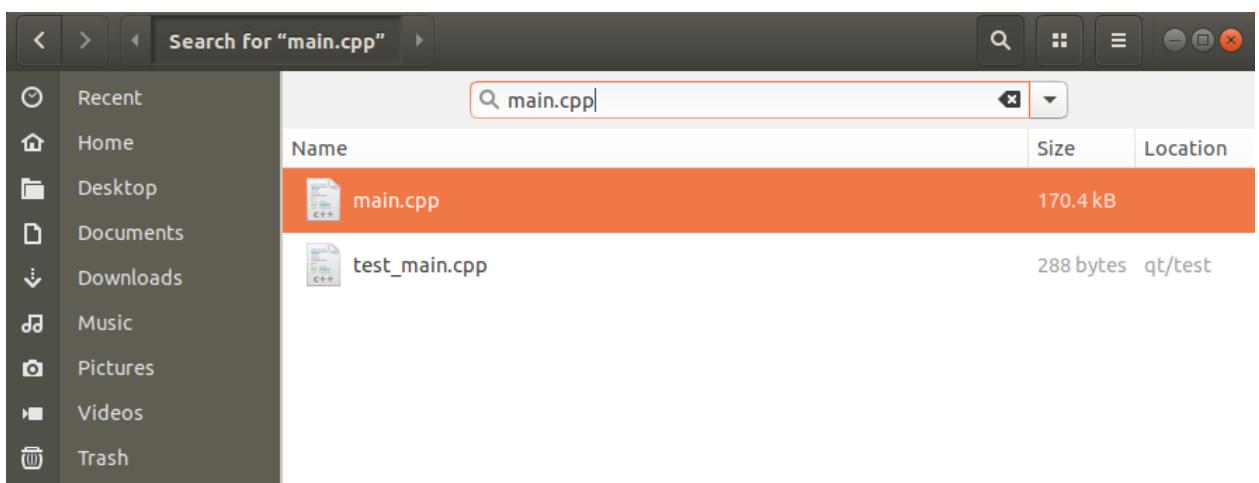


Lab 21-6-2023

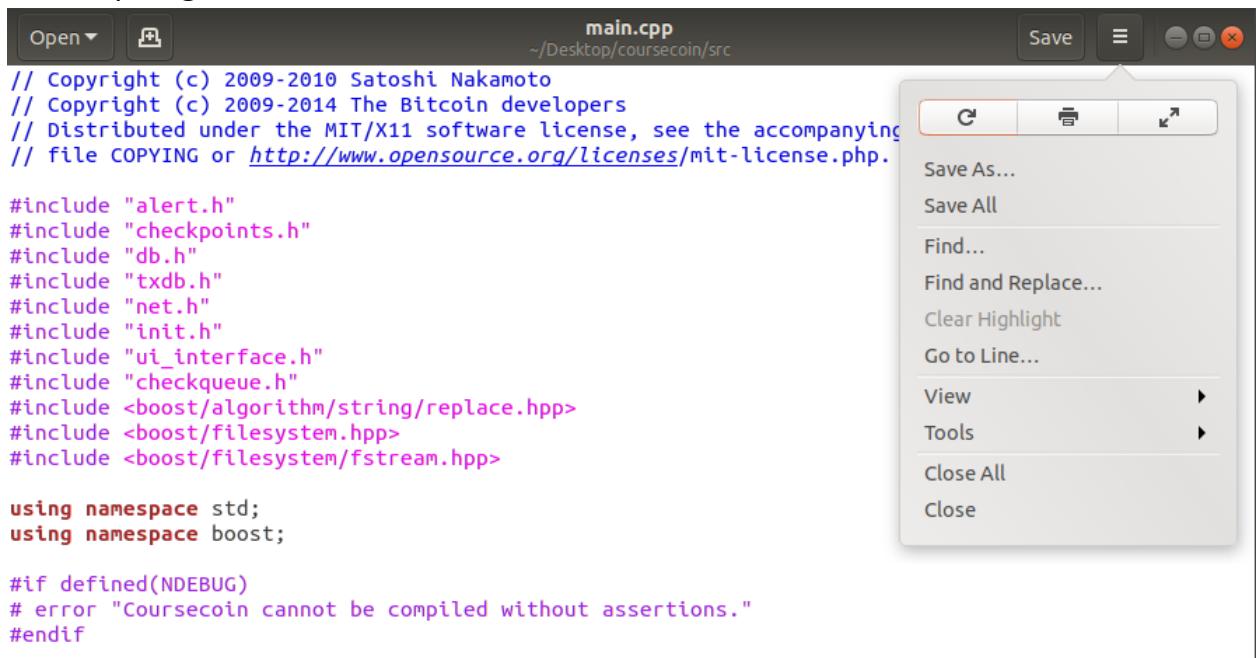
[**Note:** This hands on exercise is part of a series of ongoing labs on creating your own cryptocurrency and wallet interface and in order to do this lab please make sure that you have **completed Lab 16-6-2023, Lab 19-6-2023, Lab 20-6-2023** in a sequential order]

1. Go to your **coin** directory and inside your **src** directory using file explorer and search for a file named “**main.cpp**”. Open it by double clicking it.



```
// Copyright (c) 2009-2010 Satoshi Nakamoto
// Copyright (c) 2009-2014 The Bitcoin developers
// Distributed under the MIT/X11 software license, see the accompanying
// file COPYING or http://www.opensource.org/licenses/mit-license.php.
#include "alert.h"
#include "checkpoints.h"
#include "db.h"
#include "txdb.h"
#include "net.h"
#include "init.h"
#include "ui_interface.h"
#include "checkqueue.h"
#include <boost/algorithm/string/replace.hpp>
#include <boost/filesystem.hpp>
#include <boost/filesystem/fstream.hpp>
```

Once open go to line 1090

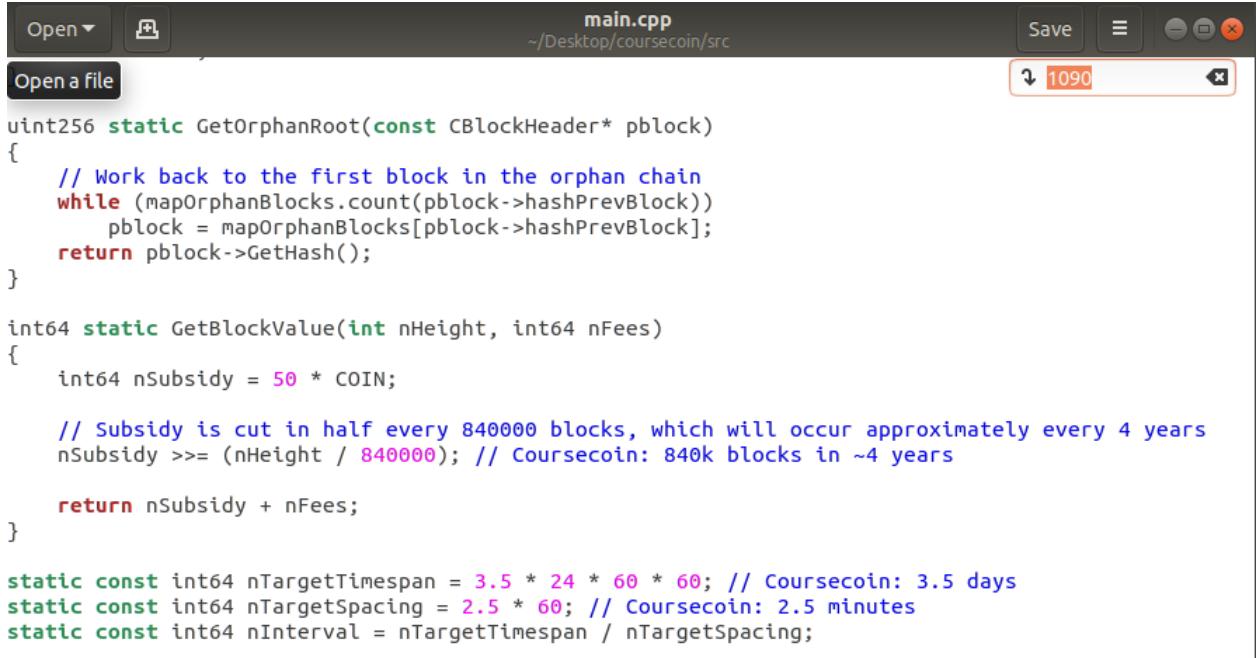


```
// Copyright (c) 2009-2010 Satoshi Nakamoto
// Copyright (c) 2009-2014 The Bitcoin developers
// Distributed under the MIT/X11 software license, see the accompanying
// file COPYING or http://www.opensource.org/licenses/mit-license.php.
//
#include "alert.h"
#include "checkpoints.h"
#include "db.h"
#include "txdb.h"
#include "net.h"
#include "init.h"
#include "ui_interface.h"
#include "checkqueue.h"
#include <boost/algorithm/string/replace.hpp>
#include <boost/filesystem.hpp>
#include <boost/filesystem/fstream.hpp>

using namespace std;
using namespace boost;

#if defined(NDEBUG)
# error "Coursecoin cannot be compiled without assertions."
#endif
```

You'll see this snippet of code



```
uint256 static GetOrphanRoot(const CBlockHeader* pblock)
{
    // Work back to the first block in the orphan chain
    while (mapOrphanBlocks.count(pblock->hashPrevBlock))
        pblock = mapOrphanBlocks[pblock->hashPrevBlock];
    return pblock->GetHash();
}

int64 static GetBlockValue(int nHeight, int64 nFees)
{
    int64 nSubsidy = 50 * COIN;

    // Subsidy is cut in half every 840000 blocks, which will occur approximately every 4 years
    nSubsidy >>= (nHeight / 840000); // Coursecoin: 840k blocks in ~4 years

    return nSubsidy + nFees;
}

static const int64 nTargetTimespan = 3.5 * 24 * 60 * 60; // Coursecoin: 3.5 days
static const int64 nTargetSpacing = 2.5 * 60; // Coursecoin: 2.5 minutes
static const int64 nInterval = nTargetTimespan / nTargetSpacing;
```

Change the block mining reward denoted by **nSubsidy from 50 to 25**, change the number of blocks after which subsidy will be halved from **840000 to 420000**, change the timespan for readjustment of difficulty level from **3.5 days to 3 days**. Lastly, change the target block time from **2.5 minutes to 2 minutes**.

```

int64 static GetBlockValue(int nHeight, int64 nFees)
{
    int64 nSubsidy = 25 * COIN;

    // Subsidy is cut in half every 420000 blocks, which will occur approximately every 2 years
    nSubsidy >>= (nHeight / 420000); // Coursecoin: 420k blocks in ~2 years

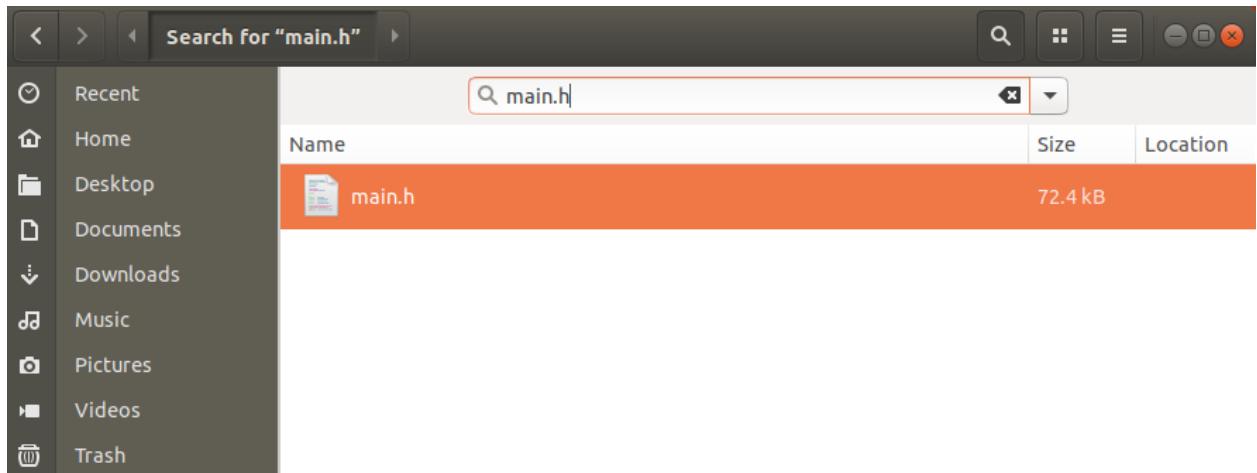
    return nSubsidy + nFees;
}

static const int64 nTargetTimespan = 3 * 24 * 60 * 60; // Coursecoin: 3 days
static const int64 nTargetSpacing = 2 * 60; // Coursecoin: 2.5 minutes
static const int64 nInterval = nTargetTimespan / nTargetSpacing;

```

Save the main.cpp and close it.

2. Go to the **src** directory within your **coin** directory and search for a file named “**main.h**”. Open it by double clicking it.



The screenshot shows a code editor window with the file "main.h" open. The file contains the following code:

```

// Copyright (c) 2009-2010 Satoshi Nakamoto
// Copyright (c) 2009-2014 The Bitcoin developers
// Distributed under the MIT/X11 software license, see the accompanying
// file COPYING or http://www.opensource.org/licenses/mit-license.php.
#ifndef BITCOIN_MAIN_H
#define BITCOIN_MAIN_H

#include "bignum.h"
#include "sync.h"
#include "net.h"
#include "script.h"
#include "scrypt.h"

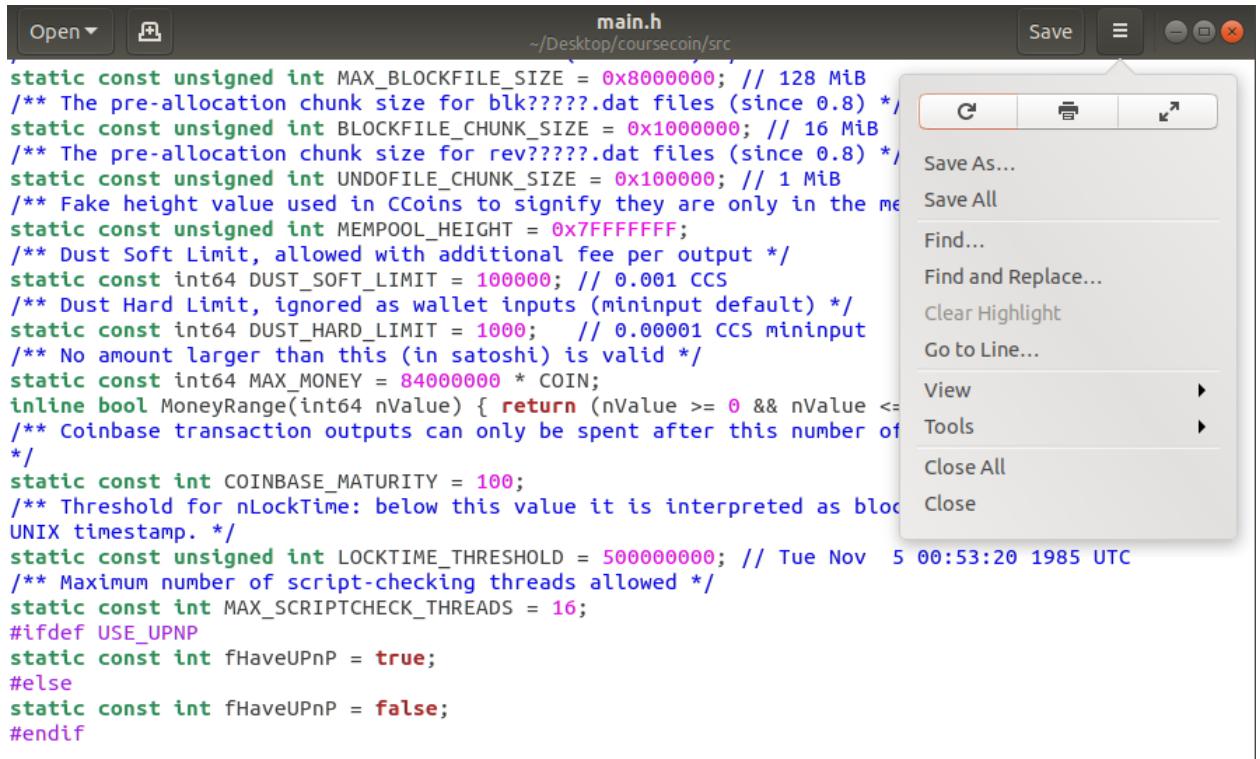
#include <list>

class CWallet;
class CBlock;
class CBlockIndex;
class CKeyItem;
class CReserveKey;

class CAddress;
class CInv;
class CNode;

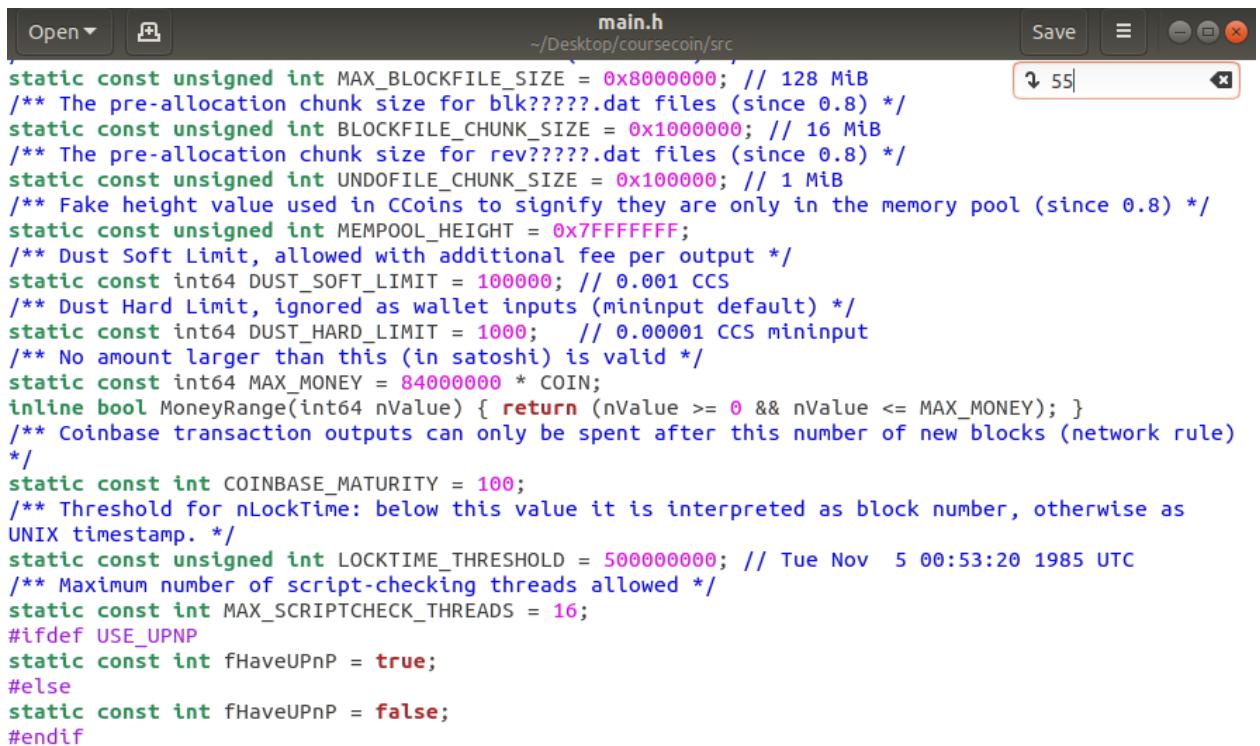
```

