NB 1: This document primarily covers a near-final version of Bitcoin Core 0.12. It will be updated later, when the final 0.12 release comes out.
NB 2: Unless there are special reasons for listing them, graphics are left out.

./.gitattributes - Settings that can be specified for a path. Used for version info stuff.
./.gitignore - File that prevents certain files from showing up in Git.
./.travis.yml - Provides project-specific info to Travis CI.
./autogen.sh - The first script called in the build process. Does a bit of setup and then calls *autoreconf* to do a proper setup of the autotools build environment.
./configure.ac - *autoconf* input file (and basically the build's starting point). Sets certain project-related values and describes tests for all features required on the build machine. Run as *./configure* once *./autogen.sh* has been run.
./COPYING - Copyright info.
./CONTRIBUTING.md - A document explaining some ways in which people can conttribute to the Core project.
./INSTALL - Just points you to doc/build-*.md for build instructions.
./libbitcoinconsensus.pc.in - Used for pkgconfig support.
./Makefile.am - *automake* input file. It's basically the recipe used to create the final Makefile. Run as *./Makefile* once *./autogen.sh* and *./configure* have been run.
./README.md - Base README file.

./.tx - Transifex (i.e., translation) stuff
./.tx/config - Transifex config file.

./build-aux - Autotools materials. Auxiliary config files (AC_CONFIG_AUX_DIR - configure.ac).
No files actually found here.
./build-aux/m4 - Additional local Autoconf macros (AC_CONFIG_MACRO_DIR - configure.ac).
Basically tests to see if certain preconditions are met before compiling Core.
./build-aux/m4/ax_boost_base.m4 - Checks if Boost C++ libraries meet a particular version.
./build-aux/m4/ax_boost_chrono.m4 - Checks for "Chrono" library from Boost.
./build-aux/m4/ax_boost_filesystem.m4 - Checks for "Filesystem" library from Boost.
./build-aux/m4/ax_boost_program_options.m4 - Checks for program options library from Boost.
./build-aux/m4/ax_boost_system.m4 - Checks for "System" library from Boost.
./build-aux/m4/ax_boost_thread.m4 - Checks for "Thread" library from Boost.
./build-aux/m4/ax_boost_unit_test_framework.m4 - Checks for unit test framework library from Boost.
./build-aux/m4/ax_check_compile_flag.m4 - Checks whether a given flag works with a compiler.
./build-aux/m4/ax_check_link_flag.m4 - Checks whether a given flag works with a linker.
./build-aux/m4/ax_check_preproc_flag.m4 - Checks whether a given flag works with a language's preprocessor.
./build-aux/m4/ax_gcc_func_attribute.m4 - Checks whether the compiler supports a given GCC function attribute (e.g., "dllimport" or "warn_unused_result").

./build-aux/m4/ax_pthread.m4 - Determines if pthreads are supported and does some setup work.
./build-aux/m4/bitcoin_find_bdb48.m4 - Checks if BerkeleyDB 4.8 is present. Core-specific.
./build-aux/m4/bitcoin_qt.m4 - Checks if Qt dependencies are met. Came about to help make Qt 5 detection more sane.
./build-aux/m4/bitcoin_subdir_to_include.m4 - I believe this is used only to indicate the location of BerkeleyDB 4.8 #include files.

./contrib - External tools that may be useful but aren't directly related to Core.
./contrib/bitcoin-qt.pro - Qt file used by *qmake* to help compile Qt-related code for Core.
./contrib/bitcoind.bash-completion - Used to help bash complete *bitcoind* commands? Looks like it's activated when doing Debian builds.
./contrib/qt_translations.py - Helps OS X include the correct translations. Apparently no longer used.
./contrib/README.md - Briefly explains many things in the subdirs.
./contrib/tidy_datadir.sh - Script that cleans up Core datadir (e.g., wallet.dat, blk00001.dat).

./contrib/debian - Contains files used to package bitcoind/bitcoin-qt for Debian-based Linux systems. Added in Nov. 2011. Included to control PPA builds.
./contrib/debian/bitcoin-qt.desktop - Desktop menu entry registration.
./contrib/debian/bitcoin-qt.install - Indicates which files need to go where when installing a package.
./contrib/debian/bitcoin-qt.lintian-overrides - Manual warning/error overrides for *lintian*, the program that inspects Debian packages.
./contrib/debian/bitcoin-qt.protocol - GNOME/KDE support for the "bitcoin" URI.
./contrib/debian/bitcoind.bash-completion - Rule that installs ./contrib/bitcoind.bash-completion in Debian systems as part of the Debian build process. Adds BASH shell completion for *bitcoind*. *Added in May 2012.*
./contrib/debian/bitcoind.examples - Contains a list of places to find examples. Used by lintian.
./contrib/debian/bitcoind.install - Used to indicate what to install for *bitcoind*.
./contrib/debian/bitcoind.lintian-overrides - Manual warning/error overrides for *lintian*.
./contrib/debian/bitcoind.manpages - Indicates where manpages should go.
./contrib/debian/bitcointx.install - Used to indicate what to install for *bitcoin-tx*.
./contrib/debian/changelog - Changelog. Required by *lintian*.
./contrib/debian/compat - *debhelper* compatibility level.
./contrib/debian/control - Values used by various packaging tools (e.g., *dpkg*, *apt-get*, etc.) to manage a package.
./contrib/debian/copyright - Copyright notice. Required by *lintian*.
./contrib/debian/gbp.conf - Config file for *git-buildpackage* & such.
./contrib/debian/README.md - Minimal README. Mostly talks about the "bitcoin" URI.
./contrib/debian/rules - Rules applied by dpkg-buikdpackage.
./contrib/debian/watch - Used by *uscan* to monitor where the source was obtained. Seems to use a qa.debian.org redirector.

./contrib/debian/examples - Related to Debian packaging. Contains examples for PPA users. *Will not discuss individual files.*

./contrib/debian/examples/bitcoin.conf

./contrib/debian/manpages - Related to Debian packaging. Package manpages. *Will not discuss individual files.*

./contrib/debian/manpages/bitcoin.conf.5
./contrib/debian/manpages/bitcoin-cli.1
./contrib/debian/manpages/bitcoind.1
./contrib/debian/manpages/bitcoin-qt.1

./contrib/debian/patches - Related to Debian packaging. Used by packagers who need to patch Core's upstream code. *Not used for now.*

./contrib/debian/patches/README - Basic info regarding patch nature.
./contrib/debian/patches/series - Order of patches to apply.

./contrib/debian/source - Related to Debian packaging. Enforces program used to apply packages (Quilt). *Not used for now.*

./contrib/debian/source/format - Enforces program used to apply packages (Quilt).

./contrib/devtools - Specific tools for devs working on the Core repo.

./contrib/devtools/clang-format - Sets the rules for *clang-format*, a program that formats C/C++/ObjC code. *Added in 0.12.*
./contrib/devtools/fix-copyright-headers.py - Script that does a mass copyright year update.
./contrib/devtools/github-merge.sh - Used to merge PRs securely and to sign the merge commits. Used only by those with commit access to the Core repo.
./contrib/devtools/git-subtree-check.sh - Verifies that subtree contents match upstream subtrees. Must run from repo root. Currently useful for libsecp256k1 and LevelDB.
./contrib/devtools/optimize-pngs.py - Optimize data in PNG files.
./contrib/devtools/README.md - Explains everything in the subdir.
./contrib/devtools/security-check.py - Confirms that certain security features (e.g., PIE, NX, RELRO, and canaries) are used for a given binary. *Added in 0.12.*
./contrib/devtools/symbol-check.py - Makes sure a Linux binary has only gcc, glibc, and libstdc++ version symbols, thereby maintaining compatibility with minimum supported Linux distro versions.
./contrib/devtools/test-security-check.py - Compiles a simple program in C and confirms that the binary has the security features tested in ./contrib/devtools/security-check.py. *Added in 0.12.*
./contrib/devtools/update-translations.py - Used to fetch new translations from Transifex.

./contrib/gitian-descriptors - Descriptor for *gitian*, the deterministic build system used to create Bitcoin Core binaries for distribution. Describes the VM that will contain the build system and the script that will run on the VM.

./contrib/gitian-descriptors/gitian-linux.yml - Linux (32-bit & 64-bit) descriptor.
./contrib/gitian-descriptors/gitian-osx.yml - OS X (64-bit only) descriptor. Uses an *Xcode* SDK (not available in the repo) to compile the OS X binary under Linux. Code not signed for Gatekeeper.
./contrib/gitian-descriptors/gitian-osx-signer.yml - Descriptor that splices in a signature for the final .dmg file. See ./contrib/macdeploy/ for the signing key tools.
./contrib/gitian-descriptors/gitian-win.yml - Windows (32-bit & 64-bit) descriptor. Uses the *MinGW* development environment to compile the Windows version under Linux. Code not signed for Windows.
./contrib/gitian-descriptors/gitian-win-signer.yml - Descriptor that splices in a signature for the final .exe file. Uses osslsigncode to attach the detached signature to the Core binary. The tool(s) & steps used to generate the signature don't appear to be listed in the Core repo.
./contrib/gitian-descriptors/README.md - Rough instructions explaining how to do Gitian builds.

./contrib/gitian-downloader - PGP keys for people who post verifications of Gitian builds. *Will not discuss list individual files.*

./contrib/init - Sample init scripts and service configuration for *bitcoind*. Used to assist packagers in creating node packages.
./contrib/init/bitcoind.conf - *Upstart* service configuration file.
./contrib/init/bitcoind.init - CentOS compatible SysV style init script.
./contrib/init/bitcoind.openrc - *OpenRC* compatible SysV style init script.
./contrib/init/bitcoind.openrcconf - *OpenRC* conf.d file.
./contrib/init/bitcoind.service - *systemd* service unit configuration.
./contrib/init/org.bitcoin.bitcoind.plist - Launch agent that OS X can use to launch *bitcoind* at startup. *Added in 0.12.*
./contrib/init/README.md - Mostly a short version of doc/init.md.

