



Blockchain Core Camp

RPCのコードを触ろう

@DG Lab - Karl-Johan Alm

このセッションについて

0104. RPCアプリの作成: 注意点とベストプラクティス(前半)

0202. RPCのコードを触ろう

0302. RPCアプリの作成: 注意点とベストプラクティス(後半)

DL: http://bc-2.jp/materials/0202_RPCコードを触ろう-na.pdf

復習 (セッションの後): [0202_RPCコードを触ろう.pdf](#)

Agenda

- ・ファイルの紹介など
- ・デバッグ
- ・ウォームアップ(タスク)
- ・getblockatheight(タスク)
- ・findblockfortx(タスク)

ファイルの紹介など

Bitcoin Coreのソースコード

コードベース外: `secp256k1`、`univalue`

- `secp256k1` = **crypto**
- `univalue` = **JSONブリッジ**

Bitcoin Coreのソースコード

比較的独立しているもの:

wallet, qt

- ・walletは説明不要
- ・qtはQTフレームワーク上のGUIのコード

Bitcoin Coreのソースコード

他:

- ネットワーク(プロトコル、TCP/IPのサーバー等)
- `src/rpc/*`:RPCサービス
- `src/consensus/*`:コンセンサスレイヤー
- アルゴリズムレイヤー(`uint256`, `arith_uint256`, `base58...`)
- 暗号化レイヤー(`pub/privkey`,)
- `src/util*`:ユーティリティーレイヤー

Bitcoin RPC ー 概要

- ・bitcoin-cli.cpp
- ・rpc/blockchain.cpp
- ・rpc/client.cpp / client.h
- ・rpc/mining.cpp
- ・rpc/misc.cpp
- ・rpc/net.cpp
- ・rpc/protocol.cpp / protocol.h
- ・rpc/rawtransaction.cpp
- ・rpc/register.h
- ・rpc/server.cpp / server.h
- ・httprpc.cpp / httprpc.h

Command Line Interfaceアプリケーション

blockに関するRPCコマンド

ユーティリティー、ヘルパー

マイニングに関するRPCコマンド

他のRPCコマンド

ネットに関するRPCコマンド

Auto/JSON request/replyなどの機能

txに関するRPCコマンド

RPCコマンドを登録する機能

RPCのサーバーの機能

HTTP RPCサーバー

Bitcoin RPC — 知っておくべきその他のクラス

• amount.cpp / amount.h

CAmount(satoshi)

• uint256.cpp / uint256.h

ハッシュなどに使うクラス

• arith_uint256.cpp/.h

数学機能の付いたuint256

コードを触る前に

自分用のbranchを作って、それを触ることが一番ベスト。

```
$ git checkout -b 名前-目的
```

例えば:

```
$ git checkout -b taro-rpc
```

デバッグ



デバッグ: bitcoindの再起動

コードを変えたら、bitcoindを再起動する必要がある。

```
$ ./bitcoind -printtoconsole  
[...]
```

^C (Ctrl + c)を押すとbitcoindが止まる。そしてmakeを入れて、
また./bitcoind -printtoconsoleを...

デバッグしたい時のヒント

MacでもLinuxでもCUIのデバッガーが利用できる。

Macユーザー

Linuxユーザー

\$ **lldb** bitcoind (bitcoin-cli、...) \$ **gdb** bitcoind (...)

注意:lldbとgdbのコマンドは別なので、それぞれ調べる必要がある。

デバッグ - lldb / gdb

lldb	gdb	結果
b <file>:<line>	break <file>:<line>	<file>の<line>行でストップ
b <function>	break <function>	<function>に入ったらストップ
bt	bt	スタックフレームを表示する
up, down	up, down	スタックフレーム内を移動する
p <var>	p <var>	<var>を表示する

コマンド対比表 : <http://lldb.llvm.org/lldb-gdb.html>

デバッグ: --enable-debug

Lldbやgdbで変数の内容を表示できなかつたりする時がある。
普通は以下のようにすれば直る:

```
$ ./configure --enable-debug
$ make clean
$ make
```

Demo

RPC

現在使えるコマンド:

```
$ ./bitcoin-cli help
```

ここからコマンドを1つ追加しよう！

ウォームアップ



コマンドを追加しよう(ウォームアップ)

