

## 65 | 知识串讲：用一个创业故事串起操作系统原理（四）

2019-08-26 刘超

趣谈Linux操作系统

[进入课程 >](#)



讲述：刘超

时长 23:21 大小 21.39M



上一节，小马的公司已经解决了生存问题，成功从小马晋升马哥。

马哥是一个有危机意识的人。尽管公司开始不断盈利，项目像流水一样，一个接一个，赚了点儿钱，但是他感觉还是有点儿像狗熊掰棒子。因为公司没有积累，永远就都是在做小生意，无法实现成倍的增长。

马哥想，公司做了这么多的项目，应该有很多的共同点，能积累下来非常多的资料。如果能够把这些资料归档、总结、积累，形成核心竞争力，就可以随着行业的飞跃，深耕一个行业，实现快速增长。

**公司发展需积累，马哥建立知识库**

这就需要我们有一个存放资料的档案库（文件系统）。档案库应该不依赖于项目而独立存在，应该井井有条、利于查询；应该长久保存，不随人员流动而损失。

公司到了这个阶段，除了周瑜和张昭，应该专门请一个能够积累核心竞争力的人来主持大局了。马哥想到了，前一阵行业交流大会上，他遇到了一个很牛的架构师——鲁肃。他感觉鲁肃在这方面很有想法，于是就请他来主持大局。

鲁肃跟马哥说，构建公司的核心技术能力，这个档案库（文件系统）也可以叫作知识库，这个需要好好规划一下。规划文件系统的时候，需要考虑以下几点。

第一点，文件系统要有严格的组织形式，使得文件能够以块为单位进行存储。

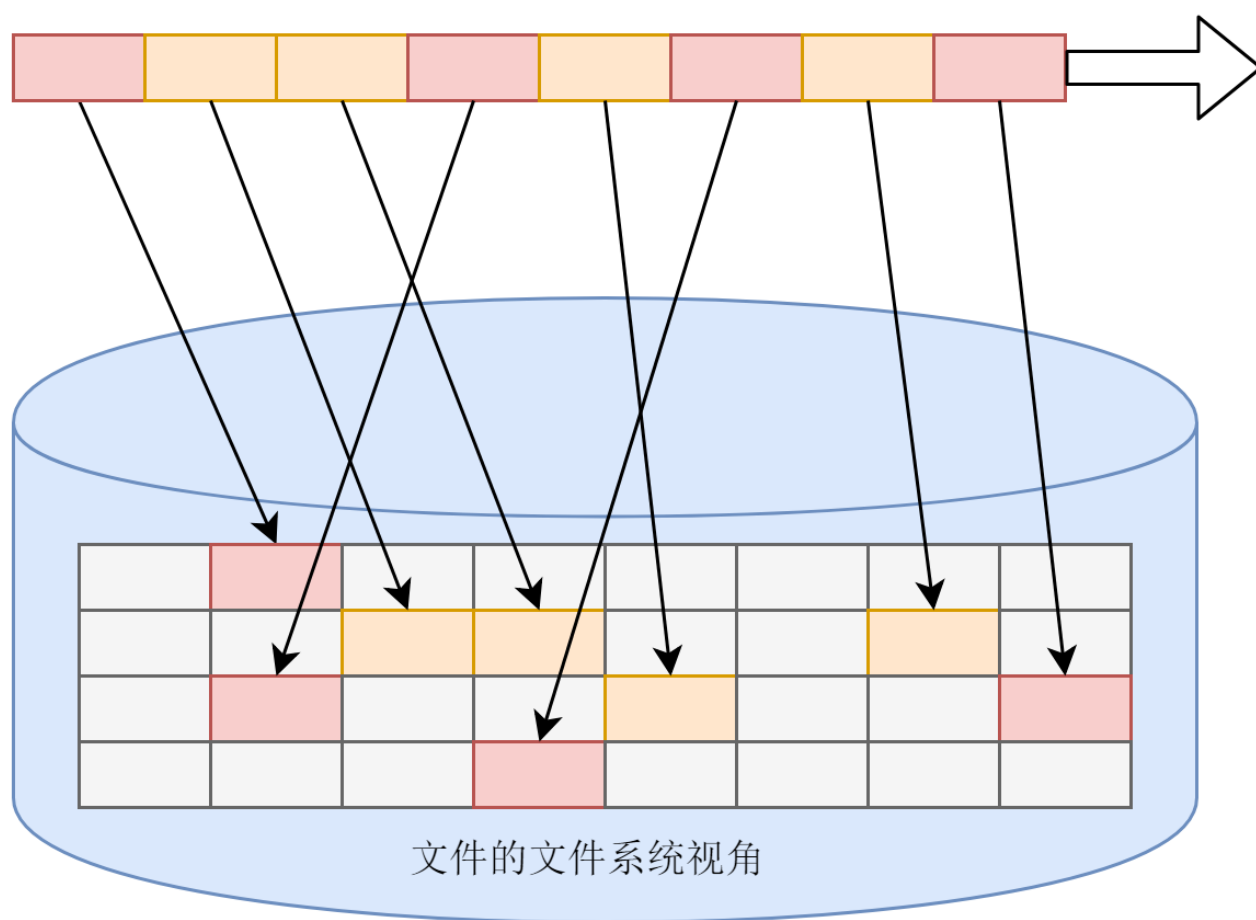
这就像图书馆里，我们会给设置一排排书架，然后再把书架分成一个个小格子。有的项目存放的资料非常多，一个格子放不下，就需要多个格子来进行存放。我们把这个区域称为存放原始资料的仓库区。对于操作系统，硬盘分成相同大小的单元，我们称为块。一块的大小是扇区大小的整数倍，默认是 4K，用来存放文件的数据部分。这样一来，如果我们像存放一个文件，就不用给他分配一块连续的空间了。我们可以分散成一个个小块进行存放。这样就灵活得多，也比较容易添加、删除和插入数据。

