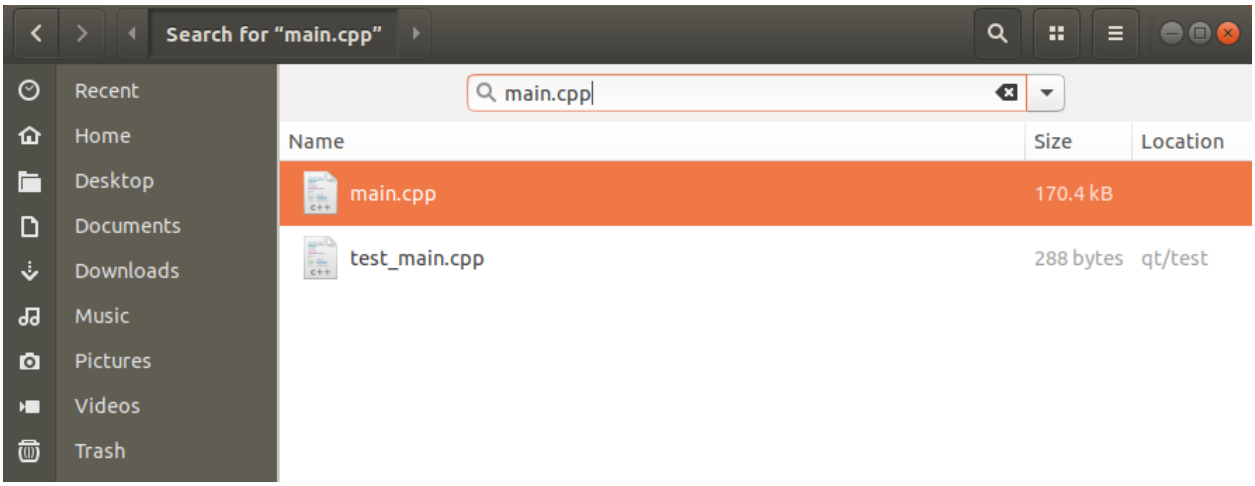
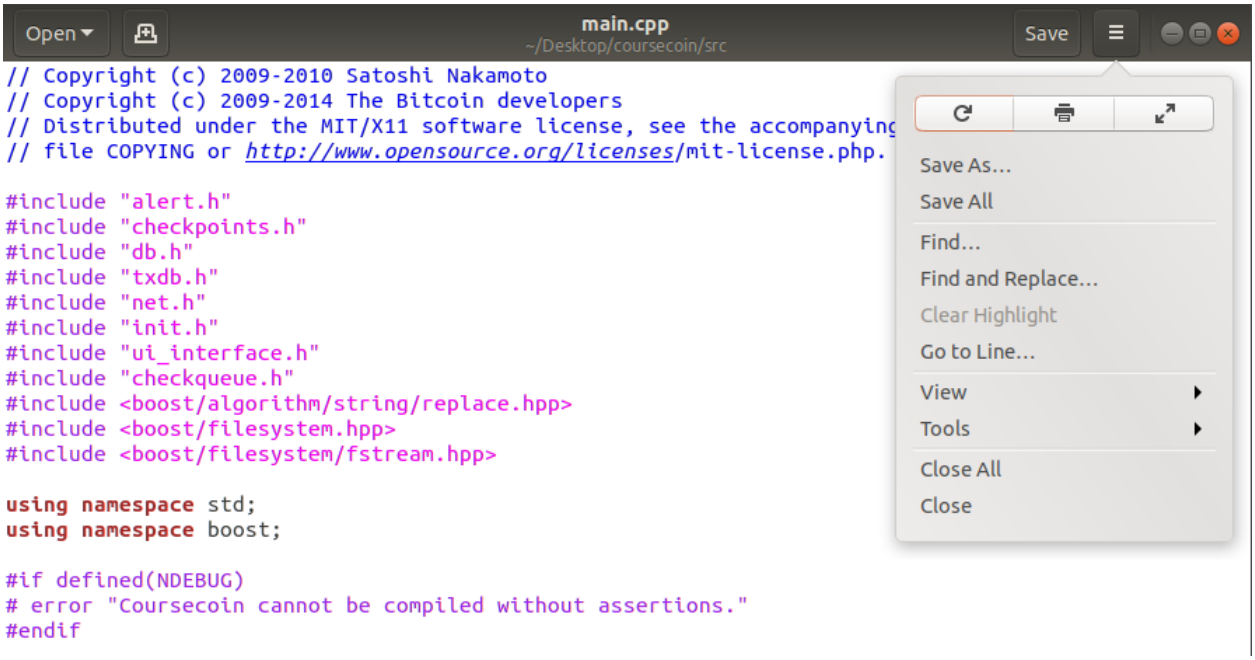
Lab 21-6-2023

[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.



Once open go to **line 1090**



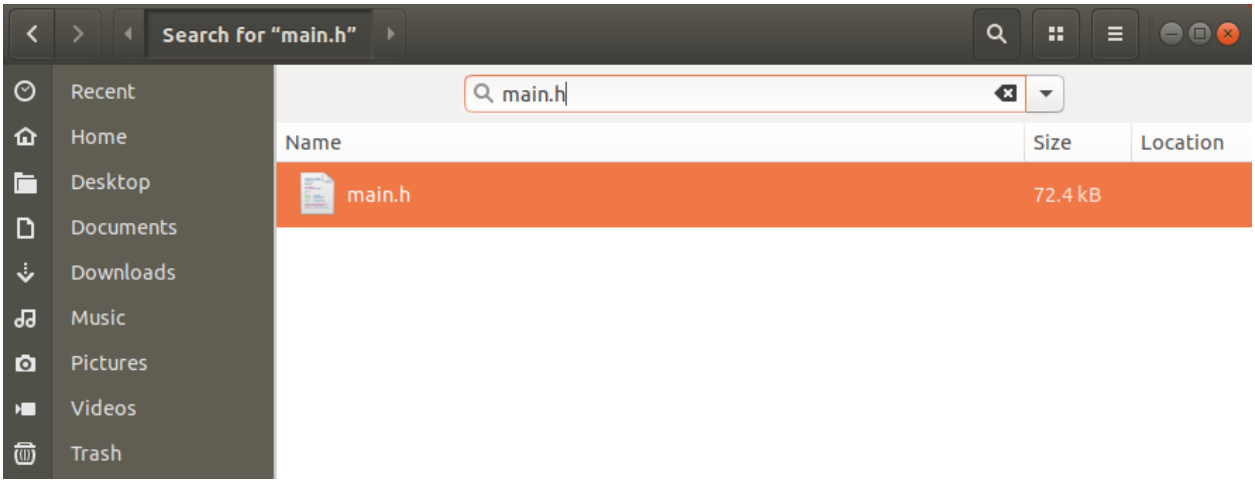
You'll see this snippet of code

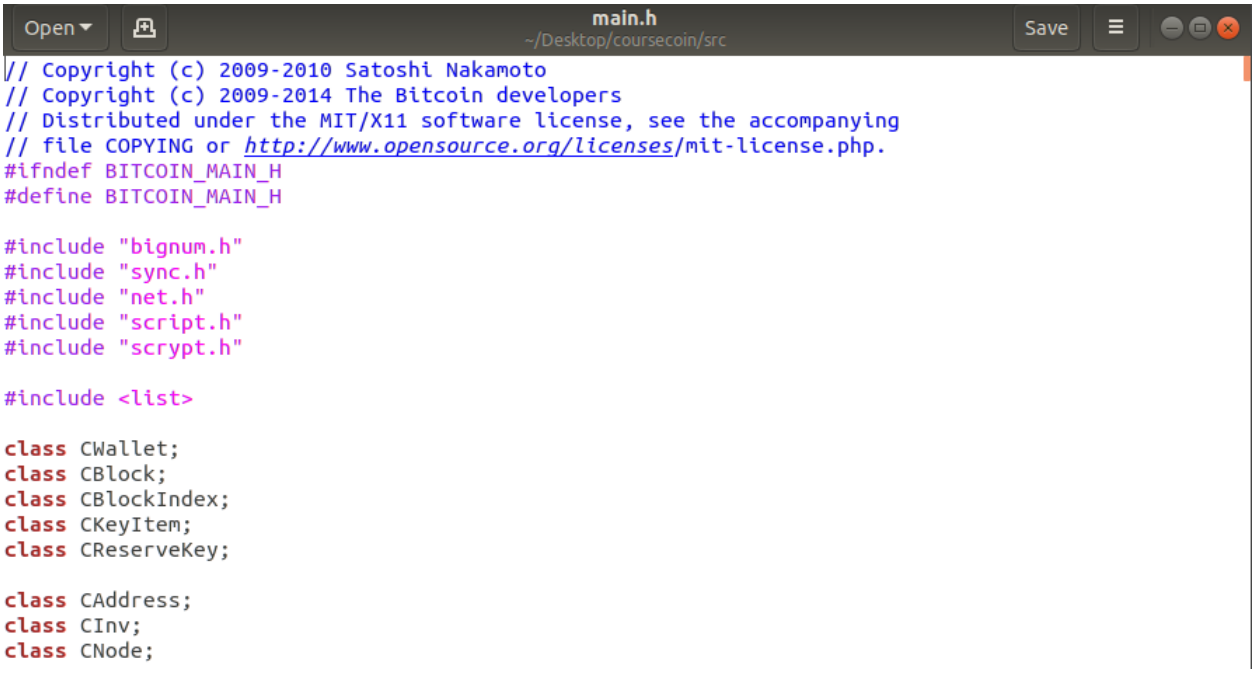
Change the block mining reward denoted by **nSubsidy from 50 to 25,** change the number of blocks after which subsidy will be halved from **840000 to 420000**, change the timespan for readjustment of difficulty level from **3.5 days to 3 days**. Lastly, change the target block time from **2.5 minutes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

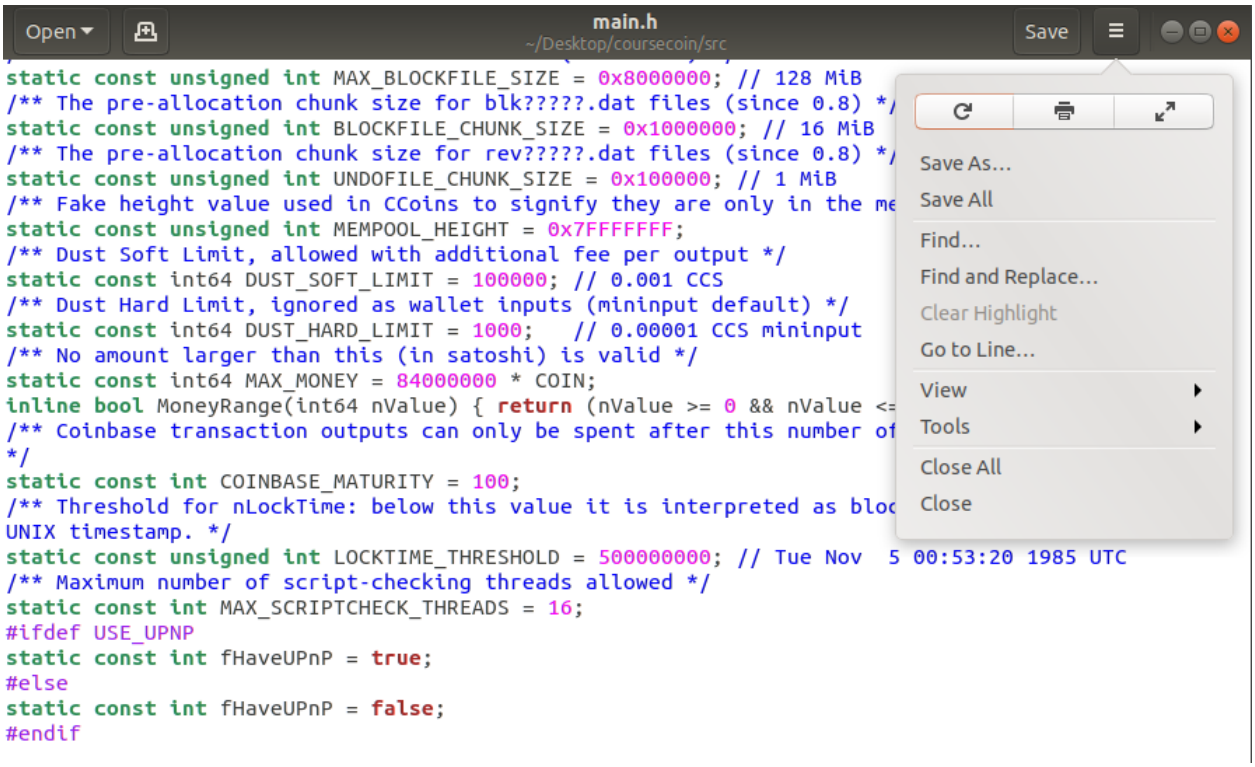
Save the main.cpp and close it.

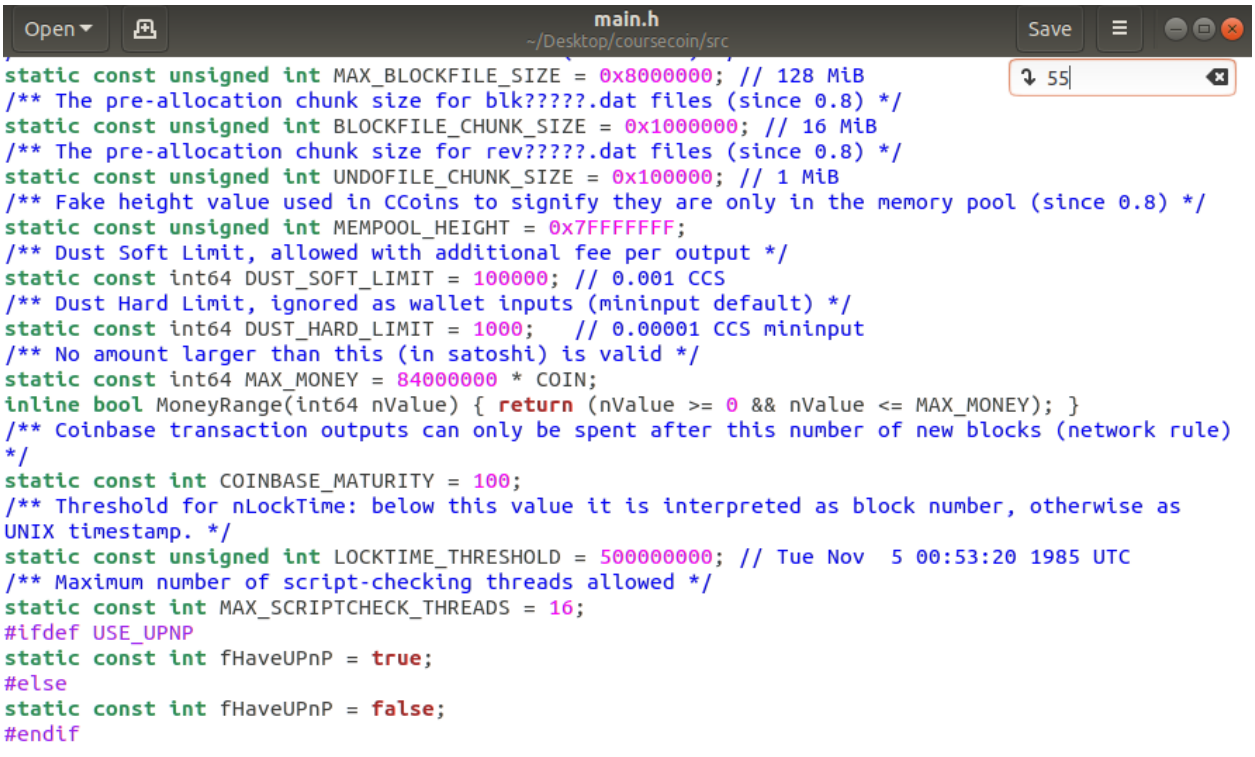
2. Go to the **src directory** within your **coin directory** and search for a file named **"main.h"**. Open it by double clicking it.





