1、Linux IO复用接口：

select

* select能监控的描述符个数由内核中的FD\_SETSIZE限制，仅为1024，这也是select最大的缺点，因为现在的服务器并发量远远不止1024。即使能重新编译内核改变FD\_SETSIZE的值，但这并不能提高select的性能。
* 每次调用select都会线性扫描所有描述符的状态，在select结束后，用户也要线性扫描fd\_set数组才知道哪些描述符准备就绪，等于说每次调用复杂度都是O（n）的，在并发量大的情况下，每次扫描都是相当耗时的，很有可能有未处理的连接等待超时。
* 每次调用select都要在用户空间和内核空间里进行内存复制fd描述符等信息。

**poll**

* poll使用pollfd结构来存储fd，突破了select中描述符数目的限制。
* 与select的后两点类似，poll仍然需要将pollfd数组拷贝到内核空间，之后依次扫描fd的状态，整体复杂度依然是O（n）的，在并发量大的情况下服务器性能会快速下降。

**epoll**

* epoll维护的描述符数目不受到限制，而且性能不会随着描述符数目的增加而下降。
* 服务器的特点是经常维护着大量连接，但其中某一时刻读写的操作符数量却不多。epoll先通过epoll\_ctl注册一个描述符到内核中，并一直维护着而不像poll每次操作都将所有要监控的描述符传递给内核；在描述符读写就绪时，通过回掉函数将自己加入就绪队列中，之后epoll\_wait返回该就绪队列。也就是说，epoll基本不做无用的操作，时间复杂度仅与活跃的客户端数有关，而不会随着描述符数目的增加而下降。
* epoll在传递内核与用户空间的消息时使用了内存共享，而不是内存拷贝，这也使得epoll的效率比poll和select更高。

2、线性表，关联数组不是线性表

* 1.循环链表是另一种形式的链式存贮结构。它的特点是表中最后一个结点的 [指针](http://baike.baidu.com/view/159417.htm) 域指向 [头结点](http://baike.baidu.com/view/549497.htm) ，整个链表形成一个环。
* （1）单循环链表——在[单链表](http://baike.baidu.com/view/178637.htm)中，将终端结点的[指针](http://baike.baidu.com/view/159417.htm)域NULL改为指向表头结点或开始结点即可。

              （2）多重链的循环链表——将表中结点链在多个环上。

* 2 队列（Queue）是只允许在一端进行插入，而在另一端进行删除的运算受限的线性表；

     3. 栈（stack）在[计算机科学](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=67589518)中是限定仅在栈顶进行插入或删除操作的线性表。

    4. “关联数组”是一种具有特殊索引方式的数组。不仅可以通过整数来索引它，还可以使用字符串或者其他类型的值（除了NULL）来索引它。   详情查看：   <http://baike.baidu.com/link?url=yYrNB5t4PrCvs-XfxfEM0ZZfALpsEi3FYopk1v0BuopUSWOr7mS0Lou8C-SzhDnSuv7BH5vKIoIblvi8GgUmGq>

       关联数组和数组类似，由以名称作为键的字段和方法组成。   它包含标量数据，可用索引值来单独选择这些数据，和数组不同的是， 关联数组的索引值不是非负的整数而是任意的标量。这些标量称为Keys，可以在以后用于检索数组中的数值。

       关联数组的元素没有特定的顺序，你可以把它们想象为一组卡片。每张卡片上半部分是索引而下半部分是数值。

5.链表（Linked list）是一种常见的基础数据结构，是一种线性表，是一种物理存储单元上非连续、非顺序的存储结构。双向链表也叫[双链表](http://baike.baidu.com/view/1850617.htm) ，是链表的一种，它的每个数据结点中都有两个 [指针](http://baike.baidu.com/view/159417.htm) ，分别指向直接后继和直接前驱。所以，从双向链表中的任意一个结点开始，都可以很方便地访问它的前驱结点和后继结点。一般我们都构造双向 [循环链表](http://baike.baidu.com/view/178643.htm) 。

3、被查找的字符串长N，模式串长P

朴素匹配算法 时间复杂度O（(N-P+1)\*P）

KMP匹配算法 时间复杂度为O（N+P）

4、