Lab 20-6-2023

[Note: This hands on exercise is a part of ongoing series on making your own cryptocurrency wallet, thus before doing it, please make sure that you have performed Lab 16-6-2023 and 19-6-2023]

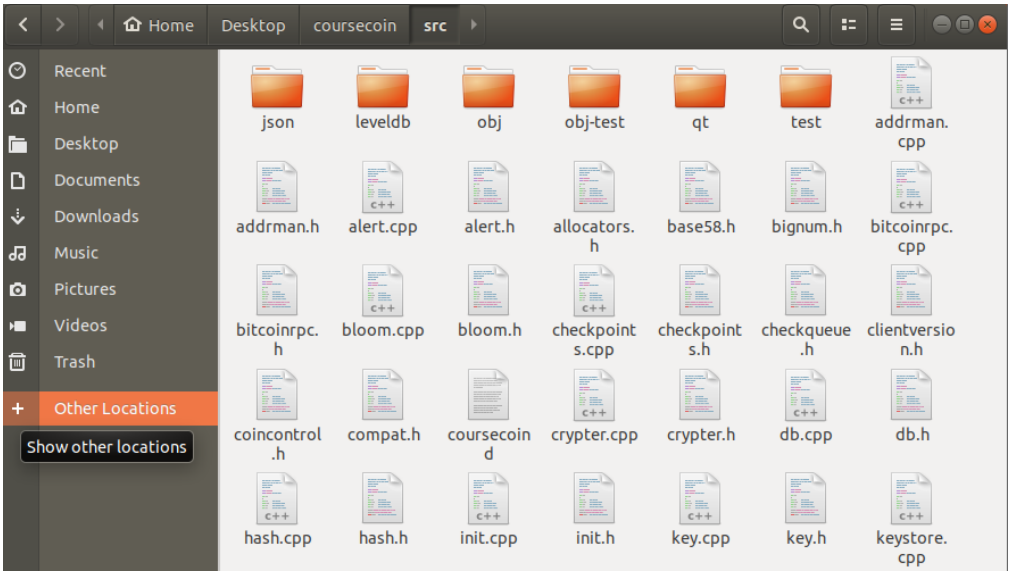
1. Open your terminal, and navigate to the directory of your coin

```
ubuntu@ubuntuVM:~$ cd Desktop/coursecoin/
ubuntu@ubuntuVM:~/Desktop/coursecoin$
```

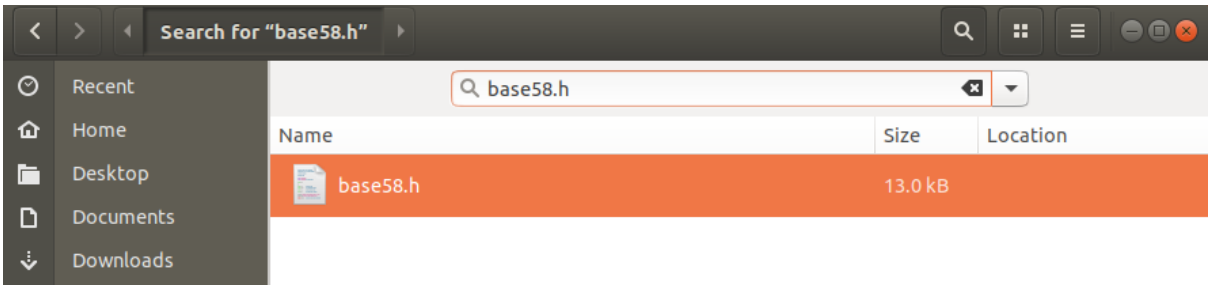
2. Type the following commands in your terminal to change the Litecoin ports to your own personal ports.
   i. **find . -type f -print0 | xargs -0 sed -i 's/9333/2333/g'**
   ii. **find . -type f -print0 | xargs -0 sed -i 's/9332/2332/g'**

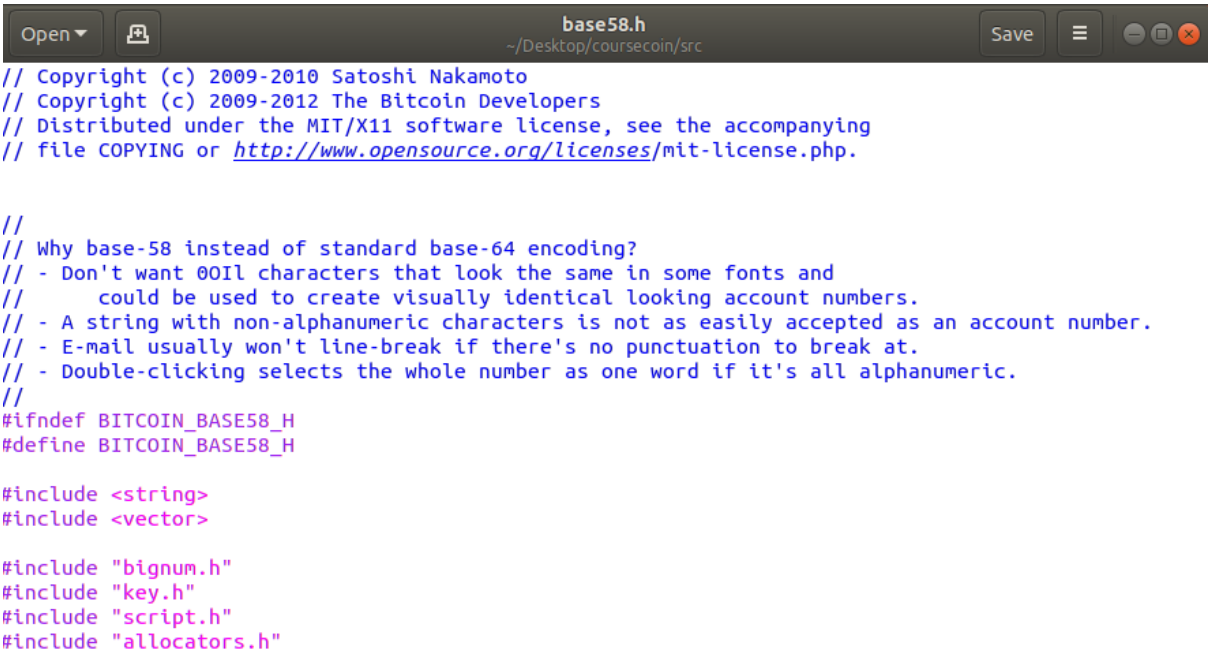
```
ubuntu@ubuntuVM:~/Desktop/coursecoin$ find . -type f -print0 | xargs -0 sed -i '
s/9333/2333/g'

ubuntu@ubuntuVM:~/Desktop/coursecoin$ find . -type f -print0 | xargs -0 sed -i '
s/9332/2332/g'
```

