ATT

1 初始化

bt\_att\_init()：注册ATT固定通道（CID\_ATT、bt\_att\_accept），初始化GATT（bt\_gatt\_init）。

bt\_att\_accept()：设置通道操作（ops）成员函数，包括连接（bt\_att\_connected）、断开连接（bt\_att\_disconnected）、接收（bt\_att\_recv）、加密改变。为当前连接的ATT请求设置操作、信号量和通道等参数。

bt\_att\_connected()：

2 接收数据

bt\_att\_recv()：根据ATT码（hdr->code）从数组中（handlers）找到对应的处理方法（handler）。若处理方法为空，则说明是不支持的ATT命令。若ATT数据包的长度不等于期望的长度，则返回无效数据包错误，否则调用处理方法的函数。

handlers：处理方法数组。

2.1 处理读组请求/响应

**att\_read\_group\_req()**：得到起始句柄（start\_handle、end\_handle），创建uuid。发送读组响应（att\_read\_group\_rsp）。

**att\_read\_group\_rsp()**：创建读组响应数据包（READ\_GROUP\_RSP），设置响应数据（data.rsp），遍历GATT数据库中处于起始句柄间的每个属性并调用读组回调函数（read\_group\_cb），将服务（包含起始句柄和服务UUID）添加到响应数据中。发送响应（bt\_att\_chan\_send\_rsp）。

**read\_group\_cb()**：若属性不是服务（主/次服务），则更新组的结束句柄（data->group->end\_handle）后返回继续（CONTIUE）。如果uuid不匹配，则将组（data->group）设置为空并返回继续。若没有剩余空间（即数据长度大于MTU），则返回停止（STOP）。将组添加到缓冲并设置组的起始句柄、属性值（att\_chan\_read），若读取的属性值不为0且组不为空，则返回继续。

**att\_chan\_read()**：调用属性的读函数（attr->read），将属性值（这里是服务的UUID）保存到缓冲中。

2.2 处理读类型请求/响应

att\_read\_type\_req()：

att\_read\_type\_rsp()：

read\_type\_cb()：

att\_handle\_read\_type\_rsp()：

注意：所有响应（读、写等）的处理方法都是一样的，就是调用发送对应请求时设置的回调函数（req->func），而发送请求一般是由用户调用Gatt.c中的函数来发起的。

2.3 处理读请求/响应

**att\_read\_req()**：获取请求（req）和句柄（handle），调用ATT读响应（att\_read\_rsp）。

**att\_read\_rsp()**：创建读响应数据包，设置读数据。遍历GATT服务链表的每个属性，根据句柄找到对应属性，调用读回调函数（read\_cb）。若返回错误，则发送错误响应（send\_err\_rsp），否则发送ATT读响应（bt\_l2cap\_send\_cb）。

**read\_cb()**：将读响应（data->rsp）添加到缓冲（data->buf）中。检查属性权限（check\_perm），调用属性的读函数（由用户实现，在声明服务时注册，attr->read）。若读取的字节数小于0，则返回错误。将读取的数据添加到缓冲中。

**att\_handle\_read\_rsp()**：ATT处理响应（att\_handle\_rsp）。

**att\_handle\_rsp()**：取消ATT的超时工作项。若ATT请求为空，说明没有未处理的请求，跳转到待定请求处理（process）。释放原始缓冲（att->req->buf，因为请求已经发送了）。调用回调函数（att->req->func，此函数由GATT传递给ATT），并重新设置回调函数。若回调函数为空，则破坏请求（att\_req\_destroy）。process：处理待定请求（att\_process）。

**att\_process()**：从请求链表（att->reqs）获取下个请求（sys\_slist\_get），若请求不为空，则发送请求（att\_send\_req）。

2.4 处理写请求/响应

**att\_write\_req()**：获取句柄（handle），调用ATT写响应（att\_write\_rsp）。

**att\_write\_rsp()**：创建写响应数据包，设置写数据（data）。遍历GATT的每个属性，根据句柄找到对应属性，调用写回调函数（write\_cb）。

write\_cb()：检查属性权限（check\_perm），调用属性的写函数（由用户实现，在声明服务时注册，attr->write），若写入的字节数为0或不等于数据长度，则返回错误。

**att\_handle\_write\_rsp()**：ATT处理响应（att\_handle\_rsp）。

2.5 处理通知

**att\_notify()**：获取连接（conn）和属性句柄（handle），调用GATT通知处理函数（bt\_gatt\_notification）。

**bt\_gatt\_notification()**：找到连接对应的订阅（gatt\_sub\_find），若订阅（sub）不为空，则遍历订阅的参数链表（sub->list），若句柄与参数句柄（params->value\_handle）相同，则调用参数的通知/指示回调函数（param->notify）。

2.6 处理指示

**att\_indicate()**：获取连接（conn）和属性句柄（handle），调用GATT通知处理函数（bt\_gatt\_notification）。发送指示确认（CONFIRM）数据包（告诉服务器已收到指示）。

**att\_confirm()**：

**注**：在客户端订阅通知/指示时，已将订阅参数添加到订阅的参数链表。

3 发送数据

**bt\_att\_req\_send()**：获取ATT通道（att\_chan\_get），若为空则返回错误。若有未完成的请求（att->req），则将当前请求先添加到请求链表（att->reqs）中，稍后再发送，否则立刻发送请求（att\_send\_req）。

**att\_send\_req()**：设置ATT的当前请求（att->req），获取发送信号量（att->tx\_sem），即等待直到信号量可用。若ATT未连接（att\_is\_connected），则返回错误并释放发送信号量。保持请求状态（net\_buf\_simple\_save），通过L2CAP发送（bt\_l2cap\_send\_cb）。

总结：通过以上分析，可以得出一个结论。分为两种情况，客户端向服务器读写数据和服务器向客户端发送数据。客户端向服务器读写数据的过程为，客户端调用GATT的函数，发送对应的ATT请求，并将请求加入到ATT的处理链表。服务器收到请求后进行处理（即发送对应的ATT响应），客户端收到响应后进行处理（即调用发送请求时注册的响应处理函数），此时一个请求才算处理完成。服务器向客户端发送数据则以通知或指示的方式进行。