第二点，文件系统中也要有索引区，用来方便查找一个文件分成的多个块都存放在了什么位置。

这就好比，图书馆的书太多了，为了方便查找，我们需要专门设置一排书架，这里面会写清楚整个档案库有哪些资料，资料在哪个架子的哪个格子上。这样找资料的时候就不用跑遍整个档案库，只要在这个书架上找到后，直奔目标书架就可以了。

在 Linux 操作系统里面，每一个文件有一个 Inode，inode 的 “i” 是 index 的意思，其实就是“索引”。inode 里面有文件的读写权限 `i_mode`，属于哪个用户 `i_uid`，哪个组 `i_gid`，大小是多少 `i_size_io`，占用多少个块 `i_blocks_io`。“某个文件分成几块、每一块在哪里”，这些信息也在 inode 里面，保存在 `i_block` 里面。

## 文件的用户视角



第三点，如果文件系统中有的文件是热点文件，近期经常被读取和写入，文件系统应该有缓存层。

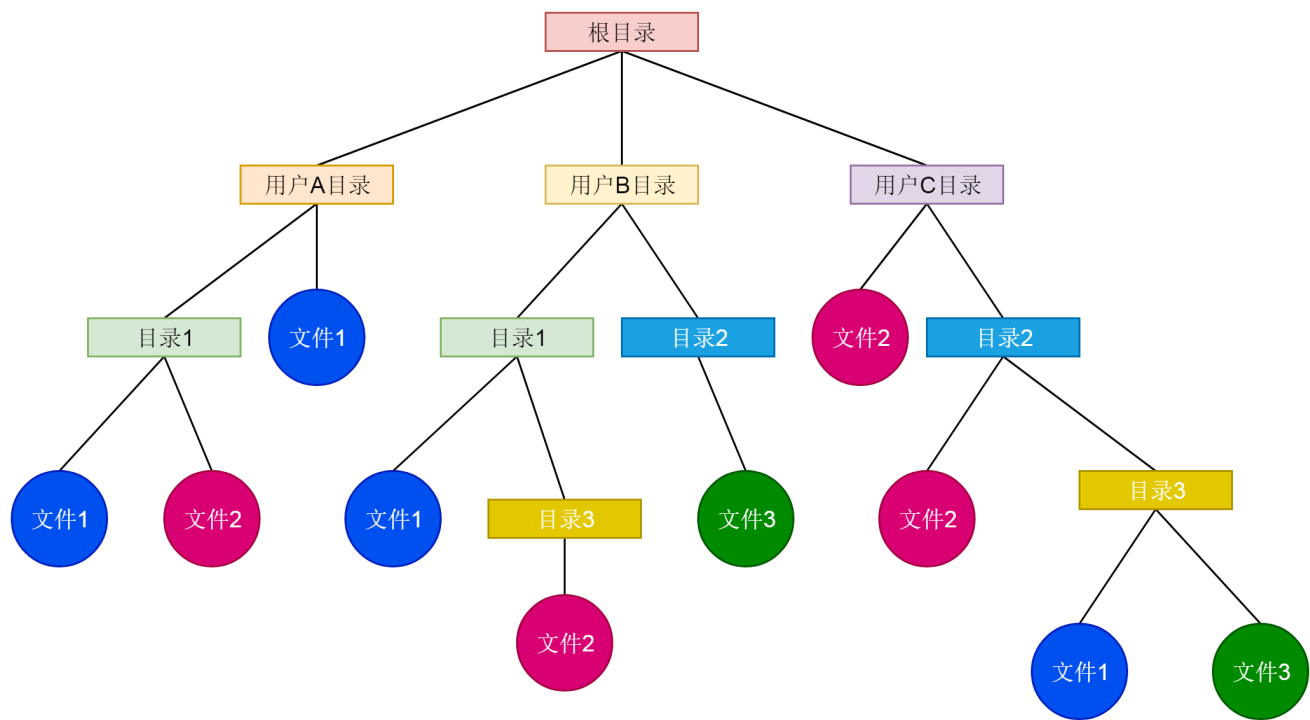
这就相当于图书馆里面的热门图书区，这里面的书都是畅销书或者是常常被借还的图书。因为借还的次数比较多，那就没有必要每次有人还了之后，还放回遥远的货架，我们可以专门开辟一个区域，放置这些借还频次高的图书。这样借还的效率就会提高。

第四点，文件应该用文件夹的形式组织起来，方便管理和查询。

这就像在图书馆里面，你可以给这些资料分门别类，比如分成计算机类、文学类、历史类等等。这样你也容易管理，项目组借阅的时候只要在某个类别中去找就可以了。

在文件系统中，每个文件都有一个名字，我们访问一个文件，希望通过他的名字就可以找到。文件名就是一个普通的文本，所以文件名经常会冲突，不同用户取相同的名字的情况会经常出现的。