3. Go into your coin directory and into src directory

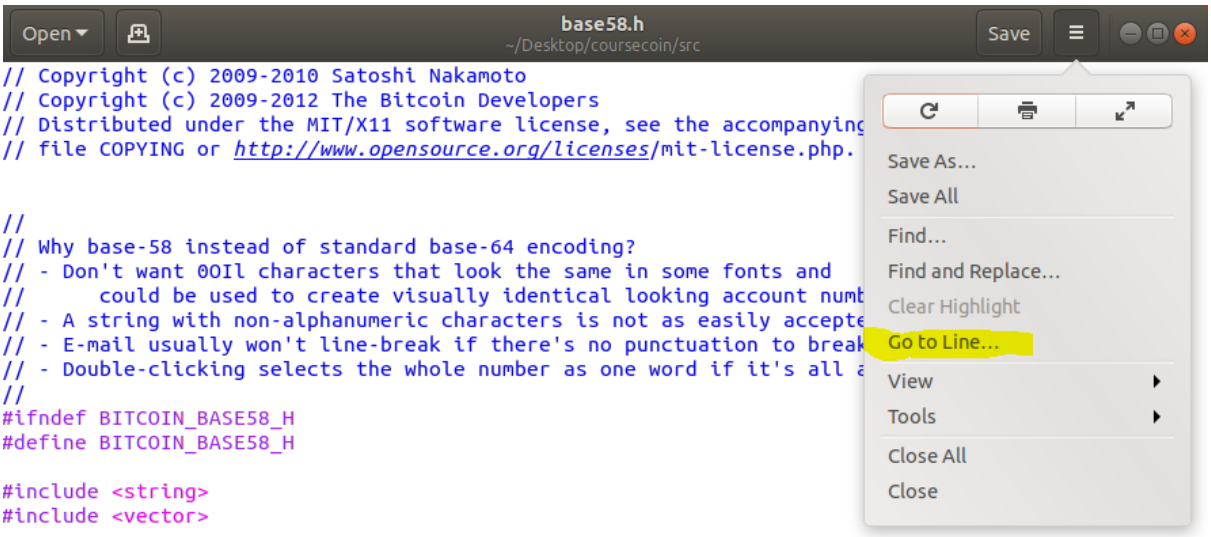


4. Search for “base58.h” in the src directory, open the file by double clicking it.

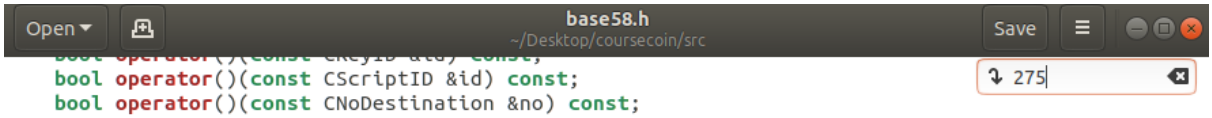




Go to line 275



You'll see this snippet of code




```
    bool operator()(const CScriptID &id) const;
    bool operator()(const CNoDestination &no) const;
};

class CBitcoinAddress : public CBase58Data
{
public:
    enum
    {
        PUBKEY_ADDRESS = 48, // Coursecoin addresses start with L
        SCRIPT_ADDRESS = 5,
        PUBKEY_ADDRESS_TEST = 111,
        SCRIPT_ADDRESS_TEST = 196,
    };

    bool Set(const CKeyID &id) {
        SetData(fTestNet ? PUBKEY_ADDRESS_TEST : PUBKEY_ADDRESS, &id, 20);
        return true;
    }

    bool Set(const CScriptID &id) {
        SetData(fTestNet ? SCRIPT_ADDRESS_TEST : SCRIPT_ADDRESS, &id, 20);
        return true;
    }
```

We are going to change the main net and test net public key address prefixes, by changing the following highlighted numbers.

```
class CBitcoinAddress : public CBase58Data
{
public:
    enum
    {
        PUBKEY_ADDRESS = 48, // Coursecoin addresses start with C
        SCRIPT_ADDRESS = 5,
        PUBKEY_ADDRESS_TEST = 111,
        SCRIPT_ADDRESS_TEST = 196,
    };
```

For that go to the link mentioned below
https://en.bitcoin.it/wiki/List_of_address_prefixes

List of address prefixes

Blockchain-based currencies use encoded strings, which are in a Base58Check encoding with the exception of Bech32 encodings. The encoding includes a prefix (traditionally a single *version byte*), which affects the leading symbol(s) in the encoded result. The following is a list of some prefixes which are in use in the reference Bitcoin codebase.[1][2][3]

Decimal prefix	Hex	Example use	Leading symbol(s)	Example
0	00	Pubkey hash (P2PKH address)	1	17VZNX1SN5NtKa8UQFxwQbFeFc3iqRYhem

In the case of **“coursecoin”** the address for main net would start with a capital C

27	B or C	34
28	C	34
29	C or D	34

and for test net it would start with small letter c.

85	b	34
86	b or c	34
87-88	c	34
89	c or d	34

Therefore, in case of **“coursecoin”** I changed the value of PUBKEY_ADDRESS to 28 and PUBKEY_ADDRESS_TEST to 87.

```
class CBitcoinAddress : public CBase58Data
{
public:
    enum
    {
        PUBKEY_ADDRESS = 28, // Coursecoin addresses start with C
        SCRIPT_ADDRESS = 5,
        PUBKEY_ADDRESS_TEST = 87,
        SCRIPT_ADDRESS_TEST = 196,
    };
```

save the base58.h file and close it.