./contrib/linearize - Construct a linear, no-fork, best version of the blockchain.
./contrib/linearize/example-linearize.cfg - Example of a config to use for both steps. No specific directions for config development seem to exist.
./contrib/linearize/linearize-data.py - Step 2.
./contrib/linearize/linearize-hashes.py - Step 1.
./contrib/linearize/README.md - Explains how to use everything.

./contrib/macdeploy - Used to create a .dmg package for Bitcoin Core. Uses a detached signature repo so that anybody can grab the appropriate sigs. (Cory Fields/"theuni" is the only person who can actually generate the sigs, which he then uploads to the sig repo.) Used to make Core compatible with Apple's "Gatekeeper" scheme. *Graphics files have been removed.*
./contrib/macdeploy/detached-sig-apply.sh - Applies a detached signature from a repo and attached the signature to an unsigned OS X build to create a signed build.
./contrib/gitian-descriptors/gitian-osx-signer.yml contains a usage example.

./contrib/macdeploy/detached-sig-create.sh - Creates a detached signature of an OS X build that can then be uploaded to a repo for downloading and attachment via the ./contrib/macdeploy/detached-sig-apply.sh script that basically uses *codesign* to create a detached signature covering Core's entire .app directory. To be used only by somebody with the Apple-supplied signing key.
./contrib/macdeploy/DS_Store - Apple-specific binary file. Used for packaging purposes.
./contrib/macdeploy/fancy.plist - Used to generate a "fancy" DMG disk image (i.e., the screen that allows the user, when a DMG file is double-clicked, to install Core).
./contrib/macdeploy/LICENSE - GNUv3 license.
./contrib/macdeploy/macdeployqtplus - The script that's used to assemble everything into an OS X-friendly app format (i.e., the .app directory).
./contrib/macdeploy/README.md - Deployment instructions. Presumably up-to-date.

./contrib/macdeploy/Base.lproj - Required for app bundle name purposes. Used to avoid a hack.
./contrib/macdeploy/Base.lproj/InfoPlist.strings - Contains strings needed by the app bundle. Can be split up according to language but isn't (for now?).

./contrib/qos - Script that can limit bandwidth used by Core.
./contrib/qos/README.md - Script explaining the code.
./contrib/qos/tc.sh - Script that uses *tc* to limit the bandwidth.

./contrib/seeds - Generates a list of fixed seed nodes. Nodes are then used to update ./src/chainparamsseeds.h
./contrib/seeds/generate-seeds.py - Script used to generate ./src/chainparamsseeds.h.
./contrib/seeds/makeseeds.py - Parses data from Pieter Wuille's website. Presumably pulls down nodes for nodes_main.txt.
./contrib/seeds/nodes_main.txt - List of mainnet nodes (IPv4, IPv6, and Onion). Tied to generate-seeds.py.
./contrib/seeds/nodes_test.txt - List of testnet nodes (Onion-only). Tied to generate-seeds.py.
./contrib/seeds/README.md - Gives a bit of context to the tool.

./contrib/spendfrom - Use the raw transactions API to send coins received on a particular address (or addresses).
./contrib/spendfrom/README.md - Explains how to use the script.
./contrib/spendfrom/setup.py - Standard Python setup.py file.
./contrib/spendfrom/spendfrom.py - The script to run.

./contrib/testgen - Utilities to generate test vectors for the data-driven Bitcoin tests.
./contrib/testgen/base58.py - Support script for Base58 encode/decode.
./contrib/testgen/gen_base58_test_vectors.py - The script to run.
./contrib/testgen/README.md - Shows how to run the script but doesn't explain what it all means.

./contrib/verify-commits - Git tool to verify that every merge commit was signed by a developer using ./contrib/devtools/github-merge.sh. It's meant only for those with commit access.
./contrib/verify-commits/allow-revsig-commits - Whitelisted commits signed by a revoked key. *Added in 0.12*.
./contrib/verify-commits/gpg.sh - The "gpg.program" Git variable is set to this script, which will act as a *GnuPG* substitute. Used instead of *GnuPG* to do the actual signature verification.
./contrib/verify-commits/pre-push-hook.sh - If copied to .git/hooks/pre-push (and BASH is installed), this script prevents the pushing of unsigned commits to *master* on bitcoin/bitcoin. Git info here.
./contrib/verify-commits/trusted-git-root - The point at which the commit tree stops looking to confirm that the code is accurate. Everything before this commit is assumed to be safe.
./contrib/verify-commits/trusted-keys - Fingerprints of trusted keys used to push commits. Used when going through commits to ensure that all signatures come from keys that are trusted.
./contrib/verify-commits/verify-commits.sh - Called by ./contrib/verify-commits/pre-push-hook.sh. Verifies that the commits before the current one have been properly signed, and that the commit about to be pushed has been signed by an accepted key.

./contrib/verifysfbinaries - Script that verifies binaries downloaded from bitcoin.org.
./contrib/verifysfbinaries/README.md - Basically explains what's going on.
./contrib/verifysfbinaries/verify.sh - The verification script!

./contrib/zmq - Support for ØMQ (ZeroMQ), an asynchronous messaging library. *Added in 0.12.*
./contrib/verifysfbinaries/zmq_sub.py - Working example of adding ØMQ support.

./depends - A shared dependency builder. See AC_CONFIG_AUX_DIR() in ./configure.ac to see how Autotools connects to these files. Added in Aug. 2014.
./depends/.gitignore - Lists files for Git to ignore.
./depends/config.guess - Used by *Autoconf* to guess a canonical system name.
./depends/config.site.in - Defines some variables. ./depends/Makefile processes the file and uses it to output ./share/config.site, which causes the build to override some build settings on the building machine.
./depends/config.sub - Used by *Autoconf* to convert system aliases into full canonical names.
./depends/description.md - General description of the depends system. Some high-level technical details of the depends setup.
./depends/funcs.mk - Seems to be what deals with everything in the ./depends/packages subdirectory. (FIX - TALK TO CORY)
./depends/Makefile - Makefile called from elsewhere.
./depends/packages.md - Steps for adding packages. Low-level "recipe" details. Good for devs.
./depends/README.md - Some general details and *make* options and such.

./depends/builders - Shared dependency builder. Compiler/Linker variables and such required by the machine building the binary. (The "builder" identity is automated and determined by ./config.guess.) Used by ./depends/Makefile. (FIX - CORY)

./depends/builders/darwin.mk - Overriding values for OS X building systems.
./depends/builders/default.mk - Default values for all builders. Called by ./depends/Makefile, along with the appropriate builder-specific .mk file.
./depends/builders/linux.mk - Overriding values for Linux building systems.

./depends/hosts - Shared dependency builder. Compiler/Linker variables and such required to build for the host machine (i.e., the machine that'll run the final binary). The host may be specified, otherwise it'll default to the builder. Used by ./depends/Makefile. (FIX - CORY)

./depends/hosts/darwin.mk - Overriding OS X build variable values.
./depends/hosts/default.mk - Default build variable values. Called by ./depends/Makefile, along with the appropriate host-specific .mk file.
./depends/hosts/linux.mk - Overriding Linux build variable values.
./depends/hosts/mingw32.mk - Overriding Windows (MinGW) build variable values.

./depends/packages - Info on packages to download and compile as Core dependencies. Almost all the packages are actually required only by Qt 4 or 5.

./depends/packages/bdb.mk - BerkeleyDB 4.8.
./depends/packages/boost.mk - Boost.
./depends/packages/dbus.mk - D-Bus.
./depends/packages/expat.mk - Expat (XML parser).
./depends/packages/fontconfig.mk - Fontconfig.
./depends/packages/freetype.mk - FreeType.
./depends/packages/libevent.mk - *libevent* (event notification) library.
./depends/packages/libICE.mk - Inter-Client Exchange (ICE) library.
./depends/packages/libSM.mk - Session Management library (SMlib).
./depends/packages/libX11.mk - X11 library.
./depends/packages/libXau.mk - X11 Authorization Protocol library.
./depends/packages/libxcb.mk - X protocol C-language Binding (XCB) library.
./depends/packages/libXext.mk - X Record Extension library.
./depends/packages/miniupnpc.mk - MiniUPnP.
./depends/packages/native_ccache.mk - C/C++ compiler cache (native).
./depends/packages/native_cctools.mk - Apple (cctools) toolchain for Linux (native).
./depends/packages/native_cdrkit.mk - *cdrkit* toolkit (native).
./depends/packages/native_comparisontool.mk - A snapshot of the *bitcoindcomparisontool* code from the *bitcoinj* library. (native) (FIX - CORY - WHAT IS THIS???)
./depends/packages/native_libdmg-hfsplus.mk - HFS+/DMG manipulation libraries (native).
./depends/packages/native_protobuf.mk - Protocol Buffers library (native).
./depends/packages/openssl.mk - OpenSSL library.

./depends/packages/packages.mk - Package variables. Lists, among other things, which packages apply to Qt only. Called by ./depends/Makefile. (FIX - GO BACK TO THIS TO UNDERSTAND SOME THINGS)
./depends/packages/protobuf.mk - Protocol Buffers variables.
./depends/packages/qrencode.mk - *libqrencode* variables.
./depends/packages/qt.mk - Qt 5 variables.
./depends/packages/qt46.mk - Qt 4.6 variables.
./depends/packages/xcb_proto.mk - XML-XCB protocol description bindings.
./depends/packages/xextproto.mk - X protocol extensions library.
./depends/packages/xproto.mk - X window system core protocol library.
./depends/packages/xtrans.mk - xtrans library.
./depends/packages/zmq.mk - ØMQ (aka ZeroMQ).