Go to line 55



```
main.h
~/Desktop/coursecoin/src

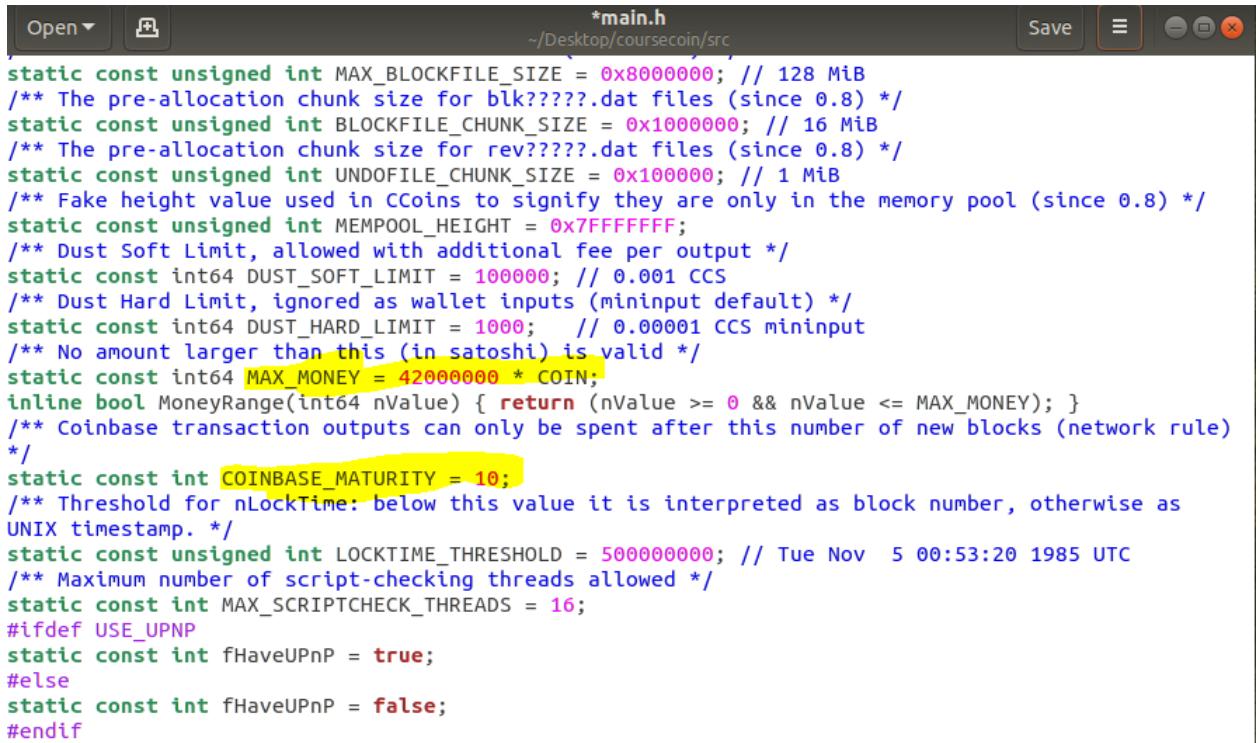
static const unsigned int MAX_BLOCKFILE_SIZE = 0x8000000; // 128 MiB
/** The pre-allocation chunk size for blk?????.dat files (since 0.8) */
static const unsigned int BLOCKFILE_CHUNK_SIZE = 0x1000000; // 16 MiB
/** The pre-allocation chunk size for rev?????.dat files (since 0.8) */
static const unsigned int UNDOFILE_CHUNK_SIZE = 0x100000; // 1 MiB
/** Fake height value used in CCoins to signify they are only in the memory pool (since 0.8) */
static const unsigned int MEMPOOL_HEIGHT = 0x7FFFFFFF;
/** Dust Soft Limit, allowed with additional fee per output */
static const int64 DUST_SOFT_LIMIT = 100000; // 0.001 CCS
/** Dust Hard Limit, ignored as wallet inputs (mininput default) */
static const int64 DUST_HARD_LIMIT = 1000; // 0.00001 CCS mininput
/** No amount larger than this (in satoshi) is valid */
static const int64 MAX_MONEY = 84000000 * COIN;
inline bool MoneyRange(int64 nValue) { return (nValue >= 0 && nValue <= MAX_MONEY); }
/** Coinbase transaction outputs can only be spent after this number of new blocks (network rule) */
static const int COINBASE_MATURITY = 100;
/** Threshold for nLockTime: below this value it is interpreted as block number, otherwise as UNIX timestamp. */
static const unsigned int LOCKTIME_THRESHOLD = 5000000000; // Tue Nov 5 00:53:20 1985 UTC
/** Maximum number of script-checking threads allowed */
static const int MAX_SCRIPTCHECK_THREADS = 16;
#ifndef USE_UPNP
static const int fHaveUPnP = true;
#else
static const int fHaveUPnP = false;
#endif
```



```
main.h
~/Desktop/coursecoin/src
↳ 55

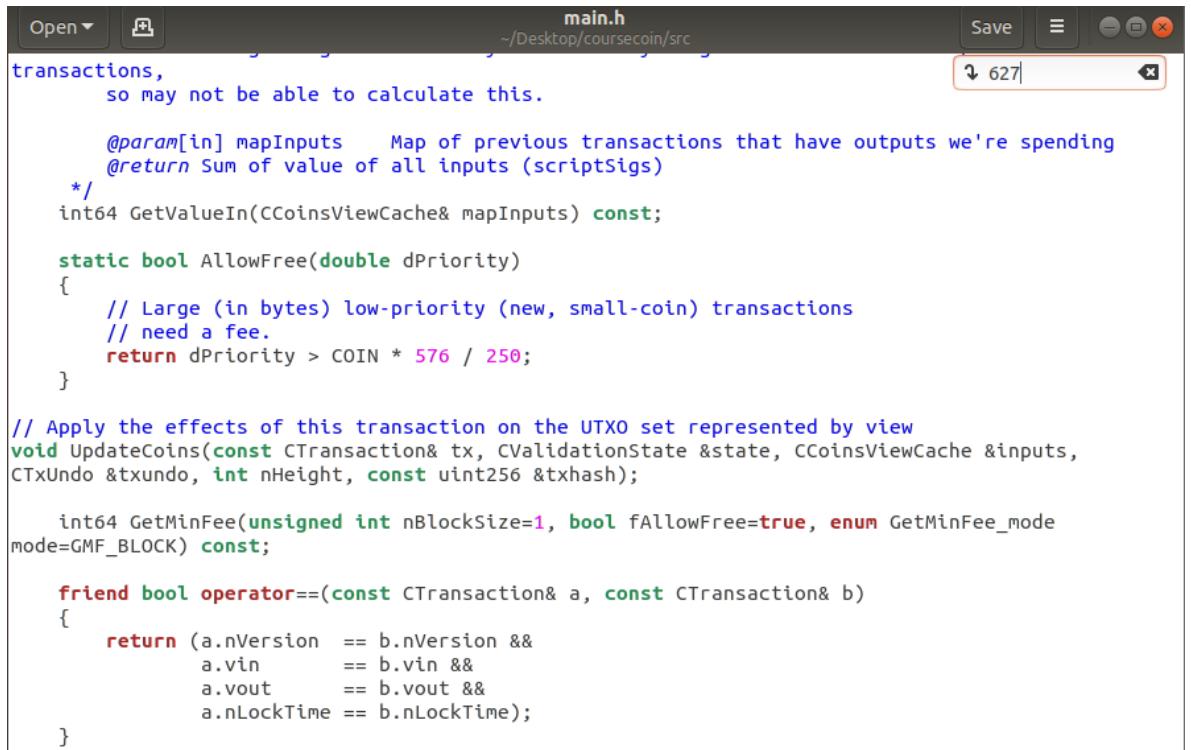
static const unsigned int MAX_BLOCKFILE_SIZE = 0x8000000; // 128 MiB
/** The pre-allocation chunk size for blk?????.dat files (since 0.8) */
static const unsigned int BLOCKFILE_CHUNK_SIZE = 0x1000000; // 16 MiB
/** The pre-allocation chunk size for rev?????.dat files (since 0.8) */
static const unsigned int UNDOFILE_CHUNK_SIZE = 0x100000; // 1 MiB
/** Fake height value used in CCoins to signify they are only in the memory pool (since 0.8) */
static const unsigned int MEMPOOL_HEIGHT = 0x7FFFFFFF;
/** Dust Soft Limit, allowed with additional fee per output */
static const int64 DUST_SOFT_LIMIT = 100000; // 0.001 CCS
/** Dust Hard Limit, ignored as wallet inputs (mininput default) */
static const int64 DUST_HARD_LIMIT = 1000; // 0.00001 CCS mininput
/** No amount larger than this (in satoshi) is valid */
static const int64 MAX_MONEY = 84000000 * COIN;
inline bool MoneyRange(int64 nValue) { return (nValue >= 0 && nValue <= MAX_MONEY); }
/** Coinbase transaction outputs can only be spent after this number of new blocks (network rule) */
static const int COINBASE_MATURITY = 100;
/** Threshold for nLockTime: below this value it is interpreted as block number, otherwise as UNIX timestamp. */
static const unsigned int LOCKTIME_THRESHOLD = 5000000000; // Tue Nov 5 00:53:20 1985 UTC
/** Maximum number of script-checking threads allowed */
static const int MAX_SCRIPTCHECK_THREADS = 16;
#ifndef USE_UPNP
static const int fHaveUPnP = true;
#else
static const int fHaveUPnP = false;
#endif
```

Change the maximum number of coins denoted by **MAX_MONEY** from **84 million to 42 million**, Change the **COINBASE_MATURITY** (Number of mined blocks after which you can spend your mining reward) from **100 to 10**.



```
*main.h
~/Desktop/coursecoin/src
static const unsigned int MAX_BLOCKFILE_SIZE = 0x8000000; // 128 MiB
/** The pre-allocation chunk size for blk?????.dat files (since 0.8) */
static const unsigned int BLOCKFILE_CHUNK_SIZE = 0x100000; // 16 MiB
/** The pre-allocation chunk size for rev?????.dat files (since 0.8) */
static const unsigned int UNDOFILE_CHUNK_SIZE = 0x10000; // 1 MiB
/** Fake height value used in CCoin to signify they are only in the memory pool (since 0.8) */
static const unsigned int MEMPOOL_HEIGHT = 0x7FFFFFFF;
/** Dust Soft Limit, allowed with additional fee per output */
static const int64 DUST_SOFT_LIMIT = 100000; // 0.001 CCS
/** Dust Hard Limit, ignored as wallet inputs (mininput default) */
static const int64 DUST_HARD_LIMIT = 1000; // 0.00001 CCS mininput
/** No amount larger than this (in satoshi) is valid */
static const int64 MAX_MONEY = 42000000 * COIN;
inline bool MoneyRange(int64 nValue) { return (nValue >= 0 && nValue <= MAX_MONEY); }
/** Coinbase transaction outputs can only be spent after this number of new blocks (network rule) */
static const int COINBASE_MATURITY = 10;
/** Threshold for nLockTime: below this value it is interpreted as block number, otherwise as UNIX timestamp. */
static const unsigned int LOCKTIME_THRESHOLD = 5000000000; // Tue Nov 5 00:53:20 1985 UTC
/** Maximum number of script-checking threads allowed */
static const int MAX_SCRIPTCHECK_THREADS = 16;
#ifndef USE_UPNP
static const int fHaveUPnP = true;
#else
static const int fHaveUPnP = false;
#endif
```

Save the file and Go to line 627, you'll see this snippet of code



```
main.h
~/Desktop/coursecoin/src
transactions,
    so may not be able to calculate this.

    @param[in] mapInputs Map of previous transactions that have outputs we're spending
    @return Sum of value of all inputs (scriptSigs)
*/
int64 GetValueIn(CCoinsViewCache& mapInputs) const;

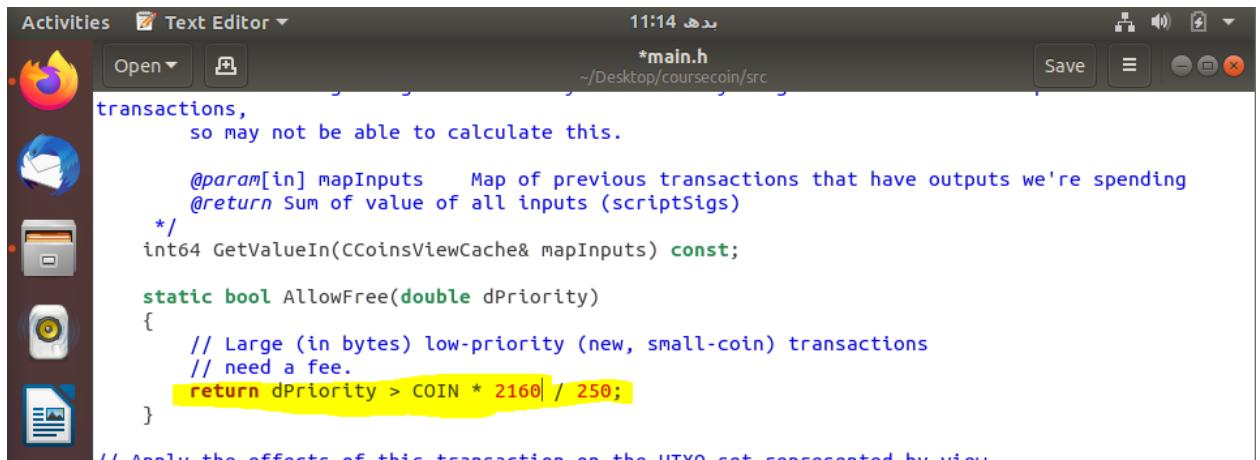
static bool AllowFree(double dPriority)
{
    // Large (in bytes) low-priority (new, small-coin) transactions
    // need a fee.
    return dPriority > COIN * 576 / 250;
}

// Apply the effects of this transaction on the UTXO set represented by view
void UpdateCoins(const CTransaction& tx, CValidationState &state, CCoinViewCache &inputs,
CTxUndo &txundo, int nHeight, const uint256 &txhash);

int64 GetMinFee(unsigned int nBlockSize=1, bool fAllowFree=true, enum GetMinFee_mode
mode=GMF_BLOCK) const;

friend bool operator==(const CTransaction& a, const CTransaction& b)
{
    return (a.nVersion == b.nVersion &&
            a.vin == b.vin &&
            a.vout == b.vout &&
            a.nLockTime == b.nLockTime);
}
```

Change the numerator value in **dPriority** to **2160** from **576**



A screenshot of a Linux desktop environment. On the left is a dock with icons for a browser, file manager, terminal, and others. In the center, a terminal window titled "Text Editor" shows code in a file named "main.h". The code includes a function "GetValueIn" with a condition that highlights the line "return dPriority > COIN * 2160 / 250;" in yellow. The number "2160" is also highlighted in yellow. The file path is "/Desktop/coursecoin/src". On the right, a file manager window titled "Search for 'main.cpp'" shows a list of files. The "main.cpp" file is selected and highlighted in orange, showing its size as 170.4 kB.

```
Activities Text Editor 11:14 11:14 ~
Open ~/Desktop/coursecoin/src *main.h Save
transactions,
    so may not be able to calculate this.

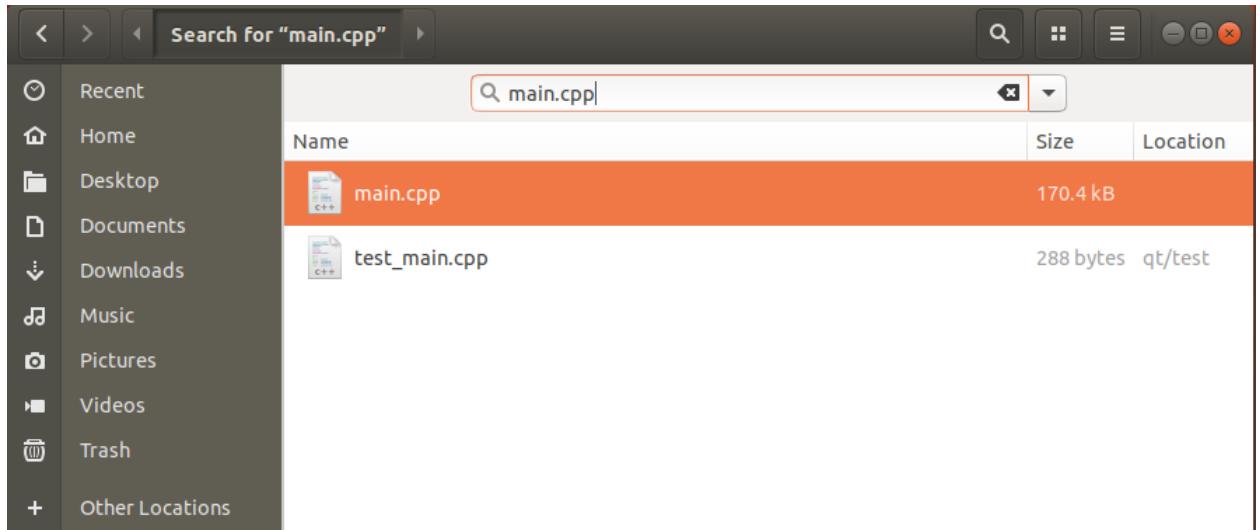
@param[in] mapInputs Map of previous transactions that have outputs we're spending
@return Sum of value of all inputs (scriptSigs)
*/
int64 GetValueIn(CCoinsViewCache& mapInputs) const;

static bool AllowFree(double dPriority)
{
    // Large (in bytes) low-priority (new, small-coin) transactions
    // need a fee.
    return dPriority > COIN * 2160 / 250;
}

// Apply the effects of this transaction on the UTXO set represented by view.
```

Save the **main.h** file and close it.

3. Go to the **src** directory within your **coin** directory using file explorer and search for a file named “**main.cpp**”. Open it by double clicking it.



