关于rabbitmq-c消息队列

接口参数

- 发布端

  - ***amqp_new_connection()***
    声明一个新的 amqp_connection，简称为 conn

  - ***amqp_tcp_socket_new(conn)***
    创建一个 TCP socket:
    conn 为先前声明的 amqp_conneciton，函数返回值记为 socket

  - ***amqp_socket_open(socket, hostname, port)***
    打开 TCP socket，获取 socket 值为 status。其中：
    socket --- 为先前创建的 TCP;
    hostname --- 为rabbitmq server 主机；
    port --- 为其监听端口

  - ***amqp_login(amqp_connection_state_t state, char const *vhost, int channel_max, int frame_max, int heartbeat, int sasl_method, ...)***
    用于登录rabbitmq sever，主要用于进行权限管理：
    state --- 如前文 conn(amqp_connection);
    vhost --- 虚拟主机;
    chnnel_max --- 最大连接数;
    frame_max --- 和客户端通信时允许的最大帧数， 默认值131072（↑提高吞吐，↓降低时延）;
    heartbeat --- 心跳帧，两次心跳间的秒数，heartbeat超时值定义了RabbitMQ及其client库在多久之后认为TCP连接不可到达。默认为0;
    sasl_method --- SSL认证;

  - ***amqp_channel_open(amqp_connection_state_t state, amqp_channel_t channel)***
    用于关联 conn 和 channel:
    state --- conn(amqp_connection);
    channel --- 进行RPC的通道；

  - ***amqp_basic_publish(amqp_connection_state_t state, amqp_channel_t channel, amqp_bytes_t exchange, amqp_bytes_t routing_key, amqp_boolean_t mandatory, amqp_boolean_t immediate, struct amqp_basic_properties_t_ const *properties, amqp_bytes_t body)***
    发布消息到代理（通过 routing key 发布消息到 exchange 上）:
    state --- conn；
    channel --- 通道标识符；
    exchange --- 代理上被发布消息的 exchange；
    routing_key --- 发布消息时需要使用的路由密匙；
    mandatory --- 此标志指示服务器如果无法将消息路由到队列，该如何反应。 如果设置了此标志，则服务器将使用Return方法返回不可路由的消息。 如果此标志为零，则服务器将静默删除该消息；
    immediate --- 该标志指示服务器如果消息不能立即路由到队列消费者，该如何反应。 如果设置了此标志，则服务器将使用Return方法返回不可传递的消息。 如果此标志为零，则服务器将对消息进行排队，但不保证它将被使用；
    properties --- 与消息相关联的属性；
    body --- 消息正文；

  - ***amqp_channel_close(amqp_connection_state_t state, amqp_channel_t channel, int code)***
    关闭通道：
    state --- conn;
    channel --- 通道标识符；
    code --- 关闭通道的原因，默认值为 AMQP_REPLY_SUCCESS;

  - ***amqp_connection_close(amqp_connection_state_t state, int code)***
    关闭整个连接：
    state --- conn;
    code --- 关闭连接的原因，默认值为 AMQP_REPLY_SUCCESS;

- 消息队列

  - ***amqp_queue_declare(amqp_conneciton_state_t state, amqp_channel_t channel, amqp_bytes_t queue, amqp_boolean_t passive, amqp_boolean_t durable, amqp_boolean_t exclusive, amqp_boolean_t auto_delete, amqp_table_t arguments)***
    声明队列:
    state --- conn;
    channel --- 进行RPC的通道；
    queue --- 队列；
    passive --- 被动？（存疑）
    durable --- 队列是否持久化，1 为持久；
    exclusive --- 是否独占（当前连接不在时，队列是否自动删除）；
    auto_delete --- 是否自动删除（无消费者时）；
    arguments --- 额外参数，一般不做处理，设置为 amqp_empty_table;

  - ***amqp_queue_bind(amqp_connection_state_t state, amqp_channel_t channel, amqp_bytes_t queue, amqp_bytes_t exchange, amqp_bytes_t routing_key, amqp_table_t arguments)***
    将消息队列与 exchange 绑定：
    state --- conn;
    channel --- 进行RPC的通道;
    queue --- 队列；
    exchange --- 与之绑定的 exchange；
    routing_key --- 路由键，队列与 exchange 绑定的标识符;
    arguments --- 一般设置为 amqp_empty_table;

- 消费端
  - ***amqp_basic_consume(amqp_connection_state_t state, amqp_channel_t channel, amqp_bytes_t queue, amqp_bytes_t consumer_tag, amqp_boolean_t no_local, amqp_boolean_t no_ack, amqp_boolean_t exclusive, amqp_table_t arguments)***
    在一个队列上开始一个消费：
    state --- conn;
    channel --- 进行RPC的通道;
    queue --- 队列；
    consumer_tag --- 消费者标签；
    no_local --- 如果设置了非本地字段，则服务器不会将消息发布到conn上；
    no_ack --- 如果设置了此字段，则服务器不会受到消息的确认；
    exclusive --- 是否独占；
    arguments --- 一般设置为 amqp_empty_table;

  - ***amqp_consume_message(amqp_connection_state_t state, amqp_envelope_t *envelope, struct timeval *timeout, int flags)***
    等待并消费一条消息：
    state --- conn；
    envelope --- 指向 amqp_envelope_t 对象的指针；
    timeout --- 等待消息传递的超时。NULL表示阻塞；
    flags --- 0为通过，表当前未使用；

  - ***amqp_basic_ack(amqp_connection_state_t state, amqp_channel_t channel, uint64_t delivery_tag, amqp_boolean_t multiple)***
    确认一条消息：
    state --- conn；
    channel --- 通道标识符；
    delivery_tag --- 需要确认的消息的传递标签；
    multiple --- 真，则确认直到此标签的所有消息； 假，则仅确认此delivery_tag对应的消息；