./depends/patches - Patches to be applied to downloaded packages before building the packages. *Nothing in the root directory.*

./depends/patches/boost - Boost patches.
./depends/patches/boost/darwin_boost_atomic-1.patch - Upstream patch. Should be dropped eventually.
./depends/patches/boost/darwin_boost_atomic-2.patch - Upstream patch. Should be dropped eventually.
./depends/patches/boost/gcc_5_no_cxx11.patch - Turns off C++11 assumptions in Boost code. Will probably be dropped once C++11 is allowed in Core.

./depends/patches/native_cdrkit - cdrkit patches.
./depends/patches/native_cdrkit/cdrkit-deterministic.patch - Makes *cdrkit* deterministic.

./depends/patches/libevent - *libevent* patches.
./depends/patches/libevent/reuseaddr.patch - Fixes a Windows socket issue. *Added in 0.12.*

./depends/patches/qt - Qt 5 patches.
./depends/patches/qt/fix-xcb-include-order.patch - Fixes various compile errors for *libxcb*.
./depends/patches/qt/mac-qmake.conf - *qmake* config variables and such.
./depends/patches/qt/mingw-uuidof.patch - Required to build with MinGW. *Added in 0.12.*
./depends/patches/qt/pidlist_absolute.patch - Fixes Qt “PIDLIST_ABSOLUTE” issue. *Added in 0.12.*

./depends/patches/qt46 - Qt 4.6 patches.
./depends/patches/qt46/stlfix.patch - Fixes compile issue under Qt 4.6.

./doc - Various documents.
./doc/assets-attribution.md - Graphics attributions. Nothing special.
./doc/bips.md - List of BIPs implemented by Core, the version where they were added, the PR #, and, if relevant, the block where the BIP was activated.

./doc/build-openbsd.md - Details regarding building under OpenBSD.
./doc/build-osx.md - Details regarding regular (non-Gitian) OSX builds.
./doc/build-unix.md - General (non-Gitian) build notes for Linux.
./doc/build-windows.md - General (non-Gitian) build notes for cross-compiling the Windows (MinGW32) build.
./doc/developer-notes.md - General dev notes. Lots of miscellaneous things.
./doc/dnsseed-policy.md - Details regarding DNS seeds. Explains the rules necessary to run a DNS seed, which becomes a CDNSSeedData entry as added in ./src/chainparams.cpp.
./doc/Doxyfile - Doxygen (documentation system) config file.
./doc/files.md - Explains which files are made by Core (e.g., blockchain files).
./doc/gitian-building.md - Details regarding Gitian building.
./doc/init.md - “Sample init scripts and service configuration for bitcoind”.
./doc/multiwallet-qt.md - “Multiwallet Qt Development and Integration Strategy”. Old.
./doc/README_osx.txt - Details regarding on the deterministic build.
./doc/README_windows.txt - Vague Windows README. Not much useful info.
./doc/README.md - General catch-all / index of sorts.
./doc/reduce-traffic.md - Ways to reduce traffic on an Internet connection when running Core.
./doc/release-notes.md - Release notes for the latest version.
./doc/release-process.md - Steps for putting out new releases.
./doc/REST-interface.md - Details regarding using the REST interface (“-test” option).
./doc/shared-libraries.md - Discusses shared libraries created and used by Core (e.g., *bitcoinconsensus*).
./doc/tor.md - Details regarding Tor support in Core.
./doc/translation_process.md - More translation notes.
./doc/translation_strings_policy.md - Details on how the translation system works.
./doc/travis-ci.txt - Details regarding the Travis CI testing system.
./doc/unit-tests.md - Details regarding running and adding unit tests.
./doc/zmq.md - Details regarding using ØMQ.

./doc/gitian-building - Graphics used in Gitian guide. *Will not list the files.*
./doc/release-notes - Release note archive. *Will not list the files.*

./qa - Core and related tests. Deals primarily with the Core binaries and how they respond to external input (flags, messages, etc.), not tests for internal source code.
./qa/README.md - Explains how to run tests.

./qa/pull-tester - A set of tools that can be used to kick off tests on a remote server. (FIX - CORY)
./qa/pull-tester/rpc-tests.py - Primary script that kicks off one or more tests. *Added in 0.12.*
./qa/pull-tester/run-bitcoind-for-test.sh.in - Appears to run an instance of *bitcoind* in regtest mode. AC_CONFIG_FILES() (./configure.ac) processes the file and outputs it as ./qa/pull-tester/run-bitcoind-for-test.sh, which does the actual running of *bitcoind*. See ./Makefile.am for examples of how to run the script. Appears to require Cory Fields's

bitcoind-comparisontool, which is a snapshot of *bitcoindcomparisontool* from the *bitcoinj* Java library at certain times. The Java tools run the same tests using *bitcoinj* and compare the results with those from the original *bitcoind* binary run. Look here for a (poorly documented) example of how the *bitcoindcomparisontool* code can be used. (FIX - IS THIS CORRECT?)
./qa/pull-tester/tests_config.py.in - Sets, via AC_CONFIG_FILES() wizardry, some variables to be used in ./qa/pull-tester/rpc-tests.py. Examples include enabling ØMQ tests.

./qa/rpc-tests - Regression tests to run. These tests focus on high-level RPC (i.e., external-facing) functionality.
./qa/rpc-tests/.gitignore - Files for Git to ignore.
./qa/rpc-tests/bipdersig.py - Confirms that the BIP 66 soft fork/switchover code works properly. Uses BitcoinTestFramework.
./qa/rpc-tests/bipdersig-p2p.py - Confirms that the BIP 66 soft fork/switchover code works properly in a P2P environment. Uses *comptool*/ComparisonTestFramework.
./qa/rpc-tests/bip65-cltv.py - Confirms that the BIP 65 soft fork/switchover code works properly. Uses BitcoinTestFramework.
./qa/rpc-tests/bip65-cltv-p2p.py - Confirms that the BIP 65 soft fork/switchover code works properly in a P2P environment. Uses *comptool*/ComparisonTestFramework.
./qa/rpc-tests/blockchain.py - Tests the *gettxoutsetinfo* RPC functionality.
./qa/rpc-tests/decodescript.py - Tests the *decodescript* RPC functionality.
./qa/rpc-tests/disablewallet.py - Confirms that Core will work properly with the -disablewallet option.
./qa/rpc-tests/forknotify.py - Tests the *alertnotify* CL flag for when a fork occurs.
./qa/rpc-tests/fundrawtransaction.py - Tests the *fundrawtransaction* RPC functionality.
./qa/rpc-tests/getblocktemplate_longpoll.py - Tests the “long polling” functionality of the *getblocktemplate* RPC function (BIP 22).
./qa/rpc-tests/getblocktemplate_proposals.py - Tests the “block proposal” functionality of the *getblocktemplate* RPC function (BIP 23).
./qa/rpc-tests/getchaintips.py - Tests the *getchaintips* RPC functionality.
./qa/rpc-tests/httpbasics.py - Tests HTTP “keep-alive” functionality.
./qa/rpc-tests/invalidateblock.py - Tests the hidden *invalidateblock* RPC function.
./qa/rpc-tests/invalidblockrequest.py - Tests P2P ability to process valid and invalid blocks received after requesting blocks.
./qa/rpc-tests/invalidtxrequest.py - Checks to make sure that P2P “reject” mesages are properly sent and handled for transactions and blocks.
./qa/rpc-tests/keypool.py - Wallet keypool tests that interact with wallet locking/unlocking.
./qa/rpc-tests/listtransactions.py - Tests the *listtransactions* RPC functionality.
./qa/rpc-tests/maxblocksinflight.py - Tests whether a node is limiting the number of in-flight block requests.
./qa/rpc-tests/maxuploadtarget.py - Confirms that Core will work properly with the -maxuploadtarget option.
./qa/rpc-tests/mempool_limit.py - Confirms that Core will work properly with the -maxmempool option.

./qa/rpc-tests/mempool_packages.py - Tests the *descendantcount*, *descendantsize*, and *descendantfees* values from the *getrawmempool* RPC functionality.
./qa/rpc-tests/mempool_reorg.py - Tests the *invalidateblock* RPC functionality. In particular, it makes sure that coins that were valid due toinvalidated blocks will no longer be valid.
./qa/rpc-tests/mempool_resurrect_test.py - Confirms that a Tx in a block that was reorg'ed out is placed back in the mempool.
./qa/rpc-tests/mempool_spendcoinbase.py - Confirms that immature coinbase spends aren't allowed.
./qa/rpc-tests/merkle_blocks.py - Tests the generation and verification of merkle blocks. In other words, it tests the *gettxoutproof* and *verifytxoutproof* RPC functionality, which relate to proofs that a given TXID is in a given block. (The proofs are serialized CMerkleBlock classes.)
./qa/rpc-tests/multi_rpc.py - Tests to make sure that multiple *rpcuser* entries in bitcoin.conf will be properly handled by Core.
./qa/rpc-tests/nodehandling.py - Tests the *setban*, *listbanned*, and *disconnectnode* RPC functionality.
./qa/rpc-tests/p2p-acceptblock.py - Tests *AcceptBlock*() functionality from ./src/main.cpp, which is basically how unrequested blocks are handled.
./qa/rpc-tests/p2p-fullblocktest.py - A partial port of FullBlockTestGenerator.java, a file driven by BitcoinJ that generates test blockchains used to test/verify the handling of the blockchain in Core and various alternative implementations (e.g., BitcoinJ and BTCD). *Added in 0.12.*
./qa/rpc-tests/prioritise_transaction.py - Tests the *prioritisetransaction*RPC functionality.
./qa/rpc-tests/proxy_test.py - Various proxy tests for the *proxy*, *onion*, and *proxyrandomize* CL arguments.
./qa/rpc-tests/pruning.py - Tests block pruning functionality.
./qa/rpc-tests/rawtransactions.py - Tests reorg scenarios w/ a mempool that contain a Tx spending (direct or indirect) a coinbase Tx.
./qa/rpc-tests/README.md - Explains some the test_framework subdirectory's contents, along with some of the underlying technical details.
./qa/rpc-tests/receivedby.py - Tests the *getreceivedbyaddress* and *listreceivedbyaddress* functionality.
./qa/rpc-tests/reindex.py - Tests reindexing with CheckBlockIndex functionality enabled, all on the CL.
./qa/rpc-tests/rest.py - Tests the REST interface functionality.
./qa/rpc-tests/rpcbind_test.py - Tests binding of RPC functionality to various interfaces.
./qa/rpc-tests/sendheaders.py - Tests the *sendheaders* P2P message.
./qa/rpc-tests/signrawtransactions.py - Tests the *signrawtransaction* RPC functionality.
./qa/rpc-tests/smartfees.py - Tests the fee estimation code.
./qa/rpc-tests/txn_clones.py - Confirms that, if a cloned Tx is malleated and sent on the network instead of the original Tx, the malleated Tx will be seen later and the original Tx ignored. *Added in 0.12.*
./qa/rpc-tests/txn_doublespend.py - Tests double spend handling functionality.
./qa/rpc-tests/wallet.py - Various wallet tests.
./qa/rpc-tests/walletbackup.py - Tests wallet backup functionality.