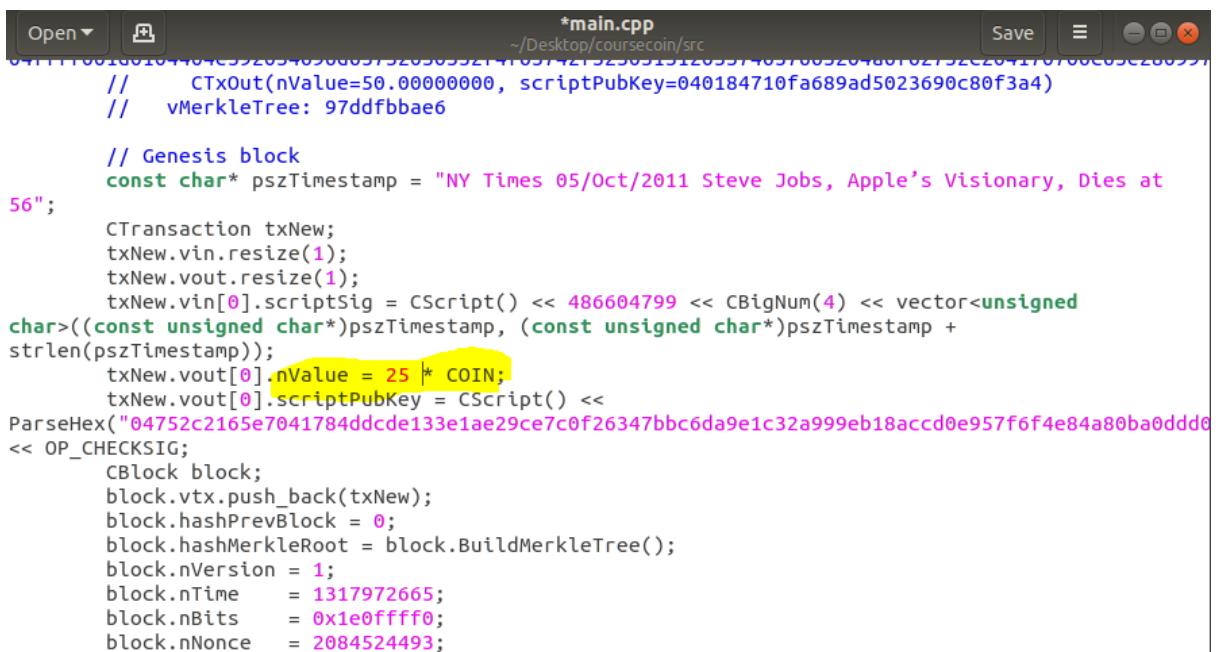
Go to **line 2787** and you'll see this snippet of code.



```
main.cpp
~/Desktop/coursecoin/src
0417100104404c5920540900057520505521410574215250515120557405700520480102752
//      CTxOut(nValue=50.00000000, scriptPubKey=040184710fa689ad5023690c: 2787
//      vMerkleTree: 97ddfbbae6

// Genesis block
const char* pszTimestamp = "NY Times 05/Oct/2011 Steve Jobs, Apple's Visionary, Dies at
56";
CTransaction txNew;
txNew.vin.resize(1);
txNew.vout.resize(1);
txNew.vin[0].scriptSig = CScript() << 486604799 << CBigNum(4) << vector<unsigned
char>((const unsigned char*)pszTimestamp, (const unsigned char*)pszTimestamp +
strlen(pszTimestamp));
txNew.vout[0].nValue = 50 * COIN;
txNew.vout[0].scriptPubKey = CScript() <<
ParseHex("04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80ba0ddd0
<< OP_CHECKSIG;
CBlock block;
block.vtx.push_back(txNew);
block.hashPrevBlock = 0;
block.hashMerkleRoot = block.BuildMerkleTree();
```

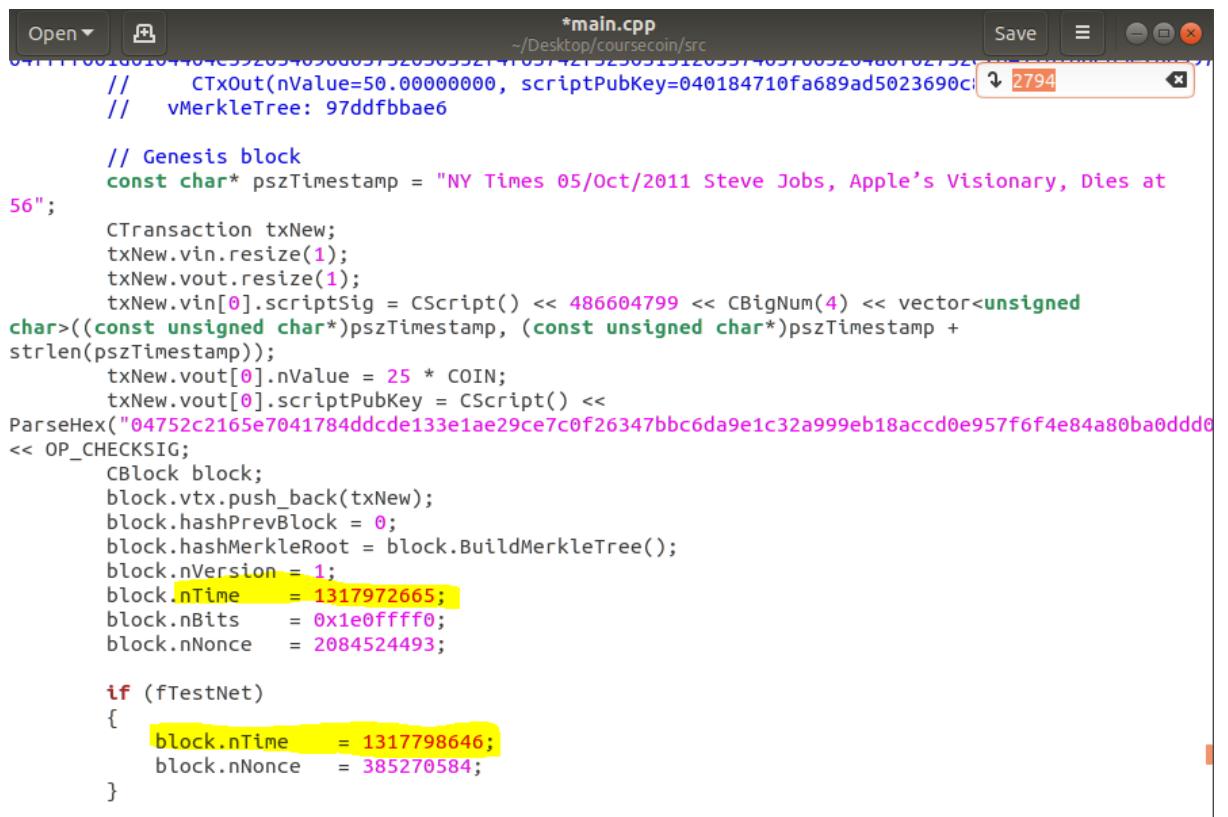
Change the nValue from 50 to 25.



```
*main.cpp
~/Desktop/coursecoin/src
0417100104404c5920540900057520505521410574215250515120557405700520480102752c2041700c05c200557
//      CTxOut(nValue=50.00000000, scriptPubKey=040184710fa689ad5023690c80f3a4)
//      vMerkleTree: 97ddfbbae6

// Genesis block
const char* pszTimestamp = "NY Times 05/Oct/2011 Steve Jobs, Apple's Visionary, Dies at
56";
CTransaction txNew;
txNew.vin.resize(1);
txNew.vout.resize(1);
txNew.vin[0].scriptSig = CScript() << 486604799 << CBigNum(4) << vector<unsigned
char>((const unsigned char*)pszTimestamp, (const unsigned char*)pszTimestamp +
strlen(pszTimestamp));
txNew.vout[0].nValue = 25 /* COIN */;
txNew.vout[0].scriptPubKey = CScript() <<
ParseHex("04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80ba0ddd0
<< OP_CHECKSIG;
CBlock block;
block.vtx.push_back(txNew);
block.hashPrevBlock = 0;
block.hashMerkleRoot = block.BuildMerkleTree();
block.nVersion = 1;
block.nTime    = 1317972665;
block.nBits    = 0x1e0ffff0;
block.nNonce   = 2084524493;
```

Go to **line 2794**, we are going to change nTime value in main net and nTime value in Test net.

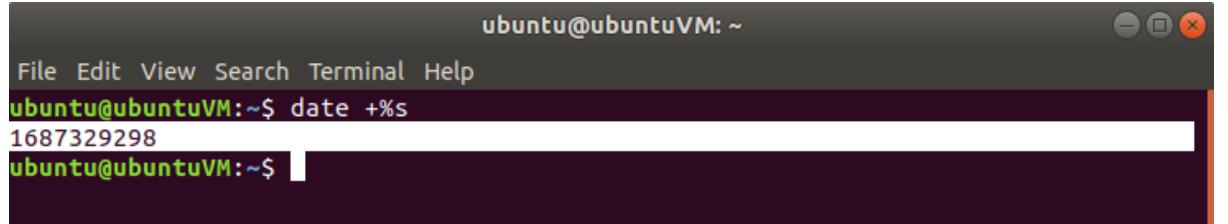


```
*main.cpp
~/Desktop/coursecoin/src
    //      CTxOut(nValue=50.00000000, scriptPubKey=040184710fa689ad5023690c: 2794
    // vMerkleTree: 97ddfbbae6

    // Genesis block
    const char* pszTimestamp = "NY Times 05/Oct/2011 Steve Jobs, Apple's Visionary, Dies at
56";
    CTransaction txNew;
    txNew.vin.resize(1);
    txNew.vout.resize(1);
    txNew.vin[0].scriptSig = CScript() << 486604799 << CBigNum(4) << vector<unsigned
char>((const unsigned char*)pszTimestamp, (const unsigned char*)pszTimestamp +
strlen(pszTimestamp));
    txNew.vout[0].nValue = 25 * COIN;
    txNew.vout[0].scriptPubKey = CScript() <<
ParseHex("04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80ba0ddd0
<< OP_CHECKSIG;
    CBlock block;
    block vtx.push_back(txNew);
    block.hashPrevBlock = 0;
    block.hashMerkleRoot = block.BuildMerkleTree();
    block.nVersion = 1;
    block.nTime = 1317972665;
    block.nBits = 0x1e0ffff0;
    block.nNonce = 2084524493;

    if (fTestNet)
    {
        block.nTime = 1317798646;
        block.nNonce = 385270584;
    }
}
```

For that open your terminal and type this command “**date +%**s”



```
ubuntu@ubuntuVM: ~
File Edit View Search Terminal Help
ubuntu@ubuntuVM:~$ date +%
1687329298
ubuntu@ubuntuVM:~$
```

Copy the current epic time from your terminal and paste it in main net **nTime** value field.

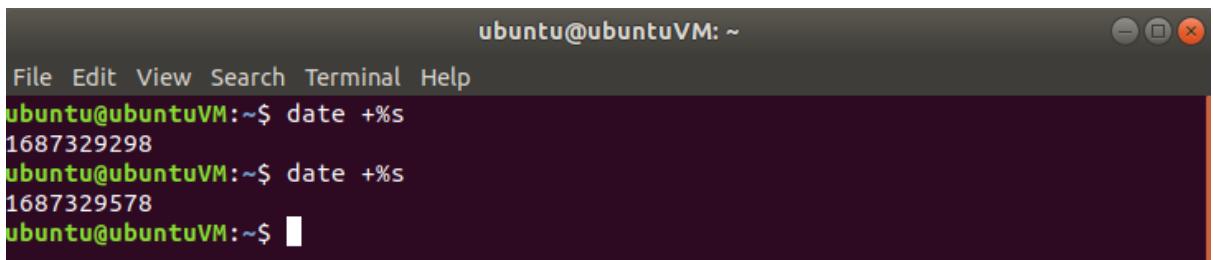
```

char>((const unsigned char*)pszTimestamp, (const unsigned char*)pszTimestamp +
strlen(pszTimestamp));
    txNew.vout[0].nValue = 25 * COIN;
    txNew.vout[0].scriptPubKey = CScript() <<
ParseHex("04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80ba0ddd0
<< OP_CHECKSIG;
    CBlock block;
    block.vtx.push_back(txNew);
    block.hashPrevBlock = 0;
    block.hashMerkleRoot = block.BuildMerkleTree();
    block.nVersion = 1;
    block.nTime = 1687329298;
    block.nBits = 0x1e0ffff0;
    block.nNonce = 2084524493;

    if (fTestNet)
    {
        block.nTime = 1317798646;
        block.nNonce = 385270584;
    }
}

```

Do the same thing in your terminal to generate an epic time for your test net **nTime**.



```

ubuntu@ubuntuVM:~$ date +%
1687329298
ubuntu@ubuntuVM:~$ date +%
1687329578
ubuntu@ubuntuVM:~$ █

```

Copy the newly generated time and paste it in the value field of test net **nTime**.

```

char>((const unsigned char*)pszTimestamp, (const unsigned char*)pszTimestamp +
strlen(pszTimestamp));
    txNew.vout[0].nValue = 25 * COIN;
    txNew.vout[0].scriptPubKey = CScript() <<
ParseHex("04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80ba0ddd0
<< OP_CHECKSIG;
    CBlock block;
    block.vtx.push_back(txNew);
    block.hashPrevBlock = 0;
    block.hashMerkleRoot = block.BuildMerkleTree();
    block.nVersion = 1;
    block.nTime = 1687329298;
    block.nBits = 0x1e0ffff0;
    block.nNonce = 2084524493;

    if (fTestNet)
    {
        block.nTime = 1687329578;
        block.nNonce = 385270584;
    }
}

```

Go to line 2782



The screenshot shows a code editor window with the file `*main.cpp` open. The line number `2782` is highlighted in a red box in the top right corner. The code itself is a C++ program for generating a genesis block. It includes comments for the genesis block's structure and its initial values. A specific line of code is highlighted in yellow:

```
    // Genesis block
    const char* pszTimestamp = "NY Times 05/Oct/2011 Steve Jobs, Apple's Visionary, Dies at
56";
```

This line defines a string constant `pszTimestamp` containing the headline about Steve Jobs' death.

and change the timestamp headline in genesis block as per your liking.

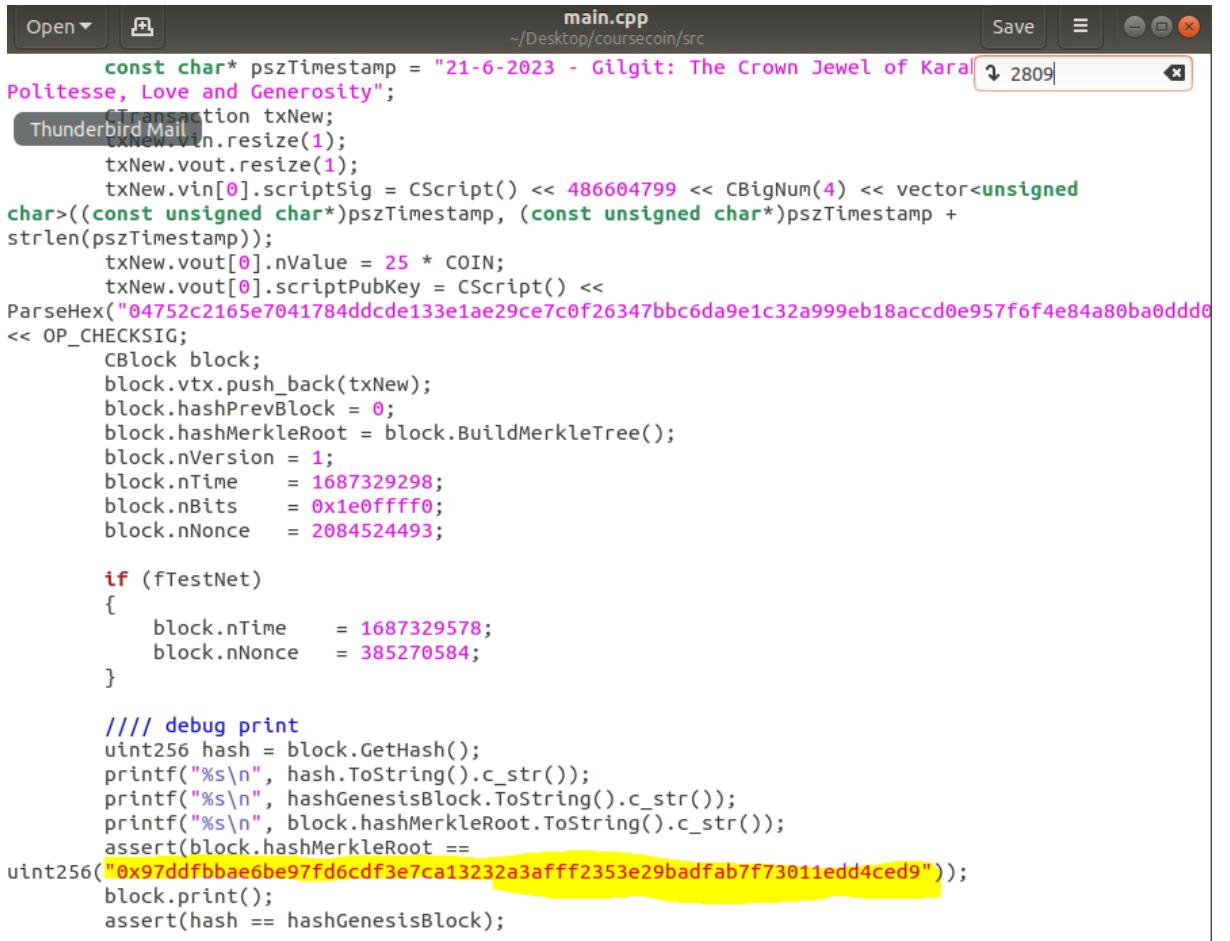


The screenshot shows the same code editor window with the file `*main.cpp` open. The line number `2782` is still highlighted in red. The timestamp headline has been changed to a new one:

```
    const char* pszTimestamp = "21-6-2023 - Gilgit: The Crown Jewel of Karakorum, A Land of
Politesse, Love and Generosity";
```

This change updates the headline to reflect a different event or location.

Go to line 2809, we are going to change the merkle root.



```
main.cpp
~/Desktop/coursecoin/src
const char* pszTimestamp = "21-6-2023 - Gilgit: The Crown Jewel of Karakorum, A Land of Politesse, Love and Generosity";
CTransaction txNew;
txNew.vin.resize(1);
txNew.vout.resize(1);
txNew.vin[0].scriptSig = CScript() << 486604799 << CBigNum(4) << vector<unsigned char>((const unsigned char*)pszTimestamp, (const unsigned char*)pszTimestamp + strlen(pszTimestamp));
txNew.vout[0].nValue = 25 * COIN;
txNew.vout[0].scriptPubKey = CScript() <<
ParseHex("04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80ba0ddd0
<< OP_CHECKSIG;
CBlock block;
block vtx.push_back(txNew);
block.hashPrevBlock = 0;
block.hashMerkleRoot = block.BuildMerkleTree();
block.nVersion = 1;
block.nTime = 1687329298;
block.nBits = 0x1e0ffff0;
block.nNonce = 2084524493;

if (fTestNet)
{
    block.nTime = 1687329578;
    block.nNonce = 385270584;
}

//// debug print
uint256 hash = block.GetHash();
printf("%s\n", hash.ToString().c_str());
printf("%s\n", hashGenesisBlock.ToString().c_str());
printf("%s\n", block.hashMerkleRoot.ToString().c_str());
assert(block.hashMerkleRoot ==
uint256("0x97ddfbbae6be97fd6cdf3e7ca13232a3afff2353e29badfab7f73011edd4ced9"));
block.print();
assert(hash == hashGenesisBlock);
```

For that delete everything in the highlighted string up until 0x



```
*main.cpp
~/Desktop/coursecoin/src
const char* pszTimestamp = "21-6-2023 - Gilgit: The Crown Jewel of Karakorum, A Land of Politesse, Love and Generosity";
CTransaction txNew;
txNew.vin.resize(1);
txNew.vout.resize(1);
txNew.vin[0].scriptSig = CScript() << 486604799 << CBigNum(4) << vector<unsigned char>((const unsigned char*)pszTimestamp, (const unsigned char*)pszTimestamp + strlen(pszTimestamp));
txNew.vout[0].nValue = 25 * COIN;
txNew.vout[0].scriptPubKey = CScript() <<
ParseHex("04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80ba0ddd0
<< OP_CHECKSIG;
CBlock block;
block vtx.push_back(txNew);
block.hashPrevBlock = 0;
block.hashMerkleRoot = block.BuildMerkleTree();
block.nVersion = 1;
block.nTime = 1687329298;
block.nBits = 0x1e0ffff0;
block.nNonce = 2084524493;

if (fTestNet)
{
    block.nTime = 1687329578;
    block.nNonce = 385270584;
}

//// debug print
uint256 hash = block.GetHash();
printf("%s\n", hash.ToString().c_str());
printf("%s\n", hashGenesisBlock.ToString().c_str());
printf("%s\n", block.hashMerkleRoot.ToString().c_str());
assert(block.hashMerkleRoot == uint256("0x"));
block.print();
assert(hash == hashGenesisBlock);
```

Save the main.cpp file, and to generate a new merkle root open your terminal and navigate to the **src** directory in your **coin** directory

```
ubuntu@ubuntuVM: ~/Desktop/coursecoin/src
File Edit View Search Terminal Help
ubuntu@ubuntuVM:~$ cd Desktop/coursecoin/src/
ubuntu@ubuntuVM:~/Desktop/coursecoin/src$
```

