UNIX BUILD NOTES

Some notes on how to build Dynamic in Unix.

Note

Always use absolute paths to configure and compile dynamic and the dependencies,
for example, when specifying the the path of the dependency:

```
../dist/configure --enable-cxx --disable-shared
--with-pic --prefix=$BDB_PREFIX
```

Here BDB_PREFIX must absolute path - it is defined using $(pwd) which ensures
the usage of the absolute path.

To Build

```
./autogen.sh
./configure
make
make install # optional
```

This will build dynamic-qt as well if the dependencies are met.

Dependencies

These dependencies are required:

Library	Purpose	Description
libssl	SSL Support	Secure communications
libboost	Boost	C++ Library
libevent	Networking	OS independent asynchronous networking

Optional dependencies:

Library	Purpose	Description
miniupnpc	UPnP Support	Firewall-jumping support
libdb4.8	Berkeley DB	Wallet storage (only needed when wallet enabled)
qt	GUI	GUI toolkit (only needed when GUI enabled)
protobuf	Payments in GUI	Data interchange format used for payment protocol (only needed when GUI enabled)
libqrencode	QR codes in GUI	Optional for generating QR codes (only needed when GUI enabled)
libzmq3	ZMQ notification	Optional, allows generating ZMQ notifications (requires ZMQ version >= 4.x)

For the versions used in the release, see release-process.md under *Fetch and build inputs*.

System requirements

C++ compilers are memory-hungry. It is recommended to have at least 1 GB of
memory available when compiling Dynamic. With 512MB of memory or less compilation will take much longer due to swap thrashing.

Dependency Build Instructions: Ubuntu & Debian

Build requirements:

```
sudo apt-get install build-essential libtool autotools-dev
autoconf pkg-config libssl-dev libcrypto++-dev
libevent-dev git
```

for Ubuntu 12.04 and later or Debian 7 and later libboost-all-dev has to be installed:

```
sudo apt-get install libboost-all-dev
```

db4.8 packages are available here.
You can add the repository using the following command:

```
    sudo add-apt-repository ppa:bitcoin/bitcoin
    sudo apt-get update
```

Ubuntu 12.04 and later have packages for libdb5.1-dev and libdb5.1++-dev, but using these will break binary wallet compatibility, and is not recommended.

for Debian 7 (Wheezy) and later:
The oldstable repository contains db4.8 packages.
Add the following line to /etc/apt/sources.list,
replacing [mirror] with any official debian mirror.

```
deb http://[mirror]/debian/ oldstable main
```

To enable the change run

```
sudo apt-get update
```

for other Debian & Ubuntu (with ppa):

```
sudo apt-get install libdb4.8-dev libdb4.8++-dev
```

Optional:

```
sudo apt-get install libminiupnpc-dev
(see --with-miniupnpc and --enable-upnp-default)
```

Optional ZMQ dependencies (provides ZMQ API 4.x):

```
sudo apt-get install libzmq3-dev
```

Dependencies for the GUI: Ubuntu & Debian

If you want to build Dynamic-Qt, make sure that the required packages for Qt development
are installed. Either Qt 4 or Qt 5 are necessary to build the GUI.
If both Qt 4 and Qt 5 are installed, Qt 4 will be used. Pass `--with-gui=qt5` to configure to choose Qt5.
To build without GUI pass `--without-gui`.

To build with Qt 4 you need the following:

```
sudo apt-get install libqt4-dev libprotobuf-dev
protobuf-compiler
```

For Qt 5 you need the following:

```
sudo apt-get install libqt5gui5 libqt5core5a libqt5dbus5
qttools5-dev qttools5-dev-tools libprotobuf-dev
protobuf-compiler
```

libqrencode (optional) can be installed with:

```
sudo apt-get install libqrencode-dev
```

Once these are installed, they will be found by configure and a dynamic-qt executable will be
built by default.

Notes

The release is built with GCC and then "strip dynamicd" to strip the debug symbols, which reduces the executable size by about 90%.

miniupnpc

miniupnpc may be used for UPnP port mapping. It can be downloaded from here. UPnP support is compiled in and
turned off by default. See the configure options for upnp behavior desired:

```
--without-miniupnpc
(No UPnP support miniupnp not required)
--disable-upnp-default
(the default) UPnP support turned off by default at runtime
--enable-upnp-default
(UPnP support turned on by default at runtime)
```

To build:

```
tar -xzvf miniupnpc-1.6.tar.gz
cd miniupnpc-1.6
make
sudo su
make install
```

