目 录

[了解rabbitmq 1](#_Toc422147077)

[安装配置 2](#_Toc422147078)

[配置说明 2](#_Toc422147079)

[接口说明 2](#_Toc422147080)

[demo源码 3](#_Toc422147081)

[附：enum amqp\_status\_enum\_ 3](#_Toc422147082)

## 了解rabbitmq

<http://www.rabbitmq.com/getstarted.html>

## 安装配置

1、到[https://github.com/alanxz/rabbitmq-c/releases](https://github.com/alanxz/rabbitmq-c/releases下载rabbitmq-c-0.6.0.tar.gz) 下载rabbitmq-c-0.6.0.tar.gz

2、编译安装

autoreconf -i

./configure

make

make install

参考来源: <https://github.com/alanxz/rabbitmq-c#building-and-installing-using-cmake>

3、配置/usr/local/lib到LD\_LIBRARY\_PATH

4、makefile文件加入-lrabbitmq

## 配置说明

详情参见rabbitmq\_config.h

**PRIORITY\_MAX** 10 //定义优先级范围(0到PRIORITY\_MAX)，数字越大优先级越高.

**log\_path** log/rabbitmq.log //独立的rabbitmq-c接口日志，可以通过rmq\_log\_set\_handler使用自定义日志接口.

**rmq\_hostname** 192.168.102.50 //安装rabbitmq的目标机器IP

**rmq\_port** 5672 //rabbitmq监听端口

**rmq\_username** tei //登陆帐户

**rmq\_passwd** tei //登陆密码

**rmq\_exchange\_queues** //配置exchange和对应的queue

**rmq\_topics** //配置queue对应的topics

说明：按照rabbitmq协议，当发送到队列，先根据约定的exchange进行路由，再根据topic的匹配规则路由到最终的queue；topic相当于queue感兴趣的专题属性，进行绑定后一旦属性匹配上，就能进入当前队列.

## 接口说明

BOOL rmq\_log\_init()

描述：初始化日志对象

返回值：TRUE/FALSE

void rmq\_log\_set\_handler()

描述：使用自定义日志接口，默认会写到log/rabbitmq.log

函数指针类型：typedef void (\*rmq\_log\_handler)(const char\*, rmq\_log\_level level);

void rmq\_log\_exit()

BOOL rmq\_init()

描述：根据配置初始化连接channel，日志对象初始化(如果还没初始化)。

BOOL rmq\_exchange\_queues\_declare()

描述：根据配置初始化exchange\queue\topic，然后后面按照这个规则发送入队。

int rmq\_send(const char\* exchange, int priority, const char\* routing\_key, const void\* sendbuf, int sendlen)

描述：向队列发送数据。

在rabbitmq\_config.h目前设置心跳时间rmq\_heartbeat，如果发现tcp连接已经失效客户端会自动切断tcp连接，rmq\_send遇到通讯层异常会尝试进行一次重连然后继续发送数据，其他异常需要应用层根据amqp.h对amqp\_status\_enum\_的定义来处理异常。

**exchange**: 发送到上面初始化的exchange。

**priority**: 优先级，目前范围是0-10，若不用，输入0即可。

**routing**\_key：匹配上面topic规则的字符串，比如zb.yy匹配到zb.#，那么上面绑定了zb.#的队列都会进队，send一次可能会匹配然后发送到多个队列。

**sendbuf**: 发送的数据部分。

**sendlen**: 数据长度。

返回值：依据amqp协议，共27个，可参见附录

void rmq\_exit()