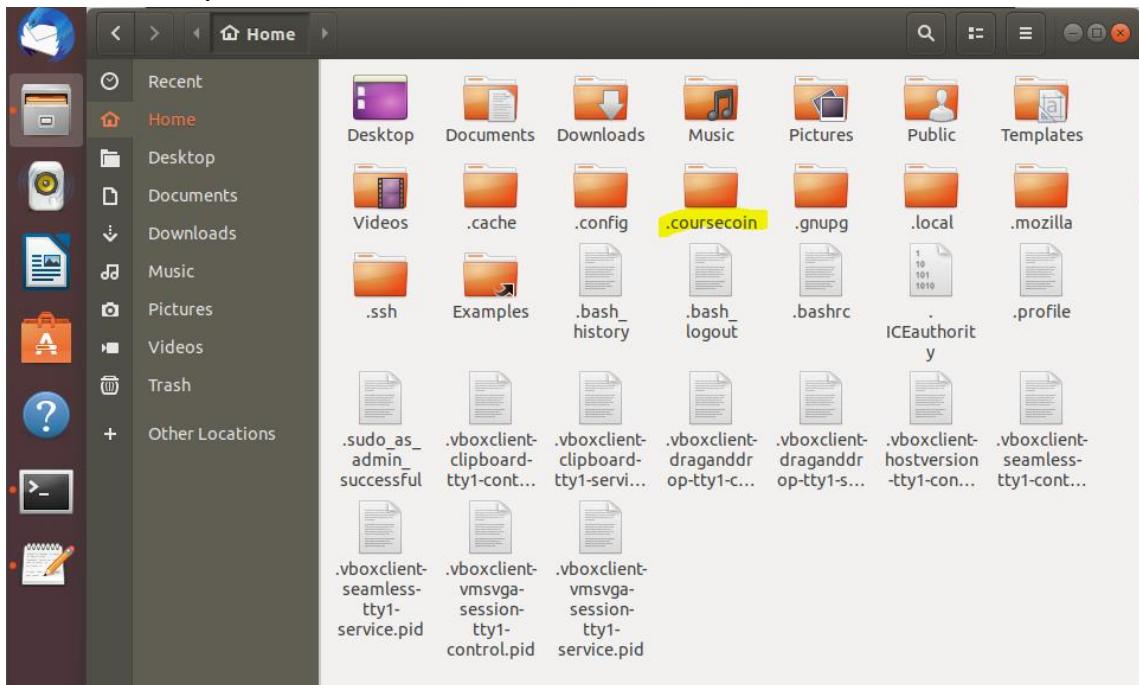
Type the command “**make -f makefile.unix**” to compile it.

```
ubuntu@ubuntuVM:~/Desktop/coursecoin/src$ make -f makefile.unix
/bin/sh ../share/genbuild.sh obj/build.h
g++ -c -O2 -pthread -Wall -Wextra -Wformat -Wformat-security -Wno-unused-parameter -g -DBOOST_SPIRIT_THREADSAFE -D_FILE_OFFSET_BITS=64 -I/home/ubuntu/Desktop/coursecoin/src -I/home/ubuntu/Desktop/coursecoin/src/obj -DUSE_UPNP=0 -DUSE_IPV6=1 -I/home/ubuntu/Desktop/coursecoin/src/leveldb/include -I/home/ubuntu/Desktop/coursecoin/src/leveldb/helpers -DHAVE_BUILD_INFO -fno-stack-protector -fstack-protector-all -Wstack-protector -U_FORTIFY_SOURCE -D_FORTIFY_SOURCE=2 -o coursecoind leveldb/libleveldb.a obj/alert.o obj/version.o obj/checkpoints.o obj/netbase.o obj/addrman.o obj/crypter.o obj/key.o obj/db.o obj/init.o obj/keystore.o obj/main.o obj/net.o obj/protocol.o obj/bitcoinrpc.o obj/rpcdump.o obj/rpcnet.o obj/rpcmining.o obj/rpcwallet.o obj/rpcblockchain.o obj/rpcrawtransaction.o obj/script.o obj/scrypt.o obj/sync.o obj/util.o obj/wallet.o obj/walletdb.o obj/hash.o obj/block.o obj/noui.o obj/leveldb.o obj/txdb.o -Wl,-z,relro -Wl,-z,now -Wl,-Bdynamic -l boost_system -l boost_filesystem -l boost_program_options -l boost_thread -l db_cxx -l ssl -l crypto -l miniupnpc -Wl,-Bdynamic -l z -l dl -l pthread /home/ubuntu/Desktop/coursecoin/src/leveldb/libleveldb.a /home/ubuntu/Desktop/coursecoin/src/leveldb/libmemenv.a
ubuntu@ubuntuVM:~/Desktop/coursecoin/src$ ./coursecoind
```

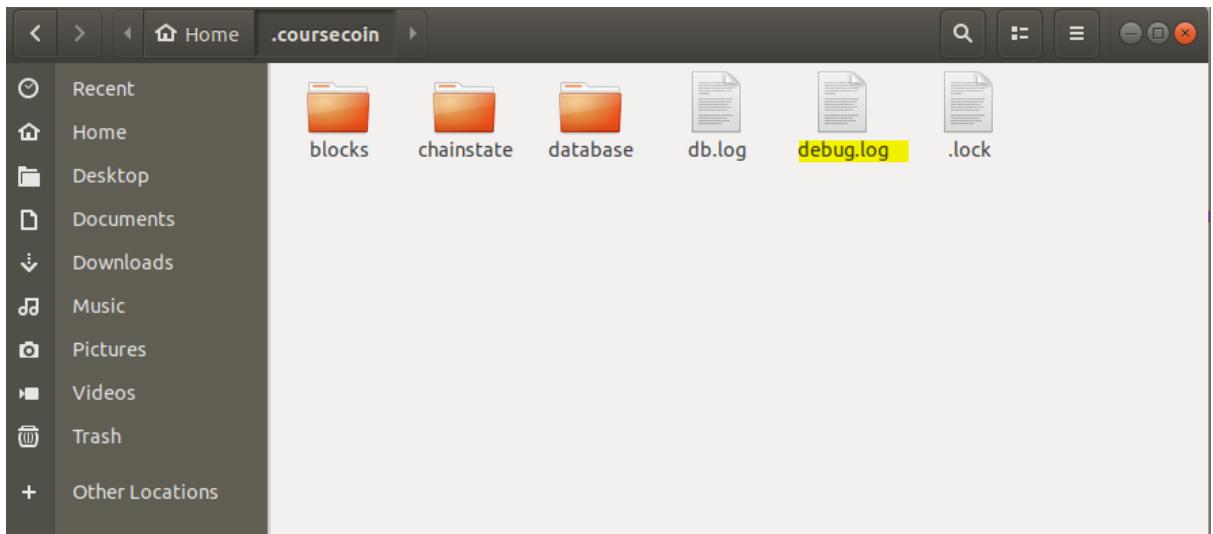
After the compilation type the command (**./yourcoinname with d at the end**) In my case it is “**./coursecoind**”. Running this command will result into a failure.

```
ubuntu@ubuntuVM: ~/Desktop/coursecoin/src
File Edit View Search Terminal Help
+11 [-Wdeprecated]
    bool Sync() throw(leveldb_error) {
        ~~~~~
g++ -O2 -pthread -Wall -Wextra -Wformat -Wformat-security -Wno-unused-parameter -g -DBOOST_SPIRIT_THREADSAFE -D_FILE_OFFSET_BITS=64 -I/home/ubuntu/Desktop/coursecoin/src -I/home/ubuntu/Desktop/coursecoin/src/obj -DUSE_UPNP=0 -DUSE_IPV6=1 -I/home/ubuntu/Desktop/coursecoin/src/leveldb/include -I/home/ubuntu/Desktop/coursecoin/src/leveldb/helpers -DHAVE_BUILD_INFO -fno-stack-protector -fstack-protector-all -Wstack-protector -U_FORTIFY_SOURCE -D_FORTIFY_SOURCE=2 -o coursecoind leveldb/libleveldb.a obj/alert.o obj/version.o obj/checkpoints.o obj/netbase.o obj/addrman.o obj/crypter.o obj/key.o obj/db.o obj/init.o obj/keystore.o obj/main.o obj/net.o obj/protocol.o obj/bitcoinrpc.o obj/rpcdump.o obj/rpcnet.o obj/rpcmining.o obj/rpcwallet.o obj/rpcblockchain.o obj/rpcrawtransaction.o obj/script.o obj/scrypt.o obj/sync.o obj/util.o obj/wallet.o obj/walletdb.o obj/hash.o obj/block.o obj/noui.o obj/leveldb.o obj/txdb.o -Wl,-z,relro -Wl,-z,now -Wl,-Bdynamic -l boost_system -l boost_filesystem -l boost_program_options -l boost_thread -l db_cxx -l ssl -l crypto -l miniupnpc -Wl,-Bdynamic -l z -l dl -l pthread /home/ubuntu/Desktop/coursecoin/src/leveldb/libleveldb.a /home/ubuntu/Desktop/coursecoin/src/leveldb/libmemenv.a
ubuntu@ubuntuVM:~/Desktop/coursecoin/src$ ./coursecoind
coursecoind: main.cpp:2809: bool InitBlockIndex(): Assertion `block.hashMerkleRoot == uint256("0x")' failed.
Aborted (core dumped)
ubuntu@ubuntuVM:~/Desktop/coursecoin/src$
```

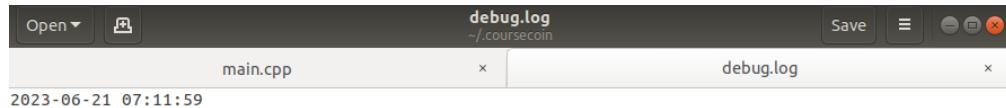
Now to find the merkle root go to the home directory in file explorer, press **Ctrl+H** to show the hidden files. You'll see a hidden directory named after your coin.



Go inside this directory, you'll see a file named **debug.log**. Open the **debug.log**



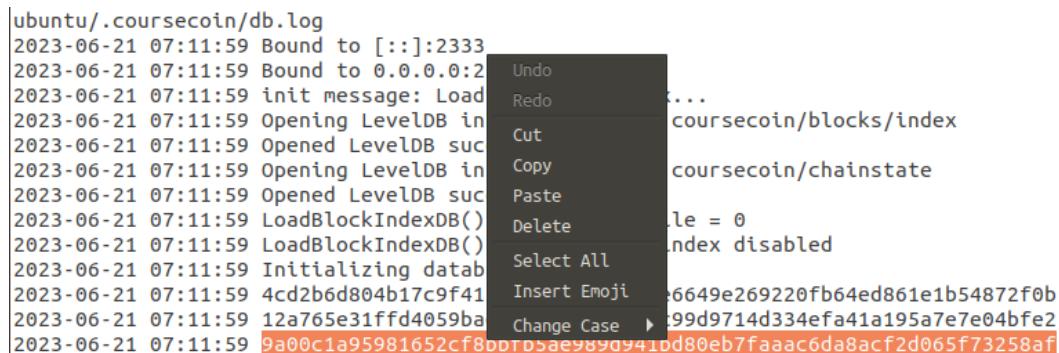
Scroll to the bottom and the last string is your merkle root.



```
Open debug.log ~/coursecoin Save main.cpp x debug.log x
2023-06-21 07:11:59
```

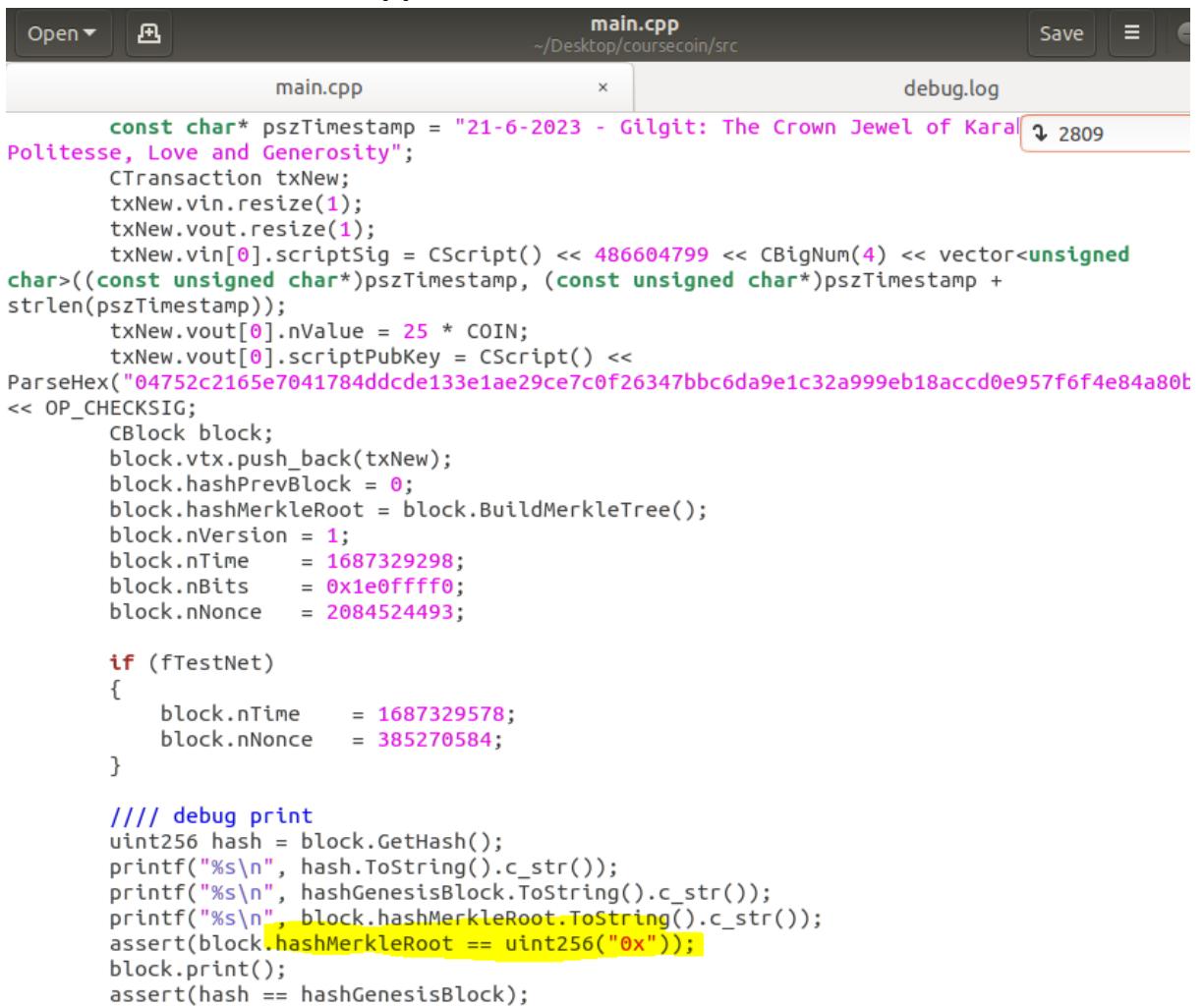
```
2023-06-21 07:11:59 Coursecoin version v0.8.7.5-unk-beta (2015-01-12 16:55:18 -1000)
2023-06-21 07:11:59 Using OpenSSL version OpenSSL 1.0.2n 7 Dec 2017
2023-06-21 07:11:59 Default data directory /home/ubuntu/.coursecoin
2023-06-21 07:11:59 Using data directory /home/ubuntu/.coursecoin
2023-06-21 07:11:59 Using at most 125 connections (1024 file descriptors available)
2023-06-21 07:11:59 Using 4 threads for script verification
2023-06-21 07:11:59 init message: Verifying wallet...
2023-06-21 07:11:59 dbenv.open LogDir=/home/ubuntu/.coursecoin/database ErrorFile=/home/ubuntu/.coursecoin/db.log
2023-06-21 07:11:59 Bound to [::]:2333
2023-06-21 07:11:59 Bound to 0.0.0.0:2333
2023-06-21 07:11:59 init message: Loading block index...
2023-06-21 07:11:59 Opening LevelDB in /home/ubuntu/.coursecoin/blocks/index
2023-06-21 07:11:59 Opened LevelDB successfully
2023-06-21 07:11:59 Opening LevelDB in /home/ubuntu/.coursecoin/chainstate
2023-06-21 07:11:59 Opened LevelDB successfully
2023-06-21 07:11:59 LoadBlockIndexDB(): last block file = 0
2023-06-21 07:11:59 LoadBlockIndexDB(): transaction index disabled
2023-06-21 07:11:59 Initializing databases...
2023-06-21 07:11:59 4cd2b6d804b17c9f413d0cae71e77cbe6649e269220fb64ed861e1b54872f0b
2023-06-21 07:11:59 12a765e31ffd4059bada1e25190f6e98c99d9714d334efa41a195a7e7e04bfe2
2023-06-21 07:11:59 9a00c1a95981652cf8bbfb5ae989d941bd80eb7faaac6da8acf2d065f73258af
```

Copy this merkle root



```
ubuntu/.coursecoin/db.log
2023-06-21 07:11:59 Bound to [::]:2333
2023-06-21 07:11:59 Bound to 0.0.0.0:2
2023-06-21 07:11:59 init message: Load
2023-06-21 07:11:59 Opening LevelDB in
2023-06-21 07:11:59 Opened LevelDB suc
2023-06-21 07:11:59 Opening LevelDB in
2023-06-21 07:11:59 Opened LevelDB suc
2023-06-21 07:11:59 LoadBlockIndexDB()
2023-06-21 07:11:59 LoadBlockIndexDB()
2023-06-21 07:11:59 Initializing datab
2023-06-21 07:11:59 4cd2b6d804b17c9f41
2023-06-21 07:11:59 12a765e31ffd4059ba
2023-06-21 07:11:59 9a00c1a95981652cf8bbfb5ae989d941bd80eb7faaac6da8acf2d065f73258af
```

Go to line 2809 in main.cpp



```
main.cpp
~/Desktop/coursecoin/src

main.cpp x debug.log ↴ 2809

const char* pszTimestamp = "21-6-2023 - Gilgit: The Crown Jewel of Karakorum, Love and Generosity";
CTransaction txNew;
txNew.vin.resize(1);
txNew.vout.resize(1);
txNew.vin[0].scriptSig = CScript() << 486604799 << CBigNum(4) << vector<unsigned char>((const unsigned char*)pszTimestamp, (const unsigned char*)pszTimestamp + strlen(pszTimestamp));
txNew.vout[0].nValue = 25 * COIN;
txNew.vout[0].scriptPubKey = CScript() <<
ParseHex("04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80t
<< OP_CHECKSIG;
CBlock block;
block.vtx.push_back(txNew);
block.hashPrevBlock = 0;
block.hashMerkleRoot = block.BuildMerkleTree();
block.nVersion = 1;
block.nTime = 1687329298;
block.nBits = 0x1e0ffff0;
block.nNonce = 2084524493;

if (fTestNet)
{
    block.nTime = 1687329578;
    block.nNonce = 385270584;
}

//// debug print
uint256 hash = block.GetHash();
printf("%s\n", hash.ToString().c_str());
printf("%s\n", hashGenesisBlock.ToString().c_str());
printf("%s\n", block.hashMerkleRoot.ToString().c_str());
assert(block.hashMerkleRoot == uint256("0x"));
block.print();
assert(hash == hashGenesisBlock);
```