Berkeley DB

It is recommended to use Berkeley DB 4.8. If you have to build it yourself:

```
DYNAMIC_ROOT=$(pwd)

# Pick some path to install BDB to,
here we create a directory within the dynamic directory
BDB_PREFIX="${DYNAMIC_ROOT}/db4"
mkdir -p $BDB_PREFIX

# Fetch the source and verify that it is not tampered with
wget
'http://download.oracle.com/berkeley-db/db-4.8.30.NC.tar.gz'

echo
'12edc0df75bf9abd7f82f821795bcee50f42cb2e5f76a6a281b85732798
364ef  db-4.8.30.NC.tar.gz' | sha256sum -c

# -> db-4.8.30.NC.tar.gz: OK
tar -xzvf db-4.8.30.NC.tar.gz

# Build the library and install to our prefix
cd db-4.8.30.NC/build_unix/
#  Note: Do a static build so that it can be embedded
into the exectuable, instead of having to find
a .so at runtime
../dist/configure --enable-cxx --disable-shared
--with-pic --prefix=$BDB_PREFIX
```

```
make install

# Configure Dynamic to use our own-built instance of BDB
cd $DYNAMIC_ROOT
./configure (other args...) LDFLAGS="-L${BDB_PREFIX}/lib/"
CPPFLAGS="-I${BDB_PREFIX}/include/"
```

Note: You only need Berkeley DB if the wallet is enabled (see the section *Disable-Wallet mode* below).

Boost

If you need to build Boost yourself:

```
sudo su
./bootstrap.sh
./bjam install
```

Security

To help make your Dynamic installation more secure by making certain attacks impossible to
exploit even if a vulnerability is found, binaries are hardened by default.
This can be disabled with:

Hardening Flags:

```
./configure --enable-hardening
./configure --disable-hardening
```

Hardening enables the following features:

- Position Independent Executable
  Build position independent code to take advantage of Address Space Layout Randomization
  offered by some kernels. An attacker who is able to cause execution of code at an arbitrary
  memory location is thwarted if he doesn't know where anything useful is located.
  The stack and heap are randomly located by default but this allows the

code section to be
randomly located as well.

On an Amd64 processor where a library was not compiled with -fPIC, this will cause an error
such as: "relocation R_X86_64_32 against `.....' can not be used when making a shared object;"

To test that you have built PIE executable, install scanelf, part of paxutils, and use:

```
scanelf -e ./dynamicd
```

The output should contain:
TYPE
ET_DYN

- Non-executable Stack
  If the stack is executable then trivial stack based buffer overflow exploits are possible if
  vulnerable buffers are found. By default, dynamic should be built with a non-executable stack
  but if one of the libraries it uses asks for an executable stack or someone makes a mistake
  and uses a compiler extension which requires an executable stack, it will silently build an
  executable without the non-executable stack protection.

  To verify that the stack is non-executable after compiling use:
  `scanelf -e ./dynamicd`

  the output should contain:
  STK/REL/PTL
  RW- R-- RW-

> The STK RW- means that the stack is readable and writeable but not executable.

Disable-wallet mode

When the intention is to run only a P2P node without a wallet, dynamic may be compiled in
disable-wallet mode with:

```
./configure --disable-wallet
```

In this case there is no dependency on Berkeley DB 4.8.

Mining is also possible in disable-wallet mode, but only using the `getblock template` RPC
call not `getwork`.

AVX2 Mining Optimisations

For increased performance when mining, AVX2 optimisations can be enabled.

At configure time:

```
--enable-avx2
```

CPU's with AVX2 support:

```
Intel
    Haswell processor, Q2 2013
    Haswell E processor, Q3 2014
    Broadwell processor, Q4 2014
    Broadwell E processor, Q3 2016
    Skylake processor, Q3 2015
    Kaby Lake processor, Q3 2016
    (ULV mobile)/Q1 2017(desktop/mobile)
    Coffee Lake processor, Q4 2017

AMD
    Carrizo processor, Q2 2015
    Ryzen processor, Q1 2017
```

AVX512 Mining Optimisations

For increased performance when mining, AVX512 optimisations can be enabled.

At configure time:

```
--enable-avx512f
```

CPU's with AVX512 support:

```
Intel
    Xeon Phi x200/Knights Landing processor, 2016
    Knights Mill processor, 2017
    Skylake-SP processor, 2017
    Skylake-X processor, 2017
    Cannonlake processor, expected in 2019
    Ice Lake processor, expected in 2019
```

GPU Mining

To enable GPU mining within the wallet, OpenCL or CUDA can be utilised. Please use GCC/G++ 6.4 or newer and for CUDA to be utilised please use CUDA 9.1 or newer and ensure you have graphics drivers installed.

For OpenCL you need the following:

```
sudo apt-get install ocl-icd-opencl-dev
```

For CUDA please visit: https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html

At configure time for OpenCL(Nvidia/AMD):

```
--enable-gpu
```

At configure time for CUDA(Nvidia):

```
--enable-gpu --enable-cuda
```

Example Build Command

Qt Wallet and Deamon, CLI version build without GPU support and without AVX support:

```
./autogen.sh && ./configure --with-gui --disable-gpu && make
```

CLI and Deamon Only build without GPU support and without AVX support:

```
./autogen.sh && ./configure --without-gui --disable-gpu && make
```