描述：关闭释放链接。

### demo源码

参考test.c

编译：g++ test.c -lrabbitmq utils.c rabbitmq\_mgr.c rabbitmq\_log.c

## 附：enum amqp\_status\_enum\_

/\*\*

\* Status codes

\*

\* \since v0.4.0

\*/

/\* NOTE: When updating this enum, update the strings in librabbitmq/amqp\_api.c \*/

typedef enum amqp\_status\_enum\_

{

AMQP\_STATUS\_OK = 0x0, /\*\*< Operation successful \*/

AMQP\_STATUS\_NO\_MEMORY = -0x0001, /\*\*< Memory allocation

failed \*/

AMQP\_STATUS\_BAD\_AMQP\_DATA = -0x0002, /\*\*< Incorrect or corrupt

data was received from

the broker. This is a

protocol error. \*/

AMQP\_STATUS\_UNKNOWN\_CLASS = -0x0003, /\*\*< An unknown AMQP class

was received. This is

a protocol error. \*/

AMQP\_STATUS\_UNKNOWN\_METHOD = -0x0004, /\*\*< An unknown AMQP method

was received. This is

a protocol error. \*/

AMQP\_STATUS\_HOSTNAME\_RESOLUTION\_FAILED= -0x0005, /\*\*< Unable to resolve the

\* hostname \*/

AMQP\_STATUS\_INCOMPATIBLE\_AMQP\_VERSION = -0x0006, /\*\*< The broker advertised

an incompaible AMQP

version \*/

AMQP\_STATUS\_CONNECTION\_CLOSED = -0x0007, /\*\*< The connection to the

broker has been closed

\*/

AMQP\_STATUS\_BAD\_URL = -0x0008, /\*\*< malformed AMQP URL \*/

AMQP\_STATUS\_SOCKET\_ERROR = -0x0009, /\*\*< A socket error

occurred \*/

AMQP\_STATUS\_INVALID\_PARAMETER = -0x000A, /\*\*< An invalid parameter

was passed into the

function \*/

AMQP\_STATUS\_TABLE\_TOO\_BIG = -0x000B, /\*\*< The amqp\_table\_t object

cannot be serialized

because the output

buffer is too small \*/

AMQP\_STATUS\_WRONG\_METHOD = -0x000C, /\*\*< The wrong method was

received \*/

AMQP\_STATUS\_TIMEOUT = -0x000D, /\*\*< Operation timed out \*/

AMQP\_STATUS\_TIMER\_FAILURE = -0x000E, /\*\*< The underlying system

timer facility failed \*/

AMQP\_STATUS\_HEARTBEAT\_TIMEOUT = -0x000F, /\*\*< Timed out waiting for

heartbeat \*/

AMQP\_STATUS\_UNEXPECTED\_STATE = -0x0010, /\*\*< Unexpected protocol

state \*/

AMQP\_STATUS\_SOCKET\_CLOSED = -0x0011, /\*\*< Underlying socket is

closed \*/

AMQP\_STATUS\_SOCKET\_INUSE = -0x0012, /\*\*< Underlying socket is

already open \*/

\_AMQP\_STATUS\_NEXT\_VALUE = -0x0013, /\*\*< Internal value \*/

AMQP\_STATUS\_TCP\_ERROR = -0x0100, /\*\*< A generic TCP error

occurred \*/

AMQP\_STATUS\_TCP\_SOCKETLIB\_INIT\_ERROR = -0x0101, /\*\*< An error occurred trying

to initialize the

socket library\*/

\_AMQP\_STATUS\_TCP\_NEXT\_VALUE = -0x0102, /\*\*< Internal value \*/

AMQP\_STATUS\_SSL\_ERROR = -0x0200, /\*\*< A generic SSL error

occurred. \*/

AMQP\_STATUS\_SSL\_HOSTNAME\_VERIFY\_FAILED= -0x0201, /\*\*< SSL validation of

hostname against

peer certificate

failed \*/

AMQP\_STATUS\_SSL\_PEER\_VERIFY\_FAILED = -0x0202, /\*\*< SSL validation of peer

certificate failed. \*/

AMQP\_STATUS\_SSL\_CONNECTION\_FAILED = -0x0203, /\*\*< SSL handshake failed. \*/

\_AMQP\_STATUS\_SSL\_NEXT\_VALUE = -0x0204 /\*\*< Internal value \*/

} amqp\_status\_enum;