Right after **0x** paste this newly obtained merkle root

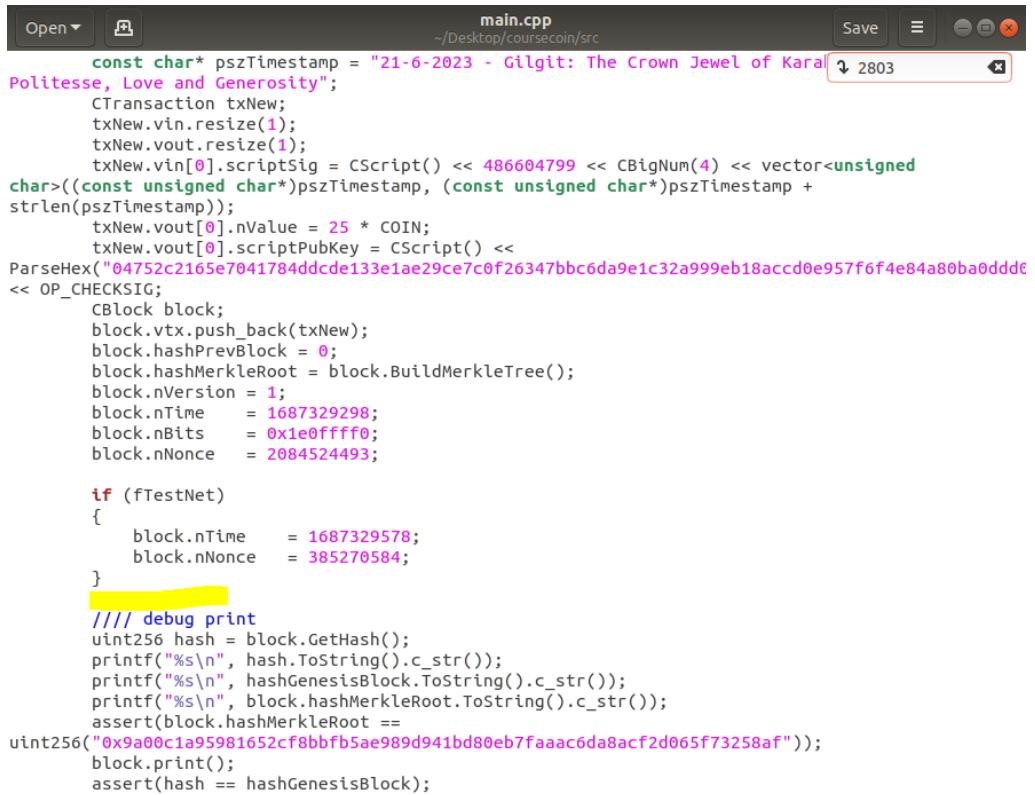
```
ParseHex("04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80ba0ddd0
<< OP_CHECKSIG;
CBlock block;
block.vtx.push_back(txNew);
block.hashPrevBlock = 0;
block.hashMerkleRoot = block.BuildMerkleTree();
block.nVersion = 1;
block.nTime = 1687329298;
block.nBits = 0x1e0ffff0;
block.nNonce = 2084524493;

if (fTestNet)
{
    block.nTime = 1687329578;
    block.nNonce = 385270584;
}

//// debug print
uint256 hash = block.GetHash();
printf("%s\n", hash.ToString().c_str());
printf("%s\n", hashGenesisBlock.ToString().c_str());
printf("%s\n", block.hashMerkleRoot.ToString().c_str());
assert(block.hashMerkleRoot ==
uint256("0x9a00c1a95981652cf8bbfb5ae989d941bd80eb7faaac6da8acf2d065f73258af"));
block.print();
assert(hash == hashGenesisBlock);
```

Save the file.

Go to **line 2803** in **main.cpp**



```
const char* pszTimestamp = "21-6-2023 - Gilgit: The Crown Jewel of Karakorum, Love and Generosity";
CTransaction txNew;
txNew.vin.resize(1);
txNew.vout.resize(1);
txNew.vin[0].scriptSig = CScript() << 486604799 << CBigNum(4) << vector<unsigned char>((const unsigned char*)pszTimestamp, (const unsigned char*)pszTimestamp + strlen(pszTimestamp));
txNew.vout[0].nValue = 25 * COIN;
txNew.vout[0].scriptPubKey = CScript() <<
ParseHex("04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80ba0ddd6
<< OP_CHECKSIG;
CBlock block;
block vtx.push_back(txNew);
block.hashPrevBlock = 0;
block.hashMerkleRoot = block.BuildMerkleTree();
block.nVersion = 1;
block.nTime = 1687329298;
block.nBits = 0x1e0ffff0;
block.nNonce = 2084524493;

if (fTestNet)
{
    block.nTime = 1687329578;
    block.nNonce = 385270584;
}

//// debug print
uint256 hash = block.GetHash();
printf("%s\n", hash.ToString().c_str());
printf("%s\n", hashGenesisBlock.ToString().c_str());
printf("%s\n", block.hashMerkleRoot.ToString().c_str());
assert(block.hashMerkleRoot ==
uint256("0x9a00c1a95981652cf8bbfb5ae989d941bd80eb7faaac6da8acf2d065f73258af"));
block.print();
assert(hash == hashGenesisBlock);
```

Right after **if (fTestNet)** block, paste the following piece of code (a text file containing this code is also available on LMS)

```
if (true && block.GetHash() != hashGenesisBlock)
{
    printf("Searching for genesis block...\n");
    // This will figure out a valid hash and Nonce if you're
    // creating a different genesis block:
    uint256 hashTarget =
CBigNum().SetCompact(block.nBits).getuint256();
    uint256 thash;
    char scratchpad[SCRYPT_SCRATCHPAD_SIZE];

    loop
    {
#ifndef USE_SSE2
        // Detection would work, but in cases where we KNOW it
        // always has SSE2,
```

```

        // it is faster to use directly than to use a function pointer or
conditional.

#if defined(_M_X64) || defined(__x86_64__) || defined(_M_AMD64)
|| (defined(MAC_OSX) && defined(__i386__))
    // Always SSE2: x86_64 or Intel MacOS X
    scrypt_1024_1_1_256_sp_sse2(BEGIN(block.nVersion),
BEGIN(thash), scratchpad);
#else
    // Detect SSE2: 32bit x86 Linux or Windows
    scrypt_1024_1_1_256_sp(BEGIN(block.nVersion),
BEGIN(thash), scratchpad);
#endif
#else
    // Generic scrypt
    scrypt_1024_1_1_256_sp_generic(BEGIN(block.nVersion),
BEGIN(thash), scratchpad);
#endif

if (thash <= hashTarget)
    break;
if ((block.nNonce & 0xFF) == 0)
{
    printf("nonce %08X: hash = %s (target = %s)\n",
block.nNonce, thash.ToString().c_str(), hashTarget.ToString().c_str());
}
++block.nNonce;
if (block.nNonce == 0)
{
    printf("NONCE WRAPPED, incrementing time\n");
    ++block.nTime;
}
printf("block.nTime = %u \n", block.nTime);
printf("block.nNonce = %u \n", block.nNonce);
printf("block.GetHash = %s\n",
block.GetHash().ToString().c_str());
}

```

After pasting, your code will look like this

```
ParseHex("04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80ba0ddd0
<< OP_CHECKSIG;
    CBlock block;
    block.vtx.push_back(txNew);
    block.hashPrevBlock = 0;
    block.hashMerkleRoot = block.BuildMerkleTree();
    block.nVersion = 1;
    block.nTime = 1687329298;
    block.nBits = 0x1e0ffff0;
    block.nNonce = 2084524493;

    if (fTestNet)
    {
        block.nTime = 1687329578;
        block.nNonce = 385270584;
    }

    if (true && block.GetHash() != hashGenesisBlock)
    {
        printf("Searching for genesis block...\n");
        // This will figure out a valid hash and Nonce if you're
        // creating a different genesis block:
        uint256 hashTarget = CBigNum().SetCompact(block.nBits).getuint256();
        uint256 thash;
        char scratchpad[SCRIPT_SCRATCHPAD_SIZE];
```



Save the file and go to **line 2749**

```
main.cpp
-/Desktop/coursecoin/src
Save 2749 ×

}

void UnloadBlockIndex()
{
    mapBlockIndex.clear();
    setBlockIndexValid.clear();
    pindexGenesisBlock = NULL;
    nBestHeight = 0;
    nBestChainWork = 0;
    nBestInvalidWork = 0;
    hashBestChain = 0;
    pindexBest = NULL;
}

bool LoadBlockIndex()
{
    if (fTestNet)
    {
        pchMessageStart[0] = 0xfa;
        pchMessageStart[1] = 0xc5;
        pchMessageStart[2] = 0xbd;
        pchMessageStart[3] = 0xdf;
        hashGenesisBlock =
uint256("0xf5ae71e26c74beacc88382716aced69cddf3dff24f384e1808905e0188f68f");
    }
}
```

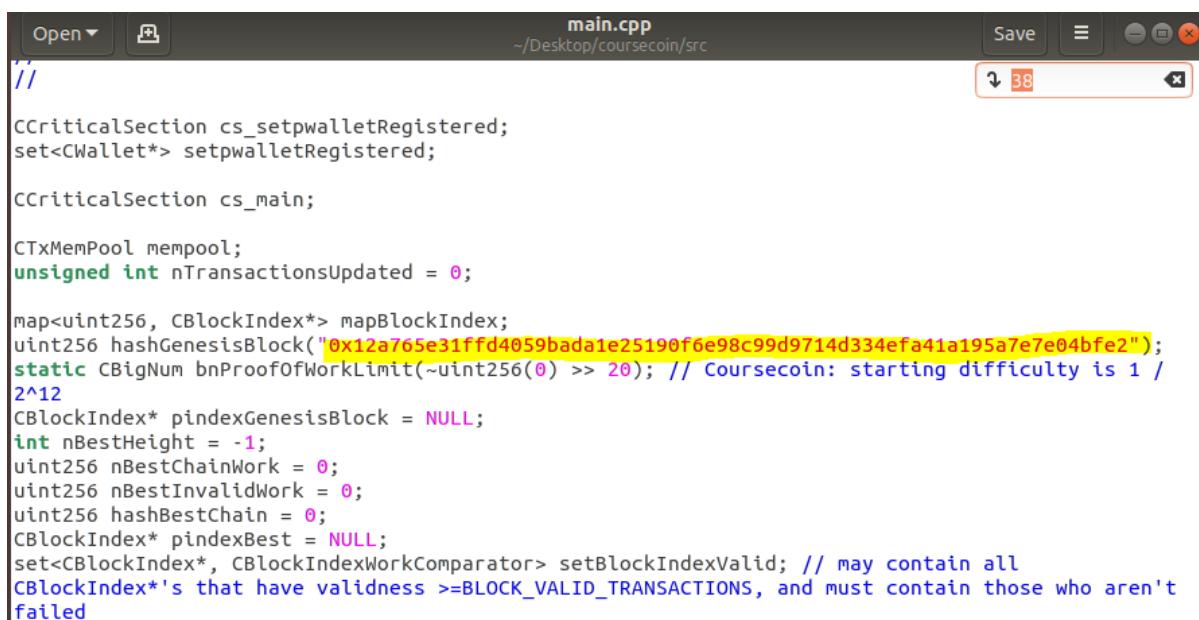
erase the test net genesis block hash till 0x

```
void UnloadBlockIndex()
{
    mapBlockIndex.clear();
    setBlockIndexValid.clear();
    pindexGenesisBlock = NULL;
    nBestHeight = 0;
    nBestChainWork = 0;
    nBestInvalidWork = 0;
    hashBestChain = 0;
    pindexBest = NULL;
}

bool LoadBlockIndex()
{
    if (fTestNet)
    {
        pchMessageStart[0] = 0xfa;
        pchMessageStart[1] = 0xc5;
        pchMessageStart[2] = 0xbd;
        pchMessageStart[3] = 0xdf;
        hashGenesisBlock = uint256("0x");
    }

    // Load block index from databases
    //
    if (!fReindex && !LoadBlockIndexDB())
        return false;
}
```

Now go to line 38, you'll see the main net genesis block hash



```
//
CCriticalSection cs_SetWalletRegistered;
set<CWallet*> setWalletRegistered;

CCriticalSection cs_main;

CTxMemPool mempool;
unsigned int nTransactionsUpdated = 0;

map<uint256, CBlockIndex*> mapBlockIndex;
uint256 hashGenesisBlock("0x12a765e31ffd4059bada1e25190f6e98c99d9714d334efa41a195a7e7e04bfe2");
static CBigNum bnProofOfWorkLimit(~uint256(0) >> 20); // Coursecoin: starting difficulty is 1 / 2^12
CBlockIndex* pindexGenesisBlock = NULL;
int nBestHeight = -1;
uint256 nBestChainWork = 0;
uint256 nBestInvalidWork = 0;
uint256 hashBestChain = 0;
CBlockIndex* pindexBest = NULL;
set<CBlockIndex*, CBlockIndexWorkComparator> setBlockIndexValid; // may contain all CBlockIndex*'s that have validness >=BLOCK_VALID_TRANSACTIONS, and must contain those who aren't failed
```

Erase it too up until **0x**

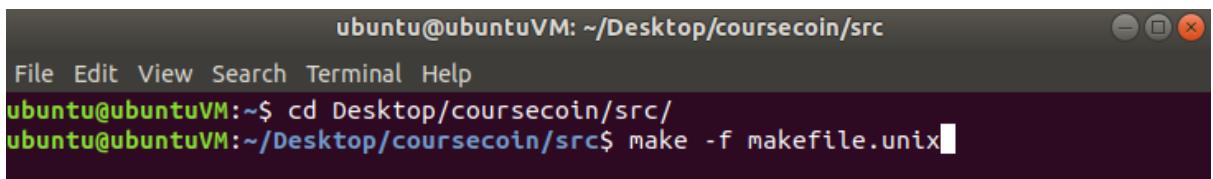


The screenshot shows a code editor window with the file 'main.cpp' open. The code is C++ and defines various global variables and structures used in a blockchain implementation. A specific line of code, 'uint256 hashGenesisBlock("0x");', is highlighted in yellow. The code editor has a dark theme with syntax highlighting.

```
//  
  
CCriticalSection cs_SetpWalletRegistered;  
set<CWallet*> setpWalletRegistered;  
  
CCriticalSection cs_main;  
  
CTxMemPool mempool;  
unsigned int nTransactionsUpdated = 0;  
  
map<uint256, CBlockIndex*> mapBlockIndex;  
uint256 hashGenesisBlock("0x");  
static CBigNum bnProofOfWorkLimit(~uint256(0) >> 20); // Coursecoin: starting difficulty is 1 /  
2^12  
CBlockIndex* pindexGenesisBlock = NULL;  
int nBestHeight = -1;  
uint256 nBestChainWork = 0;  
uint256 nBestInvalidWork = 0;  
uint256 hashBestChain = 0;  
CBlockIndex* pindexBest = NULL;  
set<CBlockIndex*, CBlockIndexWorkComparator> setBlockIndexValid; // may contain all  
CBlockIndex*'s that have validness >=BLOCK_VALID_TRANSACTIONS, and must contain those who aren't  
failed
```

Save the main.cpp file.

Now to generate the genesis block hashes of main net and test net, we are going to compile our code again. So for that open your terminal, navigate to src directory inside your coin directory. And type the command "**make -f makefile.unix**".



The screenshot shows a terminal window on an Ubuntu VM. The user has navigated to the 'src' directory and run the command 'make -f makefile.unix'. The terminal output is shown in green text.

```
ubuntu@ubuntuVM:~/Desktop/coursecoin/src$ cd Desktop/coursecoin/src/  
ubuntu@ubuntuVM:~/Desktop/coursecoin/src$ make -f makefile.unix
```

Once the compilation has been done, use the command **./yourcoinname** with d at the end, in my case it's **./coursecoind** with a **-testnet** flag e.g. **(./coursecoind -testnet)** to generate the genesis block hash for the test net.

```

g++ -O2 -pthread -Wall -Wextra -Wformat -Wformat-security -Wno-unused-parameter
-g -DBoost_SPIRIT_THREADSAFE -D_FILE_OFFSET_BITS=64 -I/home/ubuntu/Desktop/coursecoin/src -I/home/ubuntu/Desktop/coursecoin/src/obj -DUSE_UPNP=0 -DUSE_IPV6=1 -I/home/ubuntu/Desktop/coursecoin/src/leveldb/include -I/home/ubuntu/Desktop/coursecoin/src/leveldb/helpers -DHAVE_BUILD_INFO -fno-stack-protector -fstack-protect-all -Wstack-protector -U_FORTIFY_SOURCE -D_FORTIFY_SOURCE=2 -o coursecoind leveldb/libleveldb.a obj/alert.o obj/version.o obj/checkpoints.o obj/netbase.o obj/addrman.o obj/crypter.o obj/key.o obj/db.o obj/init.o obj/keystore.o obj/main.o obj/net.o obj/protocol.o obj/bitcoinrpc.o obj/rpcdump.o obj/rpcnet.o obj/rpcmining.o obj/rpcwallet.o obj/rpcblockchain.o obj/rpcrawtransaction.o obj/script.o obj/scrypt.o obj/sync.o obj/util.o obj/wallet.o obj/walletdb.o obj/hash.o obj/block.o obj/noui.o obj/leveldb.o obj/txdb.o -Wl,-z,relro -Wl,-z,now -Wl,-Bdynamic -l boost_system -l boost_filesystem -l boost_program_options -l boost_thread -l db_cxx -l ssl -l crypto -l miniupnpc -Wl,-Bdynamic -l z -l dl -l pthread /home/ubuntu/Desktop/coursecoin/src/leveldb/libleveldb.a /home/ubuntu/Desktop/coursecoin/src/leveldb/libmemenv.a
ubuntu@ubuntuVM:~/Desktop/coursecoin/src$ ./coursecoind -testnet ↵

```

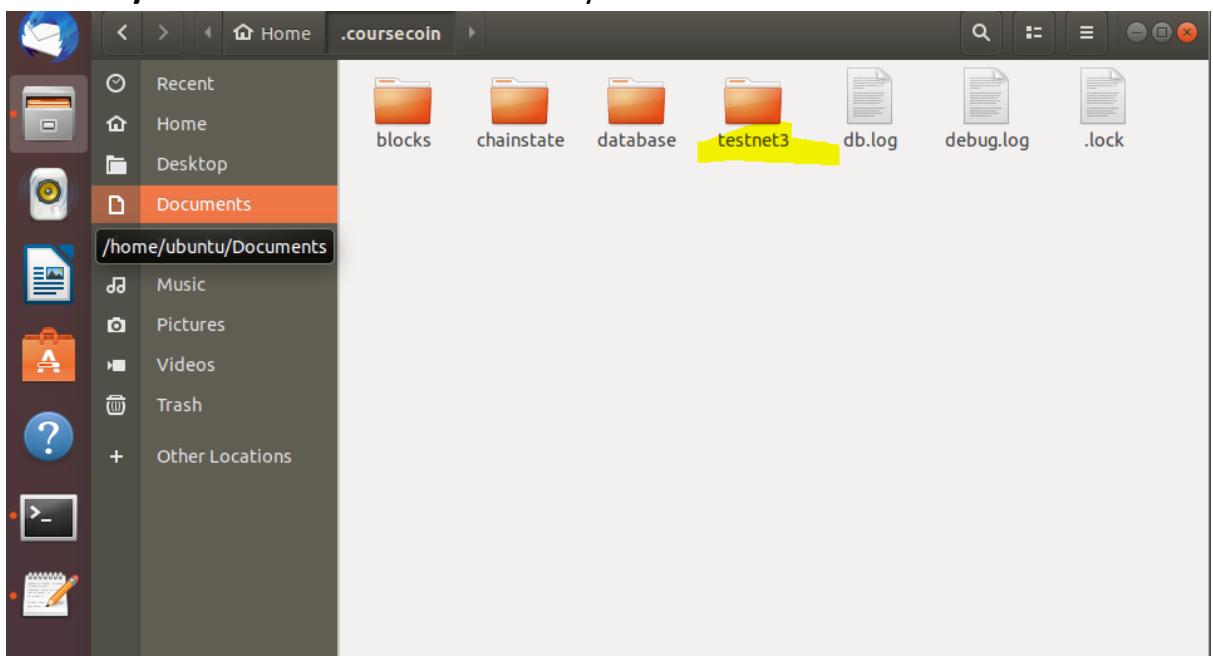
It will take some time so be patient and wait for the core to be dumped.

```

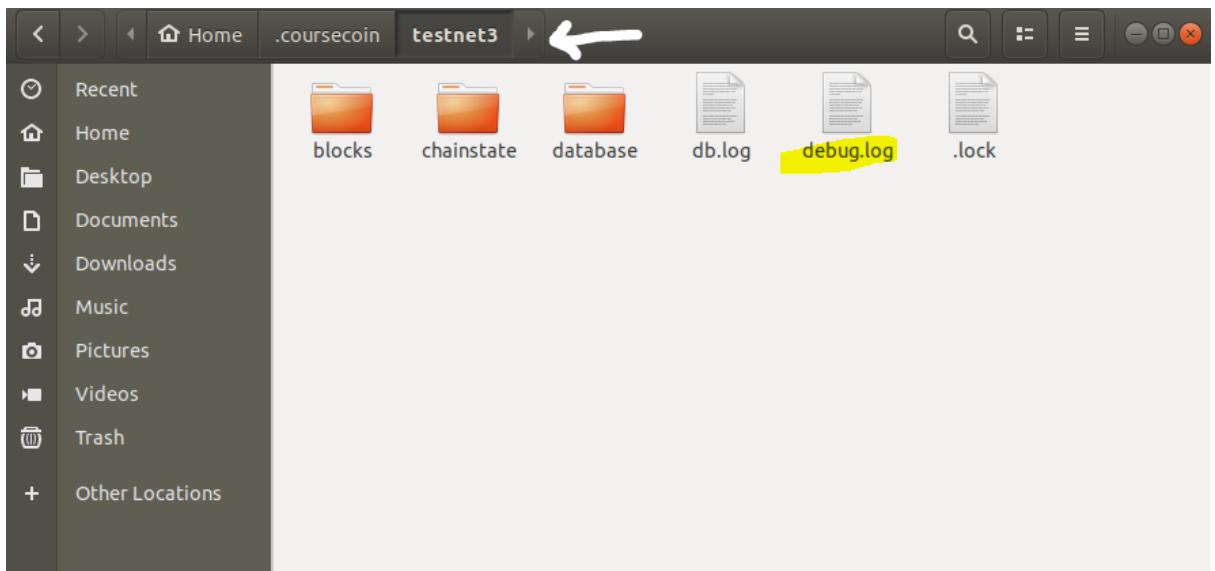
ubuntu@ubuntuVM:~/Desktop/coursecoin/src$ ./coursecoind -testnet
coursecoind: main.cpp:2853: bool InitBlockIndex(): Assertion `hash == hashGenesisBlock' failed.
Aborted (core dumped)
ubuntu@ubuntuVM:~/Desktop/coursecoin/src$ ↵

```

Now go in your home directory and inside your hidden **.yourcoin** directory. You'll see a testnet directory in it.



