文件描述符：内核给予我们的一个与文件进行数据交换的纽带，用户对文件的读写操作均是基于文件描述符进行操作， 本质上用户使用的文件描述符是文件描述符表的下标， [0, OPEN_MAX)

OPEN_MAX限制： 一个进程可以打开的最大文件个数，编译时与文件无关的限制， sysconf(_SC_OPEN_MAX)

默认的文件描述符： 0-标准输入， 1-标准输出， 2-标准错误

返回的是这个下标

0	文件描述符状态	指针
1	文件描述符状态	指针
2	文件描述符状态	指针

更复杂的结构

open： int open(const char *path, int oflag, ... /*mode_t mode*/);

int openat(int fd, const char *path, int oflag, ... /*mode_t mode*/);

打开或者创建一个文件, 出错，返回-1，并设置errno

oflag： 1. 必须提供其中之一的：O_RDONLY, O_WRONLY, O_RDWR, O_EXEC， O_SEARCH

2. 可以选择的特性：

描述写特性的： O_APPEND, O_TRUNC,
和创建文件有关的：O_CREAT, O_EXCL,
限定文件类型的：O_DIRECTORY,O_NOFOLLOW,
和终端有关的：O_NOCTTY, O_TTY_INIT,
文件阻塞模式的：O_NONBLOCK,
和读写操作特性相关的：O_SYNC,O_DSYNC, O_RSYNC
和exec后文件描述符状态有关的：O_CLOEXEC

文件名和路径名的长度： 在你新建文件的时候，文件的名称长度和路径名长度是有限制的，属于运行时与文件有关的限制。fsysconf。

_PC_PATH_MAX
_PC_NAME_MAX

如果在open的时候，文件名超出上面两者之一的限制，那么会出错，返回-1，并设置errno为ENAMETOOLONG，

creat：

int creat(const char *path, mode_t mode);

创建一个文件, 出错，返回-1，并设置errno

相当于 open(path, O_WRONLY|O_TRUNC|O_CREAT, mode);

close：

int close(int fd);

关闭一个文件,并释放记录锁 出错，返回-1，并设置errno
当一个进程终止时，内核会自动关闭它打开的所有文件。

相当于 open(path, O_WRONLY|O_TRUNC|O_CREAT, mode);
如果文件已经存在，就只读打开并截断。

lseek：

off_t lseek(int fd, off_t offset, int whence); //成功，返回文件偏移量，出错，返回-1

whence: SEEK_SET 、SEEK_CUR 、SEEK_END

1. 文件扩展;
2.文件长度测量；
3.随机化读写；
4.不引起任何IO操作；

read：

ssize_t read(int fd, void *buf, size_t nbytes)

读取数据,返回读取到的字节数Rn， 出错，返回-1，并设置errno

Rn = 0：

本次读取直接面对文件末尾

Rn<nbytes:

读普通文件， 再读取到指定字节数之前就到达文件末尾

读管道，管道内部字节数小于指定字节数

被信号中断

从终端、网络设备 、面向记录的设备等具有读取限制读

write：

ssize_t write(int fd, void *buf, size_t nbytes)