./qa/rpc-tests/zapwallettxes.py - Tests *zapwallettxes* functionality (CL argument).
./qa/rpc-tests/zmq_test.py - Tests ØMQ RPC functionality.

./qa/rpc-tests/test_framework - Test support tools.
./qa/rpc-tests/test_framework/__init__.py - Standard Python package init file.
./qa/rpc-tests/test_framework/authproxy.py - Enhanced version of *ServiceProxy* from *python-jsonrpc*. Used to handle RPC calls and such.
./qa/rpc-tests/test_framework/bignum.py - Helpers for qa/rpc-tests/test_framework/script.py.
./qa/rpc-tests/test_framework/blockstore.py - Helper that implements disk-based block & Tx storage. Useful for replying to *getheaders* & *getdata*, and constructing a *getheaders* message.
./qa/rpc-tests/test_framework/blocktools.py - Helper functs for manipulating blocks & transactions.
./qa/rpc-tests/test_framework/comptool.py - Compare two *bitcoind* instances. Useful for P2P tests.
./qa/rpc-tests/test_framework/coverage.py - Code allowing for RPC call coverage.
./qa/rpc-tests/test_framework/key.py - Wrapper around OpenSSL's *EC_Key* struct. Fixes a crash in a regression test.
./qa/rpc-tests/test_framework/mininode.py - Basic P2P connectivity to a Bitcoin node via P2P.
./qa/rpc-tests/test_framework/netutil.py - Generally useful network utilities.
./qa/rpc-tests/test_framework/script.py - Bitcoin script manipulation utilities.
./qa/rpc-tests/test_framework/socks5.py - Dummy SOCKS5 server.
./qa/rpc-tests/test_framework/test_framework.py - Basic framework references (plain-or-P2P/BitcoinTestFramework and multiple binary versions/ComparisonTestFramework). Includes descriptions of arguments recognized by test scripts. Referenced by ./qa/pull-tester/rpc-tests.py.
./qa/rpc-tests/test_framework/util.py - Generally useful utilities.

./share - External materials used by Core one way or another but not part of the source code.
./share/genbuild.sh - Used to generate build info (and ./src/build.h).
./share/setup.nsi.in - Autotools stuff. AC_CONFIG_FILES() will cause it to be processed and output as ./share/setup.nsi, which is a Windows setup file.

./share/certs - Certificate materials used to sign Core binaries.
./share/certs/BitcoinFoundation_Apple_Cert.pem - Code-signing cert from Apple (OS X).
./share/certs/BitcoinFoundation_Comodo_Cert.pem - Code-signing cert from Comodo (Windows).
./share/certs/PrivateKeyNotes.md - Code signing threat analysis.

./share/pixmaps - Graphics/Icons used outside the Qt framework. *Files not listed.*

./share/qt - Materials related to Qt but not directly used by Core.
./share/qt/extract_strings_qt.py - Converts certain strings into Qt 4-friendly strings if needed.

./share/qt/Info.plist.in - OS X bundle kickoff file. Processed by AC_CONFIG_FILES() to generate ./share/qt/Info.plist, which is included in the root of the OS X build.
./share/qt/protobuf.pri - *qmake* file integrating Payment Protocol (BIP 70) with *protoc*. Requires OpenSSL & Qt.

./src - The subdirectory with all the Core-specific code.
./src/.clang-format - Sets the rules for *clang-format*, a program that formats C/C++/ObjC code. Added in Aug. 2014.
./src/addrman.cpp - Stochastic address manager (CAddrMan) and CAddress stats (CAddrInfo).
./src/addrman.h - See the CPP file.
./src/alert.cpp - Alerts for older versions of Core. There's an unsigned alert with all the data (CUnsignedAlert) and an unsigned alert + signature (CAlert).
./src/alert.h - See the CPP file.
./src/amount.cpp - Some basic amount (CAmount, or int64_t) definitions, and a type-safe wrapper class for fee rates (CFeeRate).
./src/amount.h - See the CPP file. Includes a bit of consensus-critical code.
./src/arith_uint256.cpp - Code for handling unsigned 256-bit big integers (arith_uint256).
./src/arith_uint256.h - See the CPP file.
./src/base58.cpp - Base58 support code, including CBitcoinAddress and CSecret, along with a derived (from CBase58Data) external class (CBitcoinExtKey) that, when typedef'd, is used for external, Base58-encoded private (CBitcoinExtKeyBase) and public (CBitcoinExtPubKeyBase) keys.
./src/base58.h - See the CPP file.
./src/bitcoin-cli.cpp - Binary used to communicate with / send commands to *bitcoind*.
./src/bitcoin-cli-res.rc - Resource definition script for Windows. Needed for the *bitcoin-cli* binary.
./src/bitcoind.cpp - Command line binary that executes Bitcoin Core functionality. Created in June 2013 to accommodate refactoring.
./src/bitcoind-res.rc - Resource definition script for Windows. Needed for the *bitcoind* binary.
./src/bitcoin-tx.cpp - Command line binary that can create, parse, or modify transactions.
./src/bitcoin-tx-res.rc - Resource definition script for Windows. Needed for the *bitcoin-tx* binary.
./src/bloom.cpp - Bloom filter material. Includes bloom filters (CBloomFilter) and "rolling" filters (CRollingBloomFilter), which tracks the last N items you've seen if you can tolerate a small number of false positives. Added in Jan. 2013, with rolling filters added in May 2015.
./src/bloom.h - See the CPP file.
./src/chain.cpp - Covers critical blockchain classes, on-disk and off-disk. There's the on-disk block position (CDiskBlockPos - *struct*), an entry (final or potential) in the blockchain (CBlockIndex), an on-disk blockchain entry (CDiskBlockIndex), and an in-memory indexed blockchain (CChain). Moved out of main.cpp & main.h in Sep. 2014.
./src/chain.h - See the CPP file.
./src/chainparams.cpp - The parameters for a chain (mainnet, testnet, regtest), including things like the genesis block. CChainParams is the primary class, with a few support structs also defined. Added in June 2013.
./src/chainparams.h - See the CPP file.

./src/chainparamsbase.cpp - A skeletal set of blockchain parameters (e.g., the communications port) for mainnet, testnet, and regtest. Used for network connections. Added in June 2014.
./src/chainparamsbase.h - See the CPP file.
./src/chainparamsseeds.h - Hard-coded “seed” locations. Generated by the devs. Added in Aug. 2014.
./src/checkpoints.cpp - Compiled-in sanity checks. Makes sure a node is on the right chain when getting bootstrapped and also indicates that blocks before a given checkpoint need not be fully verified, as they’re considered too deep in the blockchain to be unwound. Also prevents a DoS attack.
./src/checkpoints.h - See the CPP file.
./src/checkqueue.h - Queue for scripts to be verified. Parallelization optimization. Added in Jan. 2013.
./src/clientversion.cpp - Basic build version numbers & string.
./src/clientversion.h - See the CPP file.
./src/coincontrol.h - A class (CCoinControl) specifying a set of coins to be used for a given Tx. (This allows the user to control which coins are used.) Added in Nov. 2013.
./src/coins.cpp - Contains code for a pruned transaction that has only metadata and basically represents UTXOs (CCoins). Also includes various helper structs/classes that do things like hash the data, abstract the “view” of the UTXOs (CCoinsView and CCoinsViewCache), etc. (FIX - GET MORE INFO) Moved to the current position in Nov. 2013.
./src/coins.h - See the CPP file.
./src/compat.h - Code that handles Windows sockets.
./src/compressor.cpp - Compressors for common scripts (CScriptCompressor) and TxOuts (CTxOutCompressor).
./src/compressor.h - See the CPP file.
./src/core_io.h - CTransaction hex decode/encode. Needed by the *bitcoin-tx* binary.
./src/core_memusage.h - Core memory computation functions.
./src/core_read.cpp - CTransaction hex decode/encode. Needed by the *bitcoin-tx* binary.
./src/core_write.cpp - CTransaction hex decode/encode. Needed by the *bitcoin-tx* binary.
./src/dbwrapper.cpp - LevelDB wrapper code. Includes various CDB* classes related to accessing LevelDB.
./src/dbwrapper.h - See the CPP file.
./src/hash.cpp - Hash160 and Hash256 functions, both classes (CHash160/CHash256) and standalone functions.
./src/hash.h - See the CPP file.
./src/httprpc.cpp - Functions that start/stop the HTTP RPC and REST subsystems. *Added in 0.12.*
./src/httprpc.h - See the CPP file.
./src/httpserver.cpp - Functions related to the HTTP server. Also includes classes for in-flight HTTP requests (HTTPRequest), events (HTTPEvent), and event closure (HTTPClosure). *Added in 0.12.*
./src/httpserver.h - See the CPP file.

