	char * path2)	件名改为 path2 所在路径的文件名
dfs_getfsize	long dfs_getfsize(int fd)	获得一个指定文件的长度
dfs_getfoffset	long dfs_getfoffset(int fd)	获得当前文件的偏移量
dfs_splitpath	int fs_splitpath(char * fd,int * drive,char * dir,char * name,char * ext);	将该完整的文件描述符 fd，分解成盘符 drive；路径 dir；文件名 name；扩展名 ext。
dfs_getcufsize	long dfs_getcufsize(int fd)	获得一个文件当前文件的长度
dfs_setfsize	int dfs_setfsize(int fd, long size)	设置一个指定文件的长度
dfs_write	int dfs_write(int fd, void * buffer, unsigned int count)	函数把 count 个字节从 buffer 所指向的缓冲区写入到 fd 指向的文件中
dfs_lseek	long dfs_lseek(int fd, long foffset, int origin)	把 fd 文件描述符所指向的指针定位到 foffset 和 origin 所指定的位置
dfs_getfattr	int dfs_getfattr(char *name,char attrp)	获得一个指定文件的属性
dfs_setfattr	int dfs_setfattr(char *name,char attrp)	设置一个指定文件的属性
dfs_rmdir	int dfs_rmdir(char * path)	删除由 path 所指定的路径名下的目录
dfs_mkdir	int dfs_mkdir(char * dir)	在 dir 所指向的路径名下建立一个新的目录

5.2.2 实现举例

到将上面提到的函数实现去描述清楚无疑是一件繁琐的事情。在这一小节中，采用自顶向下的描述方法，扼要的介绍文件系统中最典型的调用：dfs_read。

5.2.2.1 dfs_read

算法：dfs_read

功能：

函数从 fd（文件描述符）所指向的文件中读取长度为 count 的数据内容，并放到 buffer 所指向的目的缓冲区去

输入参数：

文件描述符 path

目的缓冲区 buffer

要读取的字节数 count

输出参数：

成功：返回所读数据长度

失败：

返回值	说明
DE_INVLDHNDL（-6）	无效文件
DE_INVLDACC（-12）	拒绝访问
DE_SEEK(-25)	文件指针定位出错
DE_BLKINVLD(-20)	无效块
0	可读数据块为 0 或已读到文件尾

算法描述：

{

fnp = xlt_fd(fd); /*文件描述符本身是一个序号，需要转换成它在文件描述符数组中的对应结构*/

检查该文件描述符是否有效；

检查 count 数是否非 0；

检查是否已是文件尾；

检查文件是否为只读或读写模式；

```
    while (还有数据没读完)
        if (是文件开始)
        {
            初始化文件的开始簇;
            将所有的偏移量都设置为 0;
        }else
        map_cluster 并根据其结果进行相应的处理; /*map_cluster 实现在簇链中
                                                搜索目录文件*/

        计算该读写块在簇中的扇区号和偏移量;
        进行文件尾 EOF 检查;
        设置使用该文件的设备;
        get_buffer; /*从高速缓冲区中获得需要的块*/
        检查是否获得块成功;
        将高速缓冲中的块读到指定的缓冲区中;
        修改指针;
        }
    release_buffer; /*释放掉申请的缓冲区*/
    return ret_cnt; /*返回已读字节数*/
}
```