Inside the test net directory there is a **debug.log** file.



Open the **debug.log** file in the **testnet** directory scroll to the bottom

You'll see nNonce and genesis block hash for the test net.

Copy the nNonce value

Go to line 2801 in main.cpp

main.cpp

```
transaction txNew;
txNew.vin.resize(1);
txNew.vout.resize(1);
txNew.vin[0].scriptSig = CScript() << 486604799 << CBigNum(4) << vector<unsigned char>((const unsigned char*)pszTimestamp, (const unsigned char*)pszTimestamp + strlen(pszTimestamp));
txNew.vout[0].nValue = 25 * COIN;
txNew.vout[0].scriptPubKey = CScript() <<
ParseHex("04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80ba0ddd0
<< OP_CHECKSIG;
CBlock block;
block vtx.push_back(txNew);
block.hashPrevBlock = 0;
block.hashMerkleRoot = block.BuildMerkleTree();
block.nVersion = 1;
block.nTime = 1687329298;
block.nBits = 0x1e0ffff0;
block.nNonce = 2084524493;

if (fTestNet)
{
    block.nTime = 1687329578;
    block.nNonce = 385270584;
}
```

debug.log

```
2801
```

Replace the nNonce value with the value that you copied earlier.

```
ParseHex("04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80ba0ddd0
<< OP_CHECKSIG;
    CBlock block;
    block.vtx.push_back(txNew);
    block.hashPrevBlock = 0;
    block.hashMerkleRoot = block.BuildMerkleTree();
    block.nVersion = 1;
    block.nTime = 1687329298;
    block.nBits = 0x1e0ffff0;
    block.nNonce = 2084524493;

    if (fTestNet)
    {
        block.nTime = 1687329578;
        block.nNonce = 385733459;
    }
}
```

Save the **main.cpp** file

Go back to `debug.log` and copy the genesis block hash for the testnet

Go to line 2749 in main.cpp

```
main.cpp
~/Desktop/coursecoin/src

main.cpp x debug.log x

nBestHeight = 0;
nBestChainWork = 0;
nBestInvalidWork = 0;
hashBestChain = 0;
pindexBest = NULL;
}

bool LoadBlockIndex()
{
    if (fTestNet)
    {
        pchMessageStart[0] = 0xfa;
        pchMessageStart[1] = 0xc5;
        pchMessageStart[2] = 0xbd;
        pchMessageStart[3] = 0xdf;
        hashGenesisBlock = uint256("0x");
    }

    // Load block index from databases
    //

    if (!fReindex && !LoadBlockIndexDB())
        return false;
}

// 2749
```

Paste the genesis block hash that you copied from the **debug.log** file

```
bool LoadBlockIndex()
{
    if (fTestNet)
    {
        pchMessageStart[0] = 0xfa;
        pchMessageStart[1] = 0xc5;
        pchMessageStart[2] = 0xbd;
        pchMessageStart[3] = 0xdf;
        hashGenesisBlock =
uint256("0xa52b29c7cae66b6785d06fb944011d7c9f1e1d3acb2f9b58a7cce91505faa4a7");
    }
}
```

Save the main.cpp file

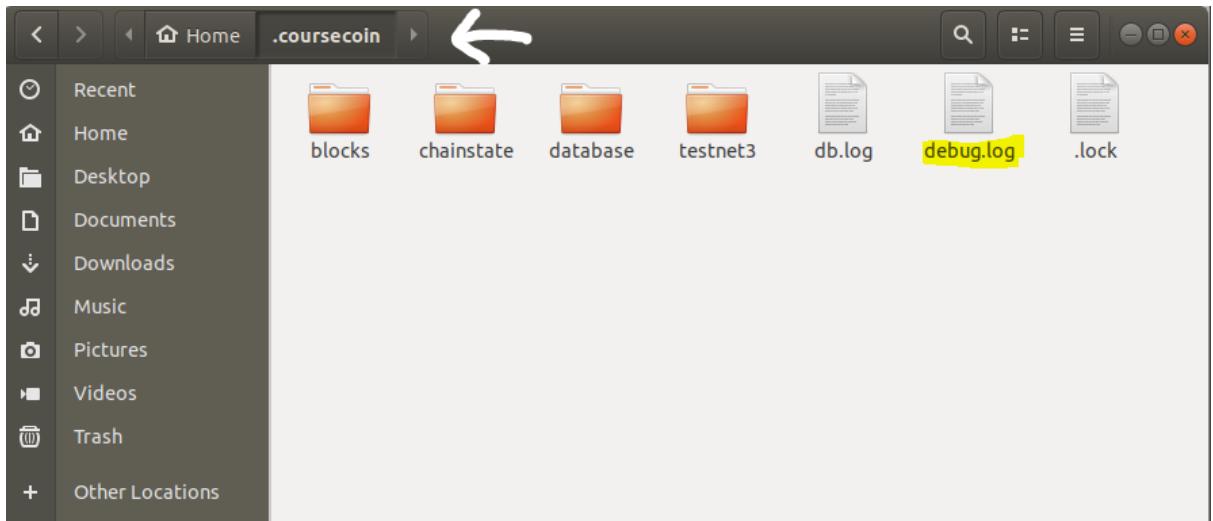
Now open your terminal and navigate to the **src directory** inside your **coin directory**. And compile the code again by using the command “make -f makefile.unix”

```
ubuntu@ubuntuVM:~/Desktop/coursecoin/src$ make -f makefile.unix
/bin/sh ../share/genbuild.sh obj/build.h
g++ -c -O2 -pthread -Wall -Wextra -Wformat -Wformat-security -Wno-unused-parameter -g -DBOOST_SPIRIT_THREADS -D_FILE_OFFSET_BITS=64 -I/home/ubuntu/Desktop/coursecoin/src -I/home/ubuntu/Desktop/coursecoin/src/obj -DUSE_UPNP=0 -DUSE_IPV6=1 -I/home/ubuntu/Desktop/coursecoin/src/leveldb/include -I/home/ubuntu/Desktop/coursecoin/src/leveldb/helpers -DHAVE_BUILD_INFO -fno-stack-protector -fstack-protector-all -Wstack-protector -U_FORTIFY_SOURCE -D_FORTIFY_SOURCE=2 -MMD -MF obj/main.d -o obj/main.o main.cpp
```

Once compiled type the command **./yourcoinname** with **d** at the end, as in this case it is **./coursecoind**

```
ubuntu@ubuntuVM:~/Desktop/coursecoin/src$ ./coursecoind
coursecoind: main.cpp:2853: bool InitBlockIndex(): Assertion `hash == hashGenesisBlock' failed.
Aborted (core dumped)
ubuntu@ubuntuVM:~/Desktop/coursecoin/src$
```

Once the core is dumped, go to the hidden .yourcoinname directory in home directory using file explorer.



Open the debug.log file for the main net, and scroll to the bottom

Copy the nNonce value

Go to line 2796



```
main.cpp
~/Desktop/coursecoin/src

main.cpp x debug.log x
strlen(psztimestamp));
    txNew.vout[0].nValue = 25 * COIN;
    txNew.vout[0].scriptPubKey = CScript() <<
ParseHex("04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80ba0ddd0
<< OP_CHECKSIG;
    CBlock block;
    block.vtx.push_back(txNew);
    block.hashPrevBlock = 0;
    block.hashMerkleRoot = block.BuildMerkleTree();
    block.nVersion = 1;
    block.nTime = 1687329298;
    block.nBits = 0x1e0ffff0;
    block.nNonce = 2084524493;

    if (fTestNet)
    {
        block.nTime = 1687329578;
        block.nNonce = 385733459;
    }

```

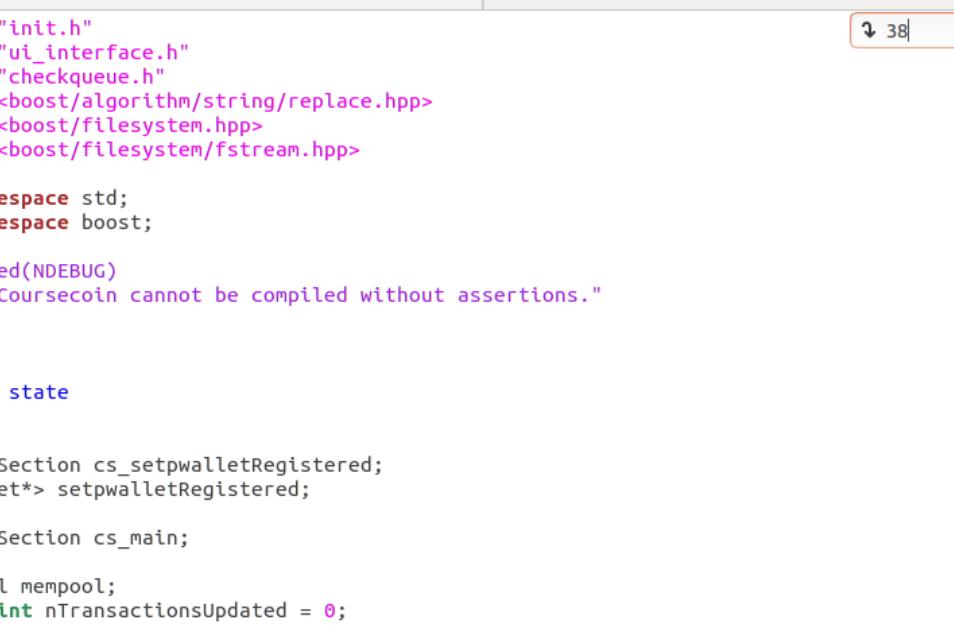
Paste the main net nNonce value that you copied earlier

```
ParseHex("04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80ba0ddd0
<< OP_CHECKSIG;
    CBlock block;
    block.vtx.push_back(txNew);
    block.hashPrevBlock = 0;
    block.hashMerkleRoot = block.BuildMerkleTree();
    block.nVersion = 1;
    block.nTime = 1687329298;
    block.nBits = 0x1e0ffff0;
    block.nNonce = 2084652351;

    if (fTestNet)
    {
        block.nTime = 1687329578;
        block.nNonce = 385733459;
    }
}
```

Now go back to debug.log file for main net, and copy the genesis block hash for main net.

Go to line 38 in main.cpp



```
main.cpp
~/Desktop/coursecoin/src

main.cpp x debug.log x
↳ 38 | ↳ 38 | ↳ 38 |

#include "init.h"
#include "ui_interface.h"
#include "checkqueue.h"
#include <boost/algorithm/string/replace.hpp>
#include <boost/filesystem.hpp>
#include <boost/filesystem/fstream.hpp>

using namespace std;
using namespace boost;

#if defined(NDEBUG)
# error "Coursecoin cannot be compiled without assertions."
#endif

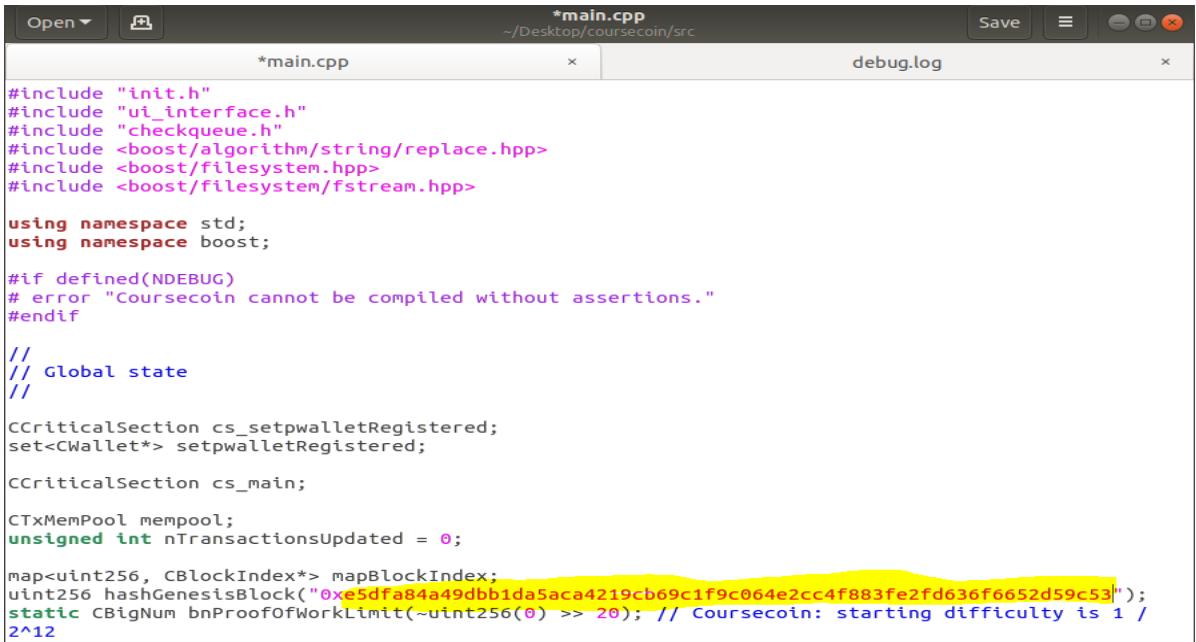
// Global state
// CCriticalSection cs_SetpWalletRegistered;
set<CWallet*> SetpWalletRegistered;

CCriticalSection cs_Main;

CTxMemPool mempool;
unsigned int nTransactionsUpdated = 0;

map<uint256, CBlockIndex*> mapBlockIndex;
uint256 hashGenesisBlock("0x");
static CBigNum bnProofOfWorkLimit(~uint256(0) >> 20); // Coursecoin: starting difficulty is 1 /
2^12
```

Paste the genesis block hash for main net that you copied after 0x



```
*main.cpp
~/Desktop/coursecoin/src
*main.cpp x debug.log x

#include "init.h"
#include "ui_interface.h"
#include "checkqueue.h"
#include <boost/algorithm/string/replace.hpp>
#include <boost/filesystem.hpp>
#include <boost/filesystem/fstream.hpp>

using namespace std;
using namespace boost;

#if defined(NDEBUG)
# error "Coursecoin cannot be compiled without assertions."
#endif

//
// Global state
//
CCriticalSection cs_SetpWalletRegistered;
set<CWallet*> setpWalletRegistered;

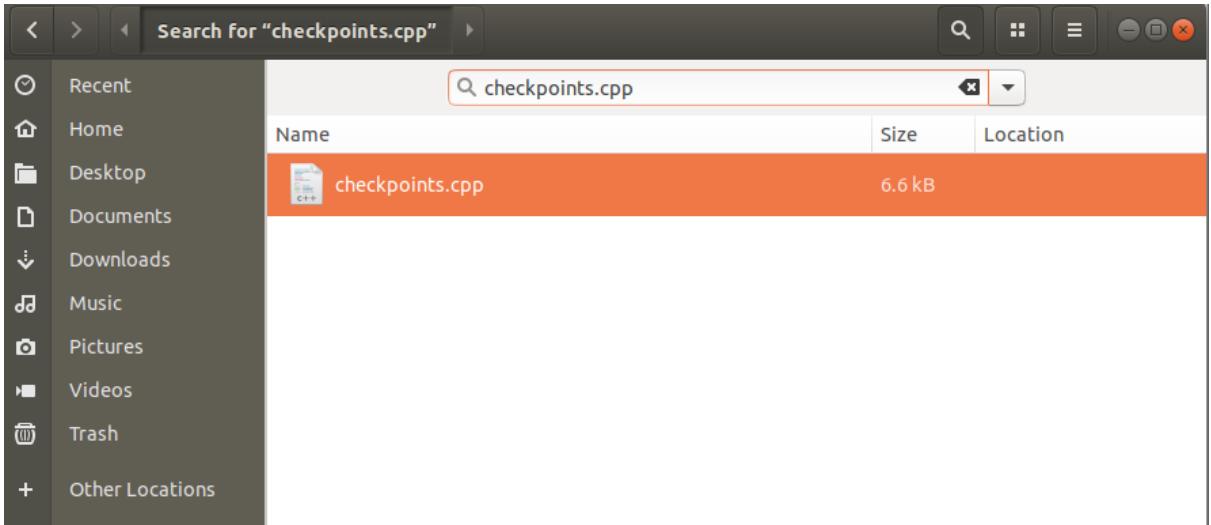
CCriticalSection cs_main;

CTxMemPool mempool;
unsigned int nTransactionsUpdated = 0;

map<uint256, CBlockIndex*> mapBlockIndex;
uint256 hashGenesisBlock("0xe5dfa84a49dbb1da5aca4219cb69c1f9c064e2cc4f883fe2fd636f6652d59c53");
static CBigNum bnProofOfWorkLimit(~uint256(0) >> 20); // Coursecoin: starting difficulty is 1 / 2^12
```

