディスクトップから端末(terminal)へドラックしたらリターンキーをたたくことである。

併行コンパイル：マルチ cpu と multi コアの Mac では、コンパイルのスピードを上げることが可能である、しかしこれは自動で行われるものではなく、いつでも"make" (しかし"make install"ではなく）とタイプしてやらなければならない、タイプする代わりに次を行う方法もある：

```
make -j [n]
```

[n] をコアの数で置き換え、もしくは自分のマックの CPU の数で置き換える。最近のモデルではハイパースレッド CPU を使っているので、物理的 CPU の数やコアの数の2倍となるであろう。

ie: Mac Pro "8 Core" model (2 quad core processors) = 8

例えば Mac Pro "Core" モデル（２クアッドコアプロセッサー）＝８　となる。

ie: Macbook Pro i5 (hyperthreading) = 2 cores X 2 = 4

また例えば Macbook Pro i5 (ハイパースレッディング）＝４となる

5.1. .dmg から Qt4をインストールする

Qt-4.4.0. の最小限が最小限必要であるから最新版を入手するように勧めたい。

Snow　Leopard(雪豹）関する注意：仮に Snow　Leopard 上でビルドを実行したい場合には古い32ビットサポートの Qt　カーボンブランチ（Carbon branch）か64ビットサポートの Qt ココアブランチ（Cocoa branch）かを決めなければならない。適切なインストーラーは Qt-4.5.2. と Qt 4.6+の 両者を利用できるココア（Cocoa）が勧められる。

PPC に関する注意：Mac の PPC 上で Qt ココア（Cocoa）をビルドすることが問題になる。というのは PPC　MAC 上では Qt カーボン（Carbon）が推奨されているためである。

訳註：PPC は PowerPC の略で、PayPerClick ではないようである。

```
http://qt.nokia.com/downloads
```

もしデバッグバージョンの frameworks が望みならば Qt はもちろん dmg でこれらを提供している。これらは非デバッグタイプの frameworks の追加として入手できる。

ひとたび、ダウンロードして、かつ dmg を開きインストーラーを実行すればよい、注意しなければならないのはインストールに際しては管理者権限がなければならない。

Qt に関する注意：Qt 4.4で開始するには libQtCLucene が追加され、Qt4.5で開始するには

libQtUiTools が追加された、両者ともに/usr/lib　に存在している。この時に、システムの SDK がこれらを使おうとすると、ライブラリーがみつからないであろう。この問題を解決するために、symlinks を/usr/local に追加する必要がある：

次を実行する。

```
sudo ln -s /usr/lib/libQtUiTools.a /usr/local/lib/

sudo ln -s /usr/lib/libQtCLucene.dylib /usr/local/lib/
```

これらのことは、Leopard もしくはそれ以上であれば、自動的にみつかるはずである。それ以前の Mac システムではそれを手伝ってやる必要がある、そのために cmake をビルドするときに'-L/usr/local/lib を

CMAKE_SHARED_LINKER_FLAGS

CMAKE_MODULE_LINKER_FLAGS　と

CMAKE_EXE_LINKER_FLAGS　とを加えてやることである。

5.2. QGIS 依存性（dependencies）に関するフレームワークス（frameworks）をインストールする。

ウイリアムキングスベリーのすぐれたイメージライブラリー GDAL からダウンロードして PROJ、GEOS, GDAL, SQLite3を含んでいる完全パッケージとフレームワークスとしてのイメージライブラリーを入手する。勿論この中には GSL framework がある。