Go to **line 55**


Open
main.h
~/Desktop/coursecoin/src
Save
Save As...
Save All
Find...
Find and Replace...
Clear Highlight
Go to Line...
View
Tools
Close All
Close
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 me
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 <=
/** Coinbase transaction outputs can only be spent after this number of
*/
static const int COINBASE_MATURITY = 100;
/** Threshold for nLockTime: below this value it is interpreted as bloc
UNIX timestamp. */
static const unsigned int LOCKTIME_THRESHOLD = 500000000; // Tue Nov 5 00:53:20 1985 UTC
/** Maximum number of script-checking threads allowed */
static const int MAX_SCRIPTCHECK_THREADS = 16;
#ifdef USE_UPNP
static const int fHaveUPnP = true;
#else
static const int fHaveUPnP = false;
#endif


Open
main.h
~/Desktop/coursecoin/src
Save
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 = 500000000; // Tue Nov 5 00:53:20 1985 UTC
/** Maximum number of script-checking threads allowed */
static const int MAX_SCRIPTCHECK_THREADS = 16;
#ifdef 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.**

Open ▾ *main.h ~/Desktop/coursecoin/src Save

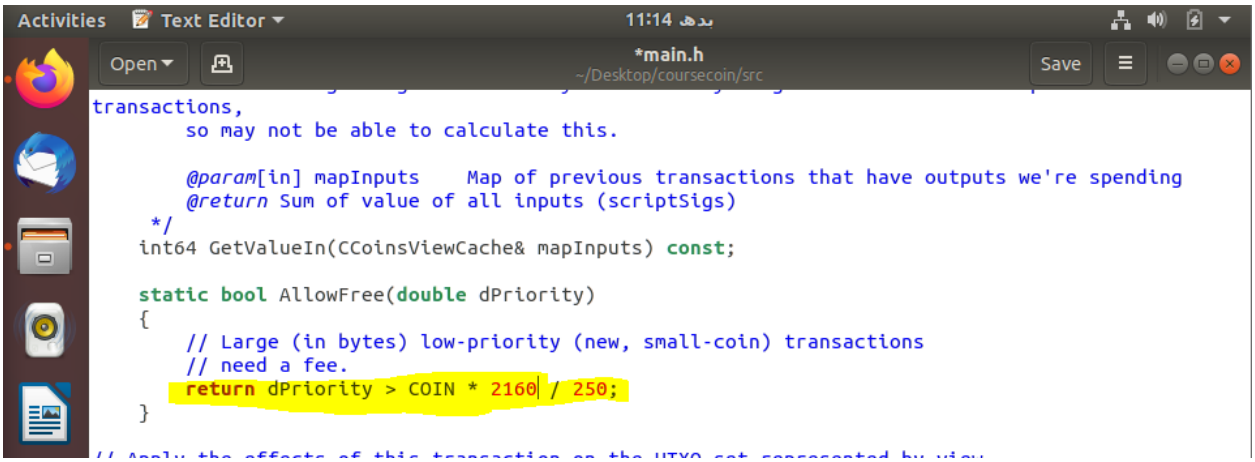
```
static const unsigned int MAX_BLOCKFILE_SIZE = 0x8000000; // 128 MiB
/** The pre-allocation chunk size for blk?????.dat files (since 0.8) */
static const unsigned int BLOCKFILE_CHUNK_SIZE = 0x1000000; // 16 MiB
/** The pre-allocation chunk size for rev?????.dat files (since 0.8) */
static const unsigned 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 = 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 = 500000000; // Tue Nov  5 00:53:20 1985 UTC
/** Maximum number of script-checking threads allowed */
static const int MAX_SCRIPTCHECK_THREADS = 16;
#ifdef 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

Open ▾ main.h ~/Desktop/coursecoin/src Save 627

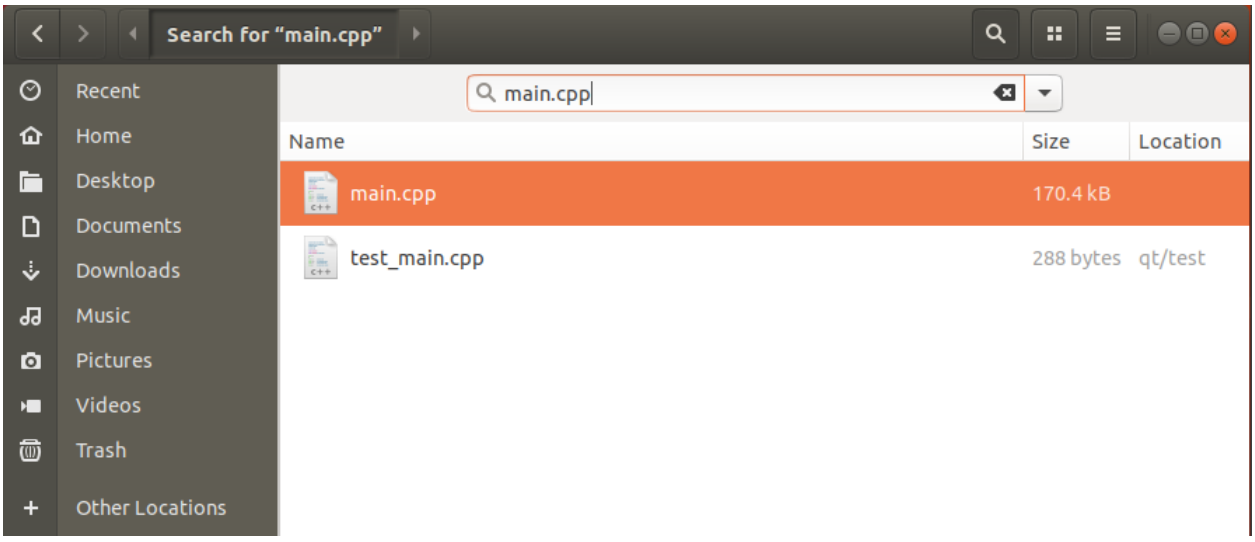
```
transactions,
        so may not be able to calculate this.

        @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, CCoinsViewCache &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**



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

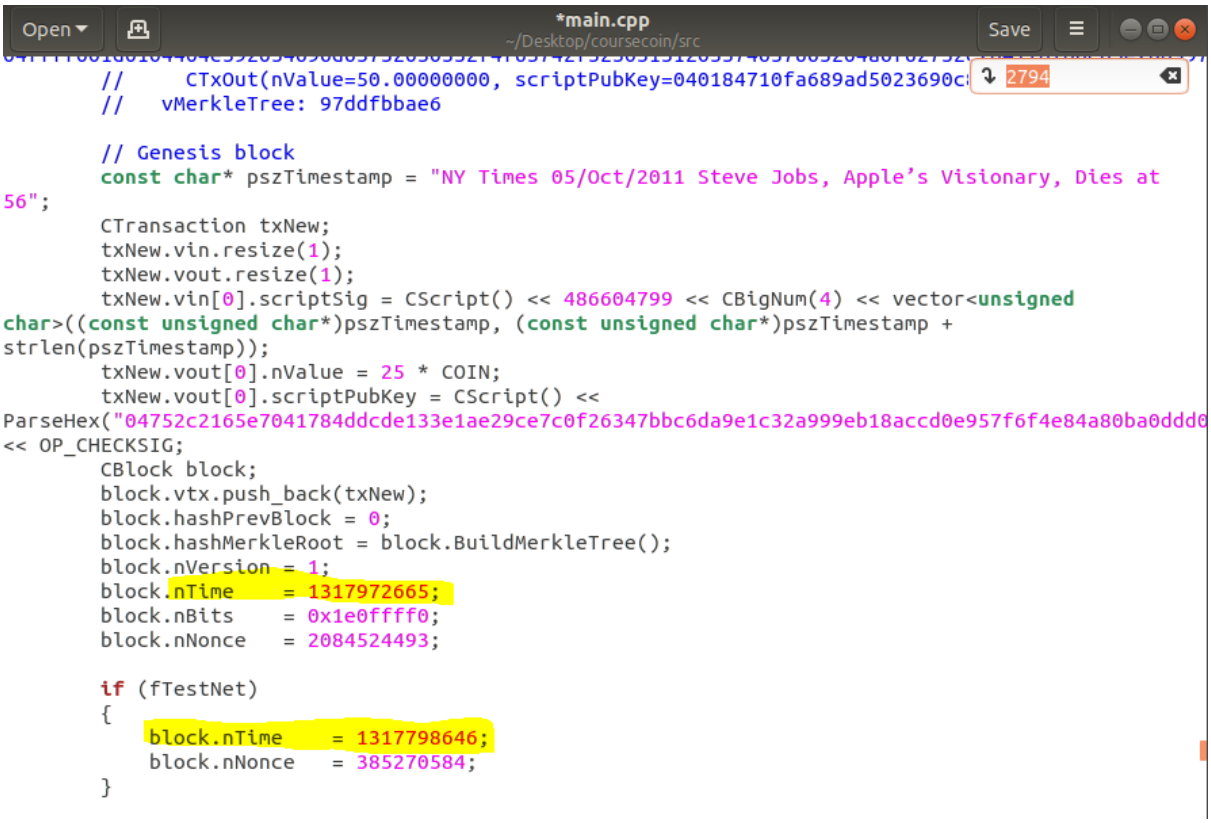
3. Go to the **src directory** within your **coin directory** using file explorer and search for a file named **“main.cpp”.** Open it by double clicking it.



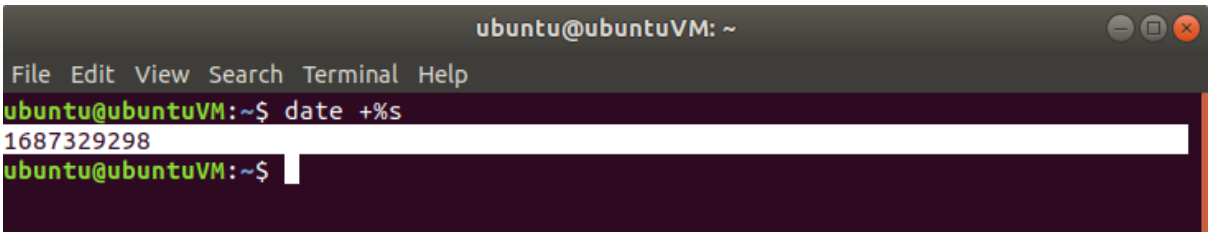
Go to **line 2787** and you’ll see this snippet of code.

Change the **nValue from 50 to 25**.

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



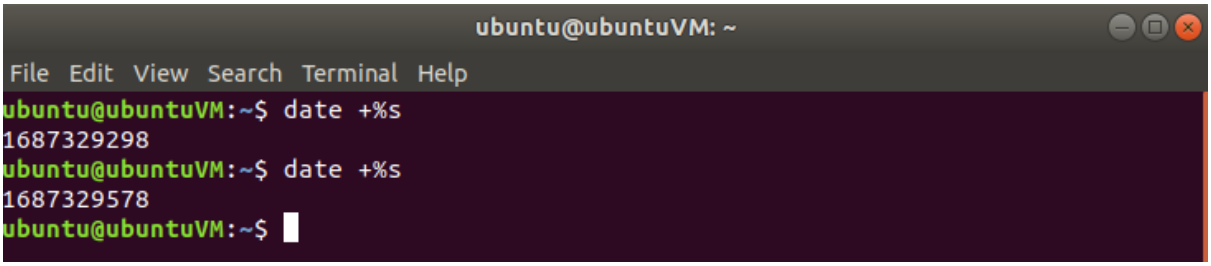
For that open your terminal and type this command **"date +%s"**



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**.



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**

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

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

main.cpp
~/Desktop/coursecoin/src

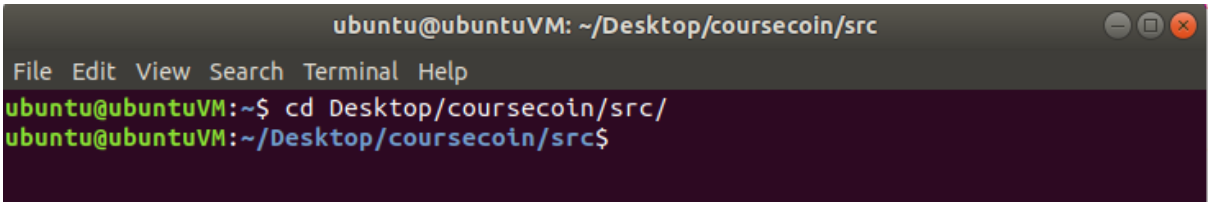
```
        const char* pszTimestamp = "21-6-2023 - Gilgit: The Crown Jewel of Karal
Politesse, Love and Generosity";
        CTransaction txNew;
        txNew.vin.resize(1);
        txNew.vout.resize(1);
        txNew.vin[0].scriptSig = CScript() << 486604799 << CBigNum(4) << vector<unsigned
char>((const 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**

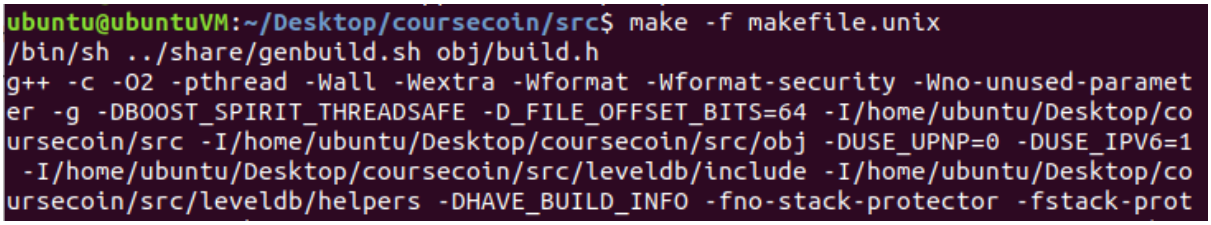
*main.cpp
~/Desktop/coursecoin/src