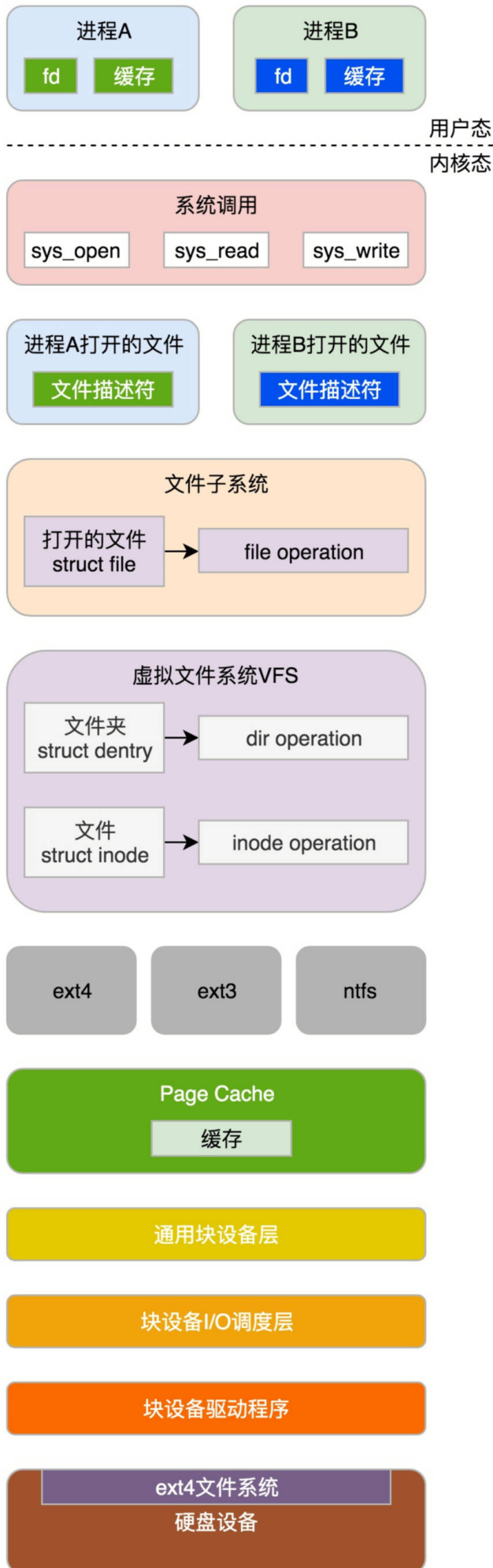
要想把很多的文件有序地组织起来，我们就需要把他们做成目录或者文件夹。这样，一个文件夹里可以包含文件夹，也可以包含文件，这样就形成了一种树形结构。我们可以将不同的用户放在不同的用户目录下，就可以一定程度上避免了命名的冲突问题。



第五点，Linux 内核要在自己的内存里面维护一套数据结构，来保存哪些文件被哪些进程打开和使用。

这就好比，图书馆里会有个图书管理系统，记录哪些书被借阅了，被谁借阅了，借阅了多久，什么时候归还。

这个图书管理系统尤为重要，如果不是很方便使用，以后项目中积累了经验，就没有人愿意往知识库里面放了。



无论哪个项目（进程），都可以通过 write 系统调用写入知识库。

对于每一个进程，打开的文件都有一个文件描述符。files\_struct 里面会有文件描述符数组。每个一个文件描述符是这个数组的下标，里面的内容指向一个 struct file 结构，表示打开的文件。这个结构里面有这个文件对应的 inode，最重要的是这个文件对应的操作 file\_operation。如果操作这个文件，就看这个 file\_operation 里面的定义了。

每一个打开的文件，都有一个 dentry 对应，虽然我们叫作 directory entry，但是他不仅仅表示文件夹，也表示文件。他最重要的作用就是指向这个文件对应的 inode。

如果说 file 结构是一个文件打开以后才创建的，dentry 是放在一个 dentry cache 里面的。文件关闭了，他依然存在，因而他可以更长期的维护内存中的文件的表示和硬盘上文件的表示之间的关系。

inode 结构就表示硬盘上的 inode，包括块设备号等。这个 inode 对应的操作保存在 inode operations 里面。真正写入数据，是写入硬盘上的文件系统，例如 ext4 文件系统。

