Building libobjc2 + gnustep

Here's the step to build the dependencies of gnustep

1. Install empty VM

We start with a empty Debian9 or FreeBSD 12 VM.

2a. Dependencies under Debian 9

we install these packages under Debian as they are coming up as dependencies anyhow

```
apt-get install dirmngr

apt-key adv --recv-keys --keyserver keyserver.ubuntu.com \
           15CF4D18AF4F7421

apt-get install build-essential git subversion  \
        libpthread-workqueue0 libpthread-workqueue-dev \
        libxml2 libxml2-dev \
        libffi6 libffi-dev\
        libicu-dev \
        libuuid1 uuid-dev uuid-runtime \
        libsctp1 libsctp-dev lksctp-tools \
        libavahi-core7  libavahi-core-dev\
        libavahi-client3 libavahi-client-dev\
        libavahi-common3 libavahi-common-dev libavahi-common-data \
        libgcrypt20 libgcrypt20-dev \
        libtiff5 libtiff5-dev \
        libbsd0 libbsd-dev \
        util-linux-locales \
        locales-all \
        libjpeg-dev \
        libtiff-dev  \
        libcups2-dev  \
        libfreetype6-dev \
        libcairo2-dev \
        libxt-dev \
        libgl1-mesa-dev \
        libpcap-dev \
        libc-dev libc++-dev libc++1 \
        python-dev swig \
        libedit-dev libeditline0  libeditline-dev  readline-common \
        binfmt-support libtinfo-dev \
        bison flex m4 wget \
        libicns1    libicns-dev \
        libxslt1.1  libxslt1-dev \
        libxft2 libxft-dev \
        libflite1 flite1-dev \
        libxmu6 libxpm4 wmaker-common\
        libgnutls30 libgnutls28-dev\
        libpng-dev libpng16-16\
        default-libmysqlclient-dev \
        libpq-dev \
        libstdc++-6-dev \
        libreadline7 libreadline-dev \
```

```
        gobjc-6 gobjc++-6 \
        libgif7 libgif-dev libwings3 libwings-dev libwraster5 \
        libwraster-dev libwutil5 \
        libcups2-dev  libicu57 libicu-dev \
        gobjc++\
        xorg \
        libfreetype6 libfreetype6-dev \
        libpango1.0-dev \
        libcairo2-dev \
        libxt-dev libssl-dev \
        libasound2-dev libjack-dev libjack0 libportaudio2 \
        libportaudiocpp0 portaudio19-dev \
        libstdc++-6-dev libstdc++-6-doc libstdc++-6-pic \
        libstdc++6 wmaker cmake cmake-curses-gui
```

2b. Dependencies under FeeBSD 12

```
pkg install git \
      autoconf \
      automake \
      cmake \
      subversion  \
      wget \
      bash \
      pkgconf \
      sudo \
      gmake \
      windowmaker \
      jpeg \
      tiff \
      png \
      libxml2 \
      libxslt \
      gnutls \
      libffi \
      icu \
      cairo \
      avahi \
      portaudio \
      flite \
      pngwriter \
      mariadb103-client \
      postgresql96-client \
      bash \
      clang\
      lldb
```

3. We install a decent clang compiler.

`clang-8:`

from clang8 repository: Add a file /etc/apt/sources.d/llvm.list with content

```
deb http://apt.llvm.org/stretch/ llvm-toolchain-stretch-8 main
deb-src http://apt.llvm.org/stretch/ llvm-toolchain-stretch-8 main
```

To install you call it like this:

```
apt-get update
apt-get install clang-8 lldb-8 lld-8
```

Note: clang-7 has a bug which makes gnustep-base fail during its ./configure phase.

(Note today clang-8 was no longer in the llvm repo for some reason. Maybe temporary issue. I also tried the bleeding edge clang-9 and that worked similarly)

Under FreeBSD we use the clang compiler which comes with FreeBSD12.

4. Download

We download the sources we need:

```
mkdir gnustep
cd gnustep
wget http://ftp.gnu.org/pub/gnu/libiconv/libiconv-1.15.tar.gz
git clone https://github.com/apple/swift-corelibs-libdispatch
git clone https://github.com/gnustep/scripts
git clone https://github.com/gnustep/make
git clone https://github.com/gnustep/libobjc2
git clone https://github.com/gnustep/base
git clone https://github.com/gnustep/corebase
git clone https://github.com/gnustep/gui
git clone https://github.com/gnustep/back
./scripts/install-dependencies
```

The last step should basically not show any additional dependency we have not already covered. On FreeBSD , the last step has to be skipped as the script is not prepared for FreeBSD yet.

5. libiconv

We compile libiconv. Debian comes standard with version 1.14 but there was an issue in 1.14 which got fixed in 1.15 which was relevant to us so using 1.15 is recommended.

```
tar -xvzf libiconv-1.15.tar.gz
cd libiconv-1.15
./configure --enable-static --enable-dynamic
make
make install
cd ..
```

6. Defaults

We set some environment defaults for the remainder of the session

```
# we set the C and C++ compiler version

export CC=/usr/bin/clang-8
export CXX=/usr/bin/clang++-8
export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
export RUNTIME_VERSION=gnustep-2.0
export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig
export LD=/usr/bin/ld.gold
export LDFLAGS=-fuse-ld=/usr/bin/ld.gold
export OBJCFLAGS="-fblocks"
```

Using the ld.gold linker is important as otherwise strange things happens. old ld does stuff wrong, newer lld from clang optimizes stuff away it shouldnt.
The RUNTIME_VERSION is overriding whatever gnustep-make uses by default (which is 1.8)