```
        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);
```

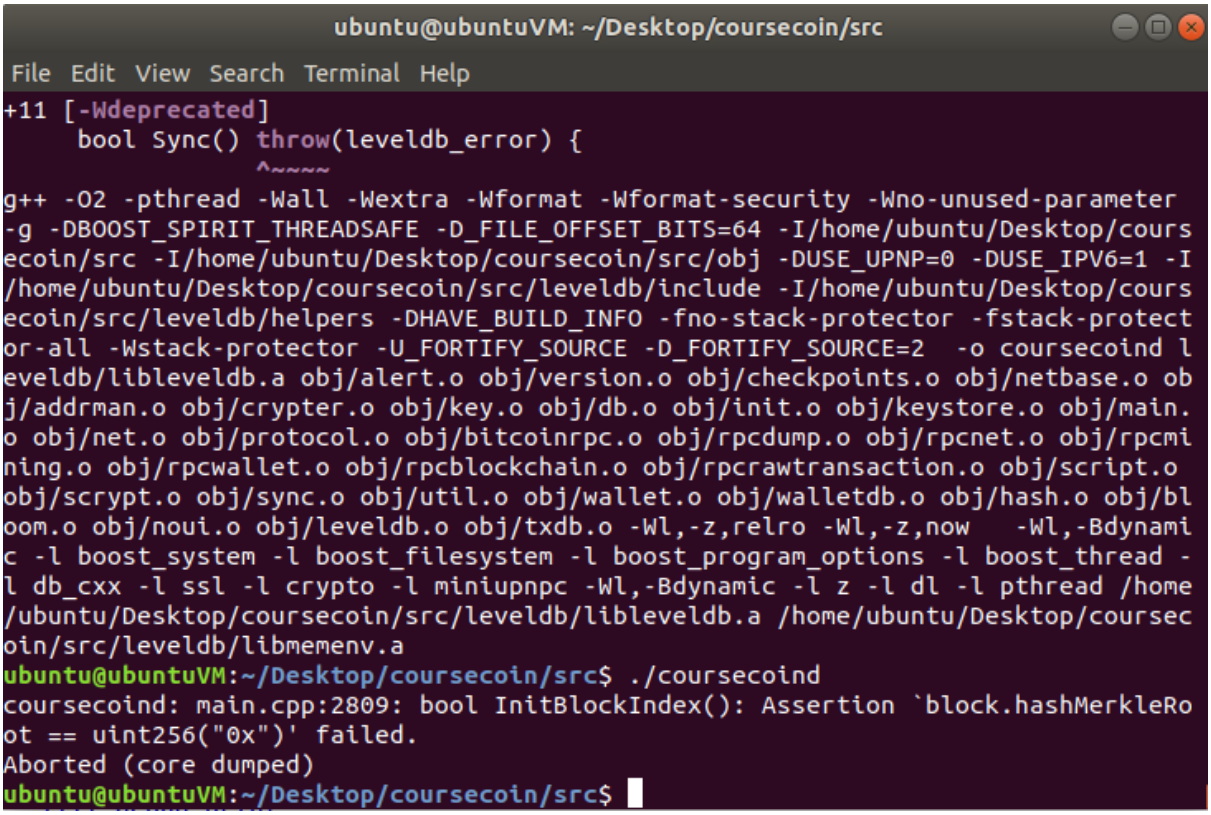
Save the main.cpp file, and to generate a new merkle root open your terminal and navigate to the **src directory** in your **coin directory**



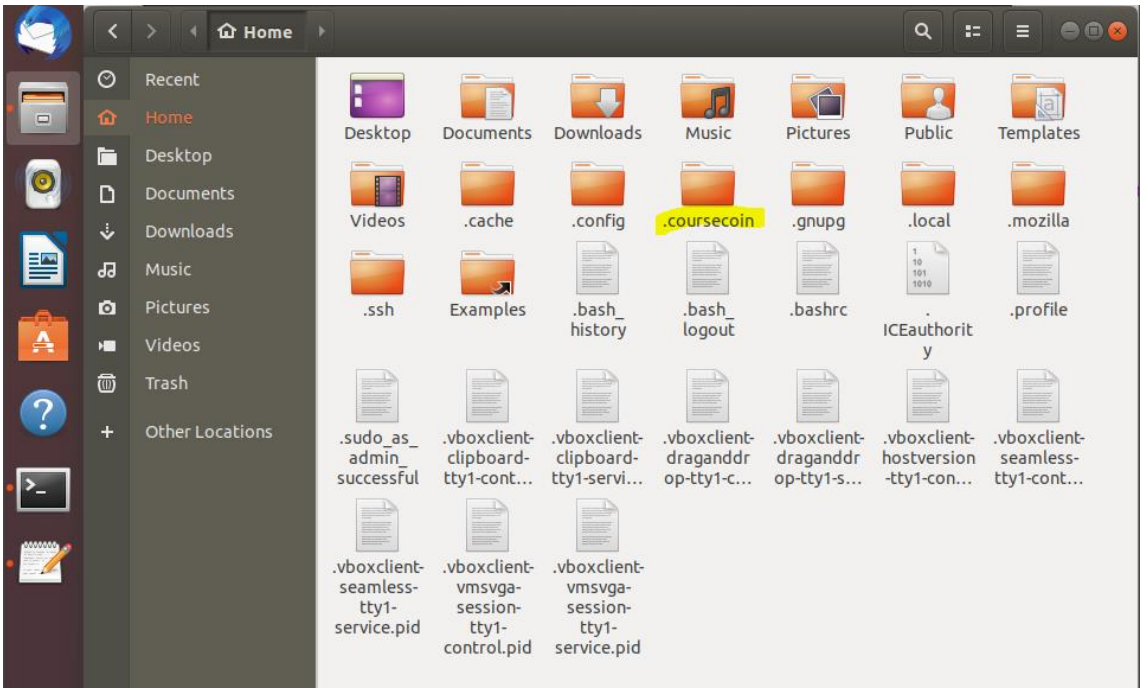
Type the command **"make –f makefile.unix"** to compile it.



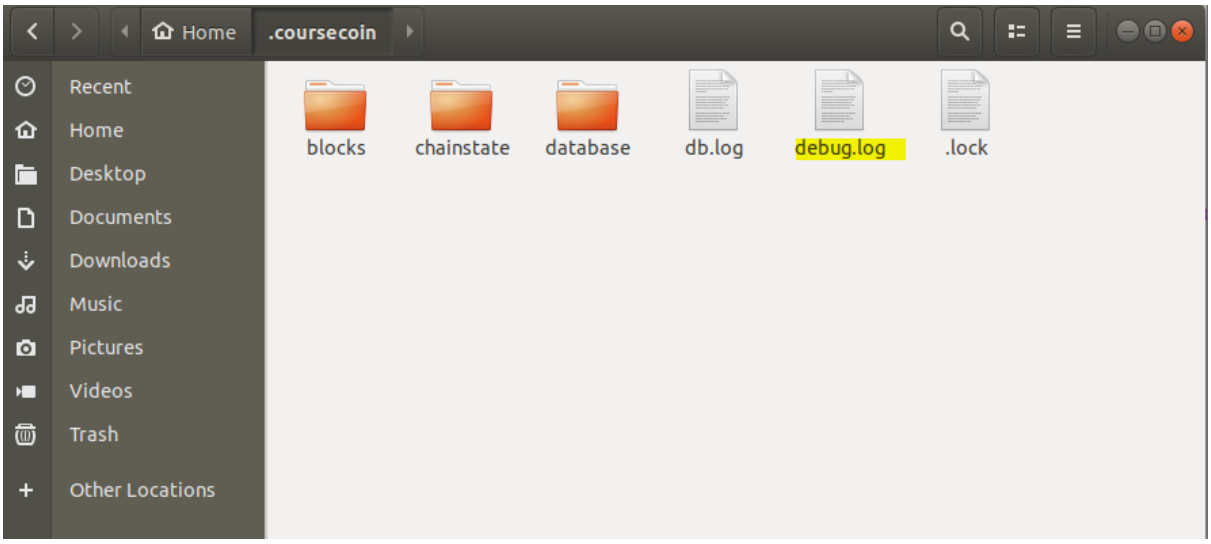
After the compilation type the command **(./yourcoinname with d at the end)** In my case it is **"./coursecoind".** Running this command will result into a failure.



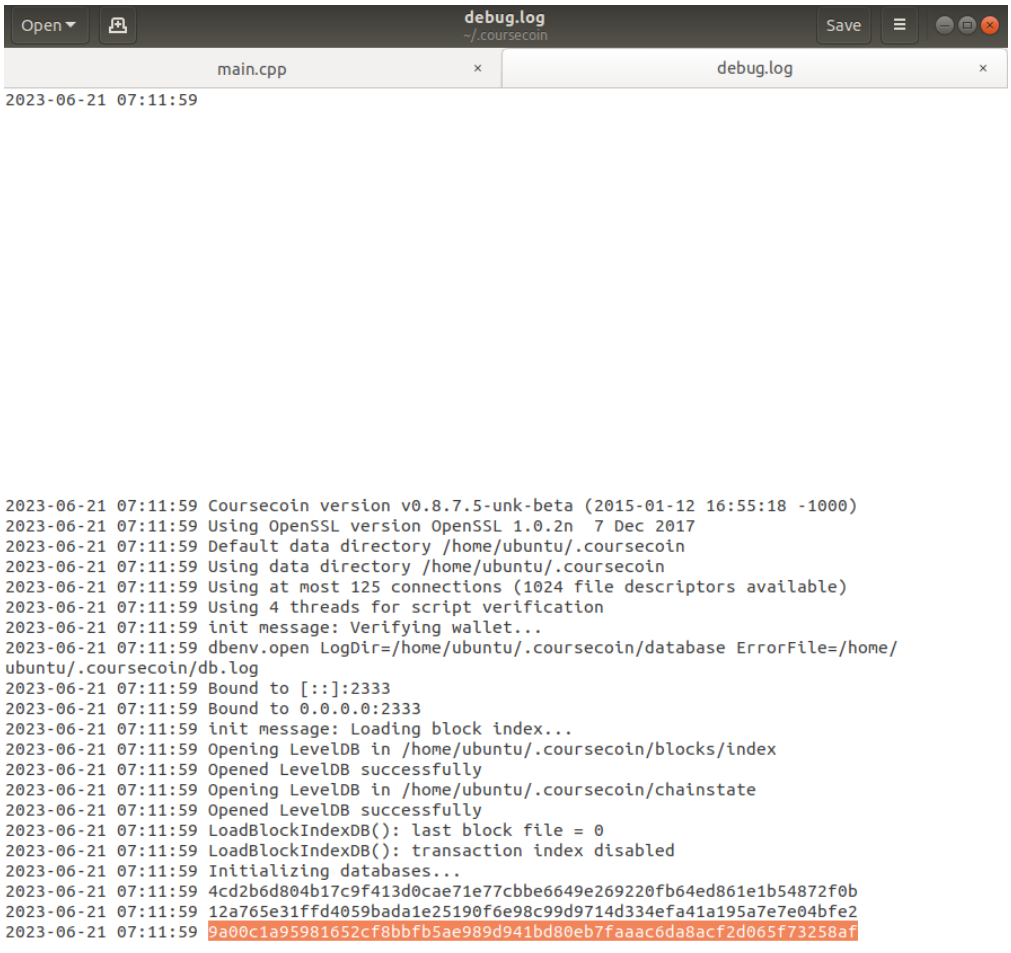
Now to find the merkle root go to the home directory in file explorer, press **Ctrl+H** to show the hidden files. You'll see a hidden directory named after your coin.



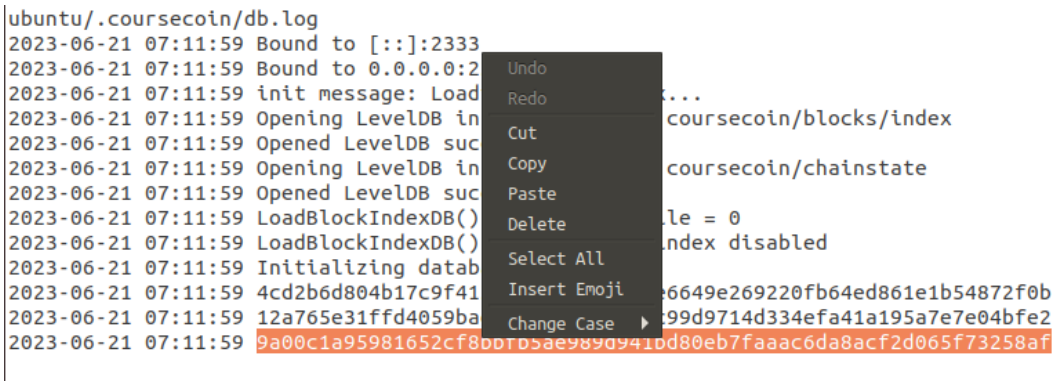
Go inside this directory, you'll see a file named **debug.log**. Open the debug.log



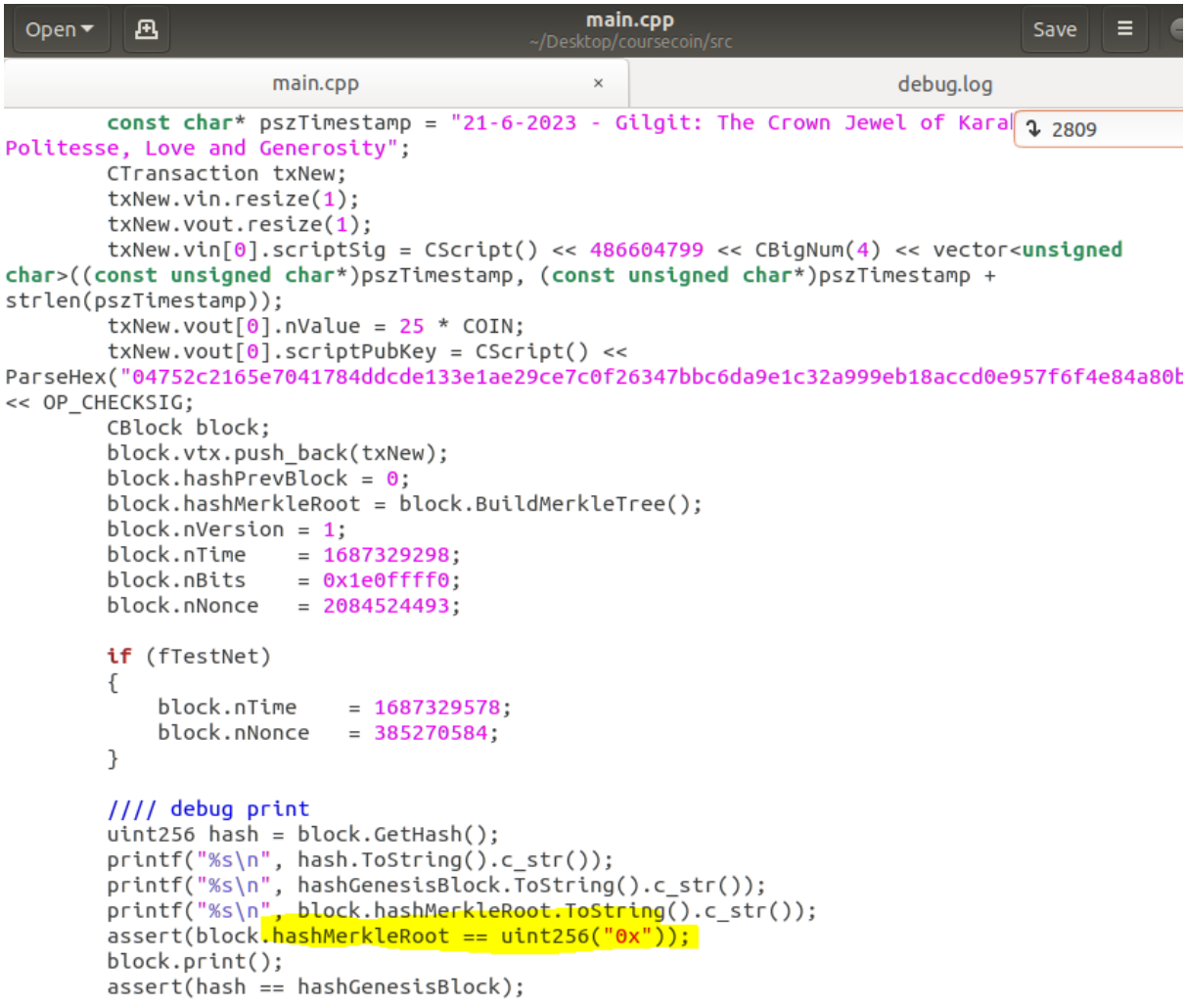
Scroll to the bottom and the last string is your merkle root.



Copy this merkle root



Go to **line 2809** in **main.cpp**

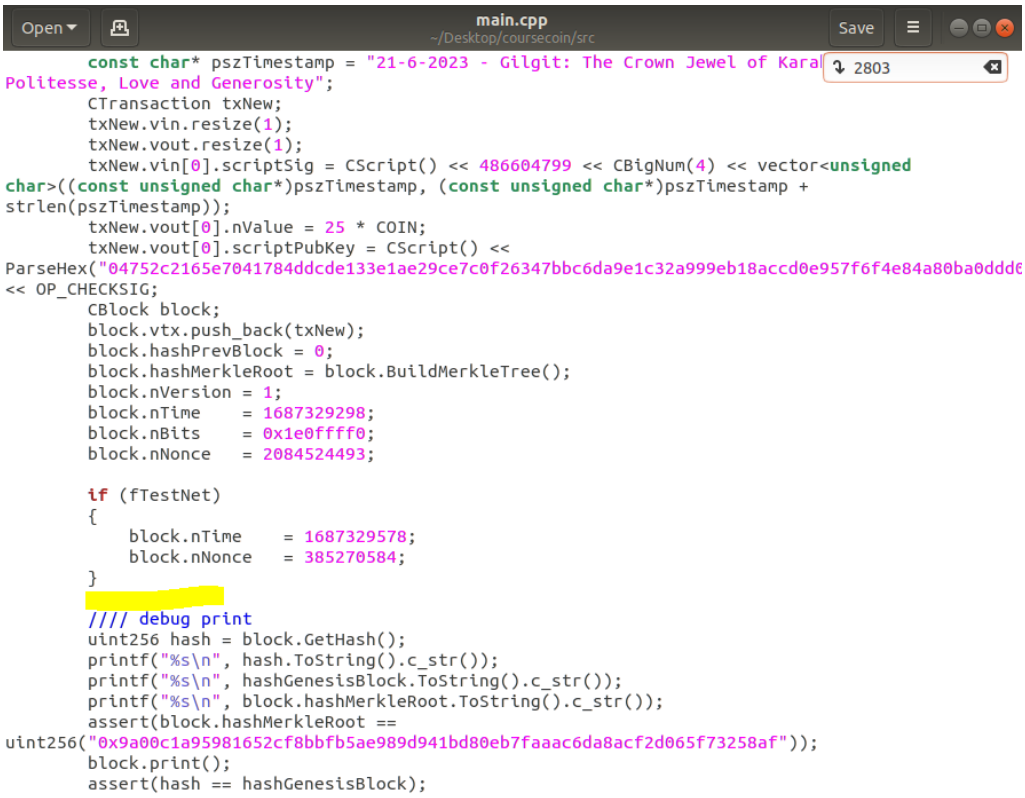


Right after **0x** paste this newly obtained merkle root