马哥听了知识库和档案库的设计，非常开心，对鲁肃说，你这五大秘籍，可是帮了我大忙了。于是马上下令实施。

## 有了积累建生态，成立渠道管理部

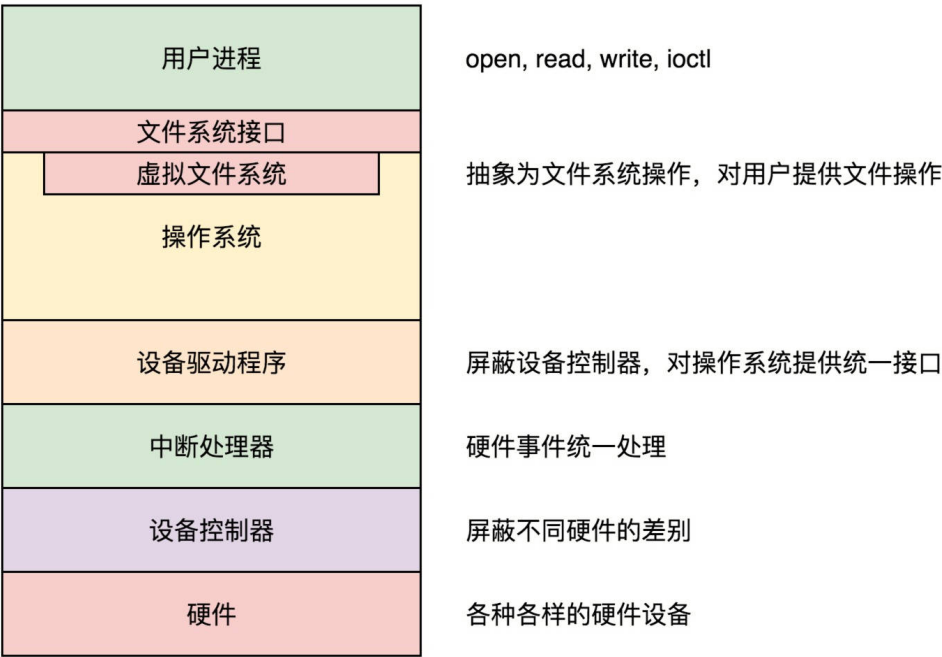
有了知识库，公司的面貌果然大为改观。

马哥发现，当知识库积累到一定程度，公司接任何项目都能找到相似的旧项目作为参考，不用重新设计，效率大大提高。而且最重要的一点是，没有知识库的时候，原来项目做的好不好，完全取决于程序员，因为所有的知识都在程序员的脑子里，所以公司必须要招聘高质量的程序员，才能保证项目的质量。一方面优秀的程序员数量很少，这大大限制了公司能够接项目的规模，一方面优秀的程序员实在太贵，大大提高了公司的成本。

有了知识库，依赖于原来积累的体系，只要找到类似的旧项目，哪怕是普通的程序员，只要会照猫画虎，结果就不会太差。

于是，马哥马上想到，现在公司只有百十来号人，能赚这些钱，现在招人门槛降低了，我要是招聘一万人，这能赚多少钱啊！

鲁肃对马哥说，“你可先别急着招人，建立知识库，降低招人成本才是第一步。公司招聘太多人不容易管理。既然项目的执行可以照猫画虎，很多项目可以不用咱们公司来，我们可以建立渠道销售体系（输入和输出系统），让供应商、渠道帮我们卖，形成一个生态。这公司的盈利规模可就不是招一万人这么点儿了，这是指数级的增长啊！”



计算机系统的输入和输出系统都有哪些呢？我们能举出来的，例如键盘、鼠标、显示器、网卡、硬盘、打印机、CD/DVD 等等，多种多样。这样，当然方便用户使用了，但是对于操作系统来讲，却是一件复杂的事情，因为这么多设备，形状、用法、功能都不一样，怎么才能统一管理起来呢？我们一层一层来看。

第一层，用设备控制器屏蔽设备差异。

马哥说，“把生意做到全国，我也想过，这个可不容易。咱们客户多种多样，众口难调，不同的地域不一样，不同的行业不一样。如果你不懂某个地方的规矩，根本卖不出去东西；如果你不懂某个具体行业的使用场景，也无法满足客户的需求。”

鲁肃说：“所以说，建议您建立生态，设置很多代理商，让各个地区和各个行业的代理商帮你屏蔽这些差异化。你和代理商之间只要进行简单的标准产品交付就可以了。”