./src/init.cpp - Some startup/shutdown functions, help message functions, and license info. Also includes the declaration of the CClientUIInterface object used throughout the CLI and GUI code. (See ./src/ui_interface.h for more info.) Also includes important notes on thread management and startup/shutdown.
./src/init.h - See the CPP file.
./src/key.cpp - Private keys, encapsulated (CKey class) and serialized (CPrivKey typedef). There's also a CExtKey and CExtPubKey structs (basically the actual, Base58-encoded private/public keys without the metadata from CBitcoinExtKeyBase) and global ECC start/stop functions.
./src/key.h - See the CPP file.
./src/keystore.cpp - A key store base class (CKeyStore) and a store with address->secret maps (CBasicKeyStore).
./src/keystore.h - See the CPP file.
./src/limitedmap.h - STL-like map container class that only keeps the N elements with the highest value (limitedmap).
./src/main.cpp - Loads of critical functionality. Examples include script op count functions (GetP2SHSigOpCount & the like), checking inputs (CheckInputs), applying Tx effects to the UTXO (UpdateCoins), validity checks (CheckTransaction), consensus-critical functions telling if a Tx is final at a given time/location (IsFinalTx) or in the next block (CheckFinalTx), etc. Also includes the CScriptCheck, CBlockFileInfo, and CVerifyDB classes.
./src/main.h - See the CPP file. Includes loads of important definitions.
./src/Makefile.am - Autotool's Makefile recipe.
./src/Makefile.bench.include - Makefile include file for the *bench_bitcoin* binary.
./src/Makefile.qt.include - Some Makefile materials related to the ./src/qt subdirectory.
./src/Makefile.qttest.include - ./src/qt/test subdirectory.
./src/Makefile.test.include - ./src/test subdirectory.
./src/memusage.h - Some tools to track basic memory usage.
./src/merkleblock.cpp - Tied to Bloom filters. Classes covering partial merkle trees that covers a subset of the TXIDs in a block such that the complete list and the merkle root can be recovered (CPartialMerkleTree), and blocks used to relay headers & vector<merkle branch> to filtered nodes (CMerkleBlock). The blocks are what are sent to bloom filters (CBloomFilter).
./src/merkleblock.h - See the CPP file.
./src/miner.cpp - Some critical miner functionality (e.g., CreateNewBlock), and the CBlockTemplate struct.
./src/miner.h - See the CPP file.
./src/net.cpp - Loads of networking functionality. Various items include signals for message handling (CNodeSignals struct), node stats (CNodeStats), net messages (CNetMessage), entries in the ban database (banlist.dat) (CBanEntry), node information (CNode), and entries in the peer database (peers.dat) (CAddrDB).
./src/net.h - See the CPP file. Includes loads of value definitions and such.
./src/netbase.cpp - Some basic network items, including IPv4/IPv6 addresses (CNetAddr), subnets (CSubNet), IP address + port (CService), and the proxy type (proxyType).
./src/netbase.h - See the CPP file.

./src/noui.cpp - It looks like this is used solely to set up logging signal handlers for *bitcoind* and the test code.
./src/noui.h - See the CPP file.
./src/pow.cpp - Functions related to difficulty. Consensus-critical.
./src/pow.h - See the CPP file.
./src/prevector.h - vector<T> replacement that allocates N elements of T, switching to heap allocation only if more elements are needed. Used only by CScript. *Added in 0.12.*
./src/protocol.cpp - P2P protocol stuff. Includes the message header (CMessageHeader), IP:Port with peer info (CAddress), *Inv* message data (CInv), and NetMsgType and nServices enums.
./src/protocol.h - See the CPP file.
./src/pubkey.cpp - Public key analogue of ./src/key.cpp. Includes a CKey reference (CKeyID), encapsulated public keys (CPubKey), external public keys (CExtPubKey struct), and a handle for the ECC verification module (ECCVerifyHandle).
./src/pubkey.h - See the CPP file.
./src/random.cpp - RNG functions & materials. Uses OpenSSL.
./src/random.h - See the CPP file.
./src/rest.cpp - HTTP REST interface to public blockchain data. Added in Nov. 2014.
./src/reverselock.h - A class replacing boost::reverse_lock with a local reverse_lock class.
./src/rpcblockchain.cpp - RPC blockchain command functionality.
./src/rpcclient.cpp - RPC client functionality.
./src/rpcclient.h - See the CPP file.
./src/rpcmining.cpp - RPC mining command functionality, including *getblocktemplate()*.
./src/rpcmisc.cpp - RPC miscellaneous command functionality.
./src/rpcnet.cpp - RPC network command functionality.
./src/rpcprotocol.cpp - RPC protocol commands/enums/etc.
./src/rpcprotocol.h - See the CPP file.
./src/rpcrawtransaction.cpp - RPC raw transaction command functionality.
./src/rpcserver.cpp - RPC server functionality.
./src/rpcserver.h - See the CPP file.
./src/scheduler.cpp - Lightweight, simple scheduler for background tasks (CScheduler). Added in May 2015.
./src/scheduler.h - See the CPP file.
./src/serialize.h - General purpose data output materials, on-and-off-network? (FIX)
./src/streams.h - Double-ended buffer combining vector and stream-like interfaces (CDataStream), non-refcounted RAII wrapper for FILE*, without (CAutoFile) and with (CBufferedFile) ring buffers that allow the rewinding of X bytes.
./src/sync.cpp - Various synchronization-related mechanisms: Classes, #defs, typedefs, etc.
./src/sync.h - See the CPP file.
./src/threadsafety.h - Macro annotation code, documenting which locks protect a given piece of data.
./src/timedata.cpp - Median filter over a stream of values (CMedianFilter). Returns the median of the last N numbers. Also includes functions to keep track of adjusted P2P time.

./src/timedata.h - See the CPP file.
./src/tinyformat.h - Typesafe drop-in for *sprintf* and logging.
./src/torcontrol.cpp - Contains functions for enabling and disabling a Tor hidden service. The CPP includes various Tor-related classes used internally. *Added in 0.12.*
./src/torcontrol.h - See the CPP file.
./src/txdb.cpp - Tx DB, which is chainstate/ (coin DB) + CCoinsView (CCoinsViewDB). Includes related #defs and such.
./src/txdb.h - See the CPP file.
./src/txmempool.cpp - Mempool materials. The actual mempool (CTxMemPool), mempool entries (CTxMemPoolEntry), a CCoinsView that apparently reaches into the mempool to get coins (CCoinsViewMemPool), and lots of supports classes and structs.
./src/txmempool.h - See the CPP file. Lots and lots of important comments.
./src/ui_interface.h - Contains CClientUIInterface, the class that handles signals for UI communication.
./src/uint256.cpp - A template class for opaque blobs (base_blob), derived classes for 160-bit and 256-bit data (uint160 and uint256), and functs for handling 256-bit data strings (uint256S).
./src/uint256.h - See the CPP file.
./src/undo.h - Undo info for CTxIn (CTxInUndo), CTransaction (CTxUndo), and CBlock (CBlockUndo).
./src/util.cpp - Code related to the client/server environment. Used for argument handling, config file parsing, logging, thread wrappers, global environment setup, Windows network setup, etc.
./src/util.h - See the CPP file.
./src/utilmoneystr.cpp - Various money utilities (e.g., *ParseMoney()*).
./src/utilmoneystr.h - See the CPP file.
./src/utilstrencodings.cpp - String utilities (e.g., *DecodeBase32()*).
./src/utilstrencodings.h - See the CPP file.
./src/utiltime.cpp - Time utilities (e.g., *SetMockTime()*).
./src/utiltime.h - See the CPP file.
./src/validationinterface.cpp - Has a class (CValidationInterface) that allows modules to hook into startup, shutdown, and node events. Also includes struct (CMainSignals) with signals to listeners. Used to interact with toolslike ØMQ/ZeroMQ.
./src/validationinterface.h - See the CPP file.
./src/version.h - Various version definitions (Core, P2P, etc.).

./src/bench - Code used to build the *bench_bitcoin* binary, which does rudimentary code benchmarking. *Added in 0.12.*

./src/bench/.gitignore - Files for Git to ignore.
./src/bench/bench.cpp - Code that can be added to other code in order to perform basic benchmarking.
./src/bench/bench.h - See the CPP file.
./src/bench/bench_bitcoin.cpp - Code that kicks off the *bench_bitcoin* binary.
./src/bench/Examples.cpp - Examples of how to apply the benchmark code.

./src/compat - Added to allow Core binaries to be compiled on older computers. *glibc* & *libstdc++*, when compiled into Core on newer machines, will have symbols that are undefined when dynamically linked on older machines. This code can be compiled in to define the newer stuff while allowing dynamic linking for *glibc* & *libstdc++*. Added in Apr. 2014. Also general compatibility code.

./src/compat/byteswap.h - Endian compatibility.
./src/compat/endian.h - Endian compatibility.
./src/compat/glibc_compat.cpp - *glibc* compatibility.
./src/compat/glibc_sanity.cpp - Tests the environment sanity.
./src/compat/glibcxx_sanity.cpp - Tests the environment sanity.
./src/compat/sanity.h - Tests the environment sanity.
./src/compat/strnlen.cpp - Provides *strnlen*() if not present on the system.