```
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**



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

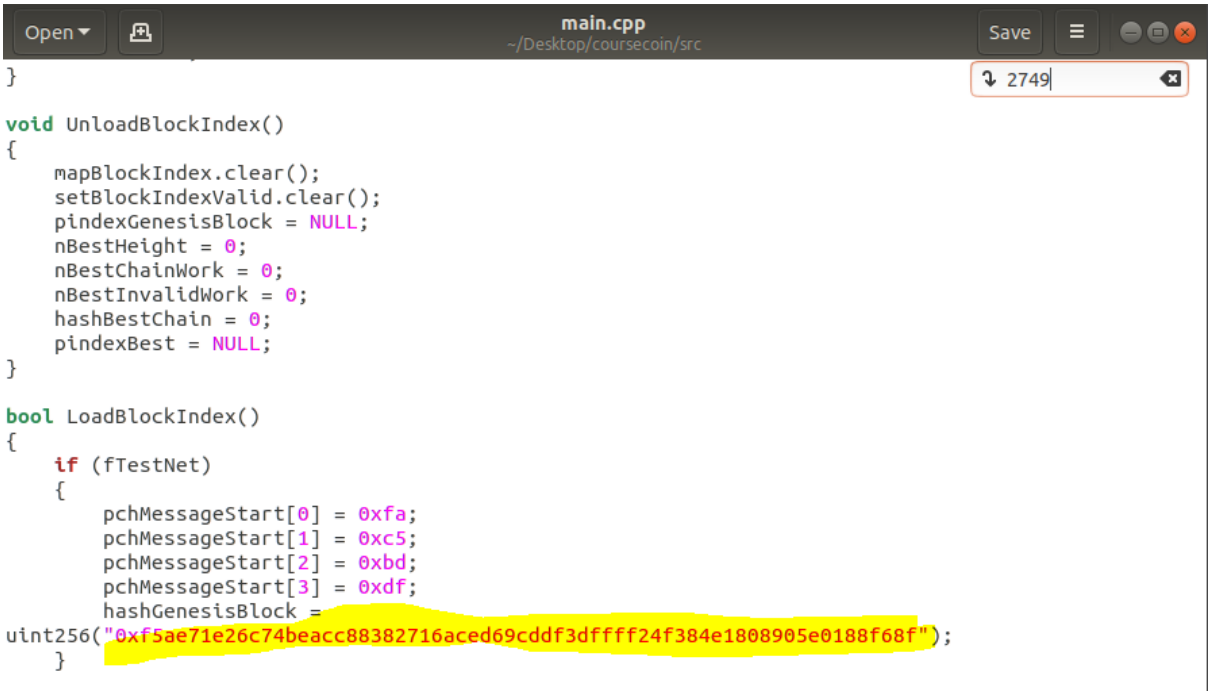
```
if (true && block.GetHash() != hashGenesisBlock)
    {
        printf("Searching for genesis block...\n");
        // This will figure out a valid hash and Nonce if you're
        // creating a different genesis block:
        uint256 hashTarget =
CBigNum().SetCompact(block.nBits).getuint256();
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

        loop
        {
#if defined(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 & 0xFFF) == 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[SCRYPT_SCRATCHPAD_SIZE];
```

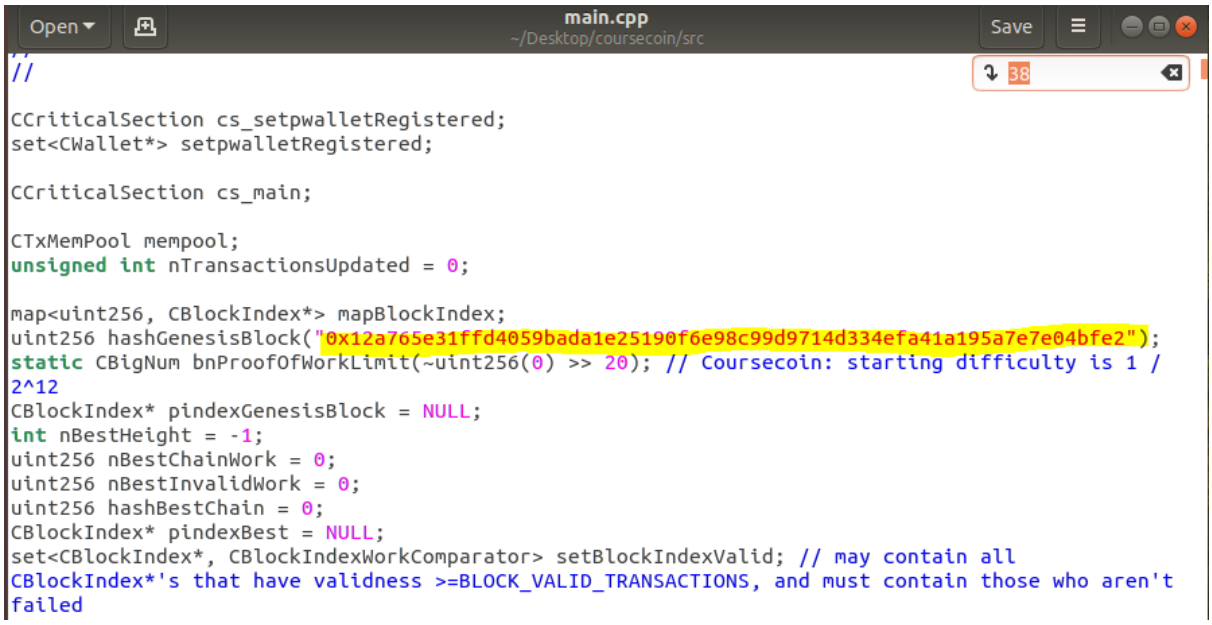
Save the file and go to **line 2749**



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;
```

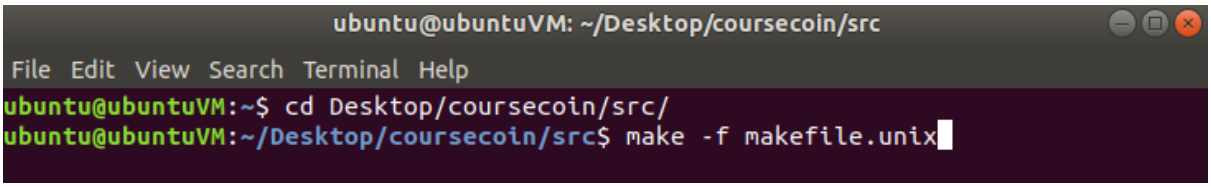
Now go to line 38, you'll see the main net genesis block hash



Erase it too up until **0x**

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”.**



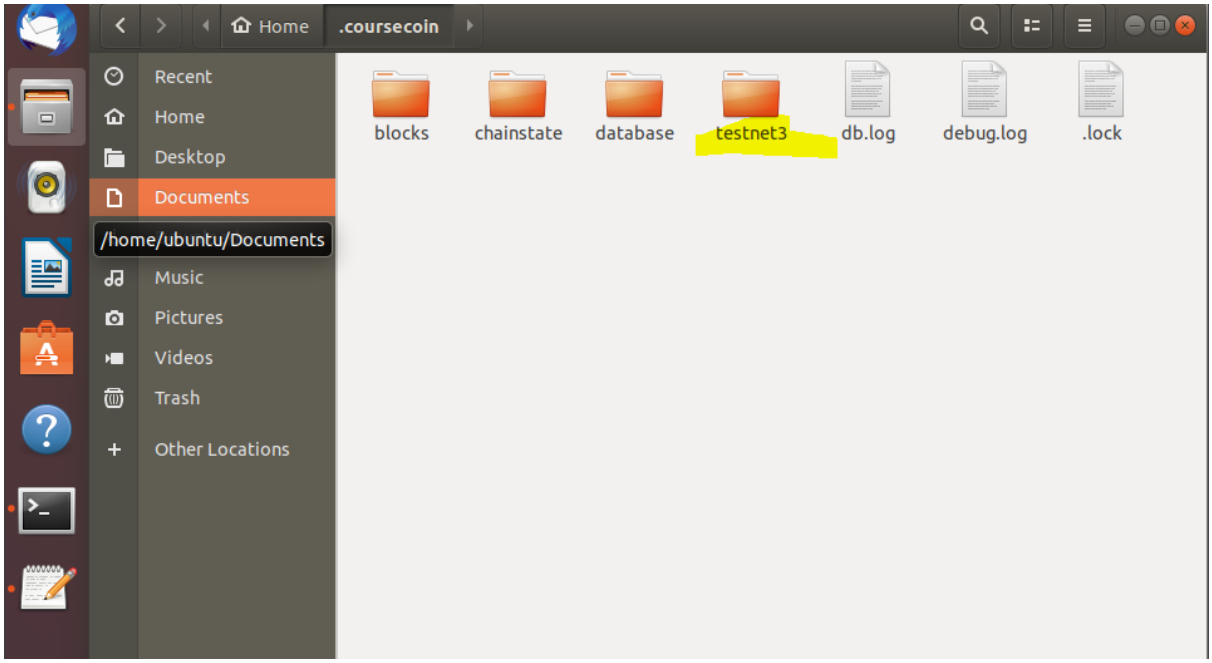
Once the compilation has been done, use the command ./yourcoinname with d at the end, in my case it’s ./coursecoind with a –testnet flag e.g. **(./coursecoind –testnet)** to generate the genesis block hash for the test net.

```
g++ -O2 -pthread -Wall -Wextra -Wformat -Wformat-security -Wno-unused-parameter
-g -DBOOST_SPIRIT_THREADSAFE -D_FILE_OFFSET_BITS=64 -I/home/ubuntu/Desktop/cours
ecoin/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/cours
ecoin/src/leveldb/helpers -DHAVE_BUILD_INFO -fno-stack-protector -fstack-protect
or-all -Wstack-protector -U_FORTIFY_SOURCE -D_FORTIFY_SOURCE=2  -o coursecoind l
eveldb/libleveldb.a obj/alert.o obj/version.o obj/checkpoints.o obj/netbase.o ob
j/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/rpcmi
ning.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/bl
oom.o obj/noui.o obj/leveldb.o obj/txdb.o -Wl,-z,relro -Wl,-z,now   -Wl,-Bdynami
c -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/coursec
oin/src/leveldb/libmemenv.a
ubuntu@ubuntuVM:~/Desktop/coursecoin/src$ ./coursecoind -testnet
```

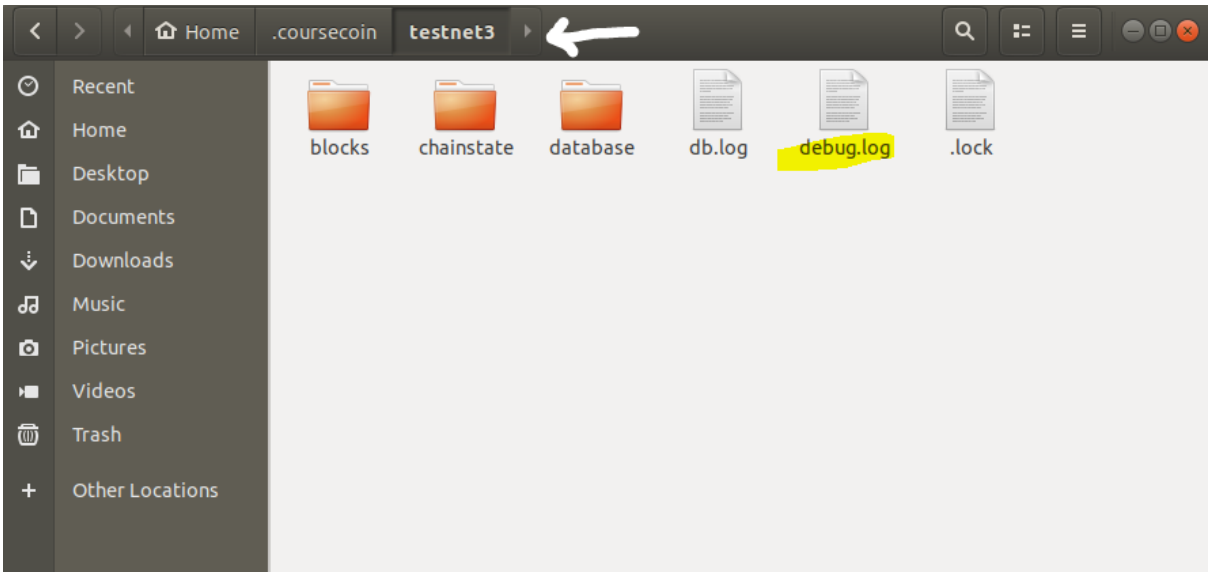
It will take some time so be patient and wait for the core to be dumped.

