1.

（1）协议设计：网络编程的服务端程序在大多数的情况下，并不只是与一个客户端进行通讯。在嵌入式行业中，设备通常被被要求至少同时需要与5-10个客户端同时通信，而对于嵌入式设备来说，其内部资源是非常有限的，我们本次使用select函数进行处理并发。本周要实现的是服务器能实现多个客户端的并发处理。即实现服务器在等待一个客户端发送下一个请求时，能够同时处理来自其它客户端的请求。

（2）并发客户端的设计方法和技术：我们是利用select函数去实现并发客户端。

int select(int maxfd + 1,fd\_set \*readset,fd\_set \*writeset, fd\_set \*exceptset,const struct timeval \* timeout);

参数一 ：最大的文件描述符 + 1。是一个整数值，是指集合中所有文件描述符的范围，即所有文件描述符的最大值加1。

参数二 ：用于检查可读性，是指向fd\_set结构的指针，这个集合中应该包括文件描述符，我们是要监视这些文件描述符的读变化的，即我们关心是否可以从这些文件中读取数据了，如果这个集合中有一个文件可读，select就会返回一个大于0的值，表示有文件可读，如果没有可读的文件，则根据timeout参数再判断是否超时，若超出timeout的时间，select返回0，若发生错误返回负值。可以传入NULL值，表示不关心任何文件的读变化。

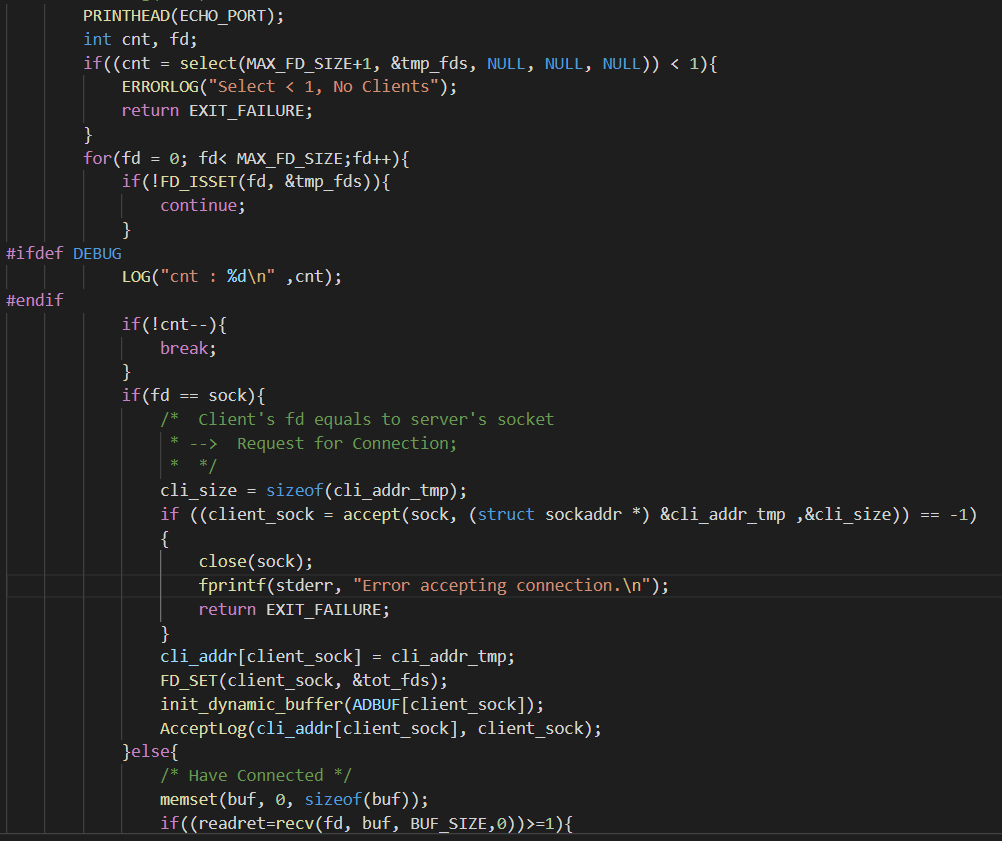
参数三：用于检查可写性，具体的解释同参数二一致。

参数四：用于检查文件错误异常，具体的解释同参数二一致。

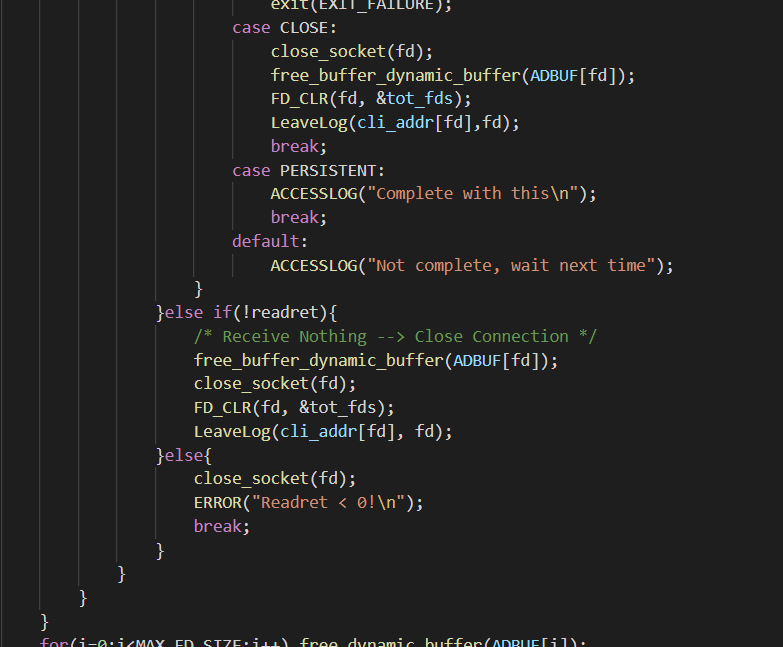
参数五:用于指定select函数运行到此处时等待的时间。它可以使select处于三种状态，第一，若将NULL以形参传入，即不传入时间结构，就是将select置于阻塞状态，一定等到监视文件描述符集合中某个文件描述符发生变化为止；第二，若将时间值设为0秒0毫秒，就变成一个纯粹的非阻塞函数，不管文件描述符是否有变化，都立刻返回继续执行，文件无变化返回0，有变化返回一个正值；第三，timeout的值大于0，这就是等待的超时时间，即select在timeout时间内阻塞，超时时间之内有事件到来就返回了，否则在超时后不管怎样一定返回，返回值同上述。

我们通过select函数实现了并发。

2.







我们先用select函数返回共监听到有多少个fd上有事件。如果为0，则直接结束。对于不在监听事件中我们就跳过，只保留符合条件的套接字留在tmp\_fds里面。如果fd的数量等于sock的数量，即监听到有客户进行连接，则我们开始连接，然后我们把client\_sock加入tot\_fds这个集合中。如果fd和sock数量不相等，即监听到已连接的客户端发来的数据，我们开始接受客户端发来的消息并作处理后回写给客户端。如果readret数量为0，则代表已经断开连接，我们要清除tot\_fds中对它的监听事件，清除相应位置。如果readret不是0，我们就利用之前已经实现的deal\_buf进行解析即可。