【 mount/umount系统调用】

功能描述：
mount挂上文件系统，umount执行相反的操作。

用法：
```
#include <sys/mount.h>
int mount(const char *source, const char *target,
      const char *filesystemtype, unsigned long mountflags, const void *data);
int umount(const char *target);
int umount2(const char *target, int flags);
```

参数：
source：将要挂上的文件系统，通常是一个设备名。
target：文件系统所要挂在的目标目录。
filesystemtype：文件系统的类型，可以
是"ext2"，"ext3","msdos"，"proc"，"nfs"，"iso9660" 。。。
mountflags：指定文件系统的读写访问标志，可能值有以下
MS_BIND：执行bind挂载，使文件或者子目录树在文件系统内的另一个点上可视。
MS_DIRSYNC：同步目录的更新。
MS_MANDLOCK：允许在文件上执行强制锁。
MS_MOVE：移动子目录树。
MS_NOATIME：不要更新文件上的访问时间。
MS_NODEV：不允许访问设备文件。
MS_NODIRATIME：不允许更新目录上的访问时间。
MS_NOEXEC：不允许在挂上的文件系统上执行程序。
MS_NOSUID：执行程序时，不遵照set-user-ID 和 set-group-ID位。
MS_RDONLY：指定文件系统为只读。
MS_REMOUNT：重新加载文件系统。这允许你改变现存文件系统的mountflag和数据，而无需使用先卸载，再挂上文件系统的方式。
MS_SYNCHRONOUS：同步文件的更新。
MNT_FORCE：强制卸载，即使文件系统处于忙状态。
MNT_EXPIRE：将挂载点标志为过时。
data：文件系统特有的参数。

返回说明：
成功执行时，返回0。失败返回-1，errno被设为以下的某个值
EACCES：权能不足，可能原因是，路径的一部分不可搜索，或者挂载只读的文件系统时，没有指定 MS_RDONLY 标志。
EAGAIN：成功地将不处于忙状态的文件系统标志为过时。

EBUSY：一. 源文件系统已被挂上。或者不可以以只读的方式重新挂载，因为它还拥有以写方式打开的文件。二. 目标处于忙状态。
EFAULT： 内存空间访问出错。
EINVAL：操作无效，可能是源文件系统超级块无效。
ELOOP ：路径解析的过程中存在太多的符号连接。
EMFILE：无需块设备要求的情况下，无用设备表已满。
ENAMETOOLONG：路径名超出可允许的长度。
ENODEV：内核不支持某中文件系统。
ENOENT：路径名部分内容表示的目录不存在。
ENOMEM： 核心内存不足。
ENOTBLK：source不是块设备。
ENOTDIR：路径名的部分内容不是目录。
EPERM ： 调用者权能不足。
ENXIO：块主设备号超出所允许的范围。