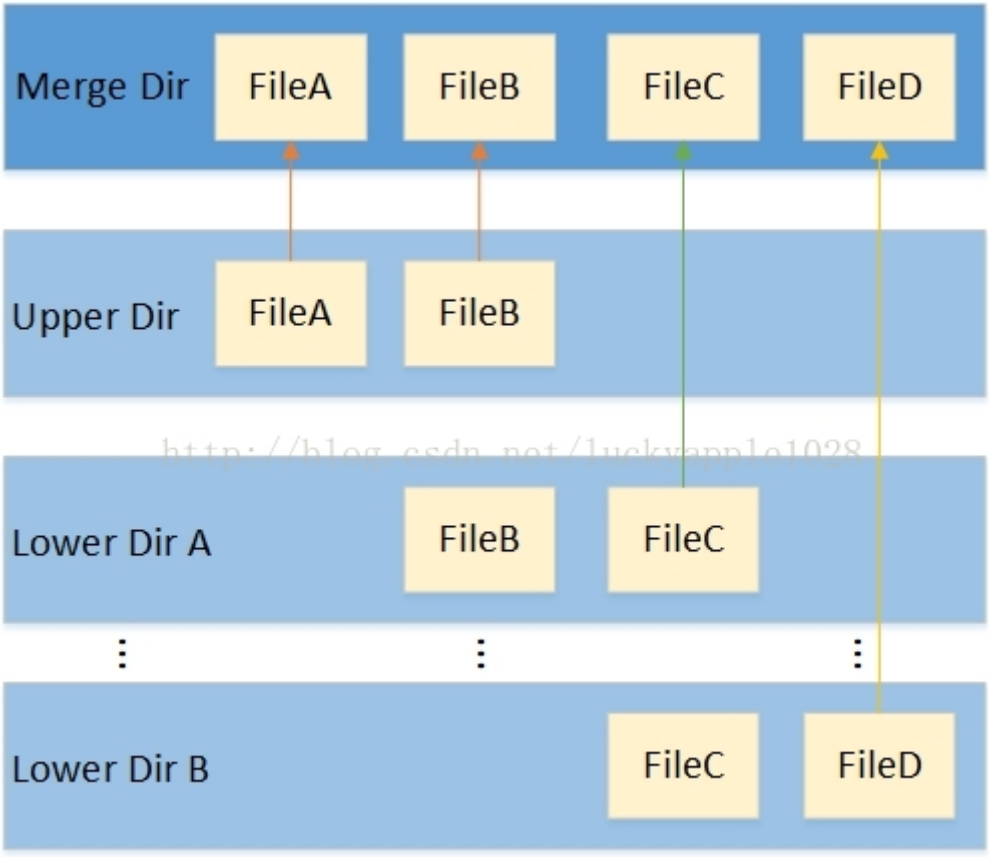
深入理解overlayfs（一） - 星火撩原 - 博客园

深入理解overlayfs（一）

Overlayfs是一种类似aufs的一种堆叠文件系统，于2014年正式合入Linux-3.18主线内核，目前其功能已经基本稳定（虽然还存在一些特性尚未实现）且被逐渐推广，特别在容器技术中更是势头难挡。本系列博文将首先介绍overlayfs的基本概念和应用场景，然后通过若干实例描述它的使用方式，最后从源码角度结合Linux VFS Layer和Ext4fs连通分析overlayfs的实现。本文先来大致认识一下什么是Overlayfs，它有什么应用场景和使用限制。

Overlayfs概述
基本概念
Overlayfs是一种堆叠文件系统，它依赖并建立在其它的文件系统之上（例如ext4fs和xfs等等），并不直接参与磁盘空间结构的划分，仅仅将原来底层文件系统中不同的目录进行“合并”，然后向用户呈现。因此对于用户来说，它所见到的overlay文件系统根目录下的内容就来自挂载时所指定的不同目录的“合集”。
见图1。



其中lower dirA / lower dirB目录和upper dir目录为来自底层文件系统的不同目录，用户可以自行指定，内部包含了用户想要合并的文件和目录，merge dir目录为挂载点。当文件系统挂载后，在merge目录下将会同时看到来自各lower和upper目录下的内容，并且用户也无法（无需）感知这些文件分别哪些来自lower dir，哪些来自upper dir，用户看见的只是一个普通的文件系统根目录而已（lower dir可以有多个也可以只有一个）。