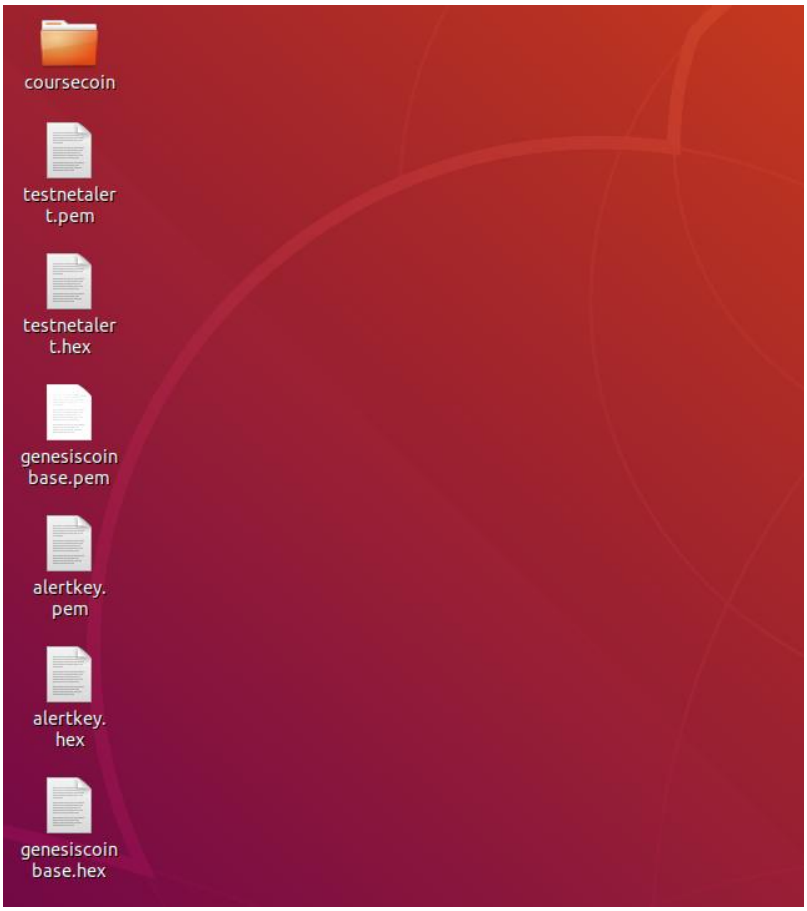
5. Open your terminal, if you are in your coin directory please navigate to your Desktop

```
ubuntu@ubuntuVM:~/Desktop/coursecoin$ cd ..
ubuntu@ubuntuVM:~/Desktop$
```

6. Type the following commands in your terminal
    i. openssl ecparam -genkey -name secp256k1 -out alertkey.pem
    ii. openssl ec -in alertkey.pem -text > alertkey.hex
    iii. openssl ecparam -genkey -name secp256k1 -out testnetalert.pem
    iv. openssl ec -in testnetalert.pem -text > testnetalert.hex
    v. openssl ecparam -genkey -name secp256k1 -out genesiscoinbase.pem
    vi. openssl ec -in testnetalert.pem -text > genesiscoinbase.hex

```
ubuntu@ubuntuVM:~/Desktop$ openssl ecparam -genkey -name secp256k1 -out alertkey.pem
ubuntu@ubuntuVM:~/Desktop$ openssl ec -in alertkey.pem -text > alertkey.hex
read EC key
writing EC key
ubuntu@ubuntuVM:~/Desktop$ openssl ecparam -genkey -name secp256k1 -out testnetalert.pem
ubuntu@ubuntuVM:~/Desktop$ openssl ec -in testnetalert.pem -text > testnetalert.hex
read EC key
writing EC key
ubuntu@ubuntuVM:~/Desktop$ openssl ecparam -genkey -name secp256k1 -out genesiscoinbase.pem
ubuntu@ubuntuVM:~/Desktop$ openssl ec -in testnetalert.pem -text > genesiscoinbase.hex
read EC key
writing EC key
ubuntu@ubuntuVM:~/Desktop$
```

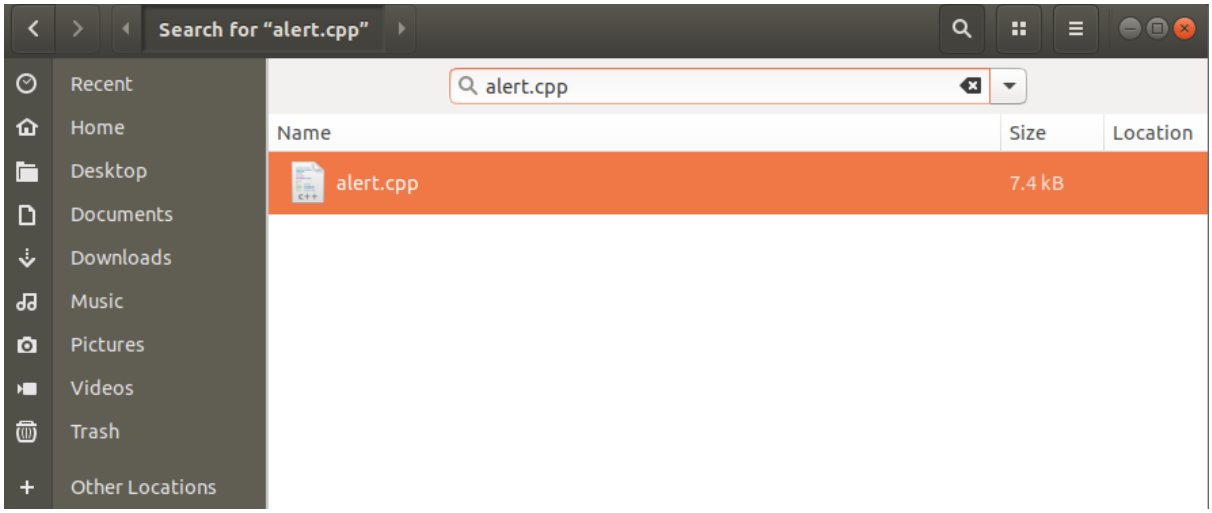
You'll see following files on your Desktop after executing these commands.



7. Stay in your terminal and show the contents of the file named alertkey.hex by using the command **“cat alertkey.hex”**. You will see that it contained a newly generated public and private key.