7. libdispatch

The libdispatch is an open source library supplied by Apple which takes care of running multiple background tasks. Apple calls it "Grand Central Dispatch". The Swift language also uses it, hence the repo name. We're not really directly using it but other objc code might use it or benefit from it so we compile it as well.

```
cd swift-corelibs-libdispatch
 mkdir build
 cd build
 cmake .. -DCMAKE_C_COMPILER=${CC} \
          -DCMAKE_CXX_COMPILER=${CXX} \
          -DCMAKE_BUILD_TYPE=Release \
          -DUSE_GOLD_LINKER=YES
 make -j40
 make install
 ldconfig
```

8. libobjc2 / gnustep-base

Now we have reached a point where the magic triple libobjc2 + gnustep-make + gnustep-base comes into play. To understand the context here. The objectiveC compiler needs a runtime. The runtime is taking care of the basic functionality of objectiveC objects such as their memory allocation and release and other stuff. The compiler calls the runtime when specific stuff happens. When a method is called, the compiler puts a call into the runtime which then looks up the dynamic methods and calls the method etc. When a block ends and certain objects are no longer needed, the compiler calls the runtime so it can release the objects accordingly.

There are different runtimes around. The original objectiveC runtime supplied by the gcc compiler. It's old, outdated and doesn't deal with automatic reference counting and a lot of other things. So it's not recommended. The libobjc2 is the "modern" rewrite for a runtime and is actually based on a project called Etoile. It's ABI has different variants. There is a fragile and a non fragile ABI. The fragile ABI which was originally used

Read this blogpost to understand what the difference is:
http://www.sealiesoftware.com/blog/archive/2009/01/27/objc_explain_Non-fragile_ivars.html

If you get this is wrong, you might end up with objects no longer using the right position of variables in the data of super or subclasses and things can get totally out of control. So the new non-fragile-ABI is really the way to go. There are also different versions of the ABI on how the compiler calls the runtime. So the compiler and the runtime are working together as a team.

So once we have a runtime library which is a requirement for any objectiveC code, there is also the base library gnustep-base which corresponds to Apple's Foundation framework. This one implements basic functionality which is used in almost all ObjC code.. It brings stuff like dealing with strings, arrays, dictionaries, and lots of other things you need day in day out. It does not supply any GUI code. All NextStep/Apple ObjectiveC code is derived from a base class called NSObject which is defined in gnustep-base. GnuStep is an open source reimplementation of NextStep.

There are 3 other libraries in GnuStep which are gnustep-gui which does all the graphical interface, gnustep-back which is the backend for the GUI (so there's a variant for X-windows, one for Windows, one for MacOS aqua etc) which is like a driver for the gui to do the displaying on

actual hardware and there is gnustep-corebase which is a glue library to use objective C objects from C and to deal with the fundamental data types of objective C like NSData, NSArray etc.

9. compiling libobjc2

```
cd libobjc2
mkdir build
cd build
cmake .. -DCMAKE_BUILD_TYPE=Release \
         -DBUILD_STATIC_LIBOBJC=1  \
         -DCMAKE_C_COMPILER=${CC} \
         -DCMAKE_CXX_COMPILER=${CXX} \
         -DCMAKE_LINKER=${LD} \
         -DCMAKE_MODULE_LINKER_FLAGS=${LDFLAGS}
make -j40
make install
ldconfig
make test
cd ../..
```

this should pass everywhere except these two which according to David Chisnal are corner cases which shouldn't affect real software:

```
18 - AssociatedObject_optimised (OTHER_FAULT)
20 - AssociatedObject_legacy_optimised (OTHER_FAULT)
```

10 installing gnustep-make

Gnustep Make is a package of makefiles to be used inside Gnustep projects. This way some common settings can be shared along all projects such as the location of certain runtime files (defaults, file layouts etc). It does not have any compiled code but its needed for the rest.

Edit the file library-combo.make and on line 47 replace

```
ifeq ($(RUNTIME_VERSION),)
  RUNTIME_VERSION=gnustep-1.8
endif
```

with

```
ifeq ($(RUNTIME_VERSION),)
  RUNTIME_VERSION=gnustep-2.0
endif
```

if you define RUNTIME_VERSION as environment variable, it overrides whats in the makefile however.

then you can install it

```
  cd make
./configure \
        --with-layout=fhs \
        --disable-importing-config-file \
        --enable-native-objc-exceptions \
        --enable-objc-arc \
        --enable-install-ld-so-conf \
        --with-library-combo=ng-gnu-gnu \
        --with-config-file=/usr/local/etc/GNUstep/GNUstep.conf \
        --with-user-config-file='.GNUstep.conf' \
        --with-user-defaults-dir='GNUstep/Library/Defaults'
make
make install
```

11 installing gnustep-base

```
cd base
./configure  --with-config-file=/usr/local/etc/GNUstep/GNUstep.conf
make -j8
make install
ldconfig
make check
cd ..
```

There is one test which will fail. For our usage this is not vital.

```
  Failed test:       basic.m:31 ... Expiration date can be retrieved
```

12 installing gnustep-corebase

```
cd corebase
./configure
make -j8
make install
ldconfig
cd ..
```

13 installing gnustep-gui

```
cd gui
./configure
make -j8
make install
ldconfig
cd ..
```

14 installing gnustep-back

```
cd back
./configure
make -j
make install
ldconfig
cd ..
```