ROVM Client Document

Weongyo Jeong

2006.3.20

Australia Melbourne
email : weongyo@gmail.org

Abstract

This document explains how ROVM Client can communicate with ROVM Server. Also, the commands and functions of ROVM Client are described.

Contents

1 Introduction

ROVM Client is a part of ROVM project and is made for communicating with ROVM Server. Using ROVM Client, you can connect ROVM Server and debug your opcodes easily.

ROVM Client parses your inputs and converts to a binary code which ROVM Server can understand. Also, ROVM Client sends the opcodes of user and processes the result which be received from ROVM Server. You can understand easily.

ROVM Client's code is python code and a look is similar with python interpreter.

2 Execution

ROVM Client consists of one python file and you can see 'prompt' if you run it.

```
C:\rovmclient> python rovmclient.py
ROVM Client 0.0.1a
Type "help" for more information.
>>>
```

You can see a help messages of ROVM Client typing 'help' command on your screen.

If you want to see a detail infomation for each commands, type 'help <command>'

```
>>> help

Normal mode commands
------------------------------------------------------------------
 help         Show this message.
 opcode       Entering OPCODE mode.
 printopcode  Print saved opcodes.
 quit         Exit rovmclient.
 req <URL>    Get Ticket.
 reqend       Remove Ticket.
 send         Send opcodes.
 stack        Print remote stack status.
------------------------------------------------------------------
If you want to see a detailed explanation, Type 'help <command>'.

>>> help req
```

The prompt of ROVM Client is same with the prompt of python. Also, there is another prompt '`...`'. This prompt is explained at 3.2 section.

3 Modes

In case of ROVM Client, Current there are two modes which are 'execute mode' and 'opcode mode' and each mode is for another purpose. 'execute mode' is a normal normal. 'opcode mode' is for inputing opcodes.

'execute mode' is used for getting a new Ticket ID, sending a reserved commands or getting the server status. These commands belongs to 'execute mode'

‘opcode mode’ is used for making opcodes of ROVM Server.

3.1 Execute Mode

HELP

Print the help messages.

Format : help
Example :

```
>>> help

Normal mode commands
-----------------------------------------------------------------
 help         Show this message.
 opcode       Entering OPCODE mode.
 printopcode  Print saved opcodes.
 quit         Exit rovmclient.
 req <URL>    Get Ticket.
 reqend       Remove Ticket.
 send         Send opcodes.
 stack        Print remote stack status.
-----------------------------------------------------------------
If you want to see a detailed explanation, Type ‘help <command>’.

>>>
```

OPCODE

Enter opcode mode.

ROVM Client has two modes. The first mode is ‘execute mode’ which has a prompt started with ‘>>>’. The second mode is ‘opcode mode’ which has a prompt started with ‘...’.

This command is used for entering opcode mode. If you want to exit from opcode mode, just type <ENTER> key.

The opcodes which have been inputed are saved automatically and will be send to ROVM Server when the user executes ‘SEND’ command.

You don’t need to type all of codes at a time. For example, you can just input 1 opcode and send your opcode to ROVM Server. By using ‘STACK’ command, you can always get the status information of the stack on Ticket ID.

Format : opcode
Example :

```
>>> opcode
... <Current you entered in opcode mode>
```

PRINTOPCODE

Print the list of opcodes which be inputed by a user.

Using this command, you can check opcodes which inputed and you can check how your input is converted to.

Format : printopcode
Example :

```
>>> opcode
... cpush 1
... cpush 2
...
>>> printopcode
#0 (02) '\x10\x01'
#1 (02) '\x10\x02'
```

QUIT

Terminate ROVM Client application.

REQ

Get a new Ticket ID.

In ROVM world, if you want to execute some opcodes on remote ROVM Server, you need a permission of server. 'REQ' command provides a method to get a permission of server. After getting a permission, all commands will enclose a ticket id at their packets.

In case of 'URL address', you must follow a format which is e://<hostname>:<port>. '<hostname>' can be *192.168.23.234* or *www.blahblah.org*. '<port>' is the port of server.

If you want to connect ROVM Server with the authentication, you should use e://<userid>@<hostname>:<port> format.

Format : req <URL address>
Example :

```
>>> req e://192.168.59.128:8889
>>> req e://rovm@192.168.58.129:4390
```

If you use the authentication, ROVM Client will ask the password.

```
C:\>python rovmclient.py
ROVM Client 0.0.1h
Type "help" for more information.
>>> req e://rovm@192.168.58.129:4390
Enter password:
```

REQEND

Destory a Ticket ID. If you send this command to server, ROVM Server releases all structures and memory allocations related with Ticket ID.

