Compiling on Linux

Instructions aimed at Debian/Ubuntu and Fedora/CentOS, but could easily be adjusted to fit other Linuxes, BSDs and Solaris.

1. Get required programs

Debian / Ubuntu / Mint etc

```
$ sudo apt-get install build-essential cmake gettext git-core gpsd gpsd-clients libgps-de
v wx-common libwxgtk3.0-dev libglu1-mesa-dev libgtk2.0-dev wx3.0-headers libbz2-dev libti
nyxml-dev libportaudio2 portaudio19-dev libcurl4-openssl-dev libexpat1-dev libcairo2-dev
libarchive-dev liblzma-dev libexif-dev libelf-dev libsqlite3-dev
```

Note for **Ubuntu Precise (12.04)** users: The wxWidgets 3.0 packages were not officially backported, use the packages from the wxFormbuilder PPA at https://launchpad.net/~wxformbuilder/+archive/ubuntu/wxwidgets (https://launchpad.net/~wxformbuilder/+archive/ubuntu/wxwidgets) using the following command:

```
$ sudo add-apt-repository ppa:wxformbuilder/wxwidgets
```

Fedora / CentOS

```
$ yum install git cmake rpm-build gcc-c++ libstdc++-devel gtk2-devel wxGTK-devel mesa-lib
GLU-devel mesa-libGL-devel gettext bzip2-devel portaudio-devel libarchive-devel liblzma-d
evel libexif-devel
```

OpenSuSE

```
$ zypper install git cmake gcc-c++ libstdc++-devel gtk2-devel wxWidgets-wxcontainer-devel
gettext-runtime gettext-tools libbz2-devel gpsd-devel portaudio-devel tinyxml-devel libcu
rl-devel freeglut-devel rpm-build libarchive-devel liblzma-devel libexif-devel
```

Slackware

see Compiling on Slackware

NOTES :

- libportaudio2 and portaudio19-dev (Ubuntu), may be called libportaudio and libportaudio-dev, or similar in other distros, must be installed, due to the sound code for linux. **Warning**, Google Earth removes portaudio19-dev, without even mentioning it. OpenCPN will compile, but sound will not work!
- From 3.3.117 is it possible to build without using OpenGL at all.
  This means that, for example “libglu1-mesa-dev” (Ubuntu), is not necessary to build a non OpenGL binary.
- **Cmake must be a version > 2.8** , which means Ubuntu 10.4 or later (9.10 with backport repository enabled), or Fedora 13 or later.
- **Important Cmake Note:** Due to a bug in Cmake, the “pixbuf-bug”, it is recommended to upgrade to Cmake version >= 2.8.3.
  Fedora 14 is updated, while on Ubuntu 10.10, it is necessary to enable the backport repository.Tick Synaptic→Settings→Repositories→Updates>Unsupported updates(maverick-backports).
- The libgps API () changed significantly between libgps19 and ..20.
  Since OCPN beta 2.6.1718 we require the libgps20 API () and include files to build OpenCPN, and a “version skew” warning will be produced if they are not found.
  The libgps interface will not function correctly if OpenCPN is built with libgps19 include files.
- Note that these packages will install a significant number of other dependencies as well…

Optional: Remove old installation

Only required if opencpn .deb / .rpm was installed previously

Debian / Ubuntu / Mint etc

```
$ sudo apt-get remove opencpn
```

Fedora / CentOS

```
$ yum remove opencpn
```

OpenSUSE

```
$ zypper remove opencpn
```

2. Download source code

Run this to get your local copy of the source code:

```
$ git clone git://github.com/OpenCPN/OpenCPN.git
```

Git will create a directory called “OpenCPN” containing all the code.

To update an existing local copy, issue the following commands:

```
$ cd OpenCPN
$ git pull
```

3. Build it

We'll build everything in a subdirectory to keep the codebase clean (easier to catch changes).

```
$ cd OpenCPN   //# unless already in this directory.//
$ mkdir build
$ cd build
$ cmake ../
$ make
```

Notes:

- Default install dir is /usr/local, you can change this by providing the appropriate option to cmake:

```
$ cmake -DCMAKE_INSTALL_PREFIX=/usr ../
```

- If you're not into debugging, change to this to generate a smaller and marginally faster binary:

```
$ cmake -DCFLAGS="-O2 -march=native" ../
```