```
ubuntu@ubuntuVM:~/Desktop/coursecoin/src$ ./coursecoind -testnet
coursecoind: main.cpp:2853: bool InitBlockIndex(): Assertion `hash == hashGenesi
sBlock' 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

```
00000ffff0000000000000000000000000000000000000000000000000000000)
2023-06-21 08:08:10 nonce 16FDC000: hash =
244d63aba4f131f9188f369b972a7c5bf8d23d333312d9e3f84dff922cfd76db (target =
00000ffff0000000000000000000000000000000000000000000000000000000)
2023-06-21 08:08:11 nonce 16FDD000: hash =
62321c76ce4a9c4f08b6b474cf957a95360750a198e961e62b4870e06d14a5e6 (target =
00000ffff0000000000000000000000000000000000000000000000000000000)
2023-06-21 08:08:11 block.nTime = 1687329578
2023-06-21 08:08:11 block.nNonce = 385733459
2023-06-21 08:08:11 block.GetHash =
a52b29c7cae66b6785d06fb944011d7c9f1e1d3acb2f9b58a7cce91505faa4a7
2023-06-21 08:08:11 a52b29c7cae66b6785d06fb944011d7c9f1e1d3acb2f9b58a7cce91505faa4a7
2023-06-21 08:08:11 0000000000000000000000000000000000000000000000000000000000000000
2023-06-21 08:08:11 9a00c1a95981652cf8bbfb5ae989d941bd80eb7faaac6da8acf2d065f73258af
2023-06-21 08:08:11 CBlock(hash=a52b29c7cae66b6785d06fb944011d7c9f1e1d3acb2f9b58a7cce91505faa4a7,
input=0100000000000000000000000000000000000000000000000000000000000000000000000000af5832f765d0f2aca86da
PoW=000009c75e9e66e9a9a31bfc17f30076cea656d54bf30e785a3b2370cb32880f, ver=1,
hashPrevBlock=0000000000000000000000000000000000000000000000000000000000000000,
hashMerkleRoot=9a00c1a95981652cf8bbfb5ae989d941bd80eb7faaac6da8acf2d065f73258af,
nTime=1687329578, nBits=1e0ffff0, nNonce=385733459, vtx=1)
2023-06-21 08:08:11
CTransaction(hash=9a00c1a95981652cf8bbfb5ae989d941bd80eb7faaac6da8acf2d065f73258af, ver=1,
vin.size=1, vout.size=1, nLockTime=0)
    CTxIn(COutPoint(0000000000000000000000000000000000000000000000000000000000000000,
4294967295), coinbase
04ffff001d01044c5a32312d362d32303233202d2047696c6769743a205468652043726f776e204a6577656c206f66204b6
    CTxOut(nValue=25.00000000, scriptPubKey=04752c2165e7041784ddcde133e1ae)
  vMerkleTree: 9a00c1a95981652cf8bbfb5ae989d941bd80eb7faaac6da8acf2d065f73258af
```

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

```
00000ffff0000000000000000000000000000000000000000000000000000000)
2023-06-21 08:08:11 block.nTime = 1687329578
2023-06-21 08:08:11 block.nNonce = 38573345
2023-06-21 08:08:11 block.GetHash =
a52b29c7cae66b6785d06fb944011d7c9f1e1d3acb2                    faa4a7
2023-06-21 08:08:11 a52b29c7cae66b6785d06fb                    3acb2f9b58a7cce9
2023-06-21 08:08:11 00000000000000000000000                    0000000000000000
2023-06-21 08:08:11 9a00c1a95981652cf8bbfb5                    7faaac6da8acf2d0
2023-06-21 08:08:11 CBlock(hash=a52b29c7cae                    011d7c9f1e1d3acb
input=0100000000000000000000000000000000000                    0000000000000000
PoW=000009c75e9e66e9a9a31bfc17f30076cea656d                    70cb32880f, ver=
hashPrevBlock=00000000000000000000000000000                    0000000000000000
hashMerkleRoot=9a00c1a95981652cf8bbfb5ae989                    c6da8acf2d065f73
nTime=1687329578, nBits=1e0ffff0, nNonce=38
2023-06-21 08:08:11
```

Undo
Redo
Cut
Copy
Paste
Delete
Select All
Insert Emoji
Change Case

Go to **line 2801** in **main.cpp**

Open | main.cpp ~/Desktop/coursecoin/src | Save

main.cpp | debug.log

2801

```
        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;
        }
```

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;
        }
```

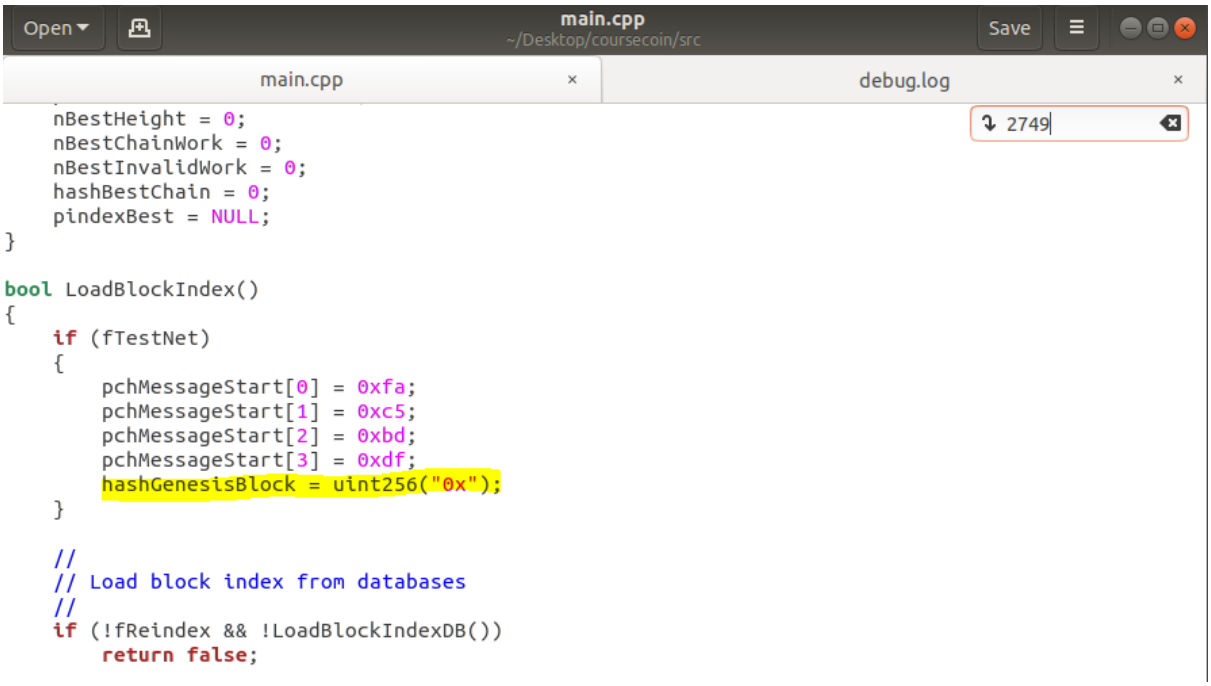
Save the **main.cpp** file

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

```
00000ffff0000000000000000000000000000000000000000000000000000000)
2023-06-21 08:08:11 block.nTime = 1687329578
2023-06-21 08:08:11 block.nNonce = 385733459
2023-06-21 08:08:11 block.GetHash =
a52b29c7cae66b6785d06fb944011d7c9f1e1d3acb2f9b58a7cce91505faa4a7
2023-06-21 08:08:11 a52b29c7cae66b6785d06fb944011d7c9f1e1d3acb2f
2023-06-21 08:08:11 0000000000000000000000000000000000000000000000000
2023-06-21 08:08:11 9a00c1a95981652cf8bbfb5ae989d941bd80eb7faaac
2023-06-21 08:08:11 CBlock(hash=a52b29c7cae66b6785d06fb944011d7c
input=0100000000000000000000000000000000000000000000000000000000000000
PoW=000009c75e9e66e9a9a31bfc17f30076cea656d54bf30e785a3b2370cb32
hashPrevBlock=0000000000000000000000000000000000000000000000000000000000
hashMerkleRoot=9a00c1a95981652cf8bbfb5ae989d941bd80eb7faaac6da8a
nTime=1687329578, nBits=1e0ffff0, nNonce=385733459, vtx=1)
2023-06-21 08:08:11
CTransaction(hash=9a00c1a95981652cf8bbfb5ae989d941bd80eb7faaac6d
vin.size=1, vout.size=1, nLockTime=0)
```

Undo
Redo
Cut
Copy
Paste
Delete
Select All
Insert Emoji
Change Case

Go to **line 2749** in main.cpp



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

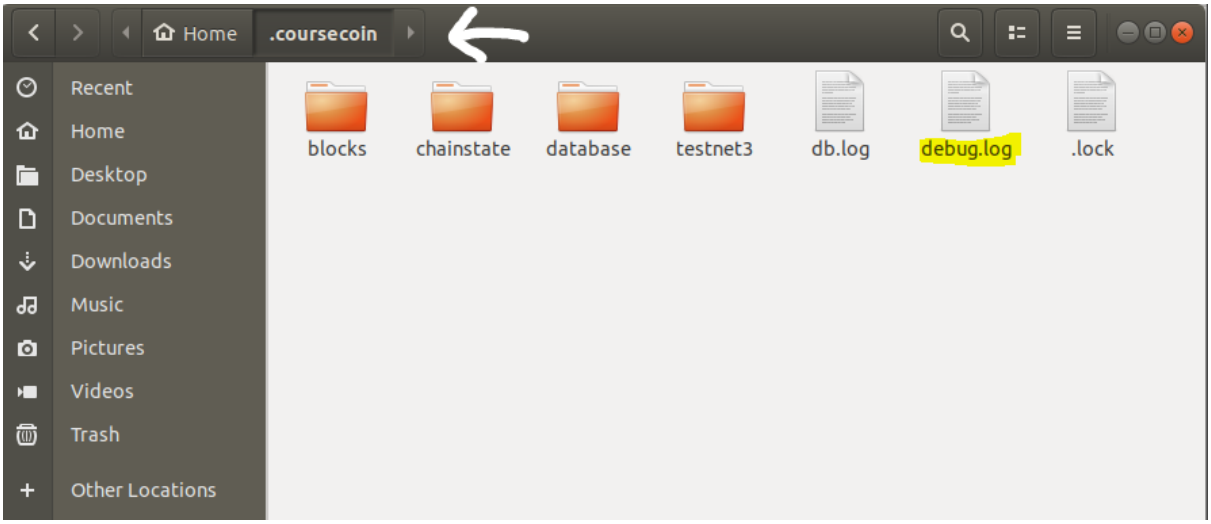
Now open your terminal and navigate to the **src directory** inside your **coin directory**. And compile the code again by using the command **“make -f makefile.unix”**

```
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-paramet
er -g -DBOOST_SPIRIT_THREADSAFE -D_FILE_OFFSET_BITS=64 -I/home/ubuntu/Desktop/co
ursecoin/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/co
ursecoin/src/leveldb/helpers -DHAVE_BUILD_INFO -fno-stack-protector -fstack-prot
ector-all -Wstack-protector -U_FORTIFY_SOURCE -D_FORTIFY_SOURCE=2  -MMD -MF obj/
main.d -o obj/main.o main.cpp
```

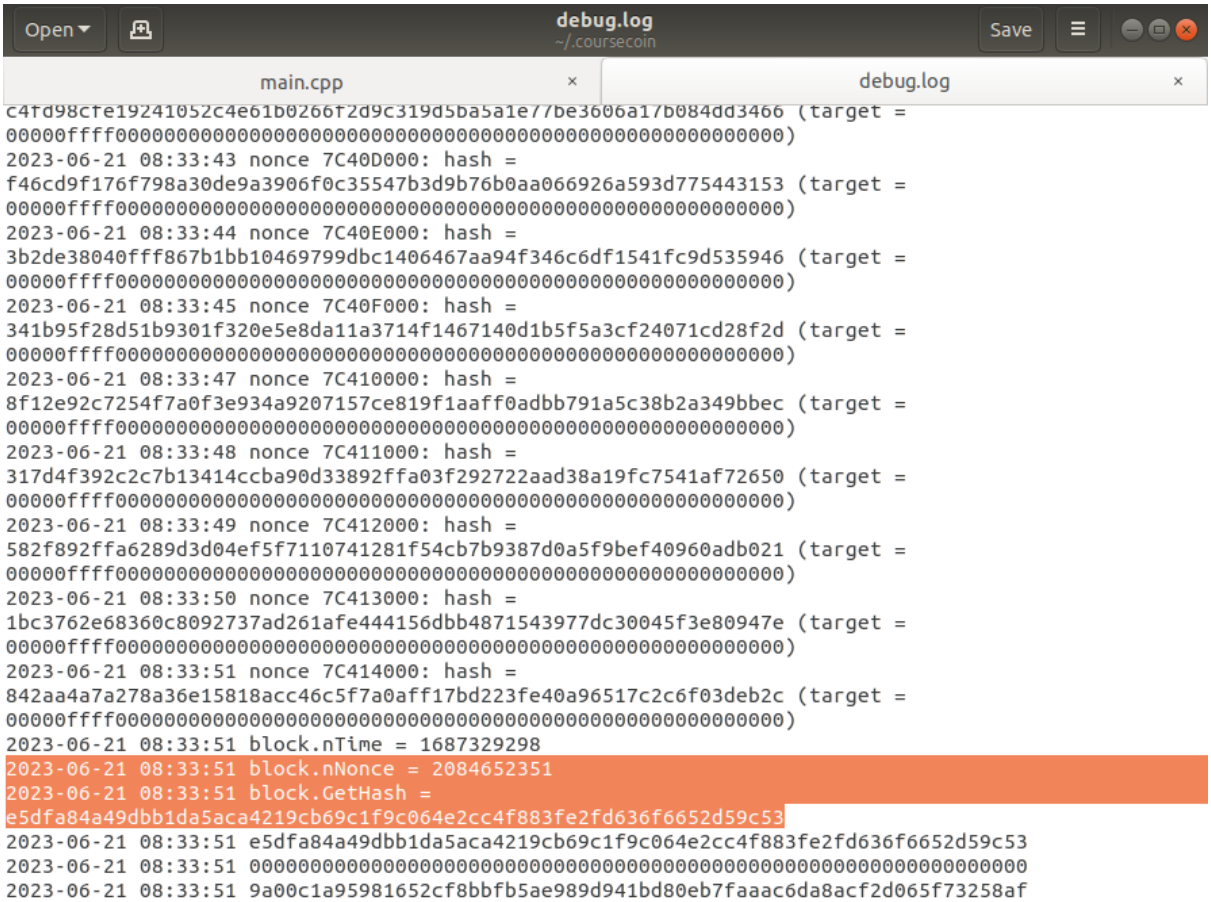
Once compiled type the command **./yourcoinname with d at the end, as in this case it is ./coursecoind**

```
ubuntu@ubuntuVM:~/Desktop/coursecoin/src$ ./coursecoind
coursecoind: main.cpp:2853: bool InitBlockIndex(): Assertion `hash == hashGenesi
sBlock' 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

```
00000ffff0000000000000000000000000000000000000000000000000000000)
2023-06-21 08:33:51 nonce 7C414000: hash =
842aa4a7a278a36e15818acc46c5f7a0aff17bd223fe40a96517c2c6f03deb2c (target =
00000ffff0000000000000000000000000000000000000000000000000000000)
2023-06-21 08:33:51 block.nTime = 1687329298
2023-06-21 08:33:51 block.nNonce = 2084652351
2023-06-21 08:33:51 block.GetHash =
```