Format : reqend
Example :

```
>>> req e://192.168.59.128:8889
>>> reqend
```

SEND

Send opcodes which be inputed by a user to ROVM Server. Before you execute this command, you mush get a ticket from server.

Format : send
Example :

```
>>> req e://192.168.59.128:8889
>>> opcode
... <Input some opcodes that you want to run.>
... <ENTER>
>>> send
```

STACK

Print the status of the stack.

Format : stack
Example :

```
>>> req e://192.168.59.128:8889
>>> opcode
... <Input some opcodes that you want to run.>
... <ENTER>
>>> send
>>> stack
<The contents of the stack>
```

3.2 Opcode Mode

If you want to enter 'opcode mode', you can do it by typing 'opcode' command at 'execute mode'.

```
>>> opcode
...
```

'. . .' prompt points that you are in 'opcode mode'. If you want to exit 'opcode mode', just type <ENTER> key.

```
... <ENTER>
>>>
```

In opcode mode, you also can see help messages about how you can use opcodes. By typing 'help' command, you can see it.

```
... help

Opcode mode commands
----------------------------------------------------------------
 cpush <Char Number>
              Push 'char type' number.
 iadd         Add int.
 <ENTER>      Escape opcode mode
----------------------------------------------------------------
If you want to see a detailed explanation, Type 'help <command>'.
```

BPUSH

Push a boolean. If 1 means TRUE, 0 means FALSE.

Number : 13 (0xd)
Format : bpush `<1 byte unsigned char>`
Stack : (...) → (..., *<boolean>*)
Example :

```
... bpush 0
... bpush 1
```

CPUSH

Push a char. It's same with C type 'char'

Number : 16 (0x10)
Format : cpush `<1 byte number>`
Stack : (...) → (..., *<byte number>*)
Example :

```
... cpush -128
... cpush 127
```

HPUSH

Push a short. It's same with C type 'short'

Number : 17 (0x11)

Format : hpush <2 byte short>
Stack : (...) → (..., <short>)
Example :

```
... hpush 12332
... hpush -123
```

IPUSH

Push a int. It's same with C type 'int'

Number : 18 (0x12)
Format : ipush <4 byte signed number>
Stack : (...) → (..., <*int number*>)
Example :

```
... ipush -2147483648
... ipush 2147483647
```

FPUSH

Push a float. It's same with C type 'float'.

Number : 14 (0xe)
Format : fpush <4 byte float>
Stack : (...) → (..., <float>)
Example :

```
... fpush 123.0
```

DPUSH

Push a double. It's same with C type 'double'.

Number : 15 (0xf)
Format : dpush <8 byte double>
Stack : (...) → (..., <double>)
Example :

```
... dpush 123123.0
```

SPUSH

Push a string. A string which be pushed changes to StringRef automatically. The size of opcode has variable size.

Number : 19 (0x13)
Format : spush “<string>”
Stack : (...) → (..., *<StringRef>*)
Example :

```
... spush "Hello! ROVM Server"
... spush "I love you."
```

DUP

Duplicate the top stack value.

Number : 89 (0x59) Format : dup Stack : (..., <value>) → (..., <value>, <value>) Example :

```
... spush "Hello! ROVM Server"
... dup
```

IADD

Add int. Both *value1* and *value2* must int type. The values are poped from the stack. The int *result* is *value1* + *value2*.

Number : 90 (0x60)
Format : iadd
Stack : (..., *<value1>*, *<value2>*) → (..., *<result>*)
Example :

```
... cpush 1
... cpush 2
... iadd
```

CALL

Call a instance method; dispatch based on class.

Number : 182 (0xb6)
Format : call <name> <type>
Stack : (..., *objectref*, [*arg1*, [*arg2* ...]]) -¿ (...)
Example :

```
... call __init__ (S)V
... call abc (SII)I
```

NEW

Create new object. Memory for a new instance of that class is allocated from the garbage-collected heap. If ROVM Server can’t find that class or can’t be loaded, Server sends ‘ERROR’ messages to the client or server.

Number : 187 (0xbb)
Format : new e://<hostname>[:<port>]<path>
Stack : (...) → (..., *<objectref>*)
Example :

```
... new e://192.168.58.129:4390/testsuite/insert
... new e://envlang.kldp.net/testsuite/insert
```

NEWARRAY

Create new array. The count must be of type int. It is popped off the stack. The count represents the number of elements in the array to be created. A new array whose components are of length *count* is allocated from the garbage-collected heap. A reference *ArrayRef* to this new array object is pushed into the operand stack.

Number : 188 (0xbc)
Format : newarray <type number>
Stack : (..., *<count>*) → (..., *<arrayref>*)
Example :

```
... ipush 256
... newarray 3
```