计算机系统就是这样的。CPU 并不直接和设备打交道，他们中间有一个叫作设备控制器（Device Control Unit）的组件。例如，硬盘有磁盘控制器、USB 有 USB 控制器、显示器有视频控制器等。这些控制器就像代理商一样，他们知道如何应对硬盘、鼠标、键盘、显示器的行为。

你的代理商往往是小公司。控制器其实有点儿像一台小电脑。他有他的芯片，类似小 CPU，执行自己的逻辑。他也有他的寄存器。这样 CPU 就可以通过写这些寄存器，对控制器下发指令，通过读这些寄存器，查看控制器对于设备的操作状态。

CPU 对于寄存器的读写，可比直接控制硬件，要标准和轻松很多。这就相当于你和代理商的标准产品交付。

第二层，用驱动程序屏蔽设备控制器差异。

马哥说：“你这么一说，还真有道理，如果我们能够找到足够多的代理商，那就高枕无忧了。”

鲁肃说：“其实事情还没这么简单，虽然代理商机制能够帮我们屏蔽很多设备的细节，但是从上面的描述我们可以看出，由于每种设备的控制器的寄存器、缓冲区等使用模式，指令都不同。对于咱们公司来讲，就需要有个部门专门对接代理商，向其他部门屏蔽代理商的差异，成立公司的渠道管理部门。”

那对于操作系统来讲，渠道管理部门就是用来对接各个设备控制器的设备驱动程序。

这里需要注意的是，设备控制器不属于操作系统的一部分，但是设备驱动程序属于操作系统的一部分。操作系统的内核代码可以像调用本地代码一样调用驱动程序的代码，而驱动程序的代码需要发出特殊的面向设备控制器的指令，才能操作设备控制器。

设备驱动程序中是一些面向特殊设备控制器的代码。不同的设备不同。但是对于操作系统其他部分的代码而言，设备驱动程序应该有统一的接口。就像下面图中的一样，不同的设备驱动程序，可以以同样的方式接入操作系统，而操作系统的其他部分的代码，也可以无视不同设备的区别，以同样的接口调用设备驱动程序。