./src/config - ./configure.ac generates ./src/config/bitcoin-config.h dynamically. AC_CONFIG_HEADERS defines the file, and all the output between AC_INIT and AC_OUTPUT is placed in the file.

./src/config/.empty - Empty file. Ensures that Git picks up the directory.

./src/consensus - Includes some consensus code *but not all code*, and also not part of *libbitcoinconsensus*.

./src/consensus/consensus.h - Has some important values (e.g., blocksize and sigops).
./src/consensus/merkle.cpp - The merkle tree computation code. *Added in 0.12.*
./src/consensus/merkle.h - See the CPP file.
./src/consensus/params.h - Introduces the Consensus namespace, which has some parameters that influence chin consensus.
./src/consensus/validation.h - Includes a class with info about the block/Tx validation state (CValidationState).

./src/crypto - Custom code for standard crypto functs (e.g., SHA-1)

./src/crypto/common.h - Endian-specific in/outs for 16, 32, and 64-bit data.
./src/crypto/hmac_sha256.cpp - HMAC-SHA-256 class (CHMAC_SHA256).
./src/crypto/hmac_sha256.h - See the CPP file.
./src/crypto/hmac_sha512.cpp - HMAC-SHA-512 class (CHMAC_SHA512).
./src/crypto/hmac_sha512.h - See the CPP file.
./src/crypto/ripemd160.cpp - RIPEMD-160 class (CRIPEMD160).
./src/crypto/ripemd160.h - See the CPP file.
./src/crypto/sha1.cpp - SHA-1 class (CSHA1).
./src/crypto/sha1.h - See the CPP file.
./src/crypto/sha256.cpp - SHA-256 class (CSHA256).
./src/crypto/sha256.h - See the CPP file.
./src/crypto/sha512.cpp - SHA-512 class (CSHA512).
./src/crypto/sha512.h - See the CPP file.

./src/leveldb - LevelDB code. *Will not list individual files (too many).*

./src/obj-test - Doesn't seem to be used anymore.
./src/obj-test/.gitignore - Files for Git to ignore.

./src/policy - Code related to various Core policies. Added in May 2015.
./src/policy/fees.cpp - Fee policies. Includes various constants, the class estimating the fee/priority required to get coins in a block (CBlockPolicyEstimator), and a class to track historical data on Tx confirmations (TxConfirmStats)
./src/policy/fees.h - See the CPP file. Loads of comments explaining the code.
./src/policy/policy.cpp - "Standard" functions (*IsStandard()*, *IsStandardTx()*, and *AreInputsStandard()*). *Added in 0.12.*
./src/policy/policy.h - Bitcoin policy constants and such. *Added in 0.12.*

./src/primitives - Some of the most basic building blocks of Bitcoin.
./src/primitives/block.cpp - Block classes (CBlock and CBlockHeader) and a struct showing where in the blockchain that a node, if it doesn't have a given branch, can find a common trunk.
./src/primitives/block.h - See the CPP file.
./src/primitives/transaction.cpp - Tx classes. Includes a Tx hash and an index *n* into the vout (COutPoint), a Tx input (CTxIn), a Tx output (CTxOut), the Tx placed on the network (CTransaction), and a struct CTransaction that's mutable (CMutableTransaction).
./src/primitives/transaction.h - See the CPP file.

./src/qt - Code related to the GUI version of Bitcoin Core. Code depends on the *Qt* framework.
./src/qt/addressbookpage.cpp - Widget showing a list of send/recv addresses (AddressBookPage).
./src/qt/addressbookpage.h - See the CPP file.
./src/qt/addresstablemodel.cpp - Qt model of the address book that allows views to access and modify the address book (AddressTableModel).
./src/qt/addresstablemodel.h - See the CPP file.
./src/qt/askpassphrasedialog.cpp - Multifunctional dialog to ask for passphrases (AskPassphraseDialog).
./src/qt/askpassphrasedialog.h - See the CPP file.
./src/qt/bantablemodel.cpp - Table showing banned nodes/services. *Added in 0.12.*
./src/qt/bantablemodel.h - See the CPP file.
./src/qt/bitcoin_locale.qrc - Locale data placed here to allow for deterministic builds.
./src/qt/bitcoin.cpp - The kickoff code for Bitcoin Core, using the GUI application's main object (BitcoinApplication).
./src/qt/bitcoin.qrc - Bitcoin app resource collection file.
./src/qt/bitcoinaddressvalidator.cpp - Validator for Base58 (BitcoinAddressEntryValidator) & Bitcoin addresses (BitcoinAddressCheckValidator).
./src/qt/bitcoinaddressvalidator.h - See the CPP file.
./src/qt/bitcoinamountfield.cpp - Widget for entering Bitcoin amounts (BitcoinAmountField).

./src/qt/bitcoinamountfield.h - See the CPP file.
./src/qt/bitcoingui.cpp - The main Bitcoin UI window (BitcoinGUI), communicating with the client & wallet models to give the user an up-to-date core state view.
./src/qt/bitcoingui.h - See the CPP file.
./src/qt/bitcoinstrings.cpp - Strings used by the GUI code.
./src/qt/bitcoinunits.cpp - Bitcoin unit definitions (BitcoinUnits).
./src/qt/bitcoinunits.h - See the CPP file.
./src/qt/clientmodel.cpp - Bitcoin network client model (ClientModel).
./src/qt/clientmodel.h - See the CPP file.
./src/qt/coincontroldialog.cpp - Coin control dialog (CoinControlDialog).
./src/qt/coincontroldialog.h - See the CPP file.
./src/qt/coincontroltreewidget.cpp - Coin control tree widget (CoinControlTreeWidget).
./src/qt/coincontroltreewidget.h - See the CPP file.
./src/qt/csvmodelwriter.cpp - Exports Qt table models to CSV files (CSVModelWriter), making data analysis via a spreadsheet easier.
./src/qt/csvmodelwriter.h - See the CPP file.
./src/qt/editaddressdialog.cpp - Dialog for editing addresses & associated info (EditAddressDialog).
./src/qt/editaddressdialog.h - See the CPP file.
./src/qt/guiconstants.h - Various GUI constants.
./src/qt/guiutil.cpp - Various GUI utility functions (GUIUtil).
./src/qt/guiutil.h - See the CPP file.
./src/qt/intro.cpp - Pre-GUI intro screen (Intro), allowing users to choose data directories & such.
./src/qt/intro.h - See the CPP file.
./src/qt/macdockiconhandler.h - See the MM file.
./src/qt/macdockiconhandler.mm - OS X-specific dock icon handler (MacDockIconHandler).
./src/qt/macnotificationhandler.h - See the MM file.
./src/qt/macnotificationhandler.mm - OS X-specific notification handler (MacNotificationHandler).
./src/qt/Makefile - Kicks off Bitcoin Core (GUI) builds in the parent directory.
./src/qt/networkstyle.cpp - Coin network-specific (e.g., testnet) style info for GUIs (Networkstyle).
./src/qt/networkstyle.h - See the CPP file.
./src/qt/notificator.cpp - Cross-platform desktop notification client (Notificator).
./src/qt/notificator.h - See the CPP file.
./src/qt/openuridialog.cpp - Open URI dialog (OpenURIDialog).
./src/qt/openuridialog.h - See the CPP file.
./src/qt/optionsdialog.cpp - Proxy address widget validator that checks for a valid proxy address (ProxyAddressValidator).
./src/qt/optionsdialog.h - See the CPP file.
./src/qt/optionsmodel.cpp - Interface from Qt to configuration data structure for Bitcoin client (OptionsModel).
./src/qt/optionsmodel.h - See the CPP file.
./src/qt/overviewpage.cpp - Home/Overview page widget (OverviewPage).
./src/qt/overviewpage.h - See the CPP file.

./src/qt/paymentrequest.proto - Payment request protocol file (Google Protocol Buffers).
./src/qt/paymentrequestplus.cpp - Wraps PaymentRequest (dumb protocol buffer) with PaymentDetails (extra methods) (PaymentRequestPlus).
./src/qt/paymentrequestplus.h - See the CPP file.
./src/qt/paymentserver.cpp - Handle payment requests from URI clicks (PaymentServer).
./src/qt/paymentserver.h - See the CPP file. Lots of important notes.
./src/qt/peertablemodel.cpp - Qt model providing info about connected peers (PeerTableModel).
./src/qt/peertablemodel.h - See the CPP file.
./src/qt/platformstyle.cpp - Class (PlatformStyle) that has coin network-specific GUI information. *Added in 0.12.*
./src/qt/platformstyle.h - See the CPP file.
./src/qt/qvalidatedlineedit.cpp - Line edit that can show input validation feedback as “invalid” (QValidatedLineEdit).
./src/qt/qvalidatedlineedit.h - See the CPP file.
./src/qt/qvaluecombobox.cpp - Combo box that can be used to select ordinal values from a model (QValueComboBox).
./src/qt/qvaluecombobox.h - See the CPP file.
./src/qt/receivecoinsdialog.cpp - Dialog for receiving Bitcoin payments (ReceiveCoinsDialog).
./src/qt/receivecoinsdialog.h - See the CPP file.
./src/qt/receiverequestdialog.cpp - Dialog for requesting Bitcoin payments (ReceiveRequestDialog) and a label widget for QR codes (QRImageWidget).
./src/qt/receiverequestdialog.h - See the CPP file.
./src/qt/recentrequeststablemodel.cpp - Model for list of recent payment request/URI generations (RecentRequestsTableModel), entries (RecentRequestEntry), and entries less than a certain value (RecentRequestEntryLessThan).
./src/qt/recentrequeststablemodel.h - See the CPP file.
./src/qt/rpcconsole.cpp - Local Bitcoin RPC console (RPCConsole).
./src/qt/rpcconsole.h - See the CPP file.
./src/qt/sendcoinsdialog.cpp - Dialog for sending Bitcoins (SendCoinsDialog).
./src/qt/sendcoinsdialog.h - See the CPP file.
./src/qt/sendcoinsentry.cpp - Entry in the “Send Bitcoins” dialog (SendCoinsEntry).
./src/qt/sendcoinsentry.h - See the CPP file.
./src/qt/signverifymessagedialog.cpp - Message sign/verification window (SignVerifyMessageDialog).
./src/qt/signverifymessagedialog.h - See the CPP file.
./src/qt/splashscreen.cpp - GUI splash screen w/ version info (SplashScreen).
./src/qt/splashscreen.h - See the CPP file.
./src/qt/trafficgraphwidget.cpp - Network traffic graph (TrafficGraphWidget).
./src/qt/trafficgraphwidget.h - See the CPP file.
./src/qt/transactiondesc.cpp - Human-readable HTML description of a Tx (TransactionDesc).
./src/qt/transactiondesc.h - See the CPP file.
./src/qt/transactiondescdialog.cpp - Dialog showing Tx details (TransactionDescDialog).
./src/qt/transactiondescdialog.h - See the CPP file.