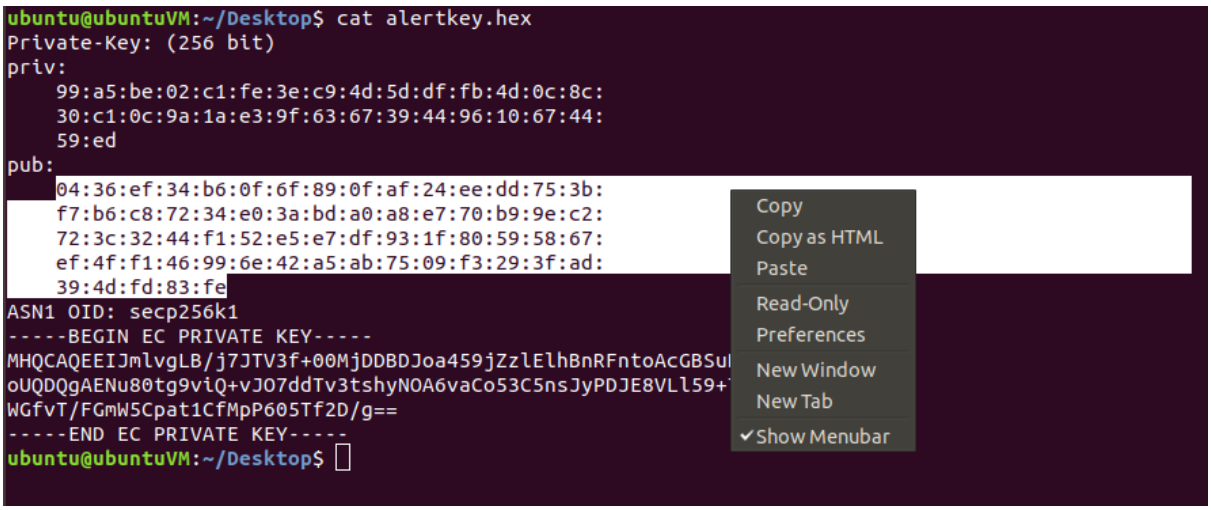
```
ubuntu@ubuntuVM:~/Desktop$ cat alertkey.hex
Private-Key: (256 bit)
priv:
    99:a5:be:02:c1:fe:3e:c9:4d:5d:df:fb:4d:0c:8c:
    30:c1:0c:9a:1a:e3:9f:63:67:39:44:96:10:67:44:
    59:ed
pub:
    04:36:ef:34:b6:0f:6f:89:0f:af:24:ee:dd:75:3b:
    f7:b6:c8:72:34:e0:3a:bd:a0:a8:e7:70:b9:9e:c2:
    72:3c:32:44:f1:52:e5:e7:df:93:1f:80:59:58:67:
    ef:4f:f1:46:99:6e:42:a5:ab:75:09:f3:29:3f:ad:
    39:4d:fd:83:fe
ASN1 OID: secp256k1
-----BEGIN EC PRIVATE KEY-----
MHQCAQEEIJmlvgLB/j7JTV3f+00MjDDBDJoa459jZzlElhBnRFntoAcGBSuBBAAK
oUQDQgAENu80tg9viQ+vJO7ddTv3tshyNOA6vaCo53C5nsJyPDJE8VLl59+TH4BZ
WGfvT/FGmW5Cpat1CfMpP605Tf2D/g==
-----END EC PRIVATE KEY-----
```

Go to the src directory within your coin directory in file explorer, and search for a file named “alert.cpp” and open it.



You'll see main net and test net public keys. Erase the main net public in the file

Copy the public key that you obtained in your terminal earlier



Paste it within the quotation marks

```
//
// Alert system
//

#include <algorithm>
#include <boost/algorithm/string/classification.hpp>
#include <boost/algorithm/string/replace.hpp>
#include <boost/foreach.hpp>
#include <map>

#include "alert.h"
#include "key.h"
#include "net.h"
#include "sync.h"
#include "ui_interface.h"

using namespace std;

map<uint256, CAlert> mapAlerts;
CCriticalSection cs_mapAlerts;

static const char* pszMainKey = "04:36:ef:34:b6:0f:6f:89:0f:af:24:ee:dd:75:3b:
    f7:b6:c8:72:34:e0:3a:bd:a0:a8:e7:70:b9:9e:c2:
    72:3c:32:44:f1:52:e5:e7:df:93:1f:80:59:58:67:
    ef:4f:f1:46:99:6e:42:a5:ab:75:09:f3:29:3f:ad:
    39:4d:fd:83:fe";
```

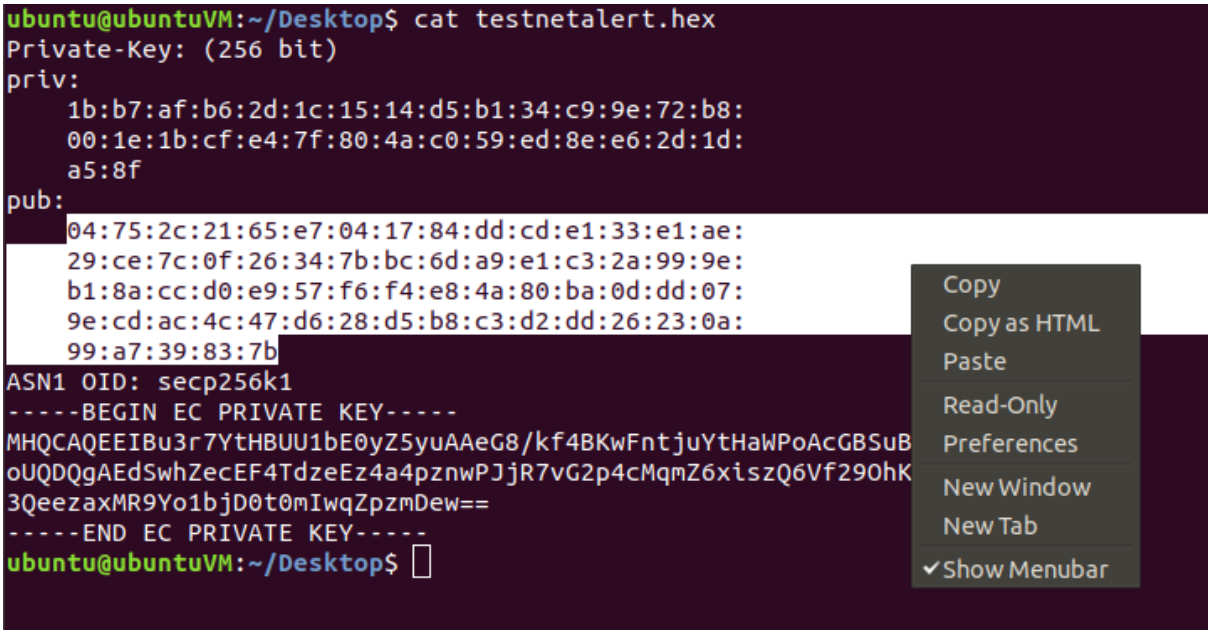
Remove every colon ":" carefully, and after removing the colons, make sure that the new main net public is of the same length as the previous test net public key below it

```
//
// Alert system
//

#include <algorithm>
#include <boost/algorithm/string/classification.hpp>
#include <boost/algorithm/string/replace.hpp>
#include <boost/foreach.hpp>
#include <map>

#include "alert.h"
#include "key.h"
#include "net.h"
#include "sync.h"
#include "ui_interface.h"

using namespace std;

map<uint256, CAlert> mapAlerts;
CCriticalSection cs_mapAlerts;

static const char* pszMainKey = "0436ef34b60f6f890faf24eedd753bf7b6c87234e03abda0a8e770b99ec2723c3244f152e5e7df931f80595867ef4ff146996e42a5ab7509f3293fad394dfd83fe";
static const char* pszTestKey = "04302390343f91cc401d56d68b123028bf52e5fca1939df127f63c6467cdf9c8e2c14b61104cf817d0b780da337893ecc4aaff1309e536162dabbdb45200ca2b0a";
```