驱动程序接口不统一



驱动程序接口统一

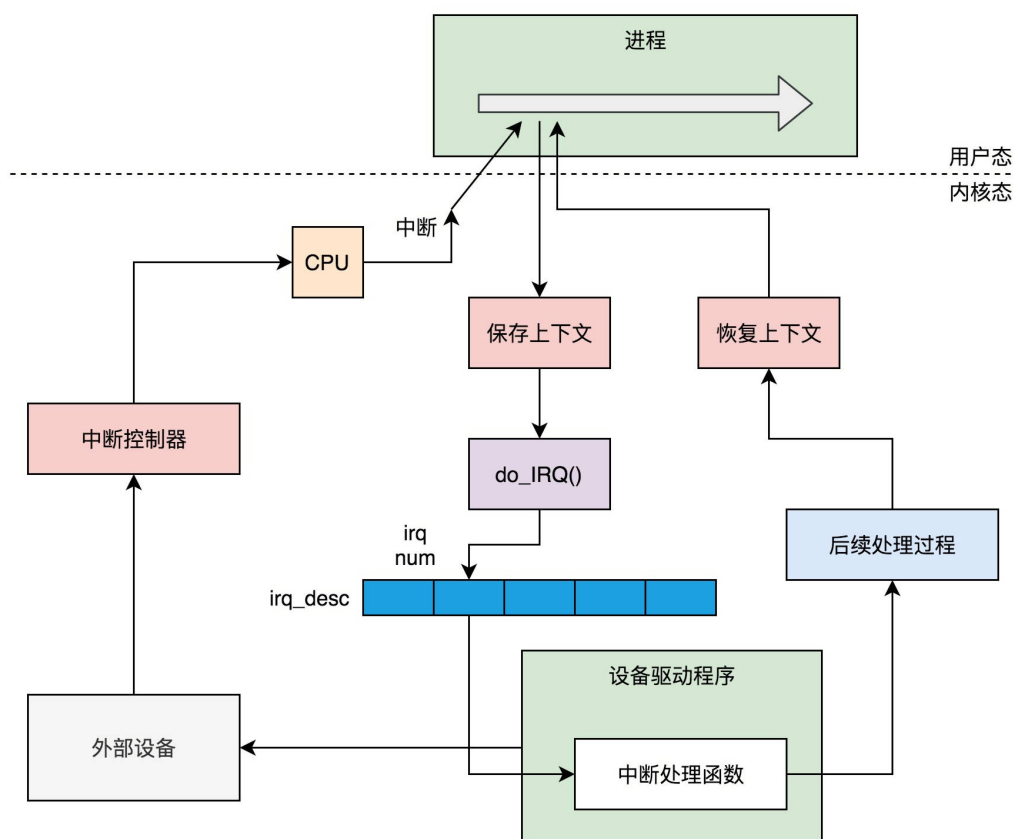
第三，用中断控制器统一外部事件处理。

马哥听了恍然大悟：“原来代理商也是五花八门，里面有这么多门道啊！”

鲁肃说：“当咱们对接的代理商多了，代理商可能会有各种各样的问题找到我们，例如代理商有了新客户，客户有了新需求，客户交付完毕等事件，都需要有一种机制通知你们公司，当然是中断，那操作系统就需要有一个地方处理这个中断，既然设备驱动程序是用来对接设备控制器的，中断处理也应该在设备驱动里面完成。”

然而，中断的触发最终会到达 CPU，会中断操作系统当前运行的程序，所以操作系统也要有一个统一的流程来处理中断，使得不同设备的中断使用统一的流程。

一般的流程是，一个设备驱动程序初始化的时候，要先注册一个该设备的中断处理函数。咱们讲进程切换的时候说过，中断返回的那一刻是进程切换的时机。中断的时候，触发的函数是 `do_IRQ`。这个函数是中断处理的统一入口。在这个函数里面，我们可以找到设备驱动程序注册的中断处理函数 `Handler`，然后执行他进行中断处理。



第四，用文件系统接口屏蔽驱动程序的差异。

马哥又问了：“对接了这么多代理商，如果咱们内部的工程师要和他们打交道，有没有一种统一的方式呢？”

鲁肃说：“当然应该了，我们内部员工操作外部设备，可以基于文件系统的接口，制定一个统一的标准。”

其实文件系统的机制是一个非常好的机制，咱们公司应该定下这样的规则，一切皆文件。