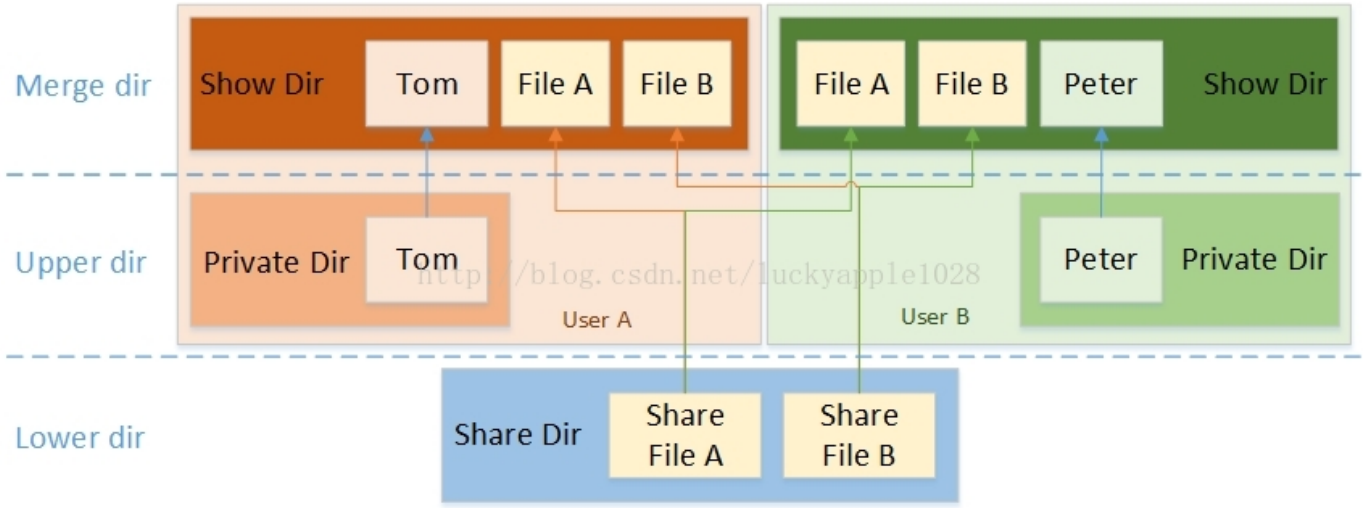
虽然overlayfs将不同的各层目录进行合并，但是upper dir和各lower dir这几个不同的目录并不完全等价，存在层次关系。首先当upper dir和lower dir两个目录存在同名文件时，lower dir的文件将会被隐藏，用户只能看见来自upper dir的文件，然后各个lower dir也存在相同的层次关系，较上层屏蔽叫下层的同名文件。除此之外，如果存在同名的目录，那就继续合并（lower dir和upper dir合并到挂载点目录其实就是合并一个典型的例子）。

各层目录中的upper dir是可读写的目录，当用户通过merge dir向其中一个来自upper dir的文件写入数据时，那数据将直接写入upper dir下原来的文件中，删除文件也是同理；而各lower dir则是只读的，在overlayfs挂载后无论如何操作merge目录中对应来自lower dir的文件或目录，lower dir中的内容均不会发生任何的改变（理论设计如此，但实际在一些极端场景存在偏差，后面我会详细介绍）。既然lower dir是只读的，那当用户想要往来自lower层的文件添加或修改内容时，overlayfs首先会的拷贝一份lower dir中的文件副本到upper dir中，后续的写入和修改操作将会在upper dir下的copy-up的副本文件中进行，lower dir原文件被隐藏。

以上就是overlayfs最基本的特性，简单的总结为以下3点：（1）上下层同名目录合并；（2）上下层同名文件覆盖；（3）lower dir文件写时拷贝。这三点对用户都是不感知的。

应用

基本了解overlayfs的基本特性以后，来了解overlayfs特性所带来的好处和应用场景。在实际的使用中，我们可能会存在以下的多用户复用共享文件和目录的场景。见图2。



在同一个设备上，用户A和用户B有一些共同使用的共享文件（例如运行程序所依赖的动态链接库等），一般是只读的；同时也有自己的私有文件（例如系统配置文件等），往往是需要能够写入修改的；最后即使用户A修改了被共享的文件也不会影响到用户B。