このコマンドを実行したら:

```
$ ./bitcoin-cli print "sample value"
```

この出力が出る:

```
sample value
```

コマンドを追加しよう

```
$ ./bitcoin-cli print "hello"
```

```
hello
```

ファイル: `src/rpc/misc.cpp`

ヒント

src/rpcの中にある.cppファイルの一番下に

```
static const CRPCCommand commands[] =  
  
{ // category  name  actor(function)  okSafeMode  
  ...
```

というところがある。そこを変えれば新しいコマンドを追加することが出来る。

解答例

解答例(1 / 4)

- ① 適切なところに新しい関数を入れる。

```
UniValue print(const UniValue& params, bool
fHelp)
{
}
```

解答例(2/4)

② printをファイル下のcommands[]のどこかに追加する。

```
{ "bc2", "print", &print, true },
```

これで、一度コンパイルして、bitcoindを実行して、
bitcoin-cli print "sample value"
を入れてみよう。

解答例(3/4)

③ まずヘルプを追加した関数の最初のところに入れる。

```
if (fHelp || params.size() != 1)
    throw runtime_error(
        "print \"message\"\n"
    );
```

もう一度コンパイルしてやってみよう。

解答例(4/4)

④ 最後に、コンソールに出力する為のコードを追加する。

```
if (fHelp || params.size() != 1)
    throw runtime_error(
        "print \"message\"\n"
    );
return params[0];
```

もう一度コンパイルしてやってみよう。うまくできました？

getblockatheight

—

役に立つであろうコマンド

ウォームアップしたので役に立ちそうなコマンドも作ろう。

問題: ブロックの高さしか知らないときに、そのブロック高に対応するブロックを見るには、現在2つのコマンドを使う必要がある。

getblockatheight

現在:

```
$ ./bitcoin-cli getblockhash 1624
```

```
0000000429a95049e...
```

```
$ ./bitcoin-cli getblock 0000000429a95049e...
```

```
{  
  "hash": "0000000429a95049e...",  
  "confirmations": 1,  
  [...]
```

getblockatheight

便利！

```
$ ./bitcoin-cli getblockatheight 1624
{
  "hash": "0000000429a95049e...",
  "confirmations": 1,
  [...]
```

getblockatheight

タスク: getblockatheightというRPCコマンドを追加

使用例:

```
$ ./bitcoin-cli getblockatheight 1624
```

```
{ "hash": [...] }
```

```
$ ./bitcoin-cli getblockatheight 1624 false
```

```
000000205c90ff6b11885582020f8798d9434e4362ac
```

```
abcd597297baa9aee083 [...]
```

getblockatheight

コマンド: `getblockatheight <height> (<verbose>)`

`getblock`と同じように、`verbose`という任意パラメーター(デフォルト=`true`)によって、ブロックのHEXを出すかJSONとして表すか決まる。ブロック1624のHEXは:

```
000000205c90ff6b11885582020f8798d9434e4362acabcd597297baa9aee08304000000c534689a02c7fe8f949f07a7d3b5e9fb98ccfa
0034c7a879765c89f00dda141448d81580c191a1dceea000001010000000001010000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
f675fc31950fa06fe907bb395f5fac000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
ebd836974e8cf90120000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
```


ヒント



ヒント①

- `getblockhash`と`getblock`という既に存在するコマンドを使ってやれば良い。(rpc/blockchain.cpp)
- `LOCK(cs_main)`はメソッドから抜ける時に自動的にアンロックされるので2回ロックしてしまうことはない。

ヒント②

・client.cppの**vRPCConvertParams**の中に以下を入れないと
パラメーターがおかしくなる:

```
{ "getblockatheight", 0 },  
{ "getblockatheight", 1 },
```

意味: パラメーター0と1を**convert**して下さい。
(さもないと、stringのままになってしまう)

ヒント③

- ・UniValueの使い方を把握しないと難しい。

- ・新しい配列(array)を作る方法:

```
UniValue arr(UniValue::VARR);
```

- ・配列に要素を入れる方法:

```
arr.push_back(var);
```

解答例

解答例 (1/5)

まずは関数とparamsの確認とヘルプ:

```
UniValue getblockatheight(const UniValue& params,
                          bool fHelp)
{
    if (fHelp || params.size() < 1 || params.size() > 2)
        throw runtime_error(
            "getblockatheight height ( verbose )\n" [...]
        );
}
```

解答例(2/5)

heightパラメータを出す:

```
int nHeight = params[0].get_int();
```

解答例(3/5)