http://www.kyngchaos.com/wiki/software/frameworks

これを、一度、ダウンロードしフレームワークスをインストールする。

ウイリアム（William)はもう一つの追加的なインストーラーパッケージを提供しているが、

これは Postgresql （PostSQL サポート）のためのパッケージである。

Qgis はまさに Libpq のクライアントライブラリーを必要としている。だからもしこれがな

ければ、完全版の Postgres +と PostGIS server をインストールする必要があった。

必要なものはクライアントパッケージだけであるが、それが、ここで入手可能である。

http://www.kyngchaos.com/wiki/software/postgres

勿論 GRASS アプリケーションも利用可能である。

http://www.kyngchaos.com/wiki/software/grass

5.2.1. 追加的な依存性: 全体的な互換性に関する注意

いくつかの追加的な依存性が存在するそれは書き込み時点の依存性であるが、これらは

frameworks としてもインストーラーとしても提供されていない、したがって、これらの依

存性はソースからビルドしてやる必要がある。

もし、Qgis の６４ビットアプリケーションをビルドしたい場合、依存性として、64ビット

サポートを生成する適切なコマンドを提供してやらなければならない。それは、64ビット

の Snow　Leopard （雪豹）に32ビットサポートをいれるのと同様の処理をする必要があ

る。デフォルトのシステム構成を上書きして、それは64ビット版であったが、各々の依存

性パッケージに関する説明に従って、上書きしてやることである。
安定したリリースバージョンが選択されている。ベータ版やそのほかの開発版は多分問題があるだろうから、自分自身で、それらのことをしなければならない。

5.2.2. 追加的な依存性: Expat(エックスパット)

訳註：C 言語で作られた XML パーサーで James Clark 氏の作、特徴は高速、認証を行わない。イベント方式の API、アプリの組み込みなどが容易（再掲載）。

Snow Leopard に関する注意：Snow　Leopard　は便利な expat が含まれているので、このステップは Snow　Leopad では必要がない。

expat のソースを入手する。

```
http://sourceforge.net/project/showfiles.php?group_id=10127
```

ソースの tarball をダブルクリックして Terminal.app に解凍し、その後、ソースフォルダーへ cd を使って移動する：

```
./configure

make

sudo make install
```

5.2.3. 追加的な依存性: Python(パイソン言語)

訳者注：Python 言語はオランダ人のグイド・ヴァンロッサムが作ったオープンソースのプログラミング言語)

Leopard と Snow Leopard に関する注意：Leopard と Snow Leopard とには便利な Python 2.5 と 2.6とが各々含まれている。

したがって、Leopard と Snow Leopard には Python をインストールする必要がない。

もし、必要ならば、python.org から Python をインストールすることができる。

もし仮に、python.org からインストールする場合は、少なくとも最新の Python 2.x より以上を確めてインストールすることである。

http://www.python.org/download/

Python 3は大規模であるが、これも互換性の確保のためである。したがってこのインストールは自己責任で行うしかない。

5.2.4. 追加的依存性: SIP

訳者注：Session Initiation Protocol のいみであろう。

次のサイトから python の組み込みツールキット SIP を入手する。

```
http://www.riverbankcomputing.com/software/sip/download
```

ソースの tarball をダブルクリックして Terminal.app に解凍し cd を使ってソースフォルダーに移動し（これはデフォールトによってこれをインストールして Python framework へ変更する。さらに、これは python.org に関する唯一適切な Python のインストールである。

```
python configure.py

make

sudo make install
```

Leopard に関する注意：

仮に Leopard にこれに組み込まれた Python を使ってシステムパスに SIP をビルドしたいとするーこれは決していい考えではないが、次のコンフィギャーコマンドをベーシックなコマンドに代わってインストールしてやることである。

```
python configure.py -n -d /Library/Python/2.5/site-packages -b /usr/local/bin \

-e /usr/local/include -v /usr/local/share/sip -s MacOSX10.5.sdk
```

Snow Leopard に関する注意：

Leopard と同様にシステムの Python パスの外にインストールしなければならない。勿論自分のほしい構成をはっきりさせておく必要がある(少なくとも SIP 4.9は必要である）各バージョンの python バイナリー（これが'arch'コマンドにと応答する、それは python がないためである）の実行を確認しなければならない。もし自分が32-bit Qt (Qt Carbon)を使うばあいである:

```
python2.6 configure.py -n -d /Library/Python/2.6/site-packages -b /usr/local/bin \

-e /usr/local/include -v /usr/local/share/sip --arch=i386 -s MacOSX10.6.sdk
```

