Intro to Bitcoin Command Line

This is assignment is structured to help you become familiar with blockchains and how they work, specifically Bitcoin. Bitcoin uses a mechanism of unspent transaction outputs (UTXOs) for keeping track of values on the blockchain. You will become familiar with wallets, address, private keys, signing transactions, as well as the UTXO model. Throughout this exercise there will be a number of ***Assignment Deliverable XX:*** . These will tell you what you need to turn in at the end of the assignment. If there are more than one items required for that deliverable add a blank line between them. The format of the submitted work should be a text file with you name, date, assignment name, and UT ID. There should be sections of indicated by Assignment Deliverable XX: followed by the deliverable information. For example:

Karl, Kreder 08/25/2019 Intro_to_Bitcoin_Command_Line UTID

Assignment Deliverable 1:

mhpt1x5Ro8SSMAQrkXwCj4F6sZd1xL9nsT

H9v014McOc0+ESiyWMWJ2lBcGi+Vd+O0/lqSR0mdt15IEK7z158PKkD8JeRe5j4n29+Gyu2m5F3qPW wHQuAPS90=

Hello, Bitcoin!

Assignment Deliverable 2:

....

Tools

First, you will need to install the bitcoin client on your machine. Directions for doing so can be found for the different operating systems in the links below. Once you have validated that you have appropriately installed bitcoin, please proceed to the directions for configuring your installation to work with the test net before syncing.

MacOS

The commands in this guide should be executed in a Terminal application. The built-in one is located in `/Applications/Utilities/Terminal.app` .

Preparation

Install the macOS command line tools:

```
xcode-select--install
```

When the popup appears, click `Install` . Then

install Homebrew.

Dependencies

```
brew install automake berkeley-db4 libtool boost miniupnpc openssl pkg-config
protobuf python qt libevent qrencode
```

Build Bitcoin Core

1. Clone the Bitcoin Core source code:

   ```
   git clonehttps://github.com/bitcoin/bitcoin cd bitcoin
   ```

2. Build Bitcoin Core:

   Configure and build the headless Bitcoin Core binaries as well as the GUI (if Qt is found).

   ```
   ./autogen.sh
   ./configure --without-gui make
   ```

3. It is recommended to build and run the unit tests:

   ```
   make check
   ```

Alternate installation:

1. Download bitcoind

   ```
   curl-Ohttps://bitcoin.org/bin/bitcoin-core-0.17.1/bitcoin-0.17.1- osx64.tar.gz
   ```

2. Extract bitcoind and its binaries

   ```
   tar -zxf bitcoin-0.17.1-osx64.tar.gz
   ```

3. Move executables into default path sudo
   mkdir -p /usr/local/bin
   sudo cp bitcoin-0.17.1/bin/bitcoin* /usr/local/bin/.

4. Clean up directory we've been working in

   ```
   rm -rf bitcoin-0.17.1*
   ```

Linux

The following instructions describe installing Bitcoin Core on Linux systems. Perform the following in your terminal or emulator.

Type the following line to add the Bitcoin Personal Package Archive (PPA) to your system:

```
sudo add-apt-repository ppa:luke-jr/bitcoincore
```

You will be prompted for your user password. Provide it to continue. Afterwards, the following text will be displayed:

```
Stable Channel of Bitcoin Core for Ubuntu.

Includes packages: bitcoin-qt, bitcoind, and bitcoin-tx (To get bitcoin-cli, install

bitcoind)
  Moreinfo:https://launchpad.net/~luke-jr/+archive/ubuntu/bitcoincore Press [ENTER] to continue or
ctrl-c to cancel adding it
```

Press enter to continue. The following text (with some variations) will be displayed, and you will be returned to the command line prompt:

```
gpg: keyring `/tmp/tmp8008p0_f/secring.gpg'created gpg: keyring
`/tmp/tmp8008p0_f/pubring.gpg'created
gpg: requesting key 6EC244E1 from hkp server keyserver.ubuntu.com gpg:
/tmp/tmp8008p0_f/trustdb.gpg: trustdb created
gpg: key 6EC244E1: public key "Launchpad PPA for Luke-Jr" imported gpg: Total number
processed: 1
gpg:                       imported: 1          (RSA: 1) OK
```

Type the following line to get the most recent list of packages:

```
sudo apt-get update
```

This step may take several minutes.

