# 文件系统的可靠性

可靠性：

抵御和预防各种物理性破坏和人为性破坏的能力

- 坏块问题
- 备份

通过转储操作，形成文件或文件系统的多个副

可以对文件形成多个副本，也可以对整个文件系统形成多个副本。

# 文件系统备份

**全量转储：**

定期将所有文件拷贝到后援存储器

**增量转储：**

只转储修改过的文件，即两次备份之间的修改，减少系统开销

**物理转储：**

从磁盘第0块开始，将所有磁盘块按序输出到磁带

**逻辑转储：**

从一个或几个指定目录开始，递归地转储自给定日期后所有更改的文件和目录

保存在一个新的介质上，这就是两种不同的转储的实现。

# 文件系统一致性

**问题的产生：**

磁盘块 → 内存 → 写回磁盘块

若在写回之前，系统崩溃，则文件系统出现不一致

**解决方案：**

设计一个实用程序，当系统再次启动时，运行该程序，检查磁盘块和目录系统

然后再对目录系统进行检查，下面我们以前一个对磁盘块进行检查为例

# 磁盘块的一致性检查

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 使用中的块 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 空闲块 |

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 使用中的块 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 空闲块 |

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 使用中的块 |
| 0 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 空闲块 |

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 2 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | |

UNIX一致性检查工作过程：

两张表，每块对应一个表中的计数器，初值为0

表一：记录了每块在文件中出现的次数

表二：记录了每块在空闲块表中出现的次数

# 文件系统的写入策略

- 通写（write-through）
  内存中的修改立即写到磁盘
  缺点：速度性能差
  例： FAT文件系统
- 延迟写（lazy-write）
  利用回写（write back）缓存的方法得到高速
  可恢复性差
- 可恢复写（transaction log）
  采用事务日志来实现文件系统的写入
  既考虑安[illegible]
  例： NTFS

考虑文件系统一致性和速度

当然了还有 Linux 的 EXT3 这样的一些文件系统。

0:00 下面我们介绍文件系统的管理存在的一些问题以及解决方案。 第一个是文件系统的可靠性问题。 所谓文件系统可靠性就是指的是这个文件系统能够抵御 或者是预防各种物理性的破坏， 或者人为性的破坏这样的能力。 一个磁盘在出厂之后呢会有一些坏块， 通常呢会把这些坏块呢把它都链接起来，标记出来。 在磁盘的使用过程中呢还会出现新的坏块。 所以坏块的处理呢虽然不是大事，但是呢 我们还要考虑到需要对它进行相应的处理。 通常的一种手段呢就是把所有的坏 块集中在一个文件，这个文件呢叫坏块文件。 就把这个坏块分配给这个文件，而这个文件呢 没有人去使用它，是系统啊创建的这么一个文件。 这就是坏块，可以这样去解决。 那么关于 可靠性呢，还有就是说如果会出现问题那我们是不是应该提前做一些准备啊？ 那么这就是备份，所谓备份就是通过一个转储的操作， 对这个文件形成它的多个副本。 可以对文件形成多个副本，也可以对整个文件系统形成多个副本。 通常文件系统的转储或者备份呢， 从两方面来考虑，第一个方面考虑呢是说 内容上，那么每次转储的内容是整个所有的文件， 还是每次转储只是保存那些 自上一次转储之后的一个修改过的文件。 这是两种不同的策略，当然了一般情况下呢 全量转储因为花费的时间比较长， 所以呢定期做，那么增量转储呢是在两个全量 转储之间呢，我们可以增加若干个增量转储。 但是它们俩的这个不同点呢是说，当你 恢复的时候，转储完之后如果系统出现问题我重新恢复的时候呢， 那么全量转储呢是容易恢复，整个重新恢复就可以了。 而增量转储呢要进行什么呀，迭代的这种恢复啊，先把第一次的 恢复了以后，在这基础上每次 delta 的修改增加 delta 的修改增加，所以呢花费的时间比较长。 另外我们要考虑在转储的这个过程当中 怎么样去实现转储呢，通常呢有物理转储呢和逻辑转储。 所谓物理转储就是按照磁盘，它从 0 块开始 将所有磁盘块一一复制到一个新的介质上，比如说磁带上。 完全是磁盘块对磁盘块的这样一个转储复制工作。 而逻辑转储呢是按一个目录结构 来保存信息的，也就是说可以从一个或者指定的几个目录开始。 然后递归的去把这个树形结构 保存在一个新的介质上，这就是两种不同的转储的实现。 下面呢我们来探讨 文件系统一致性这个概念，所谓文件系统一致性呢， 我们来看一下它呢指的是这样一个场景。 当文件系统在对它进行相应操作的时候呢， 实际上是要把磁盘块先读入内存，然后在内存里进行相应的操作。 操作完成之后呢，需要把这些修改的内容呢要写回到磁盘。 那这是一个完整的过程，先把磁盘块读入内存， 然后最后要把它写回到磁盘，如果在这个完整的过程当中， 在磁盘块没有写回磁盘之前，系统出现了故障，系统崩溃了， 那么这个时候呢你要

