第八次作业

10.1 Consider a file system where a file can be deleted and its disk space reclaimed while links to that file still exist. What problems may occur if a new file is created in the same storage area or with the same absolute path name? How can these problems be avoided?

当文件被删除后，原本指向该文件的链接会变成“悬挂链接”或“死链接”。如果在同一存储区域创建了一个新的文件，而这一新文件与之前的文件有相同的绝对路径名，那么指向原文件的链接将指向新创建的文件，这会导致错误读取的问题。

为解决该问题，可以采取以下方法：

①引用计数机制：在文件系统中使用引用计数，仅当所有链接被删除且引用计数为零时，才允许删除文件并释放空间。这种方式确保文件不会在有指向它的链接存在时被删除。

②分配新的 inode：文件系统可以在创建新文件时分配一个新的 inode（索引节点），以确保路径相同但文件内容不同的情况下不会出现指向错误文件的情况。这样，即使路径相同，新旧文件也会有不同的 inode 编号，从而避免数据混淆。

③软链接检查：当创建新文件时，可以检查系统中是否存在指向该路径的软链接，并通知用户链接失效。这可以减少误操作的可能性。

11.1 Consider a file system that uses a modified contiguous-allocation scheme with support for extents. A file is a collection of extents, with each extent corresponding to a contiguous set of blocks. A key issue in such system is the degree of variability in the size of the extents. What are the advantages and disadvantages of the following schemes?

a. **All extents are of the same size, and the size is predetermined.**
b. **Extents can be of any size and are allocated dynamically.**
c. **Extents can be of a few fixed sizes, and these sizes are predetermined.**

a.

优点：

①管理简单：由于所有 extent 大小相同，系统只需追踪固定大小的块，管理和

实现都比较简单。
②空间碎片化减少：相同大小的 extent 可以减少外部碎片，因为每个文件的扩展都是相同大小的块的简单追加。
③定位快速：文件的扩展不涉及大小变化，文件系统可以更快地找到文件的每个 extent。
缺点：
①空间利用效率低：文件大小不总是 extent 大小的倍数，因此可能会导致内部碎片——文件尾部的空间浪费。
②灵活性差：固定大小的 extent 无法适应大小变化较大的文件需求，特别是对于小文件，可能会造成大量空间浪费。
③扩展困难：当文件需要更多空间时，必须在其他位置分配新的相同大小的 extent，这可能增加文件的分散度，影响读写效率。

b.

优点：
①空间利用率高：每个 extent 的大小可以根据文件的实际需求动态分配，从而减少内部碎片，提高磁盘利用率。
②灵活性高：适合不同大小的文件和使用场景，无论是小文件还是大文件都可以灵活分配，特别适合存储大小变化较大的文件。
③减少文件分散：通过动态分配大的 extent，可以减少分散度，使文件可以更连续地存储，优化读写性能。
缺点：
①管理复杂：文件系统需要记录每个 extent 的大小和位置，增加了管理的复杂性。
②外部碎片化：由于 extent 大小不固定，随着文件增删，会产生外部碎片化的问题，降低整体性能。

c.

优点：
①平衡灵活性和管理复杂度：预定的几种固定大小可以兼顾小文件和大文件需求，既有灵活性又能保持一定的管理简便性。
②减少外碎片：尽管不是完全相同大小的 extent，但可选择的固定大小可以减少外部碎片化，并且比动态大小的方案更易管理。
③提高空间利用率：多种大小选择可以适应不同大小的文件需求，减少内部碎片，相对于固定大小的 extent 更高效。
缺点：
①实现较为复杂：系统需要管理多种大小的 extent，因此在设计和实现上要考虑不同大小的分配和回收策略，增加了系统复杂性。
②潜在碎片化：当文件大小不符合 extent 选项时，可能会选择大于所需的 extent，导致一些空间浪费。

11.2 What are the advantages of the variant of linked allocation that uses a FAT to chain together the blocks of a file?

①提高磁盘空间利用率、避免外部碎片：FAT 表允许文件块在磁盘上非连续存储，因此不需要找到连续的物理空间，相比连续分配方案，FAT 避免了因连续空间不足而无法分配空间的情况，从而有效减少了外部碎片，提高了磁盘空间的利用率。
②方便文件插入和删除、支持文件动态扩充：在 FAT 结构中，文件的每个块位置仅需要通过表中的链式结构指向下一个块，因此插入新块或删除现有块时，只需更新表中的指针关系，不涉及物理空间的重新分配或移动，这种方式大大简化了管理，特别适合需要频繁扩展、修改的文件。
③内存缓存支持快速访问：FAT 表可以被加载到内存中，以便文件系统快速访问文件中的任意块。对于中间块的访问，直接从内存中的 FAT 表定位到所需块，无需在磁盘上进行多次顺序查找。这样不仅减少了 I/O 操作的次数，还显著提升了文件系统的响应速度。
④快速直接存取任意文件块：由于 FAT 表保存了每个块的链表结构信息，文件系统可以通过表中的指针快速找到目标块的物理位置，实现对任意文件块的快速直接存取。

11.3 Consider a system where free space is kept in a free-space list.

a. **Consider a file system similar to the one used by UNIX with indexed allocation. How many disk I/O operations might be required to read the contents of a small local file at /a/b/c? Assume that none of the disk blocks is currently being cached.**
b. **Suggest a scheme to ensure that the pointer to the free space list is never lost as a result of memory failure.**

a.读取本地小文件/a/b/c 需要 4 个磁盘 I/O 操作：
①读取根目录：读取根目录来查找目录 a，1 次 I/O 操作。
②读取子目录的索引块：对于路径/a/b/c，需要分别读取目录 a 和 b 的索引块。每读取一个目录，便需要 1 次 I/O 操作：
Ⅰ读取目录 a 的索引块，1 次 I/O 操作。
Ⅱ读取目录 b 的索引块，1 次 I/O 操作。
③读取文件内容块：找到文件 c 后，直接读取该文件的内容块，1 次 I/O 操作。
b.为了保证空闲空间列表的指针不会因内存故障而丢失，可以采取以下几种方案：
①使用超级块存储指针：在磁盘的超级块中保留空闲空间列表指针的位置，并在文件系统启动时将超级块内容加载到内存中，如果内存发生故障，可以重新读取超级块中的指针以恢复。
②多重超级块备份：可以在磁盘上多个位置备份超级块，即使超级块的主备份丢失或损坏，文件系统仍可以从其他位置恢复空闲空间列表的指针。
③定期写入更新到磁盘：文件系统可以在空闲空间列表发生变化（例如文件删除、创建时）时，将更新的指针定期写入磁盘。