# 文件系统基础设施

对象：面向一切可存储数据的对象

比如：内存（虚拟内存文件系统）磁盘（传统文件系统） 软盘 虚拟磁盘 网络设备

除非必须针对特定设备设计特定的的文件系统，否则一个文件系统不能假设是基于哪一种具体设备。

它也不必知道特殊机制（如缓存机制）存在。

# 文件系统基础结构

文件系统使用metadata（元数据）存储文件的信息，基础的元数据包括：

1. 创建时间
2. 修改时间
3. 文件大小

基础操作：

创建文件

创建文件夹

修改

# EXT4文件系统

将存储器视为逻辑块（logical block）的数组，通常，逻辑块大小为4KB。（介于1~64KB，且为2的整数幂），块的大小通常在mkfs时指定，block的索引是32位，所以最多支持2^32个块寻址，如果”64bit”被启用，则可以使用2^64。

而逻辑块通常也被分成逻辑块组，一个文件通常被尽量分配在一个组内。通常一个组的默认大小为128MB。

ext4文件系统支持动态增长，增长大小为1024的倍数。

每个block group的典型布局如下。



GDT是为了以后扩展文件系统使用的，第0组的前1024个字节通常不用，保留给其他启动程序使用。只有超级快、组描述符的顺序一定处在开头，其他4个组的顺序可以不同。

# 开发一个小型文件系统:x2fs

基本思想：沿用内存管理使用的技术。

文件体系无论如何总是构成一棵树，根就是根目录，名称为/。

分配一个空间用于管理树结构的节点，这些节点的空间是固定的（如果有不固定的项，比如文件名，可以引用一个地址，那个空间的内容是不定长的，可以使用链表管理，但是情况有所简化，内存只分为两类：已经分配的和未分配的）。

磁盘结构：（扇区作为单位）

0 - 1： 保留

2 - 5： 文件名

6 – 20 ： 文件结构表（目录结构表,第一个节点必须是根，其他节点由根寻找）

21 - ： 文件内容存放区

因为需要经常用到文件名，所以文件名最好统一读入。文件名以0标志结束。

文件结构表项构成： 文件类型（目录，文件） 文件位置（对目录无效，指向文件所占的区域起始） 文件名指针 创建时间 修改时间 文件内容起始（相对于 文件位置的正偏移，前面的区域可能用于多余的属性）。

//实现和验证

# 参考

ext4 documentation：<https://ext4.wiki.kernel.org/index.php/Ext4_Disk_Layout>

<https://www.kernel.org/doc/Documentation/filesystems/ext4.txt> 来自linux官方