**linux设备驱动中file\_operations结构体分析**

[**http://blog.csdn.net/sunsea1026/article/details/6586143**](http://blog.csdn.net/sunsea1026/article/details/6586143)

# struct module \*owner

第一个 file\_operations 成员根本不是一个操作; 它是一个指向拥有这个结构的模块的指针. 这个成员用来在它的操作还在被使用时阻止模块被卸载. 几乎所有时间中, 它被简单初始化为 THIS\_MODULE, 一个在 <linux/module.h> 中定义的宏.

# loff\_t (\*llseek) (struct file \*, loff\_t, int);

llseek 方法用作改变文件中的当前读/写位置, 并且新位置作为(正的)返回值. loff\_t 参数是一个"long offset", 并且就算在 32位平台上也至少 64 位宽. 错误由一个负返回值指示. 如果这个函数指针是 NULL, seek 调用会以潜在地无法预知的方式修改 file 结构中的位置计数器( 在"file 结构" 一节中描述).

# ssize\_t (\*read) (struct file \*, char \_\_user \*, size\_t, loff\_t \*);

用来从设备中获取数据. 在这个位置的一个空指针导致 read 系统调用以 -EINVAL("Invalid argument") 失败. 一个非负返回值代表了成功读取的字节数( 返回值是一个 "signed size" 类型, 常常是目标平台本地的整数类型).

# ssize\_t (\*aio\_read)(struct kiocb \*, char \_\_user \*, size\_t, loff\_t);

初始化一个异步读 -- 可能在函数返回前不结束的读操作. 如果这个方法是 NULL, 所有的操作会由 read 代替进行(同步地).

# ssize\_t (\*write) (struct file \*, const char \_\_user \*, size\_t, loff\_t \*);

发送数据给设备. 如果 NULL, -EINVAL 返回给调用 write 系统调用的程序. 如果非负, 返回值代表成功写的字节数.

# ssize\_t (\*aio\_write)(struct kiocb \*, const char \_\_user \*, size\_t, loff\_t \*);

初始化设备上的一个异步写.

# int (\*readdir) (struct file \*, void \*, filldir\_t);

对于设备文件这个成员应当为 NULL; 它用来读取目录, 并且仅对文件系统有用.

# unsigned int (\*poll) (struct file \*, struct poll\_table\_struct \*);

poll 方法是 3 个系统调用的后端: poll, epoll, 和 select, 都用作查询对一个或多个文件描述符的读或写是否会阻塞. poll 方法应当返回一个位掩码指示是否非阻塞的读或写是可能的, 并且, 可能地, 提供给内核信息用来使调用进程睡眠直到 I/O 变为可能. 如果一个驱动的 poll 方法为 NULL, 设备假定为不阻塞地可读可写.

# int (\*ioctl) (struct inode \*, struct file \*, unsigned int, unsigned long);

ioctl 系统调用提供了发出设备特定命令的方法(例如格式化软盘的一个磁道, 这不是读也不是写). 另外, 几个 ioctl 命令被内核识别而不必引用 fops 表. 如果设备不提供 ioctl 方法, 对于任何未事先定义的请求(-ENOTTY, "设备无这样的 ioctl"), 系统调用返回一个错误.

int (\*mmap) (struct file \*, struct vm\_area\_struct \*);

mmap 用来请求将设备内存映射到进程的地址空间. 如果这个方法是 NULL, mmap 系统调用返回 -ENODEV.

int (\*open) (struct inode \*, struct file \*);

尽管这常常是对设备文件进行的第一个操作, 不要求驱动声明一个对应的方法. 如果这个项是 NULL, 设备打开一直成功, 但是你的驱动不会得到通知.

int (\*flush) (struct file \*);

flush 操作在进程关闭它的设备文件描述符的拷贝时调用; 它应当执行(并且等待)设备的任何未完成的操作. 这个必须不要和用户查询请求的 fsync 操作混淆了. 当前, flush 在很少驱动中使用; SCSI 磁带驱动使用它, 例如, 为确保所有写的数据在设备关闭前写到磁带上. 如果 flush 为 NULL, 内核简单地忽略用户应用程序的请求.

int (\*release) (struct inode \*, struct file \*);

在文件结构被释放时引用这个操作. 如同 open, release 可以为 NULL.

int (\*fsync) (struct file \*, struct dentry \*, int);

这个方法是 fsync 系统调用的后端, 用户调用来刷新任何挂着的数据. 如果这个指针是 NULL, 系统调用返回 -EINVAL.

int (\*aio\_fsync)(struct kiocb \*, int);

这是 fsync 方法的异步版本.

int (\*fasync) (int, struct file \*, int);

这个操作用来通知设备它的 FASYNC 标志的改变. 异步通知是一个高级的主题, 在第 6 章中描述. 这个成员可以是NULL 如果驱动不支持异步通知.

int (\*lock) (struct file \*, int, struct file\_lock \*);

lock 方法用来实现文件加锁; 加锁对常规文件是必不可少的特性, 但是设备驱动几乎从不实现它.

ssize\_t (\*readv) (struct file \*, const struct iovec \*, unsigned long, loff\_t \*);

ssize\_t (\*writev) (struct file \*, const struct iovec \*, unsigned long, loff\_t \*);

这些方法实现发散/汇聚读和写操作. 应用程序偶尔需要做一个包含多个内存区的单个读或写操作; 这些系统调用允许它们这样做而不必对数据进行额外拷贝. 如果这些函数指针为 NULL, read 和 write 方法被调用( 可能多于一次 ).

ssize\_t (\*sendfile)(struct file \*, loff\_t \*, size\_t, read\_actor\_t, void \*);

这个方法实现 sendfile 系统调用的读, 使用最少的拷贝从一个文件描述符搬移数据到另一个. 例如, 它被一个需要发送文件内容到一个网络连接的 web 服务器使用. 设备驱动常常使 sendfile 为 NULL.

ssize\_t (\*sendpage) (struct file \*, struct page \*, int, size\_t, loff\_t \*, int);

sendpage 是 sendfile 的另一半; 它由内核调用来发送数据, 一次一页, 到对应的文件. 设备驱动实际上不实现 sendpage.

unsigned long (\*get\_unmapped\_area)(struct file \*, unsigned long, unsigned long, unsigned long, unsigned long);

这个方法的目的是在进程的地址空间找一个合适的位置来映射在底层设备上的内存段中. 这个任务通常由内存管理代码进行; 这个方法存在为了使驱动能强制特殊设备可能有的任何的对齐请求. 大部分驱动可以置这个方法为 NULL.[[10](http://oss.org.cn/kernel-book/ldd3/ch03s03.html" \l "ftn.id415066)]

int (\*check\_flags)(int)

这个方法允许模块检查传递给 fnctl(F\_SETFL...) 调用的标志.

int (\*dir\_notify)(struct file \*, unsigned long);

这个方法在应用程序使用 fcntl 来请求目录改变通知时调用. 只对文件系统有用; 驱动不需要实现 dir\_notify.

scull 设备驱动只实现最重要的设备方法. 它的 file\_operations 结构是如下初始化的:

struct file\_operations scull\_fops = {

.owner = THIS\_MODULE,

.llseek = scull\_llseek,

.read = scull\_read,

.write = scull\_write,

.ioctl = scull\_ioctl,

.open = scull\_open,

.release = scull\_release,

};