Go to line 2796

Paste the main net nNonce value that you copied earlier

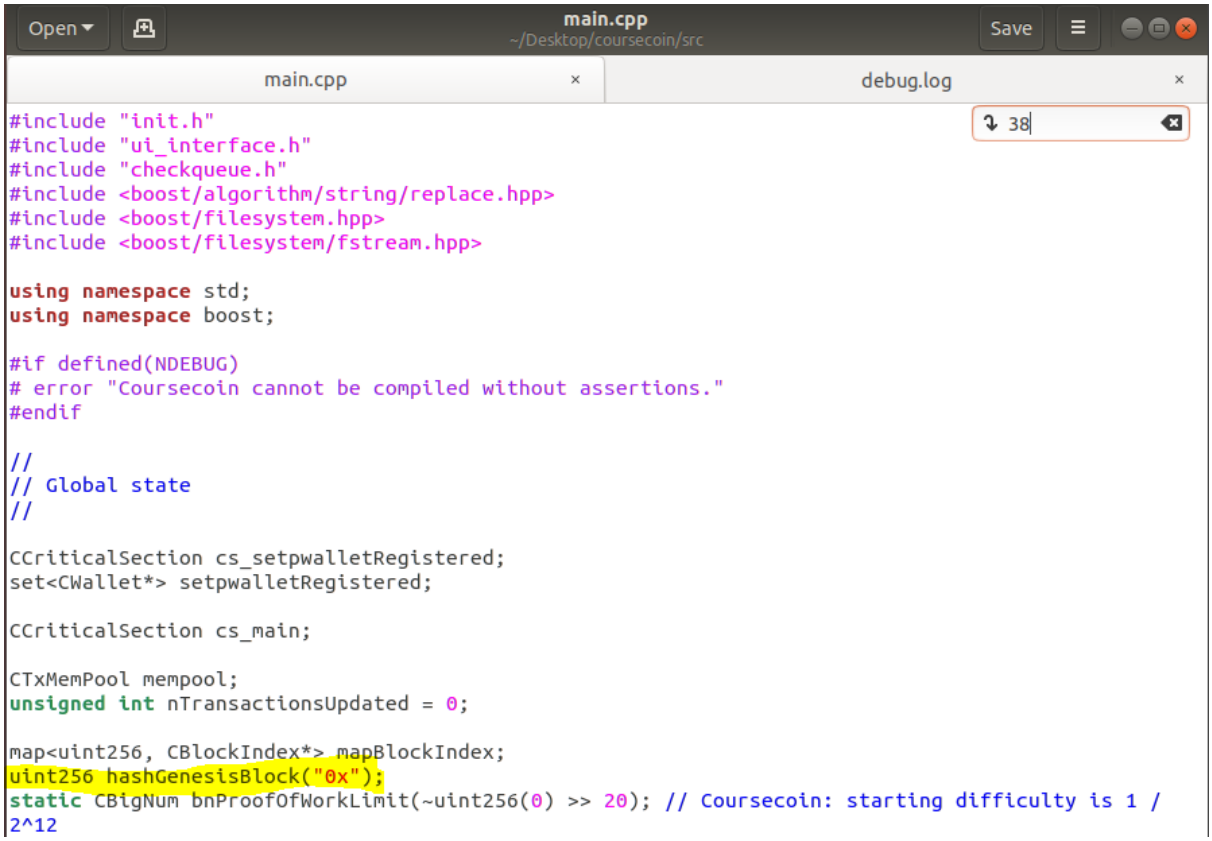
```
ParseHex("04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80ba0ddd0
<< OP_CHECKSIG;
        CBlock block;
        block.vtx.push_back(txNew);
        block.hashPrevBlock = 0;
        block.hashMerkleRoot = block.BuildMerkleTree();
        block.nVersion = 1;
        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.

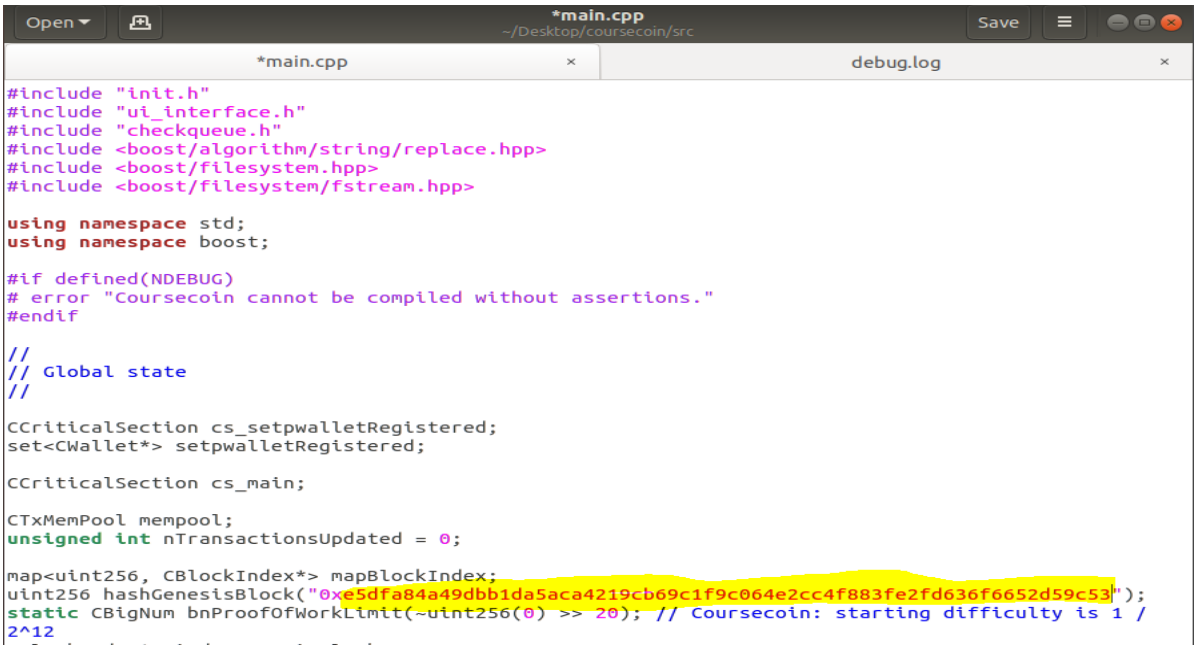
```
00000ffff0000000000000000000000000000000000000000000000000000000)
2023-06-21 08:33:51 block.nTime = 1687329298
2023-06-21 08:33:51 block.nNonce = 2084652351
2023-06-21 08:33:51 block.GetHash =
e5dfa84a49dbb1da5aca4219cb69c1f9c064e2cc4f883fe2fd636f6652d59c53
2023-06-21 08:33:51 e5dfa84a49dbb1da5aca4219cb69c1f9c064e2cc4f88                d59c53
2023-06-21 08:33:51 0000000000000000000000000000000000000000000000            000000
2023-06-21 08:33:51 9a00c1a95981652cf8bbfb5ae989d941bd80eb7faaac                3258af
2023-06-21 08:33:51 CBlock(hash=e5dfa84a49dbb1da5aca4219cb69c1f9                e2fd636f6652d59c53,
input=0100000000000000000000000000000000000000000000000000000000000000      af5832f765d0f2aca86da
PoW=00000327cc57eb22d97e81853fbceb3b63a30d2f002a64e00ef731c8e83f
hashPrevBlock=0000000000000000000000000000000000000000000000000000000000000
hashMerkleRoot=9a00c1a95981652cf8bbfb5ae989d941bd80eb7faaac6da8a                f,
nTime=1687329298, nBits=1e0ffff0, nNonce=2084652351, vtx=1)
2023-06-21 08:33:51
CTransaction(hash=9a00c1a95981652cf8bbfb5ae989d941bd80eb7faaac6d                58af, ver=1,
vin.size=1, vout.size=1, nLockTime=0)
```

Undo
Redo
Cut
Copy
Paste
Delete
Select All
Insert Emoji
Change Case

Go to **line 38** in **main.cpp**

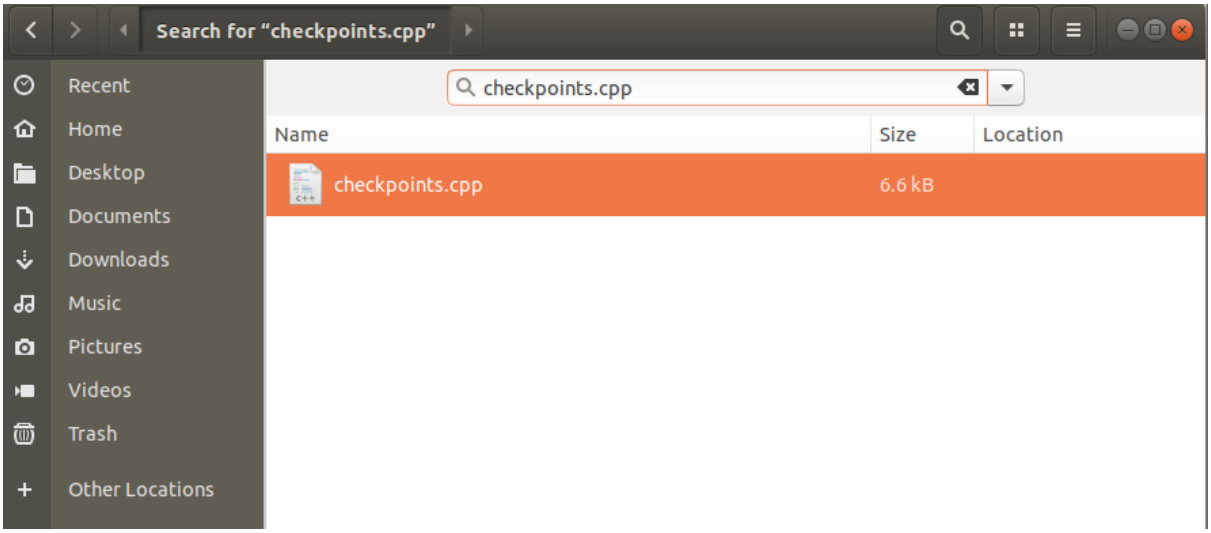


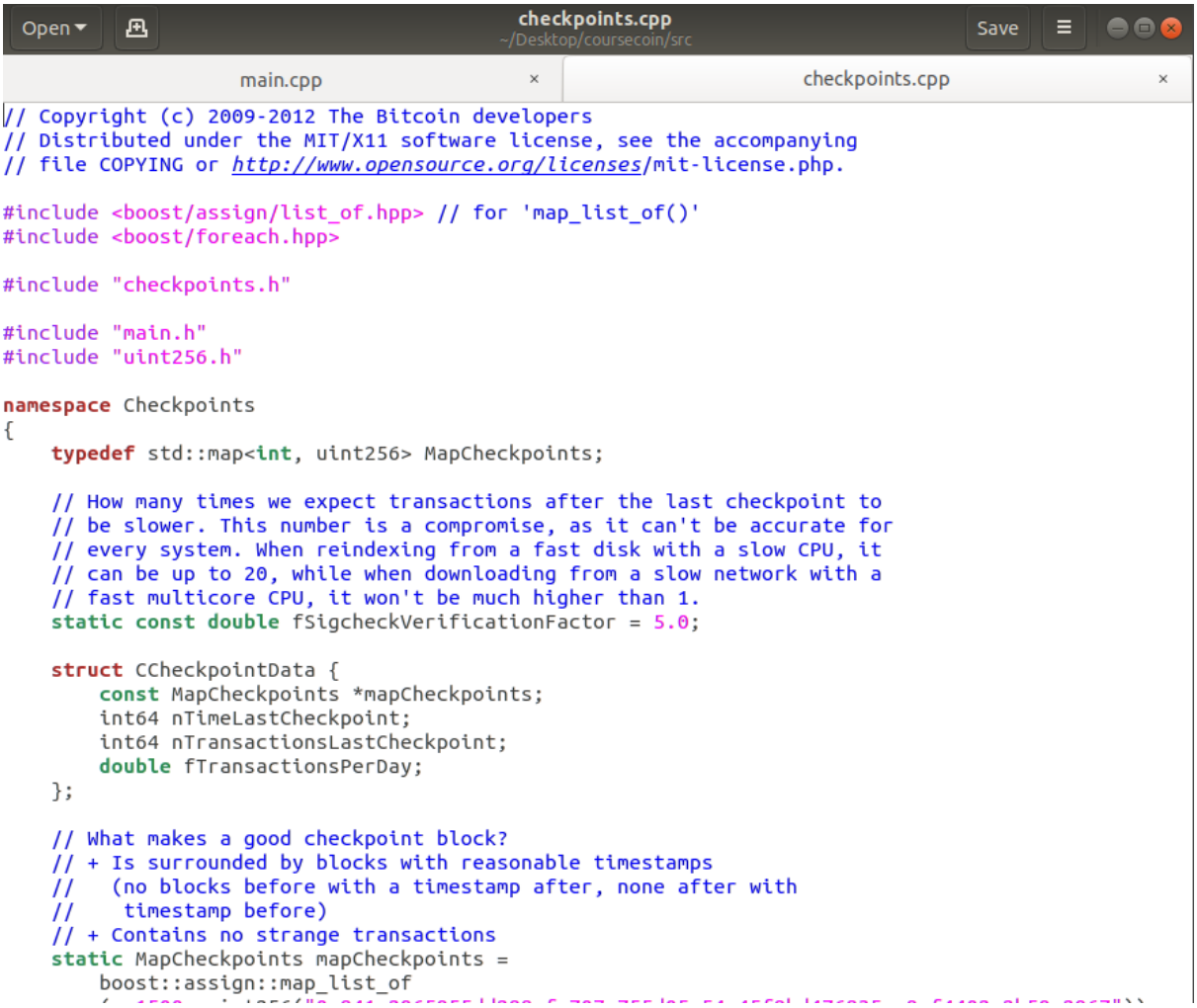
Paste the genesis block hash for main net that you copied after **0x**



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.





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("0x841a2965955dd288cfa707a755d05a54e45f8bd476835ec9af4402a2b59a2967"))
    (  4032, uint256("0x9ce90e427198fc0ef05e5905ce3503725b80e26afd35a987965fd7e3d9cf0846"))
    (  8064, uint256("0xeb984353fc5190f210651f150c40b8a4bab9eeeff0b729fcb3987da694430d70"))
    ( 16128, uint256("0x602edf1859b7f9a6af809f1d9b0e6cb66fdc1d4d9dcd7a4bec03e12a1ccd153d"))
    ( 23420, uint256("0xd80fdf9ca81afd0bd2b2a90ac3a9fe547da58f2530ec874e978fce0b5101b507"))
    ( 50000, uint256("0x69dc37eb029b68f075a5012dcc0419c127672adb4f3a32882b2b3e71d07a20a6"))
    ( 80000, uint256("0x4fcb7c02f676a300503f49c764a89955a8f920b46a8cbecb4867182ecdb2e90a"))
    (120000, uint256("0xbd9d26924f05f6daa7f0155f32828ec89e8e29cee9e7121b026a7a3552ac6131"))
    (161500, uint256("0xdbe89880474f4bb4f75c227c77ba1cdc024991123b28b8418dbbf7798471ff43"))
    (179620, uint256("0x2ad9c65c990ac00426d18e446e0fd7be2ffa69e9a7dcb28358a50b2b78b9f709"))
    (240000, uint256("0x7140d1c4b4c2157ca217ee7636f24c9c73db39c4590c4e6eab2e3ea1555088aa"))
    (383640, uint256("0x2b6809f094a9215bafc65eb3f110a35127a34be94b7d0590a096c3f126c6f364"))
    (409004, uint256("0x487518d663d9f1fa08611d9395ad74d982b667fbdc0e77e9cf39b4f1355908a3"))
    (456000, uint256("0xbf34f71cc6366cd487930d06be22f897e34ca6a40501ac7d401be32456372004"))
    (541794, uint256("0x1cbccbe6920e7c258bbce1f26211084efb19764aa3224bec3f4320d77d6a2fd2"))
    (585010, uint256("0xea9ea06840de20a18a66acb07c9102ee6374ad2cbafc71794e576354fea5df2d"))
    (638902, uint256("0x15238656e8ec63d28de29a8c75fcf3a5819afc953dcd9cc45cecc53baec74f38"))
    ;
```

