1-2 ZMQ编译安装和测试

零声学院 Darren QQ 326873713

C/C++Linux服务器开发/高级架构师 https://ke.qq.com/course/420945

网页版本：https://www.yuque.com/docs/share/91aae8cf-f3bb-468f-82d6-3ebefd0e23ad?#《ZMQ编译安装和测试》

1 下载编译安装zmq

ZeroMQ官方文档网址：http://zguide.zeromq.org/page:all

源码下载地址：http://download.zeromq.org/

1.1 安装必须的库

Bash 复制代码

```
sudo apt-get install libtool
sudo apt-get install pkg-config
sudo apt-get install build-essential
sudo apt-get install autoconf
sudo apt-get install automake
```

1.2 安装加密库

Sodium一个易于使用的可为我们提供加密、解密、签名，密码哈希等功能的软件库。除了自身强大的功能外，它还为我们提供了一个兼容API和一个外部API，以进一步的帮助我们提高其可用性。Sodium的目标是提供构建更高级别加密工具所需的所有核心操作。

若命令行不能安装，则去这个github网址手动下载并解压

```bash
git clone git://github.com/jedisct1/libsodium.git
cd libsodium
./autogen.sh -s
./configure && make check
sudo make install
sudo ldconfig
cd ..
```

1.3 下载、编译、安装libzmq

```bash
# 下载
git clone https://github.com/zeromq/libzmq.git
cd libzmq
# 查看tag
git tag
# 版本 获取指定的版本，不要用主分支，可能有bug
 git checkout v4.3.4
./autogen.sh
./configure && make check
sudo make install
sudo ldconfig
cd ..
```

编译debug版本时使用 ./configure --enable-debug

sudo make install的是时候可以看到具体的.so和.a
libtool: install: /usr/bin/install -c src/.libs/libzmq.lai /usr/local/lib/libzmq.la
libtool: install: /usr/bin/install -c src/.libs/libzmq.a /usr/local/lib/libzmq.a
libtool: install: chmod 644 /usr/local/lib/libzmq.a
libtool: install: ranlib /usr/local/lib/libzmq.a
libtool: finish:
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin:/sbin" ldconfig -n /usr/local/lib
我们在编译的时候需要加上libzmq库，比如gcc -o bin file.c -lzmq

2 测试

让我们从简单的代码开始，一段传统的Hello World程序。我们会创建一个客户端和一个服务端，客户端发送Hello给服务端，服务端返回World。下文是C语言编写的服务端，它在5555端口打开一个ZMQ套接字，等待请求，收到后应答World。

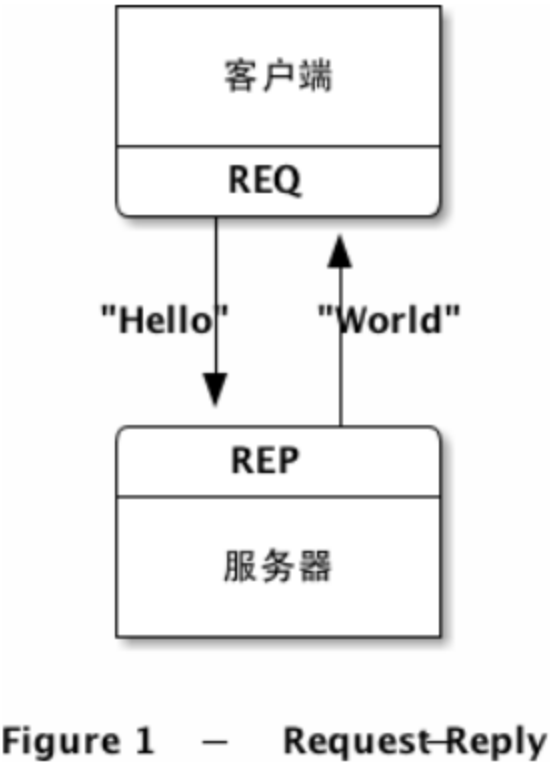

Figure 1 — Request-Reply

hwserver.c: Hello World server

C 复制代码

```c
//
//  Hello World 服务端
//  绑定一个REP套接字至tcp://*:5555
//  从客户端接收Hello，并应答World
//

#include <zmq.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <assert.h>
// gcc -o hwserver hwserver.c -lzmq
int main (void)
{
    //  Socket to talk to clients
    void *context = zmq_ctx_new ();
    //  与客户端通信的套接字
    void *responder = zmq_socket (context, ZMQ_REP);
    int rc = zmq_bind (responder, "tcp://*:5555");
    assert (rc == 0);

    while (1) {
        //  等待客户端请求
        char buffer [10];
        zmq_recv (responder, buffer, 10, 0);
        printf ("收到 Hello\n");
        sleep (1);          //  Do some 'work'
        //  返回应答
        zmq_send (responder, "World", 5, 0);
    }
    return 0;
}
```

编译：gcc -o hwserver hwserver.c -lzmq

使用REQ-REP套接字发送和接受消息是需要遵循一定规律的。客户端首先使用zmq_send()发送消息，再用zmq_recv()接收，如此循环。如果打乱了这个顺序（如连续发送两次）则会报错。类似地，服务端必须先进行接收，后进行发送。

下面是客户端的代码：

hwclient: Hello World client in C

```c
//
//  Hello World 客户端
//  连接REQ套接字至 tcp://localhost:5555
//  发送Hello给服务端，并接收World
//
//  Hello World client
#include <zmq.h>
#include <string.h>
#include <stdio.h>
#include <unistd.h>
//编译：gcc -o hwclient hwclient.c -lzmq
int main (void)
{
    printf ("Connecting to hello world server...\n");
    void *context = zmq_ctx_new ();
    //  连接至服务端的套接字
    void *requester = zmq_socket (context, ZMQ_REQ);
    zmq_connect (requester, "tcp://localhost:5555");

    int request_nbr;
    for (request_nbr = 0; request_nbr != 10; request_nbr++) {
        char buffer [10];
        printf ("正在发送 Hello %d...\n", request_nbr);
        zmq_send (requester, "Hello", 5, 0);
        zmq_recv (requester, buffer, 10, 0);
        printf ("接收到 World %d\n", request_nbr);
    }
    zmq_close (requester);
    zmq_ctx_destroy (context);
    return 0;
}
```

编译：gcc -o hwclient hwclient.c -lzmq

这看起来是否太简单了？ZMQ就是这样一个东西，你往里加点儿料就能制作出一枚无穷能量的原子弹，用它来拯救世界吧！

先运行服务端，再运行客户端。

服务端

```
lqf@ubuntu:/mnt/hgfs/ubuntu/vip/20210701-zeromq/src/test$ ./hwserver
收到 Hello
收到 Hello
收到 Hello
收到 Hello
收到 Hello
收到 Hello
收到 Hello
收到 Hello
收到 Hello
收到 Hello
```

客户端

```
lqf@ubuntu:/mnt/hgfs/ubuntu/vip/20210701-zeromq/src/test$ ./hwclient
Connecting to hello world server...
正在发送 Hello 0...
接收到 World 0
正在发送 Hello 1...
接收到 World 1
正在发送 Hello 2...
接收到 World 2
正在发送 Hello 3...
接收到 World 3
正在发送 Hello 4...
接收到 World 4
正在发送 Hello 5...
接收到 World 5
正在发送 Hello 6...
接收到 World 6
正在发送 Hello 7...
接收到 World 7
正在发送 Hello 8...
接收到 World 8
正在发送 Hello 9...
接收到 World 9
```