64ビットの Qt(Qt Cocoa)に関して次の設定用の各行を使う：

```
python2.6 configure.py -n -d /Library/Python/2.6/site-packages -b /usr/local/bin \

-e /usr/local/include -v /usr/local/share/sip --arch=x86_64 -s MacOSX10.6.sdk
```

5.2.5. 追加 依存性：PyQt

訳者注：TROLLTECH という会社の開発した、主に Linux 界でよく使われている GUI 部

品等を持ったツールキット

Qt に関する python の組み込みツールキットを以下より入手する

```
http://www.riverbankcomputing.com/software/pyqt/download
```

ソースの tarball をダブルクリックして Terminal.app へ解凍し、cd をつかってソースフォルダーへ移動してそしてデフォルトを使って Python framework をインストールする、そしてこれが python.org にかんして唯一適切な Python インストールである。

```
python configure.py

yes
```

ここで、構成上のある問題がある。それは、ここで修正される（これは PyQwt のコンパイルを後で行う時に生じる問題である）pyqtconfig.py を編集して qt_dir へと変更してやることである。

```
    'qt_dir': '/usr',
```

そして、コンパイルを継続してインストールを行う（ここは、もしできればだが、併行処理のコンパイルを行う丁度いいところかもしれない）：

```
make

sudo make install
```

Leopard にかんする注意：

もし、Leopard 上でこれに組み込まれた Python を使って PyQt をシステムパスにインストールしたい場合、これは必ずしもいい考えではないが。この構成コマンドをベーシックな構成コマンドに代わって使ってやることになる：

```
python configure.py -d /Library/Python/2.5/site-packages -b /usr/local/bin
```

仮に Leopard の QtOpenGL の中で未定義の符号が発生する問題が発生した場合は、

QtOpenGL/makefile を編集して、未定義の dynamic_lookup を LFLAGS に追加することである。そしてもう一度 make を実行することである。

Snow Leopard に関するノート

Leopard と同様にシステムの Python パスの外にインストールしなければならない。勿論自分が望む構成を特定しておく(最低でも PyQt 4.6）必要がある。更に python バイナリを確かに走らせる必要がある(これが、 'arch' コマンドに応答するはずである、これは pyuic4 に関して重要である、というのは'python' コマンドが応答しないためである)。:

```
python2.6 configure.py -d /Library/Python/2.6/site-packages -b /usr/local/bin --use-arch i386
```

64ビット版の Qt（Qt Cocoa)には次の設定ファイルの各行を使用すればよい。

```
python2.6 configure.py -d /Library/Python/2.6/site-packages -b /usr/local/bin --use-arch x86_64
```

5.2.6.追加の依存性: Qwt/PyQwt

訳者註：Qwt は ノルウェー Trolltech 社から提供される、Qt の GUI アプリケーションフレームワークで、グラフィックスの拡張である。これは、2次元描画ウィジェットを提供する。

GPS のトラッキング機能は Qwt を使っている。いくつかのサードパーティのプラグインは PyQwt を使っている。PyQwt のソースを使って両者を入手することができるサイトは次の

通り：

```
http://pyqwt.sourceforge.net/
```

tarball をダブルクリックして解凍して PyQwt5.2.0（これは Qwt5.2.1を使ってできるものである）を仮定する。通常のコンパイルでは Qwt と PyQwt とを同時に行う、しかし Qwt は静的（ダイナミックではなく）に PyQwt の中にリンクされている、そして Qgis はそれを使うことができない。
ここで必要なことはビルドを分離することである。

まず、最初に qwt-5.2のサブディレクトリーにある qwtconfig.pri を編集して、次にいくつかの設定を変更して、bloated 静的ライブラリーデバッグを得てはならない(もっと悪い場合には、qmake から再設定できない)。スクロールを下って'release/debug mode' のブロックに辿り着いたら、末尾の 'CONFIG +=' の行で'else' ブロックの'debug' を'release'.に変更する。次のようにである。

```
else {
    CONFIG                += release        # release/debug
}
```