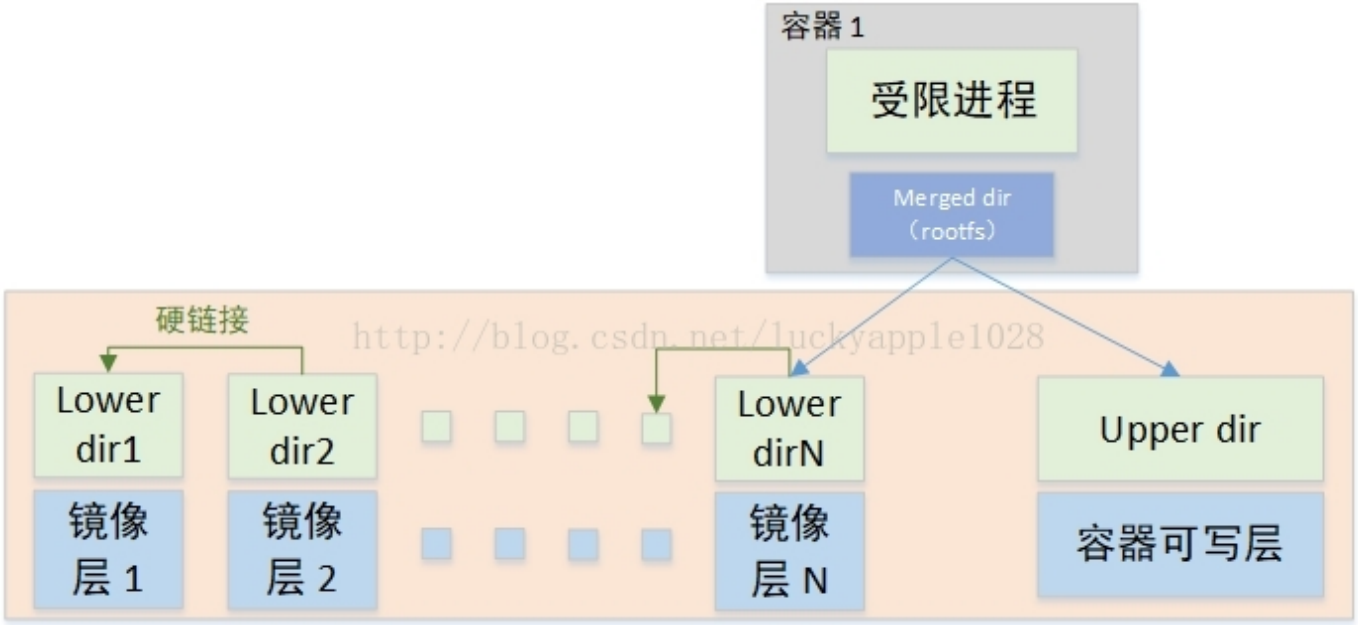
对于以上的需求场景，我们并不希望每个用户都有一份完全一样的文件副本，因为这样不仅带来空间的浪费也会影响性能，因此overlayfs是一个较为完美的解决方案。我们将这些共享的文件和目录所在的目录设定为lower dir (1~n)，将用户私有的文件和目录所在的目录设定为upper dir，然后挂载到用户指定的挂载点，这样即能够保证前面列出的3点需求，同时也能够保证用户A和B独有的目录树结构。最后最为关键的是用户A和用户B在各自挂载目录下看见的共享文件其实是同一个文件，这样磁盘空间的节省自是不必说了，还有就是共享同一份cache而减少内存的使用和提高访问性能，因为只要cache不被回收，只需某个用户首次访问时创建cache，后续其他所有用户都可以通过访问cache来提高IO性能。

上面说的这种使用场景在容器技术中应用最为广泛，下面以docker容器为例来介绍overlay的两种应用方式：Overlay和Overlay2.