所有设备都在 `/dev/` 文件夹下面，创建一个特殊的设备文件。这个设备特殊文件也有 `inode`，但是他不关联到硬盘或任何其他存储介质上的数据，而是建立了与某个设备驱动程序连接。

有了文件系统接口之后，我们不但可以通过文件系统的命令行操作设备，也可以通过程序，调用 `read`、`write` 函数，像读写文件一样操作设备。

对于块设备来讲，在驱动程序之上，文件系统之下，还需要一层通用设备层。比如，咱们讲的文件系统，里面的逻辑和磁盘设备没有什么关系，可以说是通用的逻辑。在写文件的最底

层，我们看到了 BIO 字眼的函数，但是好像和设备驱动也没有什么关系。

是的，因为块设备类型非常多，而 Linux 操作系统里面一切是文件。我们也不想文件系统以下，就直接对接各种各样的块设备驱动程序，这样会使得文件系统的复杂度非常高。所以，我们在中间加了一层通用块层，将与块设备相关的通用逻辑放在这一层，维护与设备无关的块的大小，然后通用块层下面对接各种各样的驱动程序。

虚拟文件系统VFS

ext4

ext3

ntfs

Page Cache

缓存

通用块设备层

块设备I/O调度层

块设备驱动程序

ext4文件系统

硬盘设备

鲁肃帮助马哥建立了这套体系之后，果真业务有了很大起色。原来公司只敢接华东区的项目，毕竟比较近，沟通交付都很方便。后来项目扩展到所有一线城市、二线城市、省会城市，项目数量实现了几十倍的增长。