さらに、'CONFIG += QwtDll'の行のコメントを外す（ 行頭の#を消す）次のように行う。

```
CONFIG                += QwtDll
```

仮に、Snow Leopard で Qt Carbon の32bit 版をビルドしつつある時には行の末尾に次の一行を加える。

```
CONFIG += x86
```

それを保存して、終了する。

ここで、cd を使って Terminal のサブディレクトリーにある qwt-5.2に移る。

```
qmake -spec macx-g++

make

sudo make install

sudo install_name_tool -id /usr/local/qwt-5.2.1-svn/lib/libqwt.5.dylib \

/usr/local/qwt-5.2.1-svn/lib/libqwt.5.dylib
```

Qwt の共有ライブラリーが/usr/local/qwt-5.x.x[-svn] （x.x はマイナーなバージョンポイントでかつそれは SVN 版である）中にインストールされた。これが QGIS と PyQwt の設定であることを覚えておいたほうが良い、いまや PyQwt に関していまだ Terminal の中にある：

```
cd ../configure

python configure.py --extra-include-dirs=/usr/local/qwt-5.2.1-svn/include \

--extra-lib-dirs=/usr/local/qwt-5.2.1-svn/lib --extra-libs=qwt

make

sudo make install
```

qwt のインストールパスが上記の Qwt　ビルドのパスから変わっていることを確認するべきである。

Snow Leopard に関する注意

もし、Qt Carbon を使う場合、自分でどのような構成をビルドするかを特定しなければなら

ない、そうでなければ、うまく動かない組み合わせのデフォルトを使わざろうえない（すなわち、Carbon Qt にかんして x86_64 はうまく働かない)。ここでは Qt　Cocoa は必要がない。この設定は次のように行う。

```
python configure.py --extra-cflags="-arch i386" --extra-cxxflags="-arch i386" \

--extra-lflags="-arch i386" --extra-include-dirs=/usr/local/qwt-5.2.1-svn/include \

--extra-lib-dirs=/usr/local/qwt-5.2.1-svn/lib --extra-libs=qwt
```

5.2.7. 追加の依存性 Bison(バイソン)

訳者註 :Bison はパーサーを生成するためのプログラムで、パーサーの C プログラムを生成する。

Leopard と Snow Leopard に関する注意: Leopard と Snow Leopard は Bison2.3を含んでいるので、このステップは Leopard と Snow Leopard に関して、読み飛ばすことができる。

bison のこのバージョンは Mac OS X 10.4 ではドフォルトとして利用できるが、あまりにも古く、より最新の物を自分のシステムに導入したいと感じるだろう。

少なくともバージョン2.3以上を次からダウンロードできるだろう。

```
ftp.gnu.org/gnu/bison/
```

今や、/usr/local.Ê　のプレフィックスへそれをビルドしてインストールすることにする。

ソースの tarball をダブルクリックして解凍し cd を使ってソウスフォルダーへ移動して次のコマンドを実行する。

```
./configure --prefix=/usr/local
```

```
make

sudo make install
```

5.3. OSX に CMake をインストールする

（cmake のビルドに関してのみ必要なことである）

最新のソースはここからリリースされている。

```
http://www.cmake.org/cmake/resources/software.html
```

OS X のバイナリーインストーラーが利用できる、しかし推奨されない（2.4バージョンは /usr/local ではなく/usr にインストールされる。また2.6バージョンはやや風変わりなアプリケーションである）ソースをダウンロードする代わりに、ソースの tarball をダブルクリックして、cd　を使ってソースフォルダーに移動する。

```
./bootstrap --docdir=/share/doc/CMake --mandir=/share/man

make

sudo make install
```