Install the Bitcoin Core daemon (bitcoind) using the following:

```
sudo apt-get install bitcoind
```

If there are no errors during installation proceed to configuration to configure the installation to operate on the Bitcoin testnet and then test the installation.

Configuration

You will now need to create a configuration file to run the Bitcoin daemon connected to the testnet. The following are the default locations for configuration files for both MacOS and Linux.

Linux `$HOME/.bitcoin/bitcoin.conf`

Mac OSX `$HOME/Library/Application\ Support/Bitcoin/bitcoin.conf`

Use your favorite text editor or IDE to create the bitcoin.conf file in the appropriate location. Write the following to the bitcoin.conf file and save. The location may not exist, you will have to create it yourself.

```
# Run on the test network instead of the real bitcoin network.

testnet=1

txindex=1
```

Add aliases

Aliases will allow us to access the bitcoind and bitcoin-cli without having to specify the full path. To set the aliases, execute the following to update your .bash_profile. **Note:** aliases are different for Mac and Linux.

MacOS

```
cat >> ~/.bash_profile <<EOF
alias btcdir="cd ~/Library/Application Support/Bitcoin" #MacOS default bitcoind path
alias bc="~/bitcoin/src/bitcoin-cli"
alias bd="~/bitcoin/src/bitcoind"
alias btcinfo='bc getwalletinfo | egrep "\"balance\""; bc getnetworkinfo | egrep "\"version\"|connections"; bc getmininginfo | egrep"\"blocks\"|errors"'
EOF
```

Linux

```
cat >> ~/.bash_profile <<EOF
alias btcdir="cd ~/.bitcoin/" #linux default bitcoind path
alias bc="bitcoin-cli"
alias bd="bitcoind"
alias btcinfo='bc getwalletinfo | egrep "\"balance\""; bc getnetworkinfo | egrep "\"version\"|connections"; bc getmininginfo | egrep"\"blocks\"|errors"'
EOF
```

Reload your .bash_profile for aliases to take effect.

```
source ~/.bash_profile
```

Start bitcoind

Use the following command to start the bitcoin daemon:

```
bd
```

Monitor the status of bitcoind in a new terminal tab using the tails command:

MacOS:

```
tail -f ~/Library/Application\ Support/Bitcoin/testnet3/debug.log
```

Linux:

```
tail -f ~/.bitcoin/testnet3/debug.log
```

You should see new blocks being written to the log.

Check bitcoin-cli installation

To check to see if bitcoin-cli is working correctly with bitcoind, open another new terminal tab and type the following command:

```
bc help
```

If you see a list of commands you are good to go. If you get an error something went wrong. This could be caused by one of the following things.

1) The blockchain is not yet synced. Check the status by using the tails command above.

2) The bitcoin.conf file is not correctly written. Make sure this is done correctly and in the correct path.

3) bitcoind is not currently running. Look at your system processes and make sure you see bitcoind.

4) The alias paths are not correctly setup.

5) If all else fails, try google, a friend, or ask your friendly TA.

You are now ready to start using bitcoin-cli. However, you will need to wait for you bitcoind instance to sync before you can make transactions. Syncing will take ~1 hour. You can just leave you machine on and come back after dinner.

Here are some example commands to try out with bitcoin-cli:

```
bcgetblockchaininfo bc
getmininginfo
bcgetnetworkinfo bc
getnettotals bc
getwalletinfo
```

Assignment Deliverable 1: To test whether your bitcoind syncs successfully:

1. open https://live.blockcypher.com/btc-testnet/ which lists all recent testnet transactions
2. choose one of the latest transactions and click its transaction hash, like 'bc65dd62170f...'
3. in the new page, make sure its confirmation is '0/6' and copy its full transaction hash, like 'bc65dd62170fa7c4650c37a4937917151e97f49b3e7388d46c23c8a073966de5'
4. use the command 'bitcoin-cli -testnet getrawtransaction <here_is_your_transaction_id>' and submit the output, like 'bitcoin-cli -testnet getrawtransaction bc65dd62170fa7c4650c37a4937917151e97f49b3e7388d46c23c8a073966de5'