# 第一章 测试存储器框架

dos\_test（存储器设备描述结构体）先定义需要读取的文件名fn[ ]，调用函数part\_get\_info\_dos（）获取并打印存储器分区的信息，调用函数fat\_set\_blk\_dev（）进行识别，如大容量存储器不是文件格式就退出，否则调用函数fat\_exists（）再判断存储器中是否存在需要读取的文件，如果找到该文件的话调用函数file\_fat\_read（）读取内容并显示。

# 第二章 获取并打印存储器分区信息

本章的章节结构：

1

2.1

2.2

3.1

3.2

3.3

☆

☆

1 part\_get\_info\_dos（块存储器设备描述信息，分区号，磁盘分区）首先调用part\_get\_info\_extended（）函数获取存储器信息设置通用类型名字，如果获取成功的话调用part\_print\_dos（）函数打印获取到的信息。

2.1 part\_get\_info\_extended（USB设备，扩展分区，相对性，分区号，哪一个分区，磁盘签名）获取扩展分区信息。调用函数blk\_dread（☆）将块存储器分区表等信息读到缓存区，并在读取之后检查扇区签名。

将指针pt指向数据缓存区中存放的分区表表头，即指向分区表中第一个分区。遍历分区表先确保该分区不是拓展分区，然后调用函数part\_set\_generic\_name（）等设置所有主/逻辑分区信息。

重新将指针pt指向数据缓存区中存放的分区表表头。再遍历分区采用递归调用part\_get\_info\_extended（）函数的方法设置拓展分区的信息。

如果一直未找到分区就调用test\_block\_type（）函数检查。

2.2 part\_print\_dos（块存储器设备描述信息），只调用了print\_partition\_extended（块存储器设备描述信息，扩展分区，相对性，分区号，磁盘签名）来进行进一步配置，指定了无扩展分区、分区号为1以及磁盘签名为0。调用blk\_dread（☆）读取块存储器分区表，调用test\_block\_type（）确保分区类型是DOS\_MBR，并设置磁盘签名。

循环调用print\_one\_part（）函数，遍历打印所有主分区以及逻辑分区。再递归调用print\_partition\_extended（）函数打印扩展分区。

3.1 part\_set\_generic\_name（块存储器设备描述信息，分区号，名字）根据块存储器设备描述信息中的接口类型dev\_desc->if\_type，识别出设备类型devtype。

3.2 test\_block\_type（数据缓存区）首先检查有没有DOS签名，其次检查引导指示灯是否有效并计算分区数。如果分区表无效或为空，请检查这是否是 DOS PBR。最后检查是不是DOS MBR。

3.3 print\_one\_part（分区指针，拓展分区，分区号，磁盘）打印磁盘信息：分区号、起始扇区、扇区数量、UUID以及类型。

# 第三章 判别块存储器设备

fat\_set\_blk\_dev（块存储器设备描述信息，磁盘分区）首先调用disk\_read（☆）函数读取磁盘并判断该存储设备具备有效的文件配置表（FAT）标头，再判断其是不是一个磁盘操作系统（DOS）卷，最后检查文件系统格式是不是FAT。

# 第四章 搜索目标文件

本章的章节结构：

1

2.1

2.2

3.1

3.2

3.3

3.4

4.1

☆

☆

★

经过第三章的判别之后，我们确定了此磁盘符合读取条件。故在磁盘中搜索目标文件，之后的过程涉及文件系统知识。

1 fat\_set\_blk\_dev（文件名称字符串）给迭代器itr分配一段空间记录文件结构，调用fat\_itr\_root（）函数将迭代器初始化为从根目录开始，调用fat\_itr\_resolve（）函数遍历目录结构解析请求路径，最后调用aligned\_free（★）函数释放文件数据缓存区和迭代器itr。

2.1 fat\_itr\_root（迭代器，文件数据）将迭代器初始化为从根目录开始。先调用get\_fs\_info（）函数获取分区的文件系统数据，初始化迭代器的分区文件系统数据、开始集群、当前集群、下一个集群、当前目录项、当前集群中剩余的凹痕、读取了最后一个集群的标志以及根目录的迭代器标志。

2.2 fat\_itr\_resolve（迭代器，文件名字[文件路径]，允许的文件类型的位掩码）遍历目录结构解析请求路径。首先移除文件路径中额外的前导斜杠，设置下一个需要查找的路径next。

先判断迭代器是否在根目录，如果是的话用下一路径next递归调用函数fat\_itr\_resolve（递归）。如果迭代器不在根目录，用while循环调用函数fat\_itr\_next（）跳转到目录中的下一个条目，检查路径名称并用函数fat\_itr\_isdir（）核验迭代器是不是在文件夹中，调用fat\_itr\_child（）初始化一个迭代器进入子目录并递归调用fat\_itr\_resolve（递归）函数。

3.1 get\_fs\_info（分区的文件系统数据）又调用了read\_bootsectandvi（）函数从 FAT 文件系统读取引导扇区和卷信息。根据文件配置表大小是否等于32来判断是不是FAT32格式，并配置相应的文件配置表长度以及总扇区。加载文件配置表和fat扇区以及根目录扇区，设置扇区大小、根目录扇区、根目录大小、fat缓存区数量以及fat缓存区等。

3.2 fat\_itr\_next（迭代器）跳转到下一个条目，成功返回1没有更多条目返回0。

3.3 fat\_itr\_isdir（迭代器）如果迭代器的当前目录项属性是文件夹，就代表迭代器遍历到文件夹中返回true否则返回false。

3.4 fat\_itr\_child（迭代器，父迭代器）初始化一个迭代器以下降到子目录。

4.1 read\_bootsectandvi（引导扇区，卷信息，文件配置表大小）从 FAT 文件系统读取引导扇区和卷信息。调用disk\_read（★）函数获取引导扇区数据并拷贝到boot\_sector\_t之中进行数据转换，根据引导扇区文件配置表的大小判断FAT12/FAT16/FAT32并配置引导扇区长度、标志位、根集群、扇区信息以及备份启动。

总结：有一个迭代器来遍历文件夹查找路径，该迭代器从根目录开始一共有三个主要动作以及几个辅助动作结合完成。主要动作有：①跳转到下一个条目；②初始化一个新的迭代器以下降到子目录；③递归调用函数实现遍历所有文件夹。辅助动作例如判断迭代器下一个条目是否为空、判断当前路径是否符合规范以及判断迭代器进入的是不是文件夹等。

# 第五章 读取目标文件

本章的章节结构：本章为从顶层函数到底层函数的顺序结构，读者依次阅读即可。

1 file\_fat\_read（文件名，数据缓存区，最大读取大小），直接调用函数file\_fat\_read\_at（）进行进一步配置。

2 file\_fat\_read\_at（文件名，阅读位置，缓存区，读取最大字节数，实际读取的字节数）创建迭代器itr，调用fat\_itr\_root（☆）函数将迭代器初始化为从根目录开始，调用函数fat\_itr\_resolve（☆）遍历目录结构解析请求路径，调用函数get\_contents（）获取文件内容，释放数据缓存区和迭代器。

3 get\_contents（分区文件系统数据，目录项指针，阅读位置，缓存区，读取最大字节数，实际读取的字节数），调用get\_fatent（☆）函数获取条目，调用get\_cluster（）函数将数据读取到临时缓存区tmp\_buffer，经检验后拷贝到数据缓存区buffer并释放临时缓存区，实际接收数据少于期望数据就一直交替使用上述函数继续接收。

4 get\_cluster（）该函数下有一系列磁盘读取和数据拷贝工作。