5.4. OSX に関するサブバージョンをインストールする

Leopard と Snow Leopard に関する注意: Leopard と Snow Leopard (Xcode 3+) には SVN が含まれている。したがってこのステップは Leopard と Snow Leopard では読み飛ばすことができる。

この [http://sourceforge.net/projects/macsvn/MacSVN]のプロジェクトにはダウンロード可

能な svn のビルドが含まれている。

```
curl -O http://ufpr.dl.sourceforge.net/sourceforge/macsvn/Subversion_1.4.2.zip
```

ひとたび、zip ファイルをダウンロードして開いてインストーラーを実行する。

勿論、同じ http://sourceforge.net/projects/macsvn/.サイトからバークレー版の DB

（BerkleyDB）をインストールすることができる。

ここでファイルを書き出ことはここで行われる。

```
curl -O http://ufpr.dl.sourceforge.net/sourceforge/macsvn/Berkeley_DB_4.5.20.zip
```

また、ひとたびこれを解凍してここでインストーラーを実行する。

最後に、次を確認する必要がある。すなわち svn のコマンドライン（commandline）が

このパスで実行可能であるかどうかである。

```
sudo vim /etc/bashrc
```

そして次の行を文末に追加して保存し終了する。

```
export PATH=/usr/local/bin:$PATH:/usr/local/pgsql/bin
```

/usr/local/bin をパスの最初に置かなければならない、その結果新しい bison(ソースからさら

にビルドすることができるので) 古い bison の前に発見される。これはとても古く MacOSX

によってインストールされていたものである。

ここで、一旦クローズして、再起動すれば、自分の shell が更新されたバージョンになって

いることがわかる。

5.5. SVN から QGIS をチェックアウトする

いま、ここで、QGIS に関するソースをチェックアウトしよう。まず最初にワーキングする

ディレクトリーを作る(もしくは自分で選択するフォルダー)

```
mkdir -p ~/dev/cpp cd ~/dev/cpp
```

ここでソースをチェックアウトしよう。
トランクは：

```
svn co https://svn.osgeo.org/qgis/trunk/qgis qgis
```

バージョン x.y.z（枝番）となるリリースについて：

```
svn co https://svn.qgis.org/qgis/branches/Release-x_y_z qgis-x.y.z
```

まず最初に QGIS のソースをチェックアウトしよう。たぶん次のようなメッセージをえるであろう。

```
Error validating server certificate for 'https://svn.qgis.org:443':

- The certificate is not issued by a trusted authority. Use the fingerprint to

   validate the certificate manually!   Certificate information:

- Hostname: svn.qgis.org

- Valid: from Apr   1 00:30:47 2006 GMT until Mar 21 00:30:47 2008 GMT

- Issuer: Developer Team, Quantum GIS, Anchorage, Alaska, US

- Fingerprint: 2f:cd:f1:5a:c7:64:da:2b:d1:34:a5:20:c6:15:67:28:33:ea:7a:9b

   (R)eject, accept (t)emporarily or accept (p)ermanently?
```

私が、勧めたいのは恒久的にアクセプトする意味の'p' を押すことである。

5.6. ビルドを設定す

CMake がソースのビルドからサポートしている、したがって、ビルド処理をつかって、'build' ディレクトリーを作ることにする。OS X は${HOME}/Applications をユーザーアプリケーション（app)の標準的なディレクトリーとして使っている（それはシステムアプリケーション（app)フォルダーは icon になっている)。

もし、正しい許諾された権限を持っていれば、ストレートに自分の /Applications フォルダーにビルドを作ろうとかんがえるであろう。

この手引書では、以下の過程を置いている、すなわち既存の ${HOME}/Applications ディレクトリーの中にビルドを作っていると考えている。

```
mkdir build

cd build

cmake -D CMAKE_INSTALL_PREFIX=~/Applications -D

CMAKE_BUILD_TYPE=Release \

-D CMAKE_BUILD_TYPE=MinSizeRel -D WITH_INTERNAL_SPATIALITE=FALSE \

..
```