Docker容器将镜像层（image layer）作为lower dir，将容器层（container layer）作为upper dir，最后挂载到容器merge挂载点，即容器的根目录下。遗憾的是，早期内核中的overlayfs并不支持多lower layer，在Linux-4.0以后的内核版本中才陆续支持完善。而容器中可能存在多层镜像，所以出现了两种overlayfs的挂载方式，早期的overlay不使用多lower layer的方式挂载而overlay2则使用该方式挂载。

1. Overlay Driver

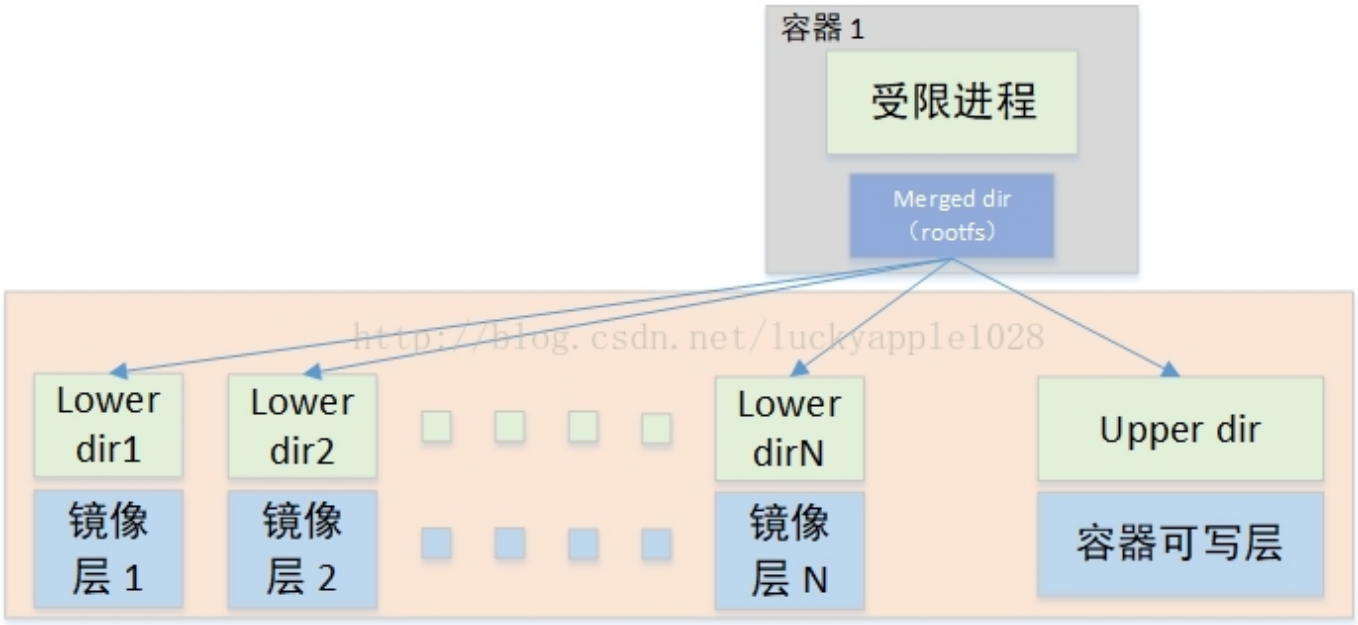
Overlay挂载方式如下。见图3（该图引用自Miklos Szeredi的《overlayfs and containers》2017 linux内核大会演讲材料）。



本图黄色框中的部分是镜像层和容器层的组织方式，各个镜像层中，每下一层中的文件以硬链接的方式出现在它的上一层中，以此类推，最终挂载overlayfs的lower dir为最上层镜像层目录imager layer N。与此同时，容器的writable dir作为upper dir，挂载成为容器的rootfs。本图中虽然只描述了一个容器的挂载方式，但是其他容器也类似，镜像层lower dir N共享，只是各个容器的upper dir不同而已。

2. Overlay2 Driver

Overlay2挂载方式如下。见图4（该图引用自Miklos Szeredi的《overlayfs and containers》2017 linux内核大会演讲材料）。



Overlay2的挂载方式比Overlay的要简单许多，它基于内核overlayfs的Multiple lower layers特性实现，不在需要硬链接，直接将镜像层的各个目录设置为overlayfs的各个lower layer即可（Overlayfs最多支持500层lower dir），对比Overlay Driver将减少inode的使用。

注意事项
尽管Overlayfs看起来是这么的优秀，但是当前它还并不是那么的完美，依然存在一些缺点和使用限制（还没有完全支持POSIX标准），这里简单列出一些，先认识一下，以后遇到也能心中有数：

