select()函数

在Linux中，我们可以使用select函数实现I/O端口的复用，**传递给** select**函数的参数**会告诉内核：

- 对每个描述符，我们所关心的状态。(我们是要想从一个文件描述符中读或者写，还是关注一个描述符中是否出现异常)
- 我们要等待多长时间。(我们可以等待无限长的时间，等待固定的一段时间，或者根本就不等待)

从 select函数返回后，内核告诉我们一下信息：

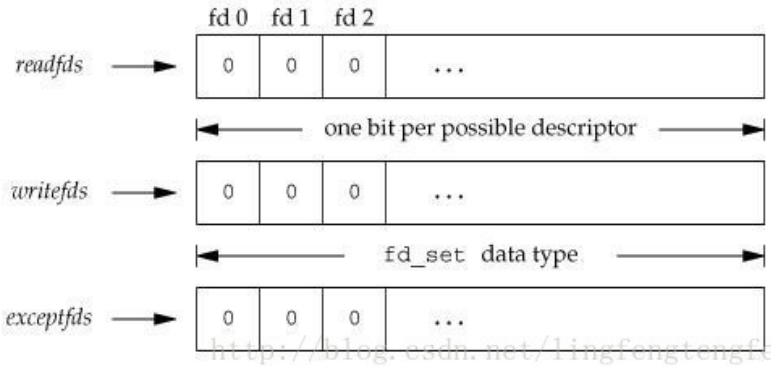
- 对我们的要求已经做好准备的描述符的个数
- 对于三种条件**哪些描述符已经做好准备**.(读，写，异常)

```
/**
 * @brief:用来监视文件句柄的变化情况
 * @Parma[1]:maxfdp1 最大的文件描述符+1，
 * @Parma[2]:readset 用来回传该描述词的读的状况
 * @Parma[3]:writeset 用来回传该描述词的写的状况
 * @Parma[4]:exceptset 用来回传该描述词的例外的状况
 * @Parma[5]:timeout 设置select函数的等待时间
 */
int select(int maxfdp1, fd_set *readset, fd_set *writeset, fd_set *exceptset, struct timeval *timeout);

/**
 * @brief:上述函数的最后一个参数，select函数要等待的时间
 * @detail:当timeout设为NULL的时候，表示select函数一直阻塞
 *         当timeout设为0的时候，则select不阻塞，直接就返回
 *         当timeout设为某个设定值，在这个时间内阻塞知道有句柄变化，如果没有，则返回0
 */
struct timeval {
    long tv_sec;   /*秒 */
    long tv_usec;  /*微秒 */
}
```

要使用select函数来监视文件描述符，首先是要将需要监视的文件描述符“set”进一个fd_set结构体中。

fd_set类型变量每一位代表了一个描述符。我们也可以认为它只是一个由很多二进制位构成的数组。



比如：描述符5和描述符1的文件需要被监视，fd为从高到低就是0001 0001（FD_SET（ ）来绑定）

```
fs_set readset;
FD_ZERO(&readset);
FD_SET(fd,&readset);
select(fd+1,&readset,NULL,NULL,NULL);
if(FD_ISSET(fd,&readset){……}
```

（基本程序流程）

备注：

#include <sys/select.h>

FD_CLR(inr fd,fd_set* set)；用来清除描述词组set中相关fd 的位
FD_ISSET(int fd,fd_set *set)；用来测试描述词组set中相关fd 的位是否为真
FD_SET（int fd,fd_set*set）；用来设置描述词组set中相关fd的位
FD_ZERO（fd_set *set）；用来清除描述词组set的全部位