これによって、以前にインストールされた frameworks と GRASS のアプリケーション（もしインストールされていれば）とをつかって自動的に発見するだろう。

そうでなければ、GRASS の Unix スタイルのビルドを使うために以下の cmake のインボケーション（invocation：プログラム間通信に使うプログラム）をつかうことになる)（最低

でも Qgis の要求事項として述べられている GRASS バージョンがあり、GRASS にパスとバージョンを代入して使うことになる）

```
cmake -D CMAKE_INSTALL_PREFIX=~/Applications -D
CMAKE_BUILD_TYPE=Release \
-D CMAKE_BUILD_TYPE=MinSizeRel -D WITH_INTERNAL_SPATIALITE=FALSE \
-D GRASS_PREFIX=/user/local/grass-6.4.0 \
..
```

Snow Leopard に関する注意: 32-ビット Qt (Carbon)を扱うために32ビットの python ラッパー（wrapper]スクリプトをつくって、設定に arch フラッグを加える。

```
sudo cat >/usr/local/bin/python32 <<EOF
#!/bin/sh
exec arch -i386 /usr/bin/python2.6 \ ${1+"\ $@"}
EOF

sudo chmod +x /usr/local/bin/python32

cmake -D CMAKE_INSTALL_PREFIX=~/Applications -D
CMAKE_BUILD_TYPE=Release \
```

```
-D CMAKE_BUILD_TYPE=MinSizeRel -D WITH_INTERNAL_SPATIALITE=FALSE \
-D CMAKE_OSX_ARCHITECTURES=i386 -D
PYTHON_EXECUTABLE=/usr/local/bin/python32 \
..
```

ビルディングに関する注意：古い Qt バージョンはいくつかの Qt　プラグインと Qgis に関しては、ある問題がある。

この問題を扱う方法は Qgis のアプリケーションの内側に Qt をバンドルすることである。

これを今行うこともできるし、もしくは Qgis を実行したときに突然すぐにクラッシュするのを見るために待つことである：

勿論、Qt をバンドルすることはいい考えである、もし、Qgis を他の Mac にコピーする必要があるばあいである（ここでは、丁度 Qt をインストールしたように Xcode をインストールする必要がある)。

Qt をバンドルするためには上記の cmake の設定行で以下の行を最終行の前に加えることである。

```
-D QGIS_MACAPP_BUNDLE=1 \
```

5.7. ビルドする

さてここで、ビルド処理を開始できる(最初に述べた、併行処理のコンパイルの注意を覚えているだろうか、もしできればここは丁度いい併行コンパイルのところである)

```
make
```

すべてのビルドがエラーなしで終われば、いよいよそれをインストールできる。

```
make install
```

もしくは、/Applications に関してビルドを実行することができる。

```
sudo make install
```

６．著者と謝辞

以下のひとびとはこの手引書書の作成に寄与している

- Windows　MINGW の章
    - ティム　サットン氏　Godofredo Contreras 2006
    - CMake additions マグナス　ホフマン氏 2007
    - Python additions マーチン　ドビアス氏 2007
    - ティシャム　ダール氏　が初期 の msys environment を作成したことに感謝する
- Windows の MSVC 章(インストールの詳細)
    - デービット　ウイリス氏 2007
    - MSVC の追加インストール　ティム サットン氏 2007
    - PostgreSQL, Qt コンパイル, SIP, Python, AutoExp 追加項　ジャーゲン フィッシャー氏　2007
- Windows MSVC 章 (インストールの単純化)
    - ティム　サットン氏 2007
    - ジャーゲン　フィッシャー氏　2007

- フローリアン　ヒルレン氏　2010

- OSX の章
  - ティム　サットン氏　　2007
  - 特別な感謝を次の両氏に to トム　エルバート氏　とウイリアムキングスバリー氏

- GNU/Linux の章
  - ティム　サットン氏　2006
  - Debian package の章: ジャーゲン　フィッシャー氏 2008