./src/qt/transactionfilterproxy.cpp - Filter Tx list according to rules (TransactionFilterProxy).
./src/qt/transactionfilterproxy.h - See the CPP file.
./src/qt/transactionrecord.cpp - UI model for Tx status (TransactionStatus).
./src/qt/transactionrecord.h - See the CPP file.
./src/qt/transactiontablemodel.cpp - UI model for a wallet's Tx table (TransactionTableModel).
./src/qt/transactiontablemodel.h - See the CPP file.
./src/qt/transactionview.cpp - Widget showing a wallet's Tx list (TransactionView).
./src/qt/transactionview.h - See the CPP file.
./src/qt/utilitydialog.cpp - "Help message" (HelpMessageDialog) and "Shutdown" (ShutdownWindow) dialog boxes.
./src/qt/utilitydialog.h - See the CPP file.
./src/qt/walletframe.cpp - Embeds all wallet-related controls (WalletFrame).
./src/qt/walletframe.h - See the CPP file.
./src/qt/walletmodel.cpp - Interface to Bitcoin wallet from Qt view (WalletModel) and coin recipients (SendCoinsRecipient).
./src/qt/walletmodel.h - See the CPP file.
./src/qt/walletmodeltransaction.cpp - Data model for wallet model Txs (WalletModelTransaction).
./src/qt/walletmodeltransaction.h - See the CPP file.
./src/qt/walletview.cpp - Handles wallet page view renderings & updates (WalletView).
./src/qt/walletview.h - See the CPP file.
./src/qt/winshutdownmonitor.cpp - Catch Windows shutdown events (WinShutdownMonitor).
./src/qt/winshutdownmonitor.h - See the CPP file.

./src/qt/forms - *Qt* Designer files (XML) used to generate .h/.cpp files for graphics layouts.
./src/qt/forms/addressbookpage.ui - Address book dialog layout.
./src/qt/forms/askpassphrasedialog.ui - Wallet passphrase dialog layout.
./src/qt/forms/coincontroldialog.ui - Coin control address selection dialog layout.
./src/qt/forms/debugwindow.ui - RPC console/debug window layout.
./src/qt/forms/editaddressdialog.ui - Bitcoin address edit dialog layout.
./src/qt/forms/helpmessagedialog.ui - Help message dialog layout.
./src/qt/forms/intro.ui - Data directory selection dialog layout.
./src/qt/forms/openuridialog.ui - "Open URI" dialog layout.
./src/qt/forms/optionsdialog.ui - Options dialog layout.
./src/qt/forms/overviewpage.ui - Home/Overview dialog layout.
./src/qt/forms/receivecoinsdialog.ui - "Receive coins" dialog layout.
./src/qt/forms/receiverequestdialog.ui - "Request coins" dialog layout.
./src/qt/forms/sendcoinsdialog.ui - "Send coins" dialog layout.
./src/qt/forms/sendcoinsentry.ui - "Send coins" entry dialog layout.
./src/qt/forms/signverifymessagedialog.ui - "Sign/Verify message" dialog layout.
./src/qt/forms/transactiondescdialog.ui - Tx description dialog layout.

./src/qt/locale - Core translations. *Files not listed.*

./src/qt/res - Qt resources.
./src/qt/res/bitcoin-qt-res.rc - Windows resource file for the Bitcoin Core GUI.

./src/qt/res/icons - PNG graphics files for Core. *Files not listed.*

./src/qt/res/movies - PNG movie/animation files for Core. *Files not listed.*

./src/qt/res/src - SVG icon "source" files for Core. *Files not listed.*

./src/qt/test - Bitcoin Core (GUI) test code.
./src/qt/test/Makefile - Makefile kicking off Bitcoin Core (GUI) test build code in the parent directory.
./src/qt/test/paymentrequestdata.h - Data for ./src/qt/tests/paymentservertests.cpp.
./src/qt/test/paymentservertests.cpp - Payment Protocol (BIP 70) test code.
./src/qt/test/paymentservertests.h - See the CPP file.
./src/qt/test/test_main.cpp - Bitcoin Core (GUI) test suite kickoff file.
./src/qt/test/uritests.cpp - Bitcoin URI test code.
./src/qt/test/uritests.h - See the CPP file.

./src/script - Code that handles Bitcoin scripts. Moved to the current location in Sep. 2014 to make the code more modular.
./src/script/bitcoinconsensus.cpp - Some consensus-critical code, centered primarily around script verification.
./src/script/bitcoinconsensus.h - See the CPP file.
./src/script/interpreter.cpp - Covers transaction signature checking. BaseSignatureChecker class, which is the parent of TransactionSignatureChecker and MutableTransactionSignatureChecker. Loads of code and comments.
./src/script/interpreter.h - See the CPP file. Also includes many script hash & verification flags.
./src/script/script.cpp - Covers classes like the serialized script used in Tx inputs & outputs (CScript), and the class that enforces the arithmetic operation semantics of opcodes (CScriptNum).
./src/script/script.h - See the CPP file. Has the script enums and related script constants.
./src/script/script_error.cpp - Defines script errors.
./src/script/script_error.h - See the CPP file.
./src/script/sigcache.cpp - Caches signatures so that a Tx's signature doesn't have to be verified twice (entry into the mempool and into the blockchain). Includes a child of TransactionSignatureChecker (CachingTransactionSignatureChecker). The CPP file has several helper classes not in the header.
./src/script/sigcache.h - See the CPP file.
./src/script/sign.cpp - Signature creation. Includes the virtual base (BaseSignatureCreator), creator for transactions (TransactionSignatureCreator), creator of empty 72 byte signatures (DummySignatureCreator), and various helper functions.
./src/script/sign.h - See the CPP file.

./src/script/standard.cpp - Code related to standard transaction types (e.g., P2PKH, P2SH, etc.). Some functions and a Hash160 of the script's serialization (CScriptID).
./src/script/standard.h - See the CPP file.

./src/secp256k1 - Downstream version of the libsecp256k1 library. This is the library that performs all cryptographic functions related to creating and verifying signatures for Bitcoin transactions. *No files listed. Consult the libsecp256k1 doc or the project website.*

./src/support - Used primarily to abstract out some low-level functionality supplied by OpenSSL and Boost. Makes code changes a lot easier since changes occur only in one place.
./src/support/cleanse.cpp - Abstracts a memory "cleanse" provided by OpenSSL.
./src/support/cleanse.h - See the CPP file.
./src/support/pagelocker.cpp - Deals with memory locking. Includes an OS-dependent memory lock/unlock class (MemoryPageLocker), a thread-safe base class that keeps track of locked (i.e., non-swappable) memory pages (LockedPageManagerBase), and a derived singleton class that tracks locked memory pages for use in std::allocator templates (LockedPageManager).
./src/support/pagelocker.h - See the CPP file.

./src/support/allocators - Secure memory allocation/deallocation code.
./src/support/allocators/secure.h - Secure memory allocation (secure_allocator struct).
./src/support/allocators/zeroafterfree.h - Safe memory deallocation (zero_after_free_allocator struct).

./src/test - Unit tests for internal (i.e., not outward-facing, like RPC calls) source code. Uses Boost's test framework. Can be built with *make check* and run with the *test_bitcoin* binary.
./src/test/accounting_tests.cpp - Wallet accounting entry tests.
./src/test/addrman_tests.cpp - Tests for the CAddrMan class. *Added in 0.12.*
./src/test/alert_tests.cpp - Alert system (CAlert) unit tests.
./src/test/allocator_tests.cpp - Memory allocation tests. Has to do with memory "stacking."
./src/test/arith_uint256_tests.cpp - uint256 ("opaque" and "arithmetic") testing.
./src/test/base32_tests.cpp - Base32 unit tests.
./src/test/base58_tests.cpp - Base58 unit tests.
./src/test/base64_tests.cpp - Base64 unit tests.
./src/test/bctest.py - Support code for *bitcoin-tx* binary test.
./src/test/bip32_tests.cpp - Hierarchical deterministic (HD) wallet unit tests.
./src/test/bitcoin-util-test.py - Kicks off *bitcoin-tx* binary test.
./src/test/bloom_tests.cpp - CBloomFilter and CMerkleBlock unit tests.
./src/test/buildenv.py.in - Helps allow Python tests to run on Windows.
./src/test/checkblock_tests.cpp - Now-obsolete test that was used to confirm that Core would handle 1 MB blocks after the temporary 500 KB soft fork imposed after the Mar. 2013 hard fork.
./src/test/Checkpoints_tests.cpp - A checkpoint-related unit test.
./src/test/coins_tests.cpp - Tests for CCoinsView and related classes.
./src/test/compress_tests.cpp - Tests for amount serializer/deserializer code.