Save the file and go to your terminal and show the contents of the file named testnetalert.hex by typing the command "cat testnetalert.hex".

```
ubuntu@ubuntuVM:~/Desktop$ cat testnetalert.hex
Private-Key: (256 bit)
priv:
    1b:b7:af:b6:2d:1c:15:14:d5:b1:34:c9:9e:72:b8:
    00:1e:1b:cf:e4:7f:80:4a:c0:59:ed:8e:e6:2d:1d:
    a5:8f
pub:
    04:75:2c:21:65:e7:04:17:84:dd:cd:e1:33:e1:ae:
    29:ce:7c:0f:26:34:7b:bc:6d:a9:e1:c3:2a:99:9e:
    b1:8a:cc:d0:e9:57:f6:f4:e8:4a:80:ba:0d:dd:07:
    9e:cd:ac:4c:47:d6:28:d5:b8:c3:d2:dd:26:23:0a:
    99:a7:39:83:7b
ASN1 OID: secp256k1
-----BEGIN EC PRIVATE KEY-----
MHQCAQEEIBu3r7YtHBUU1bE0yZ5yuAAeG8/kf4BKwFntjuYtHaWPoAcGBSuBBAAK
oUQDQgAEdSwhZecEF4TdzeEz4a4pznwPJjR7vG2p4cMqmZ6xiszQ6Vf29OhKgLoN
3QeezaxMR9Yo1bjD0t0mIwqZpzmDew==
-----END EC PRIVATE KEY-----
ubuntu@ubuntuVM:~/Desktop$
```

Copy the test net public key from your terminal



Go back to the file alert.cpp and erase the test net public key

```
using namespace std;

map<uint256, CAlert> mapAlerts;
CCriticalSection cs_mapAlerts;

static const char* pszMainKey = "0436ef34b60f6f890faf24eedd753bf7b6c87234e03abda0a8e770b99ec2723c3244f152e5e7df931f80595867ef4ff146996e42a5ab7509f3293fad394dfd83fe";
static const char* pszTestKey = "";
```

Paste the test net public key that you copied earlier

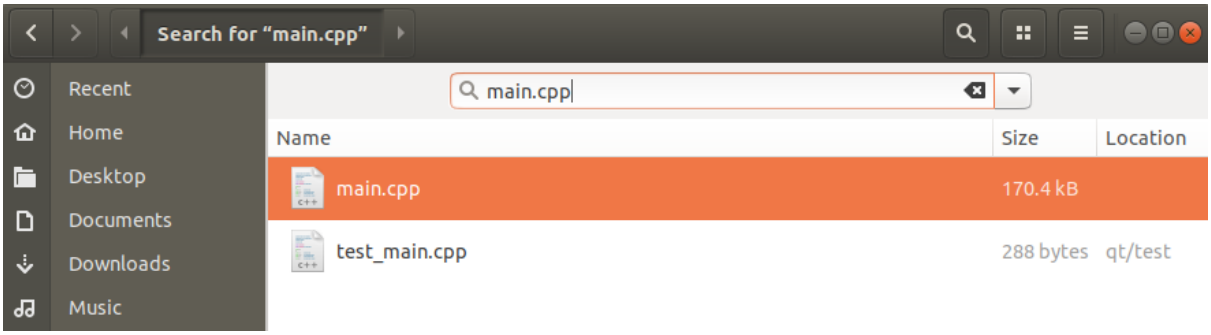
```
using namespace std;

map<uint256, CAlert> mapAlerts;
CCriticalSection cs_mapAlerts;

static const char* pszMainKey = "0436ef34b60f6f890faf24eedd753bf7b6c87234e03abda0a8e770b99ec2723c3244f152e5e7df931f80595867ef4ff146996e42a5ab7509f3293fad394dfd83fe";
static const char* pszTestKey = "04:75:2c:21:65:e7:04:17:84:dd:cd:e1:33:e1:ae:
    29:ce:7c:0f:26:34:7b:bc:6d:a9:e1:c3:2a:99:9e:
    b1:8a:cc:d0:e9:57:f6:f4:e8:4a:80:ba:0d:dd:07:
    9e:cd:ac:4c:47:d6:28:d5:b8:c3:d2:dd:26:23:0a:
    99:a7:39:83:7b
";
```

Remove every colon ":" carefully and make sure that the length of new test net public key matches the length of main net public key.

```
using namespace std;

map<uint256, CAlert> mapAlerts;
CCriticalSection cs_mapAlerts;

static const char* pszMainKey = "0436ef34b60f6f890faf24eedd753bf7b6c87234e03abda0a8e770b99ec2723c3244f152e5e7df931f80595867ef4ff146996e42a5ab7509f3293fad394dfd83fe";
static const char* pszTestKey = "04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80ba0ddd079ecdac4c47d628d5b8c3d2dd26230a99a739837b";
```

Save the file named alert.cpp and close it.