0. Mount Overlayfs之后就不允许在对原lower dir和upper dir进行操作

当我们挂载完成overlayfs以后，对文件系统的任何操作都只能在merge dir中进行，用户不允许再直接或间接的到底层文件系统的原始lower dir或upper dir目录下修改文件或目录，否则可能会出现一些无法预料的后果（kernel crash除外）。

1. Copy-up

Overlayfs的lower layer文件写时复制机制让某一个用户在修改来自lower层的文件不会影响到其他用户（容器），但是这个文件的复制动作会显得比较慢，后面我们会看到为了保证文件系统的一致性，这个copy-up实现包含了很多步骤，其中最为耗时的就是文件数据块的复制和fsync同步。用户在修改文件时，如果文件较小那可能不一定能够感受出来，但是当文件比较大或一次对大量的小文件进行修改，那耗时将非常可观。虽然自Linux-4.11起内核引入了“concurrent copy up”特性来提高copy-up的并行性，但是对于大文件也还是没有明显的效果。不过幸运的是，如果底层的文件系统支持reflink这样的延时拷贝技术（例如xfs）那就不存在这个问题了。

2. Rename directory（POSIX标准支持问题）

如果Overlayfs的某一个目录是单纯来自lower layer或是lower layer和upper layer合并的，那默认情况下，用户无法对该目录执行rename系统调用，否则会返回-EXDEV错误。不过你会发现通过mv命令重命名该目录依然可以成功，那是因为mv命令的实现对rename系统调用的-EXDEV错误进行规避（这当然是有缺点的，先暂不展开）。在Linux-4.10起内核引入了“redirect dir”特性来修复这个问题，为此引入了一个内核选项：CONFIG_OVERLAY_FS_REDIRECT_DIR，用户想要支持该特性可以在内核中开启这个选项，否则就应避免对这两类目录使用rename系统调用。

3. Hard link break（POSIX标准支持问题）

该问题源自copy-up机制，当lower dir目录中某个文件拥有多个硬链接时，若用户在merge layer对其中一个写入了一些数据，那将触发copy-up，由此该文件将拷贝到upper dir，那么和原始文件的hard link也就断开了，变成了一个单独的文件，用户在merge layer通过stat和ls命令能够直接看到这个变化。在Linux-4.13起内核引入了“index feature”来修复这个问题，同样引入了一个内核选项：
CONFIG_OVERLAY_FS_INDEX，用户想要修复该问题可以打开这个选项，不过该选项不具有向前兼容性，请谨慎使用。

4. Unconstant st_dev&st_ino（POSIX标准支持问题）

该问题同样源自copy-up机制，当原来在lower dir中的文件触发了copy-up以后，那用户在merge layer见到了将是来自upper dir的新文件，那也就意味着它俩的inode是不同的，虽然inode中很多的attr和xattr是可以copy的，但是st_dev和st_ino这两个字段却具有唯一性，是不可以复制的，所以用户可以通过ls和stat命令看到的该字段将发生变化。在Linux-4.12和Linux-4.13分别进行了部分的修复，目前在lower dir和upper dir都在同一个文件系统挂载点的场景下，问题已经修复，但lower dir和upper dir若来自不同的文件系统，问题依然存在。

5. File descriptor change（POSIX标准支持问题）

该问题也同样源自copy-up机制，用户在文件发生copy-up之前以只读方式open文件（这操作不会触发copy-up）得到的文件描述符fd1和copy-up之后open文件得到的文件描述符fd2指向不同的文件，用户通过fd2写入的新数据，将无法从fd1中获取到，只能重新open一个新的fd。该问题目前社区主线内核依然存在，暂未修复。

以上这6点列出了目前Overlayfs的主要问题和限制，将在后文中陆续展开。社区为了让Overlayfs能够更加向支持Posix标准的文件系统靠拢，做出了很多的努力，后续将进一步修复上面提到且未修复的问题，还会增加对NFS Export、freeze snapshots、overlayfs snapshots等的支持，进一步完善overlayfs。

小结
Overlayfs在以它特有的机制已经使用的越来越广泛，在Docker容器技术中以它优异的性能将会渐渐成为首选。不过overlayfs也尚存诸多限制，到目前为止，它还不是一个完全符合Posix规范的文件系统，但社区的开发人员们一直在努力完善，相信不久的将来我们会看到一个非常易用且成熟的Overlayfs。