- cmake is only required if you build source code for the first time, then running make is enough, even if you updated the source code.
- Ubuntu 10.10 fails to build…
  Change the path in this line in CMakeCache.txt:

```
GTK2_GTK_INCLUDE_DIR:PATH=/usr/include/gtk-2.0
```

4. Install it or create a package

Make sure to refer to the **Modularized Packaging** chapter of this guide to understand what gets installed/packaged for you and how to influence it. To install your build product on your local machine, simply issue

```
$ sudo make install
```

OR you could do this, to make your own .deb and .rpm packages (requires cmake > 2.8.2):

NOTE: Under Ubuntu 16.04.2 you will need to edit 'CMakeLists.txt' in the main OpenCPN folder around line 73 BEFORE building OpenCPN.

```
 SET (PACKAGE_DEPS "libc6, libwxgtk3.0-0, wx3.0-i18n, libglu1-mesa (>= 7.0.0), libgl1-mes
a-glx (>= 7.0.0), zlib1g, bzip2, libtinyxml2.6.2, libportaudio2")
```

Change 'libwxgtk3.0-0' to read 'libwxgtk3.0-0v5' and 'libtinyxml2.6.2' to read 'libtinyxml2.6.2v5'.

In the build directory issue

```
$ sudo make package
```

You may need to install the GDEBI package installer to install from the DEB package that is made.

5. IDEs for Linux to work on OpenCPN

You can use various IDEs to edit OpenCPN's code on Linux, to have an easy life, choose one of **Code::Blocks (http://www.codeblocks.org/)** , **KDevelop (http://kdevelop.org/)**and **Eclipse (http://eclipse.org/cdt/) CDT (http://eclipse.org/cdt/)**as cmake supports generating their project files. To do so, replace the cmake configuration step with one of the following, corresponding to the IDE of your choice. In all cases you will still be able to do the commandline build as described above.

```
$ cmake -G "CodeBlocks - Unix Makefiles" ../
```

```
$ cmake -G "KDevelop3 - Unix Makefiles" ../
```

```
$ cmake -G "Eclipse CDT4 - Unix Makefiles" ../
```

To prototype the GUI () parts of the application, have a look a **wxFormBuilder (https://sourceforge.net/projects/wxformbuilder/)**.

In order to be able to run OpenCPN from inside the IDE without having it installed, you must copy the following folders from the **data** subfolder of the source tree to your **build** folder: **gshhs**, **s57data**, **tcdata**. You also must create a subfolder **uidata** in the build folder and copy the following files from **src/bitmaps** into it: **styles.xml**, **toolicons_journeyman_flat.png**, **toolicons_journeyman.png**, **toolicons_traditional.png, plus.svg**. You should also copy **authors.html** and **license.html** from **/data** to your 'build' folder. Then to ensure that your 'build' folder is used as the 'source' for the run/debug session you need to ensure that OpenCPN is started with '-p' as a parameter. This sets OpenCPN into 'portable' mode and therefore looks in the location the 'opencpn' executable is run from, i.e. your 'build' directory.

6. Others

Script to make the Git/Cmake process easy.

```
#! /bin/sh
#Change this line to where you want the OpenCPN source on your computer.
cd /home/thomas/Testing/GitOpenCPN
GIT=0
test -d OpenCPN
if test $? -eq 1
  then
#   Sometimes the git port is blocked by a firewall
#   so you can use https if that happens
#    git clone https://github.com/OpenCPN/OpenCPN.git
    git clone git://github.com/OpenCPN/OpenCPN.git
    GIT=1
fi
cd OpenCPN
if test $GIT -eq 0
  then
    git pull
fi
test -d build
if test $? -eq 1
  then
    mkdir build
fi
cd build
rm -f CMakeCache.txt
cmake  ../
make
echo "Cmake OK!"
sudo make install
exit
```

Compiling older releases.

Old way from CVS, no longer maintained

```
$ cvs -z3 -d:pserver:anonymous@opencpn.cvs.sourceforge.net:/cvsroot/opencpn co -P opencpn
```

Earlier releases used the gnu automake toolchain, with the following basic commands:

```
$ aclocal
$ automake --add-missing
$ autoconf
$ ./configure
$ make
$ make install
```

Check on Plugin Availability and Versions

Go to https://launchpad.net/~opencpn/+archive/ubuntu/opencpn (https://launchpad.net/~opencpn/+archive/ubuntu/opencpn) Scroll down to the desired plugin, and look at the version number.

opencpn/developer_manual/developer_guide/compiling_linux.txt Last modified: 2020/01/05 22:02 by jongough