Save the main.cpp file

4. Go to the src directory in your coin directory and search for a file named checkpoints.cpp. Open the file by double clicking it.



```

checkpoints.cpp
-/Desktop/coursecoin/src

main.cpp x checkpoints.cpp

// Copyright (c) 2009-2012 The Bitcoin developers
// Distributed under the MIT/X11 software license, see the accompanying
// file COPYING or http://www.opensource.org/licenses/mit-license.php.
#include <boost/assign/list_of.hpp> // for 'map_list_of()'
#include <boost/foreach.hpp>

#include "checkpoints.h"

#include "main.h"
#include "uint256.h"

namespace Checkpoints
{
    typedef std::map<int, uint256> MapCheckpoints;

    // How many times we expect transactions after the last checkpoint to
    // be slower. This number is a compromise, as it can't be accurate for
    // every system. When reindexing from a fast disk with a slow CPU, it
    // can be up to 20, while when downloading from a slow network with a
    // fast multicore CPU, it won't be much higher than 1.
    static const double fSigcheckVerificationFactor = 5.0;

    struct CCheckpointData {
        const MapCheckpoints *mapCheckpoints;
        int64 nTimeLastCheckpoint;
        int64 nTransactionsLastCheckpoint;
        double fTransactionsPerDay;
    };

    // What makes a good checkpoint block?
    // + Is surrounded by blocks with reasonable timestamps
    //   (no blocks before with a timestamp after, none after with
    //     timestamp before)
    // + Contains no strange transactions
    static MapCheckpoints mapCheckpoints =
        boost::assign::map_list_of
            
```

Go to **line 38** in **checkpoints.cpp** file, you'll find a list of check points, they prevent the blockchain from brute forcing

```

static MapCheckpoints mapCheckpoints =
    boost::assign::map_list_of
        ( 1500, uint256("0x841a2965955dd288cf0707a755d05a54e45f8bd476835ec9af4402a2b59a2967"))
        ( 4032, uint256("0x9ce90e427198fc0ef05e5905ce3503725b80e26af35a987965fd7e3d9cf0846"))
        ( 8064, uint256("0xeb984353fc5190f210651f150c40b8a4bab9eeeff0b729fc3987da694430d70"))
        ( 16128, uint256("0x602edf1859b7f9a6af809f1d9b0e6cb66fdc1d4d9dc7a4bec03e12a1cc153d"))
        ( 23420, uint256("0xd80fdf9ca81af0bd2b2a90ac3a9fe547da58f2530ec874e978fce0b5101b507"))
        ( 50000, uint256("0x69dc37eb029b68f075a5012dcc0419c127672adb4f3a32882b2b3e71d07a20a6"))
        ( 80000, uint256("0x4fcfb7c02f676a300503f49c764a89955a8f920b46a8cbe3b4867182ecdb2e90a"))
        (120000, uint256("0xbd9d26924f05f6daa7f0155f32828ec89e8e29cee9e7121b026a7a3552ac6131"))
        (161500, uint256("0dbe89880474f4bb4f75c227c77ba1cdc024991123b28b8418dbbf7798471ff43"))
        (179620, uint256("0x2ad9c65c990ac00426d18e446e0fd7be2ffa69e9a7dc28358a50b2b78b9f709"))
        (240000, uint256("0x7140d1c4b4c2157ca217ee7636f24c9c73db39c4590c4e6eab2e3ea1555088aa"))
        (383640, uint256("0xb6809f094a9215bafc65eb3f110a35127a34be94b7d0590a096c3f126c6f364"))
        (409004, uint256("0x487518d663d9f1fa08611d9395ad74d982b667fbdc0e77e9cf39b4f1355908a3"))
        (456000, uint256("0xbff34f71cc6366cd487930d06be22f897e34ca6a40501ac7d401be32456372004"))
        (541794, uint256("0x1cbcbe6920e7c258bbce1f26211084efb19764aa3224bec3f4320d77d6a2fd2"))
        (585010, uint256("0xea9ea06840de20a18a66acb07c9102ee6374ad2cbafc71794e576354fea5df2d"))
        (638902, uint256("0x15238656e8ec63d28de29a8c75fcf3a5819afc953dc9cc45cecc53baec74f38"))
        ;
```

Erase all the check points except the first one

```
static MapCheckpoints mapCheckpoints =
    boost::assign::map_list_of
( 1500, uint256("0x841a2965955dd288cf707a755d05a54e45f8bd476835ec9af4402a2b59a2967"))
( 4032, uint256("0x9ce90e427198fc0ef05e5905ce3503725b80e26af35a987965fd7e3d9cf0846"))
( 8064, uint256("0xeb984353fc5190f210651f150c40b8a4bab9eeeff0b729fc3987da694430d70"))
( 16128, uint256("0x602edf1859b7ff9a6af809f1d9b0e6cb66fdc1d4d9dc7a4bec03e12a1cccd153d"))
( 23420, uint256("0xd80fdf9ca81af0bd2b2a90ac3a9fe547da58f2530ec874e978fce0b5101b507"))
( 50000, uint256("0x69dc37eb029b68f075a5012dcc0419c127672adb4f3a32882b2b3e71d07a20a6"))
( 80000, uint256("0x4fc7c02f676a300503f49c764a89955a8f920b46a8cbeccb4867182ecd2e90a"))
(120000, uint256("0xbdb9d26924f05f6daa7f0155f32828ec89e8e29cee9e7121b026a7a3552ac6131"))
(161500, uint256("0xdbe89880474f4bb4f75c227c77ba1cdc024991123b28b8418dbbf7798471ff43"))
(179620, uint256("0x2ad9c65c990ac00426d18e446e0fd7be2ffa69e9a7dc28358a50b2b78b9f709"))
(240000, uint256("0x7140d1c4b4c2157ca217ee7636f24c9c73db39c4590c4e6eab2e3ea1555088aa"))
(383640, uint256("0x2b6809f094a9215bafc65eb3f110a35127a34be94b7d0590a096c3f126c6f364"))
(409004, uint256("0x487518d663d9f1fa08611d9395ad74d982b667fbdc0e77e9cf39b4f1355908a3"))
(456000, uint256("0xbf34f71cc6366cd487930d06be22f897e34ca6a40501ac7d401be32456372004"))
(541794, uint256("0x1ccbccbe6920e7c258bbce1f26211084efb19764aa3224bec3f4320d77d6a2fd2"))
(585010, uint256("0xea9ea06840de20a18a66acb07c9102ee6374ad2cbafc71794e576354fea5df2d"))
(638902, uint256("0x15238656e8ec63d28de29a8c75fcf3a5819afc953cd9cc45cecc53baec74f38"))
;

static MapCheckpoints mapCheckpoints =
    boost::assign::map_list_of
( 1500, uint256("0x841a2965955dd288cf707a755d05a54e45f8bd476835ec9af4402a2b59a2967"))
;
static const CCheckpointData data = {
    &mapCheckpoints,
    1410516073, // * UNIX timestamp of last checkpoint block
    4896865, // * total number of transactions between genesis and last checkpoint
               // (the tx=... number in the SetBestChain debug.log lines)
    7000.0 // * estimated number of transactions per day after checkpoint
};
```

Replace the value of 1500 with 0 and erase the main net genesis block hash upto 0x.

```
// What makes a good checkpoint block?
// + Is surrounded by blocks with reasonable timestamps
//   (no blocks before with a timestamp after, none after with
//     timestamp before)
// + Contains no strange transactions
static MapCheckpoints mapCheckpoints =
    boost::assign::map_list_of
( 0, uint256("0x"))
;
static const CCheckpointData data = {
    &mapCheckpoints,
    1410516073, // * UNIX timestamp of last checkpoint block
    4896865, // * total number of transactions between genesis and last checkpoint
               // (the tx=... number in the SetBestChain debug.log lines)
    7000.0 // * estimated number of transactions per day after checkpoint
};
```

Now go to **line 38** on **main.cpp** and copy the main net genesis block hash that you generated earlier

```

main.cpp
~/Desktop/coursecoin/src
main.cpp x *checkpoints.cpp x
// ...
CCriticalSection cs_SetpWalletRegistered;
set<CWallet*> setpWalletRegistered;

CCriticalSection cs_main;

CTxMemPool mempool;
unsigned int nTransactionsUpdated = 0;

map<uint256, CBlockIndex*> mapBlockIndex;
uint256 hashGenesisBlock("0xe5dfa84a49dbb1da5aca4219cb69c1f9c064e2cc4f883fe2fd636f6652d59c53");
static CBigNum bnProofOfWorkLimit(~uint256(0) >> 20); // Coursecoin: starting difficulty is 1 / 2^12
CBlockIndex* pindexGenesisBlock = NULL;
int nBestHeight = -1;
uint256 nBestChainWork = 0;
uint256 nBestInvalidWork = 0;
uint256 hashBestChain = 0;
CBlockIndex* pindexBest = NULL;
set<CBlockIndex*, CBlockIndexWorkComparator> setBlockIndexValid; // may contain all CBlockIndex*'s that have validness >=BLOCK_VALID_TRANSACTIONS, and must contain those who aren't failed
int64 nTimeBestReceived = 0;
int nScriptCheckThreads = 0;
bool fImporting = false;
bool fReindex = false;

```

Copy this part

```

CCriticalSection cs_main;

CTxMemPool mempool;
unsigned int nTransactionsUpdated = 0;

map<uint256, CBlockIndex*> mapBlockIndex;
uint256 hashGenesisBlock("0xe5dfa84a49dbb1da5aca4219cb69c1f9c064e2cc4f883fe2fd636f6652d59c53");
static CBigNum bnProofOfWorkLimit(~uint256(0) >> 20); // Coursecoin: starting difficulty is 1 / 2^12

```

Go to **line 38** in **checkpoints.cpp** and paste the hash string that you copied from **main.cpp** after **0x**

```

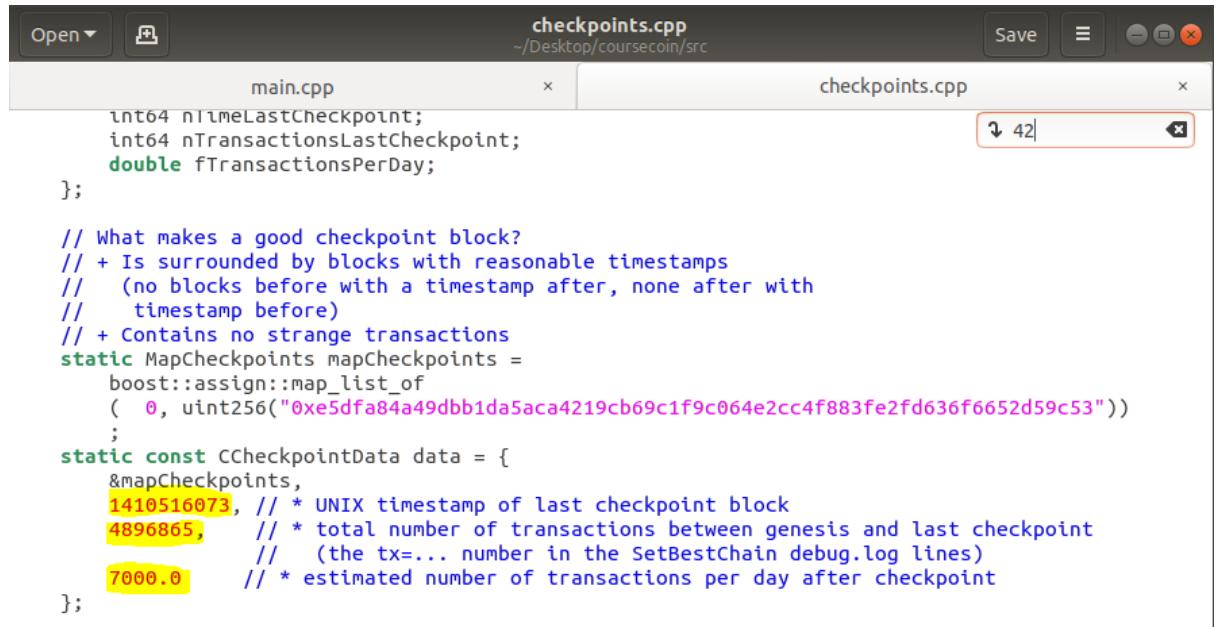
// What makes a good checkpoint block?
// + Is surrounded by blocks with reasonable timestamps
//   (no blocks before with a timestamp after, none after with
//     timestamp before)
// + Contains no strange transactions
static MapCheckpoints mapCheckpoints =
    boost::assign::map_list_of
        ( 0, uint256("0xe5dfa84a49dbb1da5aca4219cb69c1f9c064e2cc4f883fe2fd636f6652d59c53"))
;
static const CCheckpointData data = {
    &mapCheckpoints,
    1410516073, // * UNIX timestamp of last checkpoint block
    4896865, // * total number of transactions between genesis and last checkpoint
              //   (the tx=... number in the SetBestChain debug.log lines)
    7000.0 // * estimated number of transactions per day after checkpoint
};

static MapCheckpoints mapCheckpointsTestnet =
    boost::assign::map_list_of
        ( 546, uint256("0xa0fea99a6897f531600c8ae53367b126824fd6a847b2b73817a95b8e27e602"))
;
static const CCheckpointData dataTestnet = {

```

Save the checkpoints.cpp file

Now go on line 42 in checkpoints.cpp, you'll see some time stamp details.



```
checkpoints.cpp
~/Desktop/coursecoin/src

main.cpp x checkpoints.cpp x

int64 nTimeLastCheckpoint;
int64 nTransactionsLastCheckpoint;
double fTransactionsPerDay;
};

// What makes a good checkpoint block?
// + Is surrounded by blocks with reasonable timestamps
//   (no blocks before with a timestamp after, none after with
//     timestamp before)
// + Contains no strange transactions
static MapCheckpoints mapCheckpoints =
    boost::assign::map_list_of
    ( 0, uint256("0xe5dfa84a49dbb1da5aca4219cb69c1f9c064e2cc4f883fe2fd636f6652d59c53"))
;
static const CCheckpointData data = {
    &mapCheckpoints,
    1410516073, // * UNIX timestamp of last checkpoint block
    4896865, // * total number of transactions between genesis and last checkpoint
    7000.0 // * estimated number of transactions per day after checkpoint
};
```

To change these, go to line 2794 in main.cpp, copy the nTime



```
main.cpp
~/Desktop/coursecoin/src

txNew.vin[0].scriptSig = CScript() << 486604799 << CBIGNum(4) << vector<
char>((const unsigned char*)pszTimestamp, (const unsigned char*)pszTimestamp +
strlen(pszTimestamp));
txNew.vout[0].nValue = 25 * COIN;
txNew.vout[0].scriptPubKey = CScript() <<
ParseHex("04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80ba0ddd0
<< OP_CHECKSIG;
CBlock block;
block.vtx.push_back(txNew);
block.hashPrevBlock = 0;
block.hashMerkleRoot = block.BuildMerkleTree();
block.nVersion = 1;
block.nTime = 1687329298;
block.nBits = 0x1e0ffff0;
block.nNonce = 2084652351;

if (fTestNet)
{
    block.nTime = 1687329578;
    block.nNonce = 385733459;
}
```

main.cpp

```

txNew.vin[0].scriptSig = CScript() << 486604799 << CBigNum(4) << vector<unsigned char>((const unsigned char*)pszTimestamp, (const unsigned char*)pszTimestamp +
strlen(pszTimestamp));
txNew.vout[0].nValue = 25 * COIN;
txNew.vout[0].scriptPubKey = CScript() <<
ParseHex("04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80ba0ddde
<< OP_CHECKSIG;
CBlock block;
block vtx.push_back(txNew);
block.hashPrevBlock = 0;
block.hashMerkleRoot = block.BuildMerkleTree();
block.nVersion = 1;
block.nTime = 1687329298;
block.nBits = 0x1e0ffff0;
block.nNonce = 2084652352;

if (fTestNet)
{
    block.nTime = 1687329298;
    block.nNonce = 3857352;
}

if (true && block.GetHash() != hash)
{
    printf("Searching for genesis block...\n");
    // This will figure out a valid hash and Nonce if you're
    // creating a different genesis block:
    uint256 hashTarget = CBigNum().SetCompact(block.nBits).getuint256();
    uint256 hash;
    char scratchpad[SCRYPT_SCRATCHPAD_SIZE];
}

```

Go to line 42 in checkpoints.cpp

checkpoints.cpp

```

int64 nTimeLastCheckpoint;
int64 nTransactionsLastCheckpoint;
double fTransactionsPerDay;
};

// What makes a good checkpoint block?
// + Is surrounded by blocks with reasonable timestamps
//   (no blocks before with a timestamp after, none after with
//     timestamp before)
// + Contains no strange transactions
static MapCheckpoints mapCheckpoints =
    boost::assign::map_list_of
    ( 0, uint256("0xe5dfa84a49dbb1da5aca4219cb69c1f9c064e2cc4f883fe2fd636f6652d59c53"))
;
static const CCheckpointData data = {
    &mapCheckpoints,
    1410516073, // * UNIX timestamp of last checkpoint block
    4896865, // * total number of transactions between genesis and last checkpoint
    //   (the tx=... number in the SetBestChain debug.log lines)
    7000.0 // * estimated number of transactions per day after checkpoint
};

```

Paste the nTime value that you copied from **main.cpp**

```
// What makes a good checkpoint block?  
// + Is surrounded by blocks with reasonable timestamps  
//   (no blocks before with a timestamp after, none after with  
//     timestamp before)  
// + Contains no strange transactions  
static MapCheckpoints mapCheckpoints =  
    boost::assign::map_list_of  
    ( 0, uint256("0xe5dfa84a49dbb1da5aca4219cb69c1f9c064e2cc4f883fe2fd636f6652d59c53"))  
;  
static const CCheckpointData data = {  
    &mapCheckpoints,  
    1687329298, // * UNIX timestamp of last checkpoint block  
    4896865, // * total number of transactions between genesis and last checkpoint  
              //   (the tx=... number in the SetBestChain debug.log lines)  
    7000.0 // * estimated number of transactions per day after checkpoint  
};
```

Change the second value below nTime to 0, and the third value to 20.0

```
static const CCheckpointData data = {
    &mapCheckpoints,
    1687329298, // * UNIX timestamp of last checkpoint block
    0,           // * total number of transactions between genesis and last checkpoint
                 //   (the tx=... number in the SetBestChain debug.log lines)
    20.0,        // * estimated number of transactions per day after checkpoint
};
```

Save the **checkpoints.cpp** file. Now, let us do the same thing for the test net.