Erase all the check points except the first one

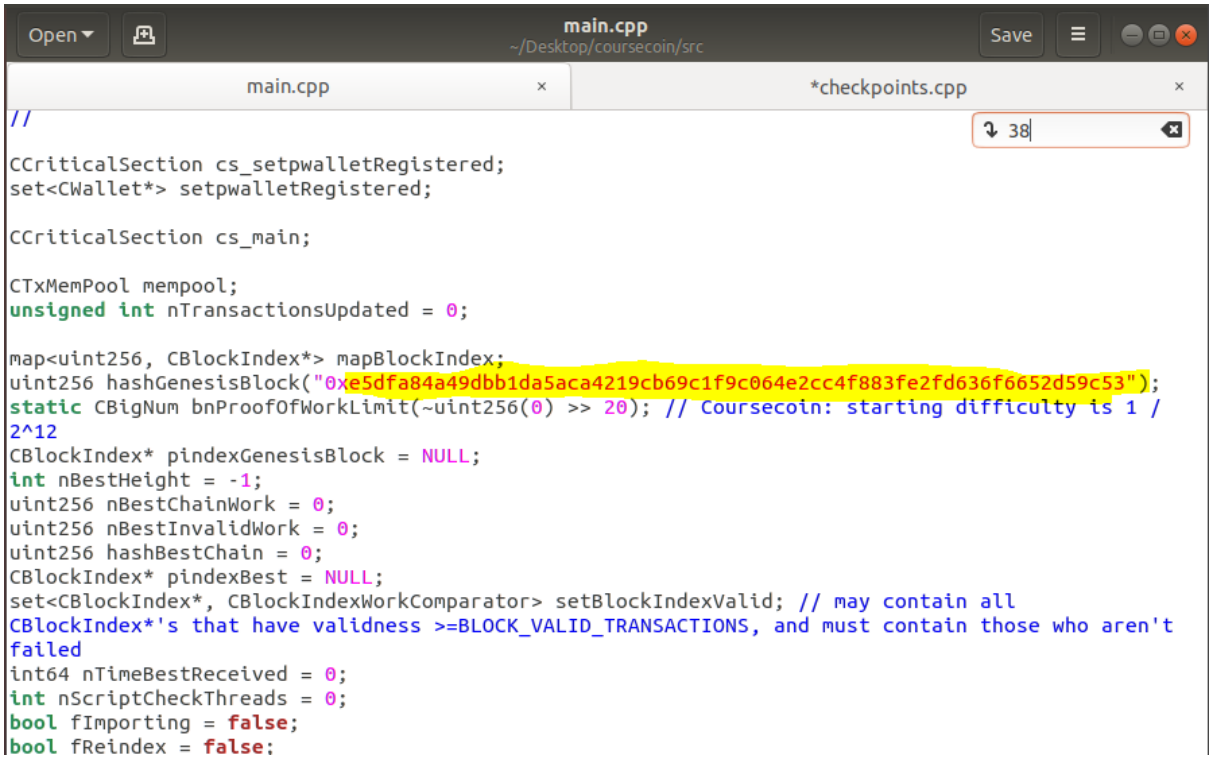
```
    static MapCheckpoints mapCheckpoints =
        boost::assign::map_list_of
        (   1500, uint256("0x841a2965955dd288cfa707a755d05a54e45f8bd476835ec9af4402a2b59a2967"))
        (   4032, uint256("0x9ce90e427198fc0ef05e5905ce3503725b80e26afd35a987965fd7e3d9cf0846"))
        (   8064, uint256("0xeb984353fc5190f210651f150c40b8a4bab9eeeff0b729fcb3987da694430d70"))
        (  16128, uint256("0x602edf1859b7f9a6af809f1d9b0e6cb66fdc1d4d9dcd7a4bec03e12a1ccd153d"))
        (  23420, uint256("0xd80fdf9ca81afd0bd2b2a90ac3a9fe547da58f2530ec874e978fce0b5101b507"))
        (  50000, uint256("0x69dc37eb029b68f075a5012dcc0419c127672adb4f3a32882b2b3e71d07a20a6"))
        (  80000, uint256("0x4fcb7c02f676a300503f49c764a89955a8f920b46a8cbecb4867182ecdb2e90a"))
        ( 120000, uint256("0xbd9d26924f05f6daa7f0155f32828ec89e8e29cee9e7121b026a7a3552ac6131"))
        ( 161500, uint256("0xdbe89880474f4bb4f75c227c77ba1cdc024991123b28b8418dbbf7798471ff43"))
        ( 179620, uint256("0x2ad9c65c990ac00426d18e446e0fd7be2ffa69e9a7dcb28358a50b2b78b9f709"))
        ( 240000, uint256("0x7140d1c4b4c2157ca217ee7636f24c9c73db39c4590c4e6eab2e3ea1555088aa"))
        ( 383640, uint256("0x2b6809f094a9215bafc65eb3f110a35127a34be94b7d0590a096c3f126c6f364"))
        ( 409004, uint256("0x487518d663d9f1fa08611d9395ad74d982b667fbdc0e77e9cf39b4f1355908a3"))
        ( 456000, uint256("0xbf34f71cc6366cd487930d06be22f897e34ca6a40501ac7d401be32456372004"))
        ( 541794, uint256("0x1cbccbe6920e7c258bbce1f26211084efb19764aa3224bec3f4320d77d6a2fd2"))
        ( 585010, uint256("0xea9ea06840de20a18a66acb07c9102ee6374ad2cbafc71794e576354fea5df2d"))
        ( 638902, uint256("0x15238656e8ec63d28de29a8c75fcf3a5819afc953dcd9cc45cecc53baec74f38"))
        ;
```

```
static MapCheckpoints mapCheckpoints =
    boost::assign::map_list_of
    (   1500, uint256("0x841a2965955dd288cfa707a755d05a54e45f8bd476835ec9af4402a2b59a2967"))
    ;
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



Copy this part

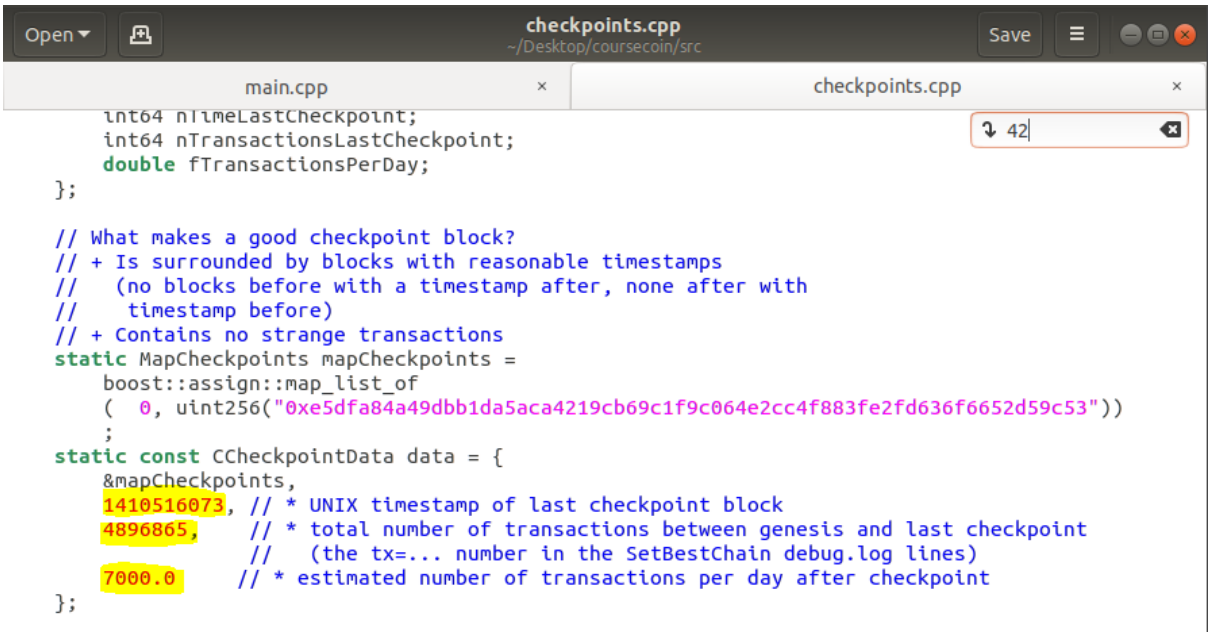
```
CCriticalSection cs_main;

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("0xa0fea99a6897f531600c8ae53367b126824fd6a847b2b2b73817a95b8e27e602"))
        ;
    static const CCheckpointData dataTestnet = {
```

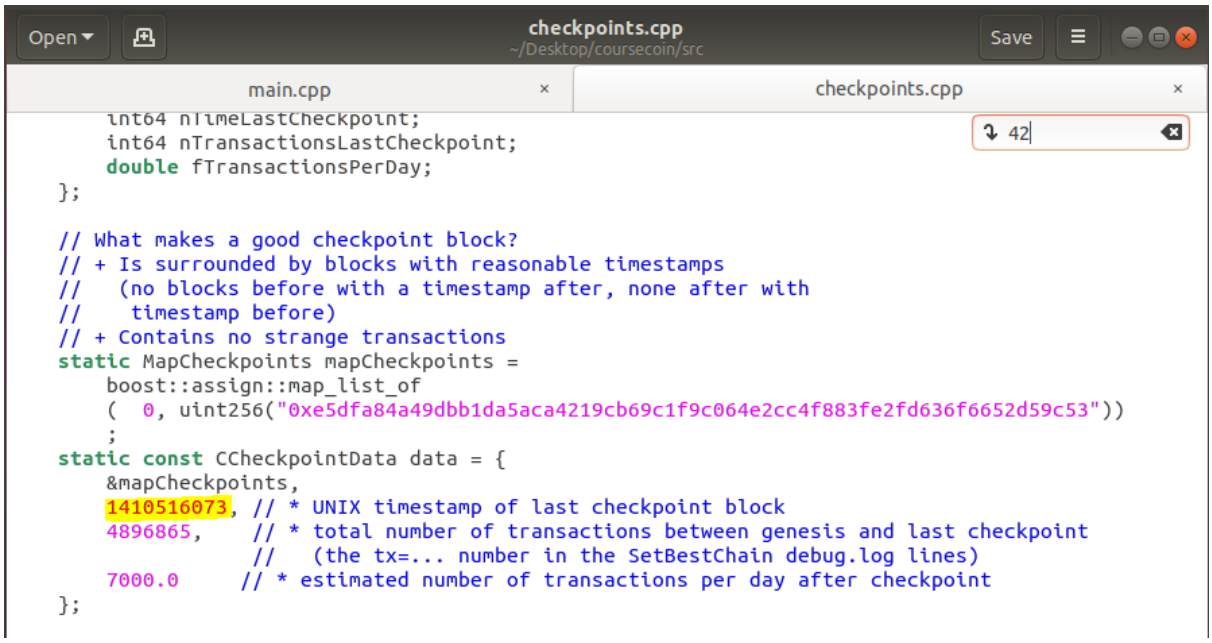
Save the checkpoints.cpp file

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



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

Go to **line 42** in **checkpoints.cpp**



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

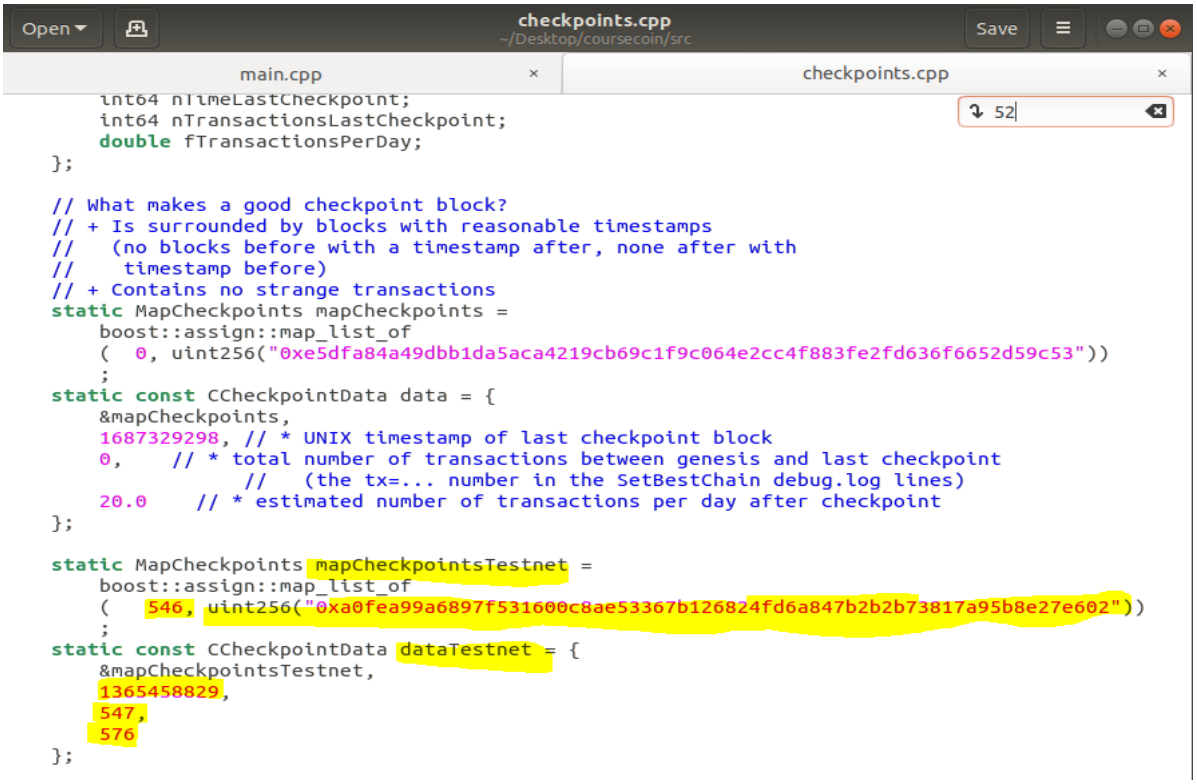
```
// What makes a good checkpoint block?
// + Is surrounded by blocks with reasonable timestamps
//   (no blocks before with a timestamp after, none after with
//    timestamp before)
// + Contains no strange transactions
static MapCheckpoints mapCheckpoints =
    boost::assign::map_list_of
    ( 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

