OpenBSD Build Guide

Updated for OpenBSD 7.0

This guide describes how to build bitcoind, command-line utilities, and GUI on OpenBSD.

Preparation

1. Install Required Dependencies

Run the following as root to install the base dependencies for building.

```
pkg_add bash git gmake libevent libtool boost
# Select the newest version of the following packages:
pkg_add autoconf automake python
```

See dependencies.md for a complete overview.

2. Clone Bitcoin Repo

Clone the Bitcoin Core repository to a directory. All build scripts and commands will run from this directory.

```
git clone https://github.com/bitcoin/bitcoin.git
```

3. Install Optional Dependencies

Wallet Dependencies

It is not necessary to build wallet functionality to run either `bitcoind` or `bitcoin-qt`.

DESCRIPTOR WALLET SUPPORT

`sqlite3` is required to support descriptor wallets.

```
pkg_add install sqlite3
```

LEGACY WALLET SUPPORT

BerkeleyDB is only required to support legacy wallets.

It is recommended to use Berkeley DB 4.8. You cannot use the BerkeleyDB library from ports. However you can build it yourself, using the installation script included in contrib/, like so, from the root of the repository.

```
./contrib/install_db4.sh `pwd`
```

Then set `BDB_PREFIX`:

```
export BDB_PREFIX="$PWD/db4"
```

GUI Dependencies

QT5

Bitcoin Core includes a GUI built with the cross-platform Qt Framework. To compile the GUI, Qt 5 is required.

```
pkg_add qt5
```

Building Bitcoin Core

Important: Use `gmake` (the non-GNU `make` will exit with an error).

Preparation:

```
# Adapt the following for the version you installed (major.minor only):
export AUTOCONF_VERSION=2.71
export AUTOMAKE_VERSION=1.16

./autogen.sh
```

1. Configuration

Note that building with external signer support currently fails on OpenBSD, hence you have to explicitly disable it by passing the parameter `--disable-external-signer` to the configure script. The feature requires the header-only library boost::process, which is available on OpenBSD, but contains certain system calls and preprocessor defines like `waitid()` and `WEXITED` that are not available.

There are many ways to configure Bitcoin Core, here are a few common examples:

Descriptor Wallet and GUI:

This enables the GUI and descriptor wallet support, assuming `sqlite` and `qt5` are installed.

```
./configure --disable-external-signer MAKE=gmake
```

Descriptor & Legacy Wallet. No GUI:

This enables support for both wallet types and disables the GUI:

```
./configure --disable-external-signer --with-gui=no \
    BDB_LIBS="-L${BDB_PREFIX}/lib -ldb_cxx-4.8" \
    BDB_CFLAGS="-I${BDB_PREFIX}/include" \
    MAKE=gmake
```

2. Compile

Important: Use `gmake` (the non-GNU `make` will exit with an error).

```
gmake # use "-j N" for N parallel jobs
gmake check # Run tests if Python 3 is available
```

Resource limits

If the build runs into out-of-memory errors, the instructions in this section might help.

The standard ulimit restrictions in OpenBSD are very strict:

```
data(kbytes)         1572864
```

This is, unfortunately, in some cases not enough to compile some `.cpp` files in the project, (see issue #6658). If your user is in the `staff` group the limit can be raised with:

```
ulimit -d 3000000
```

The change will only affect the current shell and processes spawned by it. To make the change system-wide, change `datasize-cur` and `datasize-max` in `/etc/login.conf`, and reboot.