8. Now go inside your src directory within your coin directory using file explorer, and search for a file named **"main.cpp".**



Open the file 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>
```

Go to line 2788

You'll see this snippet of code; we are going to replace "ParseHex"

```
        // 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("040184710fa689ad5023690c80f3a49c8f13f8d45b8c857fbcbc8bc4a8e4d3eb4b10f4d4604fa08dce601aaf0f470216fe1b51850b4acf21b179c45070ac7b03a9") <<
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;
```

Delete the string inside ParseHex

```
// 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("") << 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 back to your terminal and show the contents of file named genesiscoinbase.hex by using command **"cat genesiscoinbase.hex"**.

```
ubuntu@ubuntuVM:~/Desktop$ cat genesiscoinbase.hex
Private-Key: (256 bit)
priv:
    1b:b7:af:b6:2d:1c:15:14:d5:b1:34:c9:9e:72:b8:
    00:1e:1b:cf:e4:7f:80:4a:c0:59:ed:8e:e6:2d:1d:
    a5:8f
pub:
    04:75:2c:21:65:e7:04:17:84:dd:cd:e1:33:e1:ae:
    29:ce:7c:0f:26:34:7b:bc:6d:a9:e1:c3:2a:99:9e:
    b1:8a:cc:d0:e9:57:f6:f4:e8:4a:80:ba:0d:dd:07:
    9e:cd:ac:4c:47:d6:28:d5:b8:c3:d2:dd:26:23:0a:
    99:a7:39:83:7b
ASN1 OID: secp256k1
-----BEGIN EC PRIVATE KEY-----
MHQCAQEEIBu3r7YtHBUU1bE0yZ5yuAAeG8/kf4BKwFntjuYtHaWPoAcGBSuBBAAK
oUQDQgAEdSwhZecEF4TdzeEz4a4pznwPJjR7vG2p4cMqmZ6xiszQ6Vf29OhKgLoN
3QeezaxMR9Yo1bjD0t0mIwqZpzmDew==
-----END EC PRIVATE KEY-----
ubuntu@ubuntuVM:~/Desktop$
```

Copy the public key

```
ubuntu@ubuntuVM:~/Desktop$ cat genesiscoinbase.hex
Private-Key: (256 bit)
priv:
    1b:b7:af:b6:2d:1c:15:14:d5:b1:34:c9:9e:72:b8:
    00:1e:1b:cf:e4:7f:80:4a:c0:59:ed:8e:e6:2d:1d:
    a5:8f
pub:
    04:75:2c:21:65:e7:04:17:84:dd:cd:e1:33:e1:ae:
    29:ce:7c:0f:26:34:7b:bc:6d:a9:e1:c3:2a:99:9e:
    b1:8a:cc:d0:e9:57:f6:f4:e8:4a:80:ba:0d:dd:07:
    9e:cd:ac:4c:47:d6:28:d5:b8:c3:d2:dd:26:23:0a:
    99:a7:39:83:7b
ASN1 OID: secp256k1
-----BEGIN EC PRIVATE KEY-----
MHQCAQEEIBu3r7YtHBUU1bE0yZ5yuAAeG8/kf4BKwFntjuYtHaWPoAcGBSuB
oUQDQgAEdSwhZecEF4TdzeEz4a4pznwPJjR7vG2p4cMqmZ6xiszQ6Vf29OhH
3QeezaxMR9Yo1bjD0t0mIwqZpzmDew==
-----END EC PRIVATE KEY-----
ubuntu@ubuntuVM:~/Desktop$
```

Copy
Copy as HTML
Paste
Read-Only
Preferences
New Window
New Tab
✔ Show Menubar

And paste it in the ParseHex field.

```
    // 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, (co
    txNew.vout[0].nValue = 50 * COIN;
    txNew.vout[0].scriptPubKey = CScript() << ParseHex("04:75:2c:21:65:e7:04:17:84:dd:cd:e1:33:e1:ae:
29:ce:7c:0f:26:34:7b:bc:6d:a9:e1:c3:2a:99:9e:
b1:8a:cc:d0:e9:57:f6:f4:e8:4a:80:ba:0d:dd:07:
9e:cd:ac:4c:47:d6:28:d5:b8:c3:d2:dd:26:23:0a:
99:a7:39:83:7b") << OP_CHECKSIG;
    CBlock block;
```

Remove the colons ":" carefully.

```
    // 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("04752c2165e7041784ddcde133e1ae29ce7c0f26347bbc6da9e1c32a999eb18accd0e957f6f4e84a80ba0ddd079ecdac4c47d628d5b8c3d2dd26230a99a739837b") <<
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;
```

Once done, stay inside main.cpp for the next step.

9. Go to line 2745 in main.cpp, you'll see this snippet of code

```
    pindexGenesisBlock = NULL;                                          2745
    nBestHeight = 0;
    nBestChainWork = 0;
    nBestInvalidWork = 0;
    hashBestChain = 0;
    pindexBest = NULL;
}

bool LoadBlockIndex()
{
    if (fTestNet)
    {
        pchMessageStart[0] = 0xfc;
        pchMessageStart[1] = 0xc1;
        pchMessageStart[2] = 0xb7;
        pchMessageStart[3] = 0xdc;
        hashGenesisBlock =
uint256("0xf5ae71e26c74beacc88382716aced69cddf3dffff24f384e1808905e0188f68f");
    }
```

Change the last character of pchMesssageStart[0], pchMesssageStart[1], pchMesssageStart[2] and pchMesssageStart[3] randomly but please make sure that the new character is hexadecimal e.g. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F.

```
bool LoadBlockIndex()
{
    if (fTestNet)
    {
        pchMessageStart[0] = 0xfa;
        pchMessageStart[1] = 0xc5;
        pchMessageStart[2] = 0xbd;
        pchMessageStart[3] = 0xdf;
        hashGenesisBlock =
uint256("0xf5ae71e26c74beacc88382716aced69cddf3dffff24f384e1808905e0188f68f");
    }
```

Save the file and go to line 3082, you'll see the following snippet of code.