Go to line 52 in **checkpoints.cpp**, you'll see the respective values for test net

```
main.cpp x checkpoints.cpp x
int64 nTimeLastCheckpoint;
int64 nTransactionsLastCheckpoint;
double fTransactionsPerDay;
};

// What makes a good checkpoint block?
// + Is surrounded by blocks with reasonable timestamps
//   (no blocks before with a timestamp after, none after with
//     timestamp before)
// + Contains no strange transactions
static MapCheckpoints mapCheckpoints =
    boost::assign::map_list_of
    ( 0, uint256("0xe5dfa84a49dbb1da5aca4219cb69c1f9c064e2cc4f883fe2fd636f6652d59c53"))
    ;
static const CCheckpointData data = {
    &mapCheckpoints,
    1687329298, // * UNIX timestamp of last checkpoint block
    0,           // * total number of transactions between genesis and last checkpoint
                 //   (the tx=... number in the SetBestChain debug.log lines)
    20.0         // * estimated number of transactions per day after checkpoint
};

static MapCheckpoints mapCheckpointsTestnet =
    boost::assign::map_list_of
    ( 546, uint256("0xa0fea99a6897f531600c8ae53367b126824fd6a847b2b2b73817a95b8e27e602"))
    ;
static const CCheckpointData dataTestnet = {
    &mapCheckpointsTestnet,
    1365458829,
    547,
    576
};
```

Change the block height value from 546 to 0, and erase the test net genesis block hash up to **0x**.

```
static MapCheckpoints mapCheckpointsTestnet =
    boost::assign::map_list_of
    ( 0, uint256("0x"))
    ;
static const CCheckpointData dataTestnet = {
    &mapCheckpointsTestnet,
    1365458829,
    547,
    576
};
```

Go to **line 2749** in **main.cpp**, and copy the testnet genesis block hash that you created earlier.

```
main.cpp
~/Desktop/coursecoin/src

main.cpp x checkpoints.cpp x
↳ 2749

pindexGenesisBlock = NULL;
nBestHeight = 0;
nBestChainWork = 0;
nBestInvalidWork = 0;
hashBestChain = 0;
pindexBest = NULL;

}

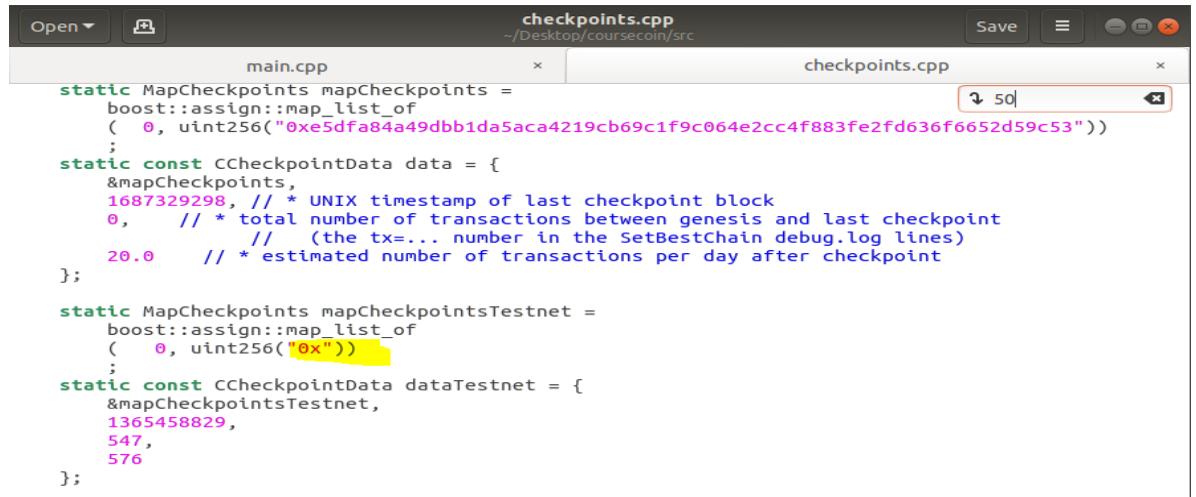
bool LoadBlockIndex()
{
    if (fTestNet)
    {
        pchMessageStart[0] = 0xfa;
        pchMessageStart[1] = 0xc5;
        pchMessageStart[2] = 0xbd;
        pchMessageStart[3] = 0xdf;
        hashGenesisBlock =
uint256("0xa52b29c7cae66b6785d06fb944011d7c9f1e1d3acb2f9b58a7cce91505faa4a7");
    }

    bool LoadBlockIndex()
    {
        if (fTestNet)
        {
            pchMessageStart[0] = 0xfa;
            pchMessageStart[1] = 0xc5;
            pchMessageStart[2] = 0xbd;
            pchMessageStart[3] = 0xdf;
            hashGenesisBlock =
uint256("0xa52b29c7cae66b6785d06fb944011d7c9f1e1d3acb2f9b58a7cce91505faa4a7");
        }

        //
        // Load block index from databases
        //
        if (!fReindex && !LoadBlockIndexDB())
            return false;

        return true;
    }
}
```

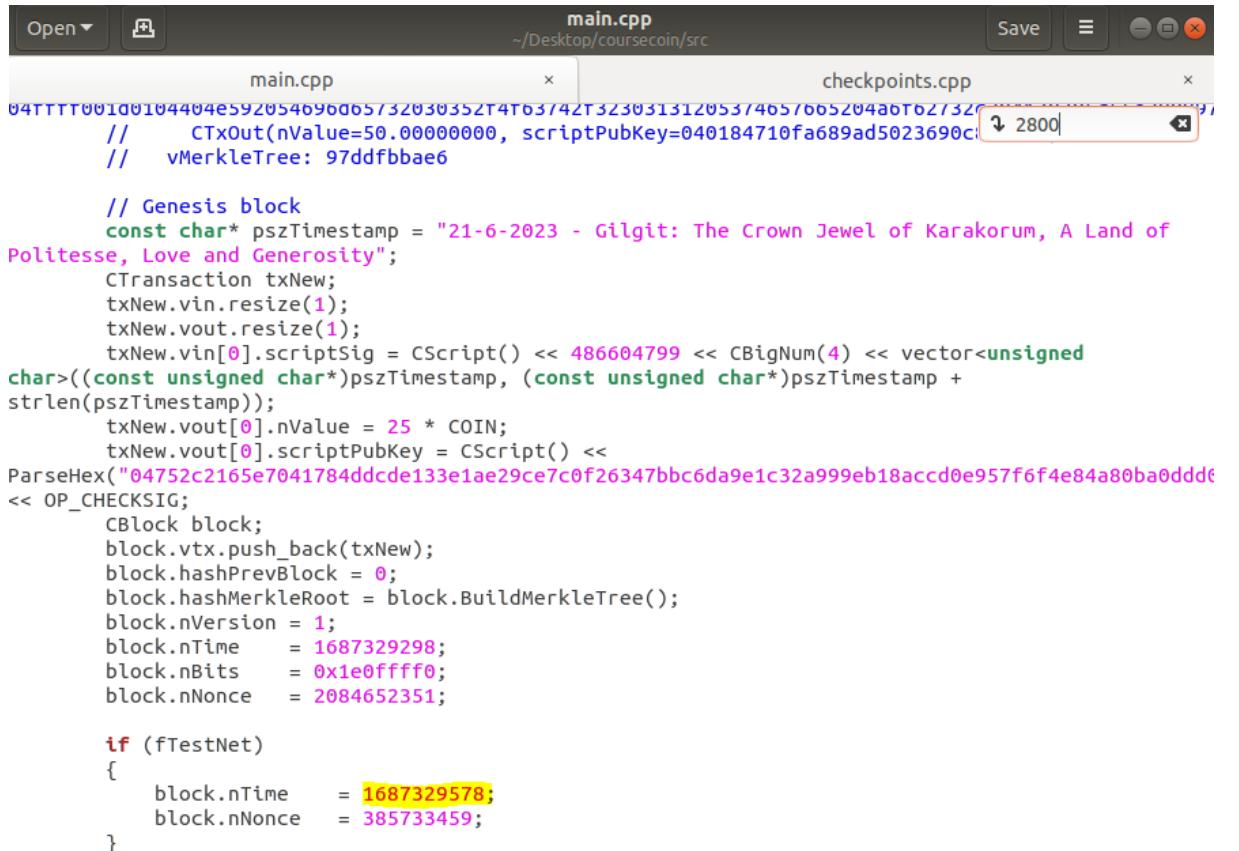
Go to **line 50** in **checkpoints.cpp** and paste the hash that you copied.



```
static MapCheckpoints mapCheckpoints =
    boost::assign::map_list_of
    ( 0, uint256("0xe5dfa84a49dbb1da5aca4219cb69c1f9c064e2cc4f883fe2fd636f6652d59c53"))
;
static const CCheckpointData data = {
    &mapCheckpoints,
    1687329298, // * UNIX timestamp of last checkpoint block
    0, // * total number of transactions between genesis and last checkpoint
        // (the tx=... number in the SetBestChain debug.log lines)
    20.0 // * estimated number of transactions per day after checkpoint
};

static MapCheckpoints mapCheckpointsTestnet =
    boost::assign::map_list_of
    ( 0, uint256("0x"))
;
static const CCheckpointData dataTestnet = {
    &mapCheckpointsTestnet,
    1365458829,
    547,
    576
};
```

Now go to line 2800, and copy the nTime value field, within the if(fTestNet) block.



```
// CTxOut(nValue=50.00000000, scriptPubKey=040184710fa689ad5023690c
// vMerkleTree: 97ddfbbae6

// Genesis block
const char* pszTimestamp = "21-6-2023 - Gilgit: The Crown Jewel of Karakorum, A Land of
Politesse, Love and Generosity";
CTransaction txNew;
txNew.vin.resize(1);
txNew.vout.resize(1);
txNew.vin[0].scriptSig = CScript() << CBigNum(4) << vector<unsigned
char>((const unsigned char*)pszTimestamp, (const unsigned char*)pszTimestamp +
strlen(pszTimestamp));
txNew.vout[0].nValue = 25 * COIN;
txNew.vout[0].scriptPubKey = CScript() <<
ParseHex("04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80ba0ddd6
<< OP_CHECKSIG;
    CBlock block;
    block.vtx.push_back(txNew);
    block.hashPrevBlock = 0;
    block.hashMerkleRoot = block.BuildMerkleTree();
    block.nVersion = 1;
    block.nTime = 1687329298;
    block.nBits = 0x1e0ffff0;
    block.nNonce = 2084652351;

    if (fTestNet)
    {
        block.nTime = 1687329578;
        block.nNonce = 385733459;
    }
```

```

ParseHex("04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80ba0ddd0
<< OP_CHECKSIG;
    CBlock block;
    block.vtx.push_back(txNew);
    block.hashPrevBlock = 0;
    block.hashMerkleRoot = block.BuildMerkleTree();
    block.nVersion = 1;
    block.nTime = 1687329298;
    block.nBits = 0x1e0ffff0;
    block.nNonce = 2084652351;

    if (fTestNet)
    {
        block.nTime = 1687329570;
        block.nNonce = 385733459
    }

    if (true && block.GetHash() != hashGenerated)
    {
        printf("Searching for genesis block...\n");
        // This will figure out a nonce if you're
        // creating a different genesis block.
        uint256 hashTarget = CBigNumber::SetCompact(block.nBits).getuint256();
        uint256 thash;
        char scratchpad[SCRIPT_SCR...

```

A context menu is open over the line of code containing 'block.nTime = 1687329570;'. The menu includes options like Undo, Redo, Cut, Copy, Paste, Delete, Select All, Insert Emoji, and Change Case.

Go to line 54 in checkpoints.cpp

```

checkpoints.cpp
-/Desktop/coursecoin/src
main.cpp x checkpoints.cpp x
static MapCheckpoints mapCheckpoints =
    boost::assign::map_list_of
    ( 0, uint256("0xe5dfa84a49dbb1da5aca4219cb69c1f9c064e2cc4f883fe2fd636f6652d59c53"))
;
static const CCheckpointData data = {
    &mapCheckpoints,
    1687329298, // * UNIX timestamp of last checkpoint block
    0,           // * total number of transactions between genesis and last checkpoint
                 // (the tx=... number in the SetBestChain debug.log lines)
    20.0         // * estimated number of transactions per day after checkpoint
};

static MapCheckpoints mapCheckpointsTestnet =
    boost::assign::map_list_of
    ( 0, uint256("0xa52b29c7cae66b6785d06fb944011d7c9f1e1d3acb2f9b58a7cce91505faa4a7"))
;
static const CCheckpointData dataTestnet = {
    &mapCheckpointsTestnet,
    1365458829,
    547,
    576
};

```

And paste the nTime value that you copied.

```

static MapCheckpoints mapCheckpointsTestnet =
    boost::assign::map_list_of
    ( 0, uint256("0xa52b29c7cae66b6785d06fb944011d7c9f1e1d3acb2f9b58a7cce91505faa4a7"))
;
static const CCheckpointData dataTestnet = {
    &mapCheckpointsTestnet,
    1687329578,
    547,
    576
};

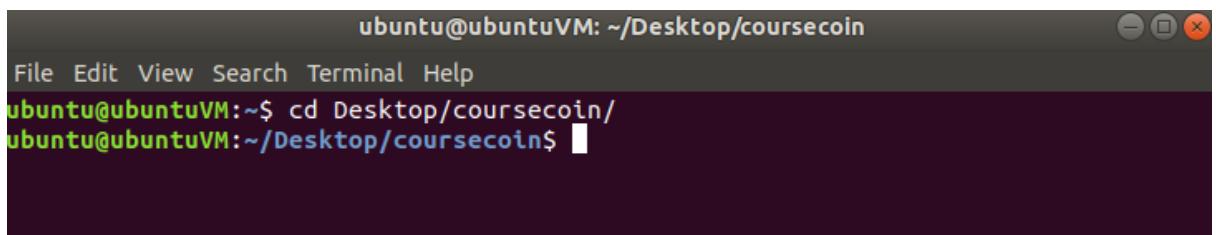
```

Change the second value from 547 to 0 and 576 to 20.0

```
static MapCheckpoints mapCheckpointsTestnet =
    boost::assign::map_list_of
    ( 0, uint256("0xa52b29c7cae66b6785d06fb944011d7c9f1e1d3acb2f9b58a7cce91505faa4a7"))
;
static const CCheckpointData dataTestnet = {
    &mapCheckpointsTestnet,
    1687329578,
    0,
    20.0
};
```

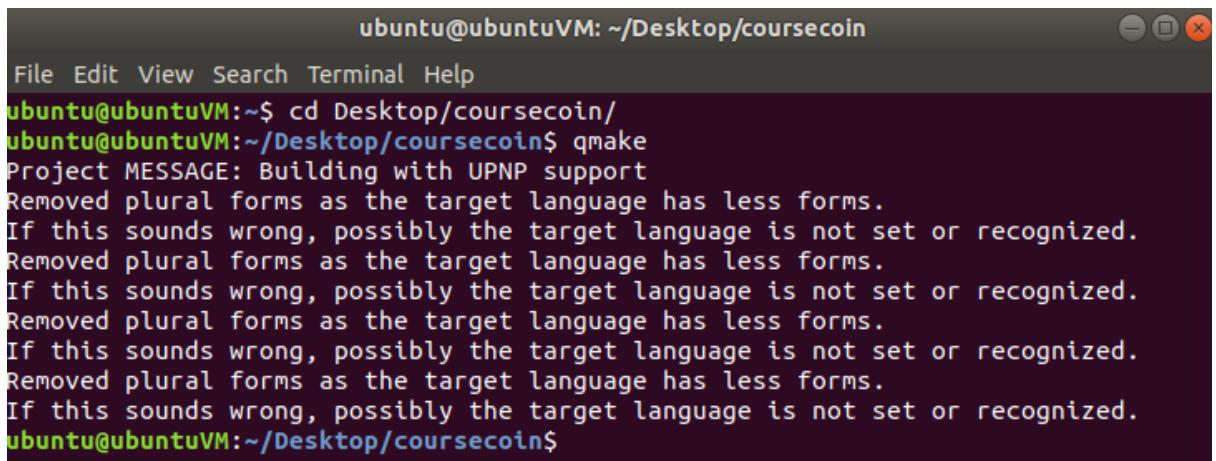
Save the checkpoints.cpp file and main.cpp file, and close all the windows.

5. Open your terminal and navigate to your coin directory (Don't have to go inside src directory)



```
ubuntu@ubuntuVM: ~/Desktop/coursecoin
File Edit View Search Terminal Help
ubuntu@ubuntuVM:~$ cd Desktop/coursecoin/
ubuntu@ubuntuVM:~/Desktop/coursecoin$
```

Type the command “qmake”



```
ubuntu@ubuntuVM: ~/Desktop/coursecoin
File Edit View Search Terminal Help
ubuntu@ubuntuVM:~$ cd Desktop/coursecoin/
ubuntu@ubuntuVM:~/Desktop/coursecoin$ qmake
Project MESSAGE: Building with UPNP support
Removed plural forms as the target language has less forms.
If this sounds wrong, possibly the target language is not set or recognized.
Removed plural forms as the target language has less forms.
If this sounds wrong, possibly the target language is not set or recognized.
Removed plural forms as the target language has less forms.
If this sounds wrong, possibly the target language is not set or recognized.
Removed plural forms as the target language has less forms.
If this sounds wrong, possibly the target language is not set or recognized.
ubuntu@ubuntuVM:~/Desktop/coursecoin$
```

After this just type the command “**make**” in your terminal

```
ubuntu@ubuntuVM:~/Desktop/coursecoin$ make
cd /home/ubuntu/Desktop/coursecoin/src/leveldb && CC=gcc CXX=g++ make OPT="-m64 -pipe -fstack-protector-all -U_FORTIFY_SOURCE -D_FORTIFY_SOURCE=2 -O2" libleveldb.a libmemenv.a
make[1]: Entering directory '/home/ubuntu/Desktop/coursecoin/src/leveldb'
g++ -I. -I./include -fno-built-in-memcmp -pthread -DOS_LINUX -DLEVELDB_PLATFORM_POSIX -m64 -pipe -fstack-protector-all -U_FORTIFY_SOURCE -D_FORTIFY_SOURCE=2 -O2 -c db/dbformat.cc -o db/dbformat.o
g++ -I. -I./include -fno-built-in-memcmp -pthread -DOS_LINUX -DLEVELDB_PLATFORM_POSIX -m64 -pipe -fstack-protector-all -U_FORTIFY_SOURCE -D_FORTIFY_SOURCE=2 -O2 -c db/db_iter.cc -o db/db_iter.o
g++ -I. -I./include -fno-built-in-memcmp -pthread -DOS_LINUX -DLEVELDB_PLATFORM_POSIX -m64 -pipe -fstack-protector-all -U_FORTIFY_SOURCE -D_FORTIFY_SOURCE=2 -O2 -c db/log_reader.cc -o db/log_reader.o
```

[Disclaimer: It will take some time, so please do not interrupt this process]

6. After it's finished type “**ls**” while staying in your coin directory.

```
build/moc_optionsdialog.o build/moc_sendcoinsdialog.o build/moc_coincontroldialog.o build/moc_coincontroltreewidget.o build/moc_addressbookpage.o build/moc_signverifymessagedialog.o build/moc_abo utdialog.o build/moc_editaddressdialog.o build/moc_bitcoinaddressvalidator.o build/moc_clientmode l.o build/moc_guiful.o build/moc_optionsmodel.o build/moc_monitoreddatamapper.o build/moc_transa ctiondesc.o build/moc_transactiondescdialog.o build/moc_bitcoinamountfield.o build/moc_transactio nfilterproxy.o build/moc_transactionview.o build/moc_walletmodel.o build/moc_walletview.o build/m oc_walletstack.o build/moc_walletframe.o build/moc_overviewpage.o build/moc_csvmodelwriter.o buil d/moc_sendcoinsentry.o build/moc_validatedlineedit.o build/moc_bitcoinunits.o build/moc_qvaluenco mbobox.o build/moc_askpassphrasedialog.o build/moc_notificator.o build/moc_paymentserver.o build/ moc_rpcconsole.o build/moc_macnotificationhandler.o build/moc_splashscreen.o build/qrc_bitcoin.o
-L/usr/lib/x86_64-linux-gnu -lminiupnpc /home/ubuntu/Desktop/coursecoin/src/leveldb/libleveldb .a /home/ubuntu/Desktop/coursecoin/src/leveldb/libmemenv.a -lrt -lssl -lcrypto -ldb_cxx -lboost_system -lboost_filesystem -lboost_program_options -lboost_thread -lpthread -lQtGui -lQtNetwork -lQtCore
ubuntu@ubuntuVM:~/Desktop/coursecoin$ ls ←
bitcoin-qt.pro contrib coursecoin-qt INSTALL qa README.md src
build COPYING doc Makefile qrc_bitcoin.cpp share
ubuntu@ubuntuVM:~/Desktop/coursecoin$
```

You'll see an executable file with graphical resources, e.g. yourcoinname-qt or in this case **coursecoin-qt**.

7. Now type the following command in your terminal **./yourcoinname-qt** and you'll see a wallet interface. In this case it is **./coursecoin-qt**.

```
ubuntu@ubuntuVM:~/Desktop/coursecoin$ ./coursecoin-qt
Gtk-Message: 15:15:23.529: Failed to load module "canberra-gtk-module"
No systemtrayicon available
```

If everything was done right, you would see a wallet interface such as below for the blockchain you just created.