./src/test/crypto_tests.cpp - SHA (regular & HMAC) and RIPEMD-160 unit tests.
./src/test/dbwrapper_tests.cpp - Tests for the LevelDB wrappers.
./src/test/DoS_tests.cpp - Denial of Service unit tests.
./src/test/getarg_tests.cpp - *GetArg()* and *GetBoolArg()* unit tests.
./src/test/hash_tests.cpp - *MurmurHash3()* unit tests. Useful for Bloom filter work.
./src/test/key_tests.cpp - EC key unit tests.
./src/test/limitedmap_tests.cpp - Tests for the *limitedmap* class.
./src/test/main_tests.cpp - Unit tests related to money supplies.
./src/test/Makefile - Used to kick off the *bitcoin_test* binary build elsewhere.
./src/test/mempool_tests.cpp - Mempool unit tests.
./src/test/miner_tests.cpp - Various miner-related unit tests.
./src/test/multisig_tests.cpp - Various multisig-related unit tests.
./src/test/netbase_tests.cpp - Some unit tests for basic network functionality (netbase).
./src/test/pmt_tests.cpp - CPartialMerkleTree (Bloom filter) unit tests.
./src/test/policyestimator_tests.cpp - Policy fee estimation unit tests.
./src/test/pow_tests.cpp - Unit tests for next difficulty calculations.
./src/test/prevector_tests.cpp - Tests for the *prevector* class.
./src/test/README.md - Explains how to run the unit tests.
./src/test/reverselock_tests.cpp - Tests for the *reverse_lock* class.
./src/test/rpc_tests.cpp - Internal RPC/JSON unit tests.
./src/test/rpc_wallet_tests.cpp - Internal RPC/JSON wallet unit tests.
./src/test/sanity_tests.cpp - Compiler sanity check unit tests.
./src/test/scriptnum10.h - Class implementing the *ScriptNum* implementation from Core 0.10.0. Used for comparison purposes.
./src/test/scheduler_tests.cpp - CScheduler unit tests.
./src/test/script_P2SH_tests.cpp - Unit tests for the inner workings of P2SH scripts.
./src/test/script_tests.cpp - Unit tests for general script validity.
./src/test/scriptnum_tests.cpp - CScriptNum unit tests.
./src/test/serialize_tests.cpp - CVarInt unit tests.
./src/test/sighash_tests.cpp - Sighash flag unit tests.
./src/test/sigopcount_tests.cpp - Sigop count unit tests.
./src/test/skiplist_tests.cpp - Skip list (related to CBlockIndex/headers-first block propagation) unit tests.
./src/test/streams_tests.cpp - Tests the *CDataStream* class. *Added in 0.12.*
./src/test/test_bitcoin.cpp - Some basic test setup code. Mainly used to configure logging and chain parameters.
./src/test/test_bitcoin.h - See the CPP file.
./src/test/timedata_tests.cpp - CMedianFilter unit tests.
./src/test/transaction_tests.cpp - Transaction unit tests.
./src/test/txvalidationcache_tests.cpp - Test ensuring that a block with a double spend in it doesn't pass validation if half of the double spend is already in the memory pool (so full-blown transaction validation is skipped) when the block is received.
./src/test/uint256_tests.cpp - Unit tests for uint256, arith_uint256 and related classes.

./src/test/univalue_tests.cpp - Unit tests for the UniValue class.
./src/test/util_tests.cpp - Unit tests for various utility-related files.

./src/test/data - Data for unit tests. Data is built into the binary.
./src/test/data/alertTests.raw - Alert data used by ./src/test/alert_tests.cpp. Data generated by ./src/Makefile.test.include.
./src/test/data/base58_encode_decode.json - Base58 test data used by ./src/tests/base58_tests.cpp to confirm that encoding & decoding work. Generated by ./contrib/testgen/gen_base58_test_vectors.py.
./src/test/data/base58_keys_invalid.json - Base58 test data used by ./src/tests/base58_tests.cpp to confirm that the given Base58 keys are invalid across various key types. Generated by ./contrib/testgen/gen_base58_test_vectors.py.
./src/test/data/base58_keys_valid.json - Base58 test data used by ./src/tests/base58_tests.cpp to confirm that the given Base58 keys are valid across various key types. Generated by ./contrib/testgen/gen_base58_test_vectors.py.
./src/test/data/bitcoin-util-test.json - Test data for the *bitcoin-tx* binary test. Used by ./src/test/bitcoin-util-test.py.
./src/test/data/blanktx.hex - Test data for the *bitcoin-tx* binary test. Used by ./src/test/data/bitcoin-util-test.json.
./src/test/data/README.md - Mentions that this is data for various tests.
./src/test/data/script_invalid.json - Invalid scripts. Used by ./src/tests/script_tests.cpp.
./src/test/data/script_valid.json - Valid scripts. Used by ./src/tests/script_tests.cpp.
./src/test/data/sighash.json - Sighash data. Used by ./src/tests/sighash_tests.cpp.
./src/test/data/tt-delin1-out.hex - *bitcoin-tx -delin=1* test data used for output comparison. Used by ./src/test/data/bitcoin-util-test.json.
./src/test/data/tt-delout1-out.hex - *bitcoin-tx -delout=1* test data used for output comparison. Used by ./src/test/data/bitcoin-util-test.json.
./src/test/data/tt-locktime317000-out.hex - *bitcoin-tx -lockout=317000* test data used for output comparison. Used by ./src/test/data/bitcoin-util-test.json.
./src/test/data/tx_invalid.json - *bitcoin-tx -delin=1* test data. Consists of invalid, deserialized transactions. Used by ./src/test/data/bitcoin-util-test.json.
./src/test/data/tx_valid.json - *bitcoin-tx -delin=1* test data. Consists of valid, deserialized transactions. Used by ./src/test/data/bitcoin-util-test.json.
./src/test/data/tx394b54bb.hex - *bitcoin-tx* input test data. Used by ./src/test/data/bitcoin-util-test.json.
./src/test/data/txcreate1.hex - *bitcoin-tx -create *loads of flags** input test data. Used by ./src/test/data/bitcoin-util-test.json.
./src/test/data/txcreate2.hex - *bitcoin-tx -create outscript=0:* (i.e., null scriptPubKey) test data used for output comparison. Used by ./src/test/data/bitcoin-util-test.json.
./src/test/data/txcreatedata1.hex - Data for tests regarding data-based outputs (OP_RETURN) from *bitcoin-tx*.
./src/test/data/txcreatedata2.hex - Data for tests regarding data-based outputs (OP_RETURN) from *bitcoin-tx*.

./src/test/data/txcreatesign.hex - *bitcoin-tx -create *loads of flags** input test data. Used by ./src/test/data/bitcoin-util-test.json, with a later change to fix some issues.

./src/univalue - Downstream version of the libunivalue library. Objects are used for parsing and encoding JSON data. Replaced JSON Spirit. *No files listed. Consult the project website.*

./src/wallet - Core's wallet functionality. Any code here should be used solely by the wallet.
./src/wallet/crypter.cpp - Includes the master key class for a wallet's private key encryption (CMasterKey), encryption/decryption context class w/ key info (CCrypter), and a keystore keeping the private keys (CCryptoKeyStore).
./src/wallet/crypter.h - See the CPP file.
./src/wallet/db.cpp - Classes providing access to a BerkeleyDB instance (CDB) and the DB's environment (CDBEnv).
./src/wallet/db.h - See the CPP file.
./src/wallet/rpcdump.cpp - RPC functions related to exporting/importing wallet info, addresses, etc.
./src/wallet/rpcwallet.cpp - RPC functionality not related to exporting/importing wallet info & such.
./src/wallet/wallet.cpp - The basic wallet functionality. Includes many constants & policy defaults, keypool entries (CKeyPool), address book data (CAddressBookData), Tx w/ merkle branch linking the Tx to the blockchain (CMerkleTx), Tx w/ only info needed by the owner (CWalletTx), wallet Tx as represented in a TxOut (COutput), private wallet key (CWalletKey), a keystore extension w/ Tx/balance info (CWallet), keys allocated from the keypool (CReserveKey), account info (CAccount), internal accounting info (CAccountEntry), and many structs. The accounting stuff is more-or-less an abandoned concept from Core's early days.
./src/wallet/wallet.h - See the CPP file.
./src/wallet/wallet_ismine.cpp - Helps determine the wallet type and how many keys are in the wallet.
./src/wallet/wallet_ismine.h - See the CPP file.
./src/wallet/walletdb.cpp - Classes that provide key metadata (CKeyMetadata) and a derived class providing access to the wallet DB (wallet.dat) (CWalletDB).
./src/wallet/walletdb.h - See the CPP file.

./src/wallet/test - Test code for the wallet.
./src/wallet/test/wallet_tests.cpp - Test code for the wallet.

./src/zmq - Code supporting ØMQ. *Added in 0.12.*
./src/zmq/zmqabstractnotifier.cpp - Contains a pure virtual notifier class (CZMQAbstractNotifier).
./src/zmq/zmqabstractnotifier.h - See the CPP file.
./src/zmq/zmqconfig.h - Minimal file. Really seems useful only for the *zmqError()* prototype.
./src/zmq/zmqnotificationinterface.cpp - Has a derived notification interface class (CZMQNotificationInterface).
./src/zmq/zmqnotificationinterface.h - See the CPP file.

./src/zmq/zmqpublishnotifier.cpp - Covers publishing for each covered event type. Has a derived virtual class (CZQAbstractPublishNotifier) that's used as the base for the classes covering the four event types: block hash (CZMQPublishHashBlockNotifier), raw block (CZMQPublishRawBlockNotifier), Tx hash (CZMQPublishHashTransactionNotifier), and raw Tx (CZMQPublishRawTransactionNotifier).
./src/zmq/zmqpublishnotifier.h - See the CPP file.