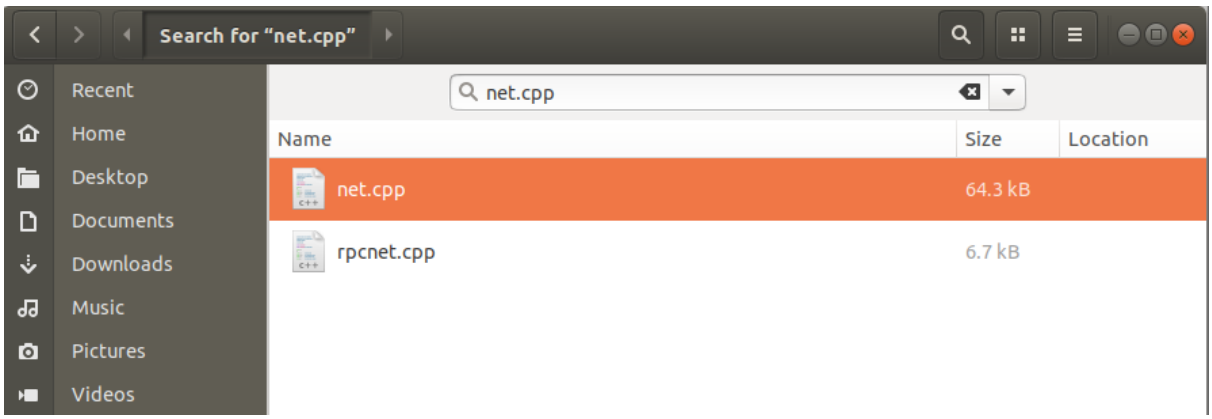
```
Open    *main.cpp                                                   Save
        ~/Desktop/coursecoin/src
            mapOrphanBlocks.count(inv.hash);                        3082
    }
    // Don't know what it is, just say we already got one
    return true;
}



// The message start string is designed to be unlikely to occur in normal data.
// The characters are rarely used upper ASCII, not valid as UTF-8, and produce
// a large 4-byte int at any alignment.
unsigned char pchMessageStart[4] = { 0xfb, 0xc0, 0xb6, 0xdb }; // Coursecoin: increase each by
adding 2 to bitcoin's value.
```

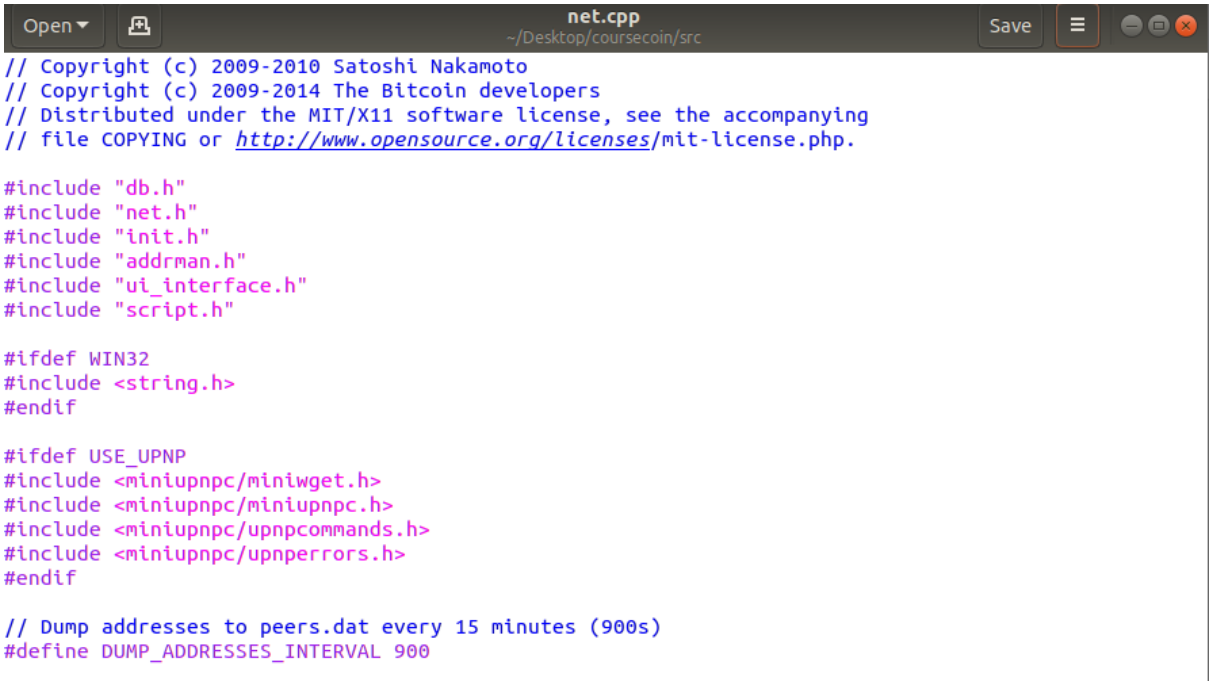
Change the last letter of pchMessageStart[4] randomly, such that it is unique to your coin. Make sure again that it is a hex number.

```
// The message start string is designed to be unlikely to occur in normal data.
// The characters are rarely used upper ASCII, not valid as UTF-8, and produce
// a large 4-byte int at any alignment.
unsigned char pchMessageStart[4] = { 0xf1, 0xcd, 0xb5, 0xdf }; // Coursecoin: increase each by
adding 2 to bitcoin's value.
```

Save the main.cpp file and close it.

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





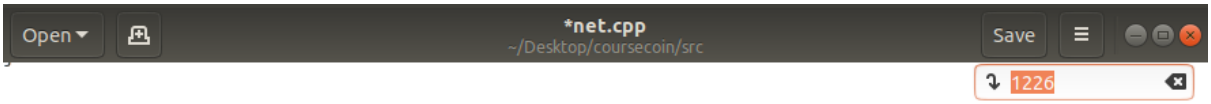
Go to line 1175 and you'll see some main net DNS seeds and test net DNS seeds.

```
// DNS seeds
// Each pair gives a source name and a seed name.
// The first name is used as information source for addrman.
// The second name should resolve to a list of seed addresses.
static const char *strMainNetDNSSeed[][2] = {
    {"coursecointools.com", "dnsseed.coursecointools.com"},
    {"coursecoinpool.org", "dnsseed.coursecoinpool.org"},
    {"xurious.com", "dnsseed.ltc.xurious.com"},
    {"koin-project.com", "dnsseed.koin-project.com"},
    {"weminemnc.com", "dnsseed.weminemnc.com"},
    {NULL, NULL}
};

static const char *strTestNetDNSSeed[][2] = {
    {"coursecointools.com", "testnet-seed.coursecointools.com"},
    {"xurious.com", "testnet-seed.ltc.xurious.com"},
    {"wemine-testnet.com", "dnsseed.wemine-testnet.com"},
    {NULL, NULL}
};
```