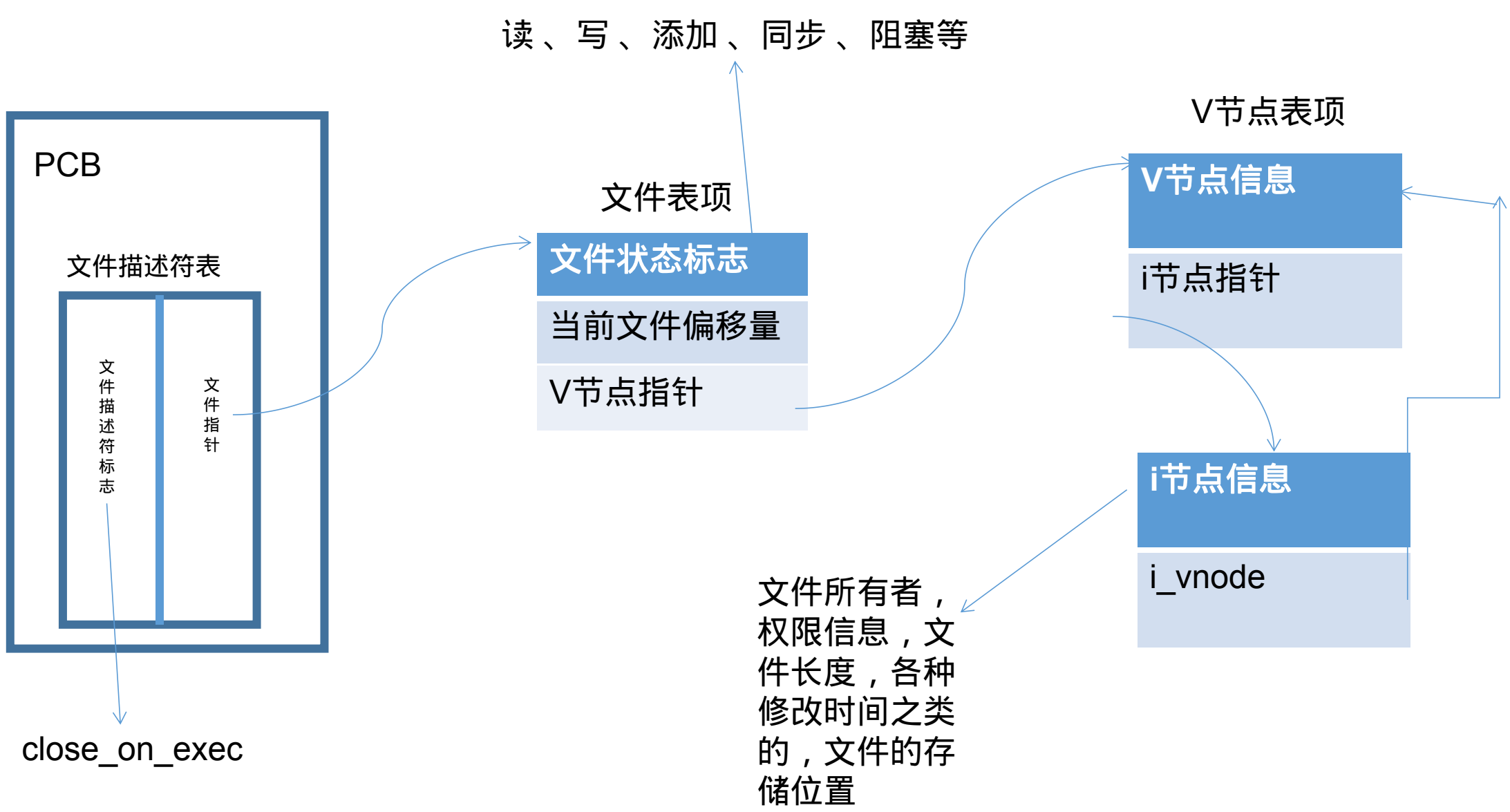
写数据,返回写入的字节数Rn， 出错，返回-1，并设置errno

Rn<nbytes:

一般理解为出错

IO的效率：　　预读技术

文件共享：


读、写、添加、同步、阻塞等
PCB
文件描述符表
文件描述符标志
文件指针
close_on_exec
文件表项
文件状态标志
当前文件偏移量
V节点指针
V节点表项
V节点信息
i节点指针
i节点信息
i_vnode
文件所有者，权限信息，文件长度，各种修改时间之类的，文件的存储位置

原子操作：　　不使用O_APPEND实现追加写操作？

1. 单进程，正常工作
2. 多进程， 时序问题？

如何解决？

```
ssize_t pread(int fd, void *buf, size_t nbytes, off_t offset);

ssize_t pwrite(int fd, const void *buf, size_t nbytes, off_t offset);
```

不会修改读写偏移量

dup和dup2：

int dup(int fd)

int dup2(int fd, int fd2)

复制文件描述符， 成功返回新的文件 描述符，失败返回-1

sync、fdatasync、fsync：

void sync(void)

将所有修改过的缓冲区排入写队列，然后返回，并不等待实际的写磁盘操作结束

int fsync(int fd)

等待写磁盘操作结束才返回，并且同步更新文件的属性

int fdatasync(int fd)

只影响文件的数据部分，并不对文件的属性更新

成功返回0，出错返回-1

fcntl：

int fcntl(int fd, int cmd, .../*int arg*/);

成功，依赖于cmd返回， 失败，返回-1

1. 复制文件描述符： F_DUPFD, F_DUPFD_CLOEXEC

2. 文件描述符标志相关：F_GETFD F_SETFD

3. 文件状态标志相关：F_GETFL F_SETFL

4. 获取设置异步IO所有权：F_GETOWN F_SETOWN

5. 记录锁(文件锁)相关： F_GETLK F_SETLK F_SETLKW

/dev/fd：

该目录下的文件为编号0，1，2等格式命名的文件

打开下面的文件等于复制文件描述符

open(“/dev/fd/0”, mode) == dup(0)

一般会忽略mode