该写的内容没有写回去，那么文件系统就会出现了不一致。啊不一致，当然我们这里头强调一下 文件系统的不一致主要是指源数据的不一致。因为我们知道系统的文件卷当中有系统的卷的信息。有哪些空闲的磁盘块，还有这个文件的目录，还有 分给文件的啊这样这样一些块的信息，记录在了这个 FCB 里头，那么这些内容呢 如果出现了不一致，那么就导致这个文件系统的这个 出现了问题，所以我们所强调的文件系统 的一致性指的是文件系统源数据的一致性。那至于说，你的一个 word 文件 在使用过程中没有保存，没有存盘，然后你呢，想说我把它这个 如果系统崩溃了我把它能恢复，那么这个呢是不在文件系统一致性 这个范畴里讨论，它是属于其他的这个相关的这个功能。那么怎么来解决文件系统一致性的这个检查呢，就是如果系统崩溃了 之后重新引导系统之后通常会花一定的时间进行相应的检查。其中一部分检查工作呢就要花费在文件系统一致性的检查上。具体的方案呢就是说我们设计一个实用程序，这个实用程序当系统再次启动的时候去执行它，通过去检查磁盘块检查目录系统，来确保文件系统的一致性。啊当然这里头有两方面，先对磁盘块进行检查，然后再对目录系统进行检查，下面我们以前一个对磁盘块进行检查为例 来介绍啊对文件系统一致性的这样一个相关的这个 检查和修复工作。下面我们来介绍一下磁盘块的一致性检查。在 UNIX 文件系统当中呢，针对磁盘块的一致性检查呢设计了两张表。每个表项呢实际上就对应了磁盘上的一个磁盘块。第一张表呢是记录了这个磁盘块 在文件当中出现的次数。第二张表呢是记录了这个磁盘块 在空闲区表当中出现的次数。因此呢当重新引导系统之后呢，那么就运行这个实用程序，这个实用程序呢就去检查空闲区 表，去检查每一个文件，然后把 这个文件对应的块儿在这个表里头进行相应的计数。那么没有分出去的空闲块儿呢也在 另外一张表里进行相应的计数。等到把所有的文件系统都查完之后，也就是把所有的空闲块查了一遍，所有的 文件查了一遍之后，这两张表就得到了不同的结果。那我们来看一下，可能的结果有四种。第一种呢实际上是一个一致性的结果，也就是说 某一个磁盘块要么它是在 使用中分配给了某个文件，要么呢它是在空闲块表当中出现，所以呢它是一个空闲块。所以如果上面是 0，下面一定是 1。啊如果下面是 0，上面就一定是 1 ，所以它是一致的。但是也会出现一些不一致情况，我们来看一下。在这个结果当中，我们可以看到，有这样一个磁盘块，既没有在空闲区，这个表里头找到它，也没有分配给某个文件，那么这一块儿就丢失了，

在整个数据结构中找不到它了。 那么怎么解决呢，很好办，那么就是要在 这个空闲区表中把这个块标记成空闲的就可以了。 我们再来看看这种情况， 这种情况是说某一个磁盘块在 第二张表中出现了两次，那么这个呢也不合理。 那么如果出现这种情况就把这个 2 变成 1 就可以了，所以呢要修改一下这个 空闲区表，空闲区表。 那么最后一种情况呢比较复杂一点。 我们可以看到某一个磁盘块，比如说块 5， 它在两个文件当中都出现了。 那么如果出现了这种情况，我们只能够在 空闲的磁盘块中找一个新的磁盘块，然后把相应的内容拷贝过去， 分配给其中一个文件，而另一个文件呢用原来的磁盘块。 那么这就是属于磁盘块的一致性。 除了磁盘块的一致性，我们还知道还有目录系统的一致性。 相关的内容呢，大家可以看教材上所介绍的内容。 下面我们来探讨一下文件系统的写入策略。 文件系统的内容要读入内存， 经过了相应的处理再写回到磁盘，我们就谈什么时候写回去，这就是 我们下面要讨论的写入策略，在考虑这个策略的时候呢第一要考虑文件系统的一致性。 因为如果我不及时写回去，如果系统出现问题，那么 实际情况和我系统的数据就不相符。 就出现了文件系统的一致性问题。 但是呢我们还要考虑另外一个 目标就是速度，性能，如果频繁地往磁盘上写， 势必会降低文件系统的性能，所以我们要考虑两个因素，进行折中权衡。 第一种写入策略呢我们叫做通写，直写， write-through ，它指的是内存的修改 立即反馈到磁盘，也就是所有的修改 只要在内存改了，立刻写回到磁盘。 完全保持一致性，它的缺点 很显然，它的性能会下降，因为频繁地写盘。 但是呢有些操作系统是这么做的，像 FAT 文件系统。 就是每次修改的内容立刻反馈到磁盘上，第二种 写入策略呢叫延迟写 lazy-write, 它是呢利用回写缓存的方法 来得到性能上的提高。 也就是说对内存的修改呢把它写到一个缓存里头， 定期将缓存里的内容 写回到磁盘，当然这种写入策略呢，它的可恢复性略差。 因为我们是定期写回，所以在两次 写回之间如果出现了系统的故障 就会导致一些信息的丢失。 就可能造成文件系统的不一致性。 但是它的优点呢是对性能上， 有所提高，因为它不有每次修改都往回写。 它是积攒了一段时间统一的写回去一次。 第三种写入策略叫做可恢复写， 那么它通常是利用了事务 日志这样的一个手段来实现文件系统的写入。 这种日志文件系统呢既考虑了安全性， 也考虑了速度和性能的问题。 这个典型的代表呢是 Windows 的 NTFS。 当然了还有 Linux 的 EXT3 这样的一些文件系统。