Delete all the seeds except for {NULL, NULL} in both main net and test net.

```
// DNS seeds
// Each pair gives a source name and a seed name.
// The first name is used as information source for addrman.
// The second name should resolve to a list of seed addresses.
static const char *strMainNetDNSSeed[][2] = {
    {NULL, NULL}
};

static const char *strTestNetDNSSeed[][2] = {
    {NULL, NULL}
};
```

Save the file and go to line 1226, you'll see some pnSeeds relating to Litecoin



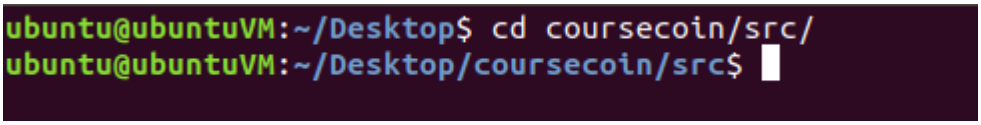

```
unsigned int pnSeed[] =
{
    0x38a9b992, 0x73d4f3a2, 0x43eda52e, 0xa1c4a2b2, 0x73c41955, 0x6992f3a2, 0x729cb992,
0x8b53b205,
    0xb651ec36, 0x8b422e4e, 0x0fe421b2, 0x83c1a2b2, 0xbd432705, 0x2e11b018, 0x281544c1,
0x8b72f3a2,
    0xb934555f, 0x2ba02e4e, 0x6ab7c936, 0x8728555f, 0x03bfd143, 0x0a73df5b, 0xcd2b5a50,
0x746df3a2,
    0x7481bb25, 0x6f4d4550, 0x78582f4e, 0xa03a0f46, 0xe8b0e2bc, 0xa2d17042, 0x718a09b0,
0xdaffd4a2,
    0xbb1a175e, 0xb21f09b0, 0xb5549bc0, 0xe404c755, 0x95d882c3, 0xfff3692e, 0x3777d9c7,
0x425b2746,
    0x497990c6, 0xb2782dcc, 0xf9352225, 0xa75cd443, 0x4c05fb94, 0x44c91c2e, 0x47c6a5bc,
0xd606fb94,
    0xc1b9e2bc, 0x32acd23e, 0x89560da2, 0x5bebdad8, 0x3a210e08, 0xbdc5795b, 0xcc86bb25,
0xbe9f28bc,
    0xef3ff3a2, 0xca29df59, 0xe4fd175e, 0x1f3eaa6b, 0xacdbaa6b, 0xb05f042e, 0x81ed6cd8,
0x9a3c0cc3,
    0x4200175e, 0x5a017ebc, 0x42ef4c90, 0x8abfd143, 0x24fbf3a2, 0x140846a6, 0x4f7d9553,
0xeea5d151,
    0xe67c0905, 0x52d8048e, 0xcabd2e4e, 0xe276692e, 0x07dea445, 0xdde3f3a2, 0x6c47bb25,
0xae0efb94,
    0xf5e15a51, 0xaebdd25b, 0xf341175e, 0x46532705, 0xc47728bc, 0xe4e14c90, 0x9dc8f752,
0x050c042e,
    0x1c84bb25, 0x4f163b25, 0x1a017ebc, 0xa5282e4e, 0x8c667e60, 0xc7113b25, 0xf0b44832,
0xf1a134d0,
    0x973212d4, 0xd35cbb25, 0xd5123b25, 0x68220254, 0x7ad43e32, 0x9268e32e, 0xdf143b25,
0xaf04c436,
    0xaded0051, 0xfa86d454, 0x09db048e, 0x26003b25, 0x58764c90, 0x9a2f555f, 0x0c24ec97,
0x92123b25,
    0x0526d35f, 0x17db048e, 0xd2e42f4e, 0x38cca5bc, 0xc6320ab9, 0xe28ac836, 0xc560aa6b,
0xa5c16041,
    0x70a6f1c0, 0x011ec8c1, 0xd6e9c332, 0x131263c0, 0xa15a4450, 0xef218abc, 0x2729f948,
0x02835443,
    0x5614336c, 0xb12aacb2, 0xe368aa6b, 0x3cc6ffad, 0x36206494, 0x2c90e9c1, 0x32bb53d4,
```

Delete all pnSeeds and replace them by typing **“0x0”** to avoid any errors.

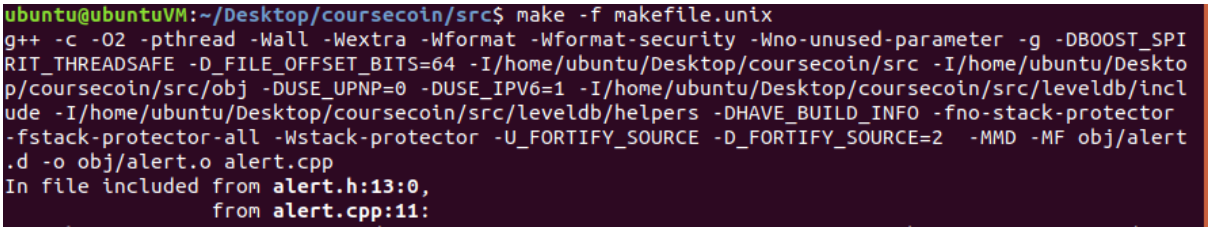
```
unsigned int pnSeed[] =
{
    0x0
};

void DumpAddresses()
{
    int64 nStart = GetTimeMillis();

    CAddrDB adb;
    adb.Write(addrman);

    printf("Flushed %d addresses to peers.dat  %"PRI64d"ms\n",
           addrman.size(), GetTimeMillis() - nStart);
}
```

Save the net.cpp file and close it.

11. Open up your terminal, navigate to the src directory inside your coin directory.



Now to test that all of the changes that we made are working we have to compile all the files by typing the command **"make –f makefile.unix"**



If it compiles without any major error, then congratulations you are good to go for the next part. Otherwise please repeat the whole process.

12. After compilations close your VM but remember to save the machine state.