ブロックハッシュを手に入れる為に新しいparamsを作る必要がある。ただしparamsをそのまま使うとverboseのパラメーターが使えなくなる。

(getBlockhashには2つ目のパラメーターがないので):

```
UniValue getblockhashP (UniValue::VARR) ;  
getBlockhashP.push_back (nHeight) ;
```


解答例(4/5)

ブロックハッシュを入手し、getblockのparamsを作成:

```
UniValue getblockhashRes =  
    getblockhash(getblockhashP, false);  
UniValue getblockP(UniValue::VARR);  
getblockP.push_back(getblockhashRes);
```

解答例 (5/5)

このままgetblockを呼び出すとverboseが無視されるので、getblockのparamsに追加する。

```
if (params.size() > 1)
    getblockP.push_back(params[1]);
```

最後にgetblockの結果をそのまま返す：

```
return getblock(getblockP, false);
```

findblockfortx



findblockfortx

トランザクションを送信した後に、結局どのブロックに入ったのかわからない時の為のコマンドを作ろう。

パラメーター: txid maxdepth (さかのぼる深さの制限)

maxdepthはデフォルト=100。つまり、最新ブロックから順番に100ブロック前まで行ってtxidを探す。

findblockfortx

大まかなやり方:

- ① currCount=チェーンの今の高さ
- ② ループ: currCountからcurrCount - maxdepth -1 まで
- ③ ブロックを取り出して、tx配列の中のtx.GetHash() == txid
を探す
- ④ 見つからなかったらthrow runtime_error("...")

ヒント



ヒント①: 今のブロックチェーン

現在のブロックチェーンは

chainActive

という変数で管理されている。

chainActive.Height() = 現在の高さ

ヒント②: ブロックの取り出し方

getBlockを見て、ReadBlockFromDiskの使い方を真似る。

```
if (!ReadBlockFromDisk(block,  
                        pblockindex,  
                        params().GetConsensus()))  
    // エラー  
    // ...
```


ヒント③: ハッシュを比べる方法

パラメーターをstringからuint256にする。

```
uint256 hash(uint256S(strHash));
```

txというCTransactionと比べる時:

```
if (tx.GetHash() == hash) [...]
```

解答例

解答例(1/10)

先ずは関数とparamの数とヘルプを入れる:

```
UniValue findblockfortx(const UniValue& params,
                        bool fHelp)
{
    if (fHelp || params.size() < 1 ||
        params.size() > 2)
        throw runtime_error([...]
```

解答例(2/10)

client.cppに変更の指示を入れる:

```
{ "findblockfortx", 1 },
```

解答例(3/10)

ハッシュを取り出して、`uint256`に変更。

```
std::string strHash = params[0].get_str();  
uint256 hash(uint256S(strHash));
```

解答例(4/10)

maxdepth(デフォルト=100)を取り出す。

```
int nMaxDepth = 100;
if (params.size() > 1)
    nMaxDepth = params[1].get_int();

if (nMaxDepth < 1 || nMaxDepth > chainActive.Height())
    throw JSONRPCError(RPC_INVALID_PARAMETER,
        "Maxdepth out of range");
```

解答例(5/10)

max depth(深さの制限)からbottom height(ブロック高さの底)に変更する。

ブロック高さの底 = 現在の高さ - 深さの制限

```
int nBottomHeight = chainActive.Height() - nMaxDepth;
```

解答例(6/10)

ループ。

```
for (int i = chainActive.Height();  
     i > nBottomHeight;  
     i--) {  
    // ...  
}
```


解答例(7/10)

ブロックを取り出す為に、blockindexを取って、ReadBlockFromDiskを使う。

```
CBlock block;
CBlockIndex* pBlockIndex = chainActive[i];
if (!ReadBlockFromDisk(block, pBlockIndex,
                        params().GetConsensus()))
    throw JSONRPCError(RPC_INTERNAL_ERROR,
                        "Can't read block from disk");
```

解答例(8/10)

ブロックのvtxをループし、txを探す。

```
for (const CTransaction& tx : block.vtx) {  
    if (tx.GetHash() == hash) {  
        // found tx!    }  
}
```

解答例(9/10)

ブロックのハッシュではなく、ブロックのJSONを返したいので、getblockを見ながら真似る。

```
if (tx.GetHash() == hash) {  
    // found tx!  
    return blockToJSON(block, pBlockIndex);  
}
```

解答例(10/10)

最後までループして見つからなかったらエラーをthrowする。

```
throw runtime_error(  
    "Transaction not found within last blocks  
of given depth."  
);
```



Blockchain Core Camp



@DG Lab - Karl-Johan Alm