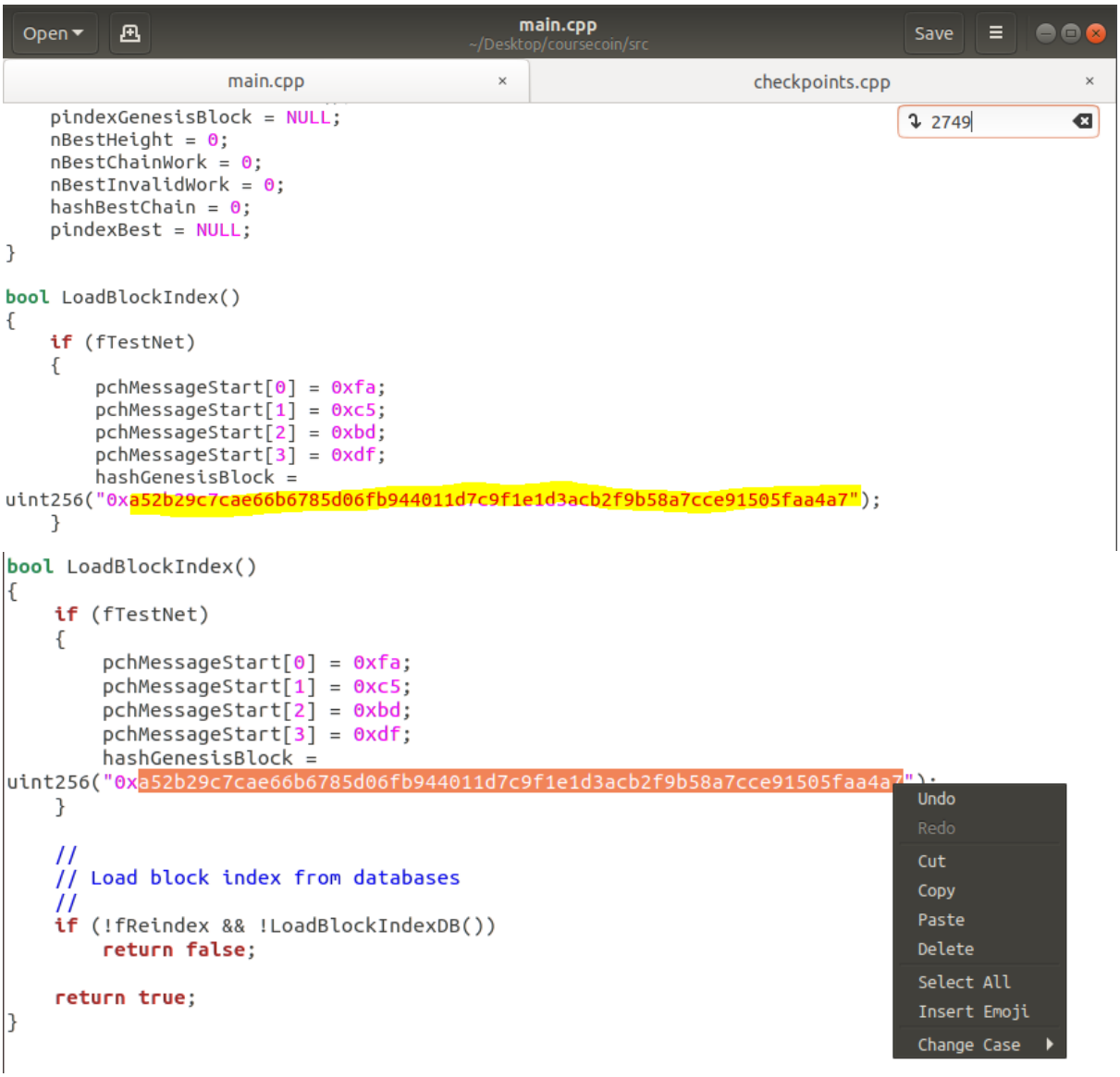
Go to line 52 in **checkpoints.cpp**, you'll see the respective values for test net



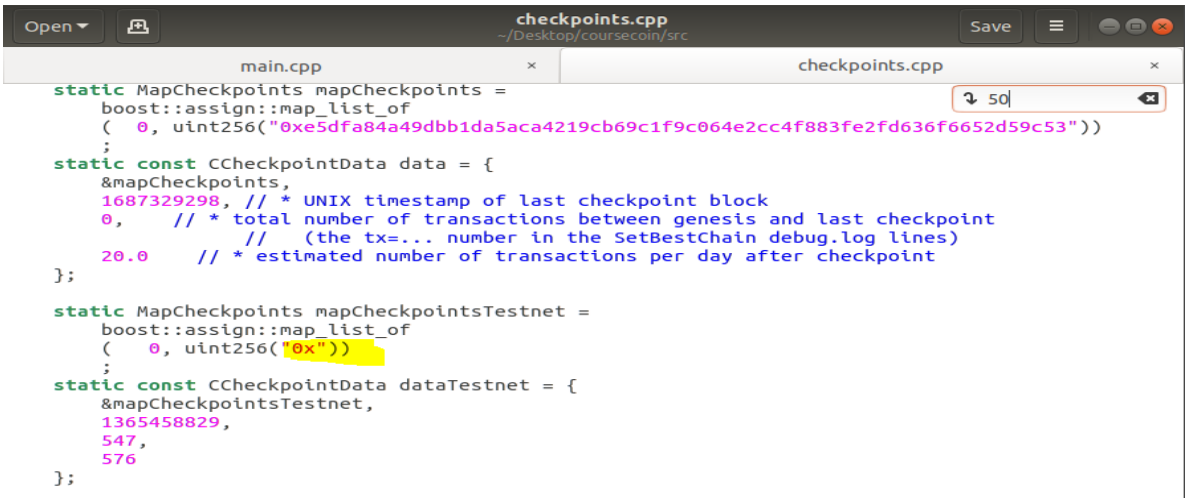
Change the block height value from 546 to 0, and erase the test net genesis block hash up to **0x**.

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

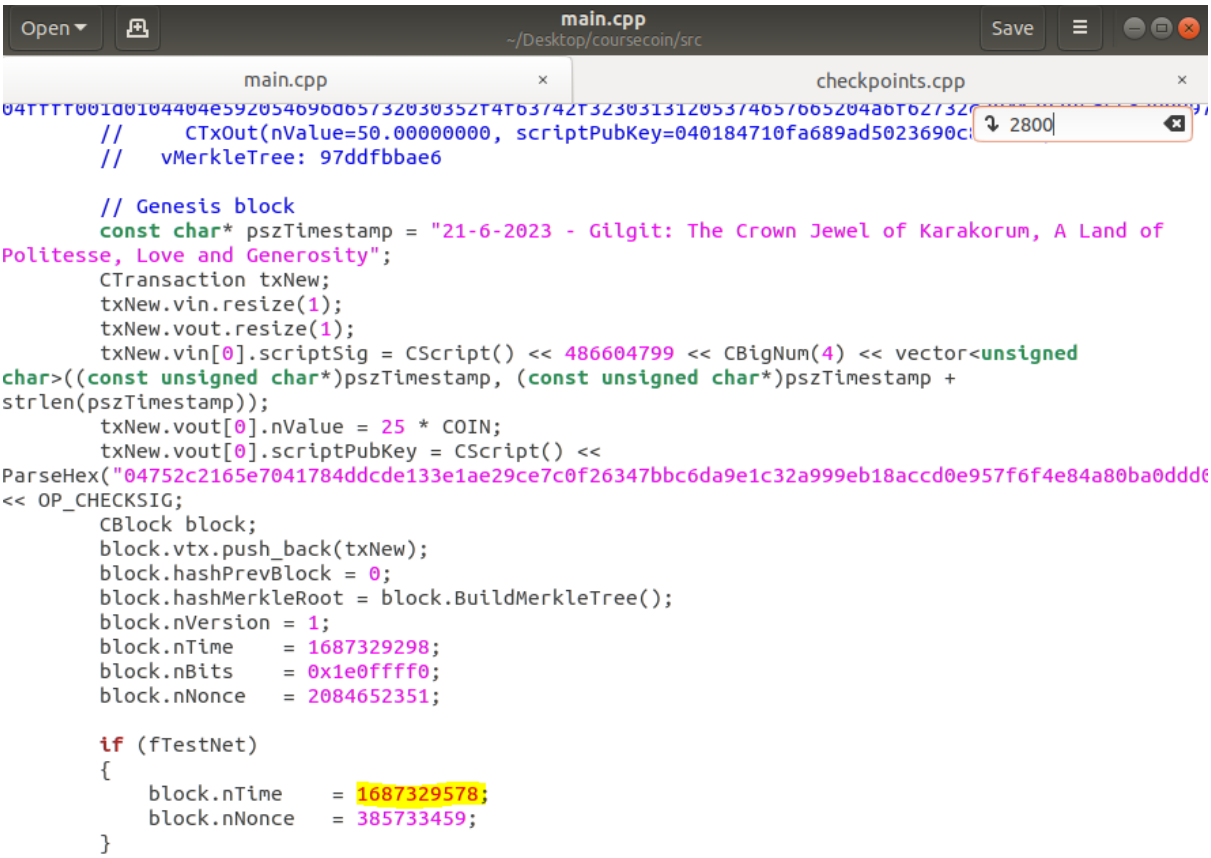
Go to **line 2749** in **main.cpp**, and copy the testnet genesis block hash that you created earlier.



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



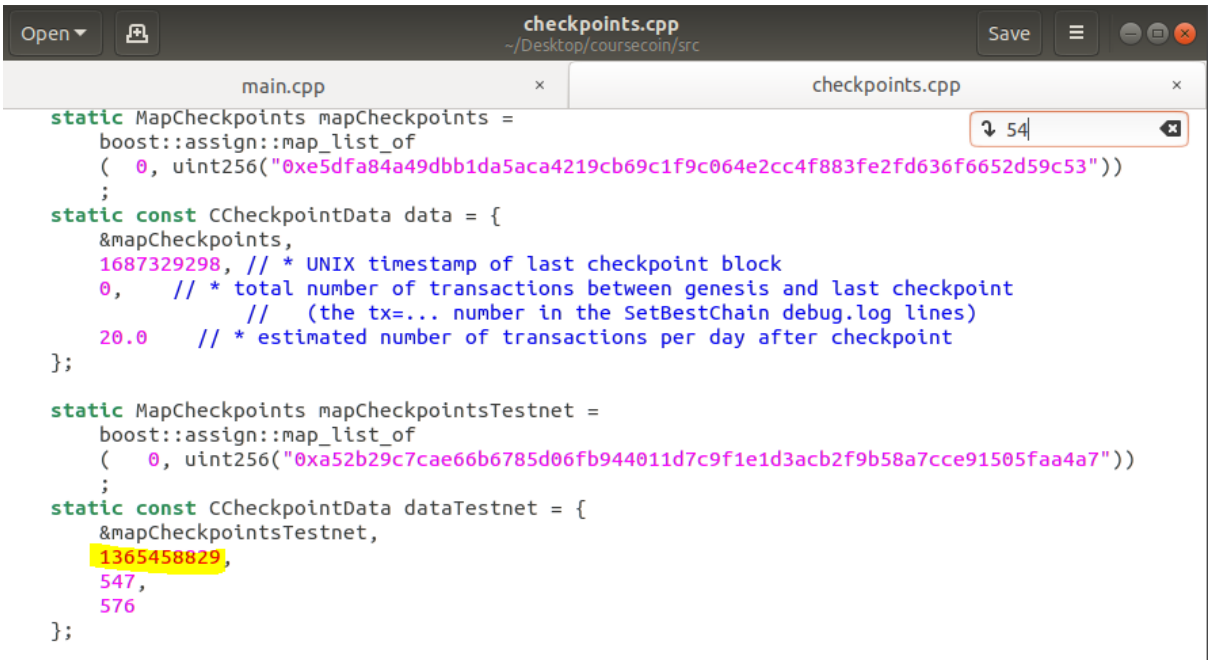
Now go to line 2800, and copy the nTime value field, within the if(fTestNet) block.



```
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    = 168732957
            block.nNonce   = 385733459
        }

if (true && block.GetHash() != hashGen
        {
            printf("Searching for gene                    );
            // This will figure out a                     Nonce if you're
            // creating a different ge
            uint256 hashTarget = CBigN                    (block.nBits).getuint256();
            uint256 thash;
            char scratchpad[SCRYPT_SCR
```

Undo
Redo
Cut
Copy
Paste
Delete
Select All
Insert Emoji
Change Case

Go to **line 54** in **checkpoints.cpp**



And paste the nTime value that you copied.

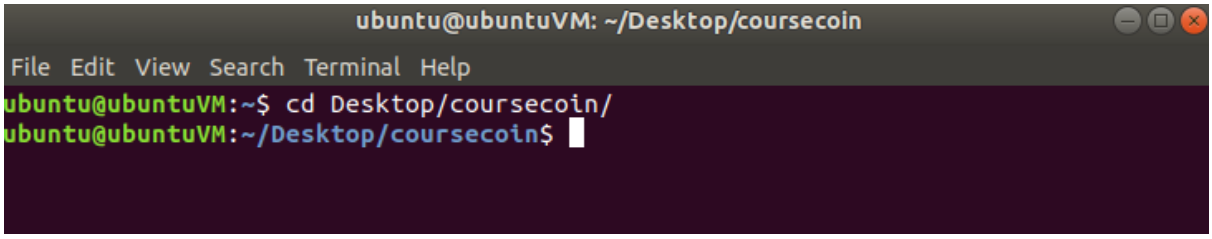
```
static MapCheckpoints mapCheckpointsTestnet =
    boost::assign::map_list_of
    (   0, uint256("0xa52b29c7cae66b6785d06fb944011d7c9f1e1d3acb2f9b58a7cce91505faa4a7"))
    ;
static const CCheckpointData dataTestnet = {
    &mapCheckpointsTestnet,
    1687329578,
    547,
    576
};
```

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

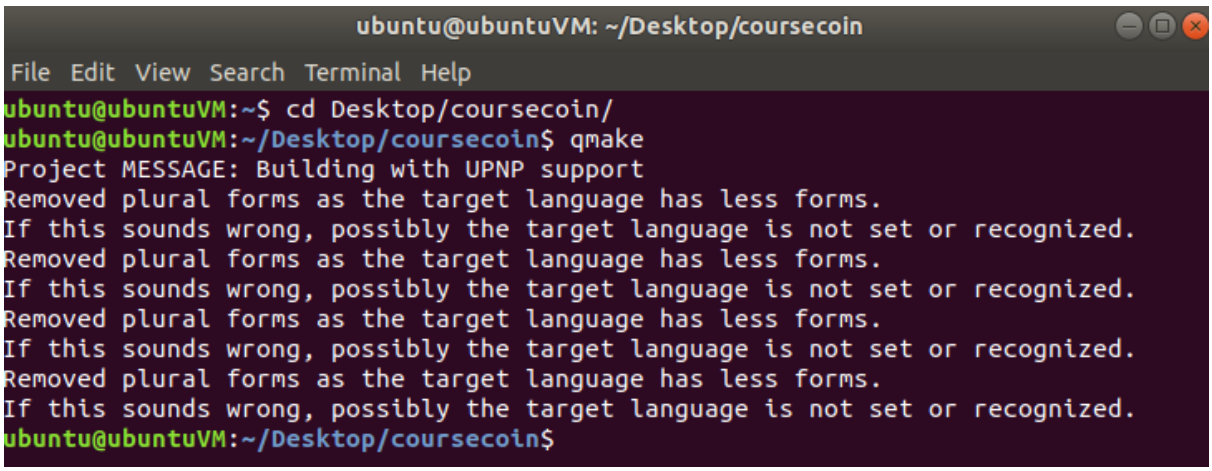
```
static MapCheckpoints mapCheckpointsTestnet =
    boost::assign::map_list_of
    (   0, uint256("0xa52b29c7cae66b6785d06fb944011d7c9f1e1d3acb2f9b58a7cce91505faa4a7"))
    ;
static const CCheckpointData dataTestnet = {
    &mapCheckpointsTestnet,
    1687329578,
    0,
    20.0
};
```

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)



Type the command **"qmake"**



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-pro
tector-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-builtin-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-builtin-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-builtin-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_read
er.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.

```
uild/moc_optionsdialog.o build/moc_sendcoinsdialog.o build/moc_coincontroldialog.o build/moc_coin
controltreewidget.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_guiutil.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_qvalidatedlineedit.o build/moc_bitcoinunits.o build/moc_qvalueco
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_s
ystem -lboost_filesystem -lboost_program_options -lboost_thread -lpthread -lQtGui -lQtNetwork -lQ
tCore
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.