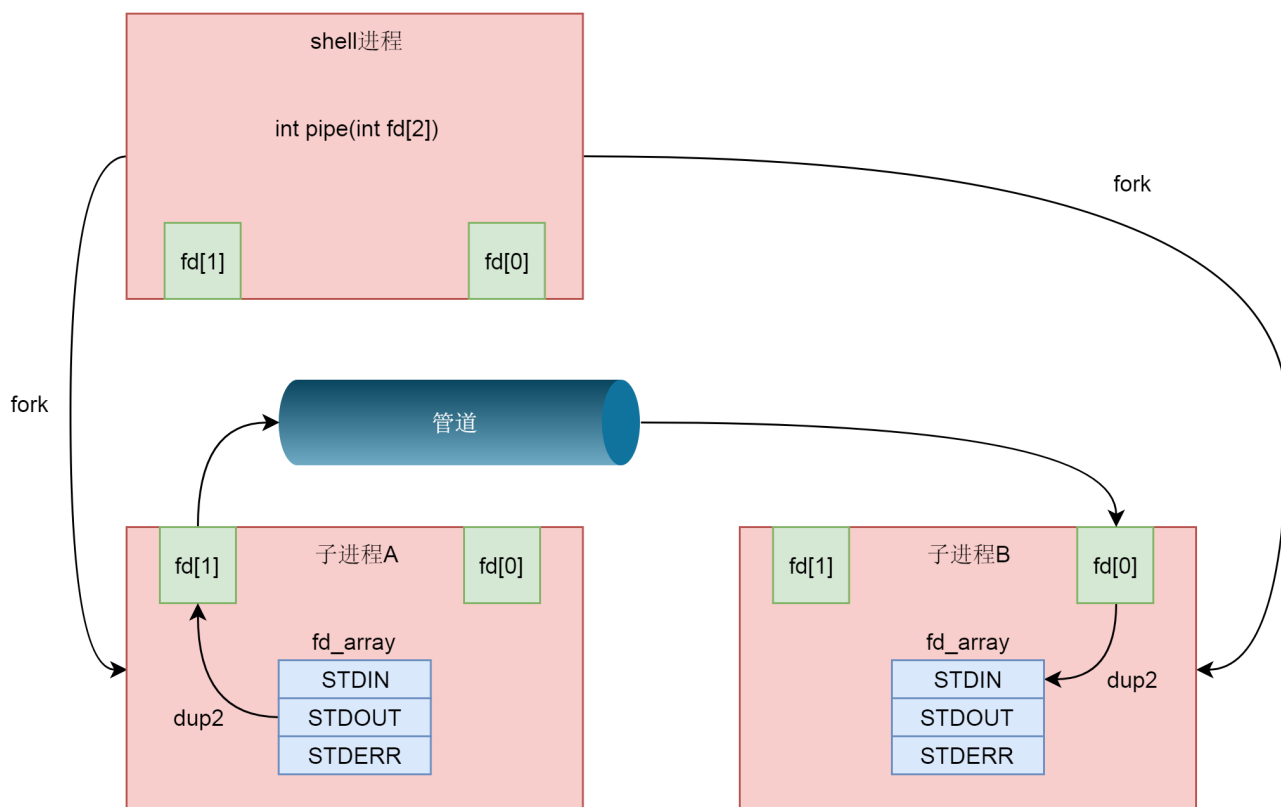
## 千万项目难度大，集体合作可断金

项目接的多了，就不免有大型的项目，涉及多个行业多个领域，需要多个项目组进行合作才能完成。那两个项目组应该通过什么样的方式，进行沟通与合作呢？作为老板，马哥应该如何设计整个流程呢？

马哥叫来周瑜、张昭、鲁肃，一起商量团队间的合作模式。大家一起献计献策。好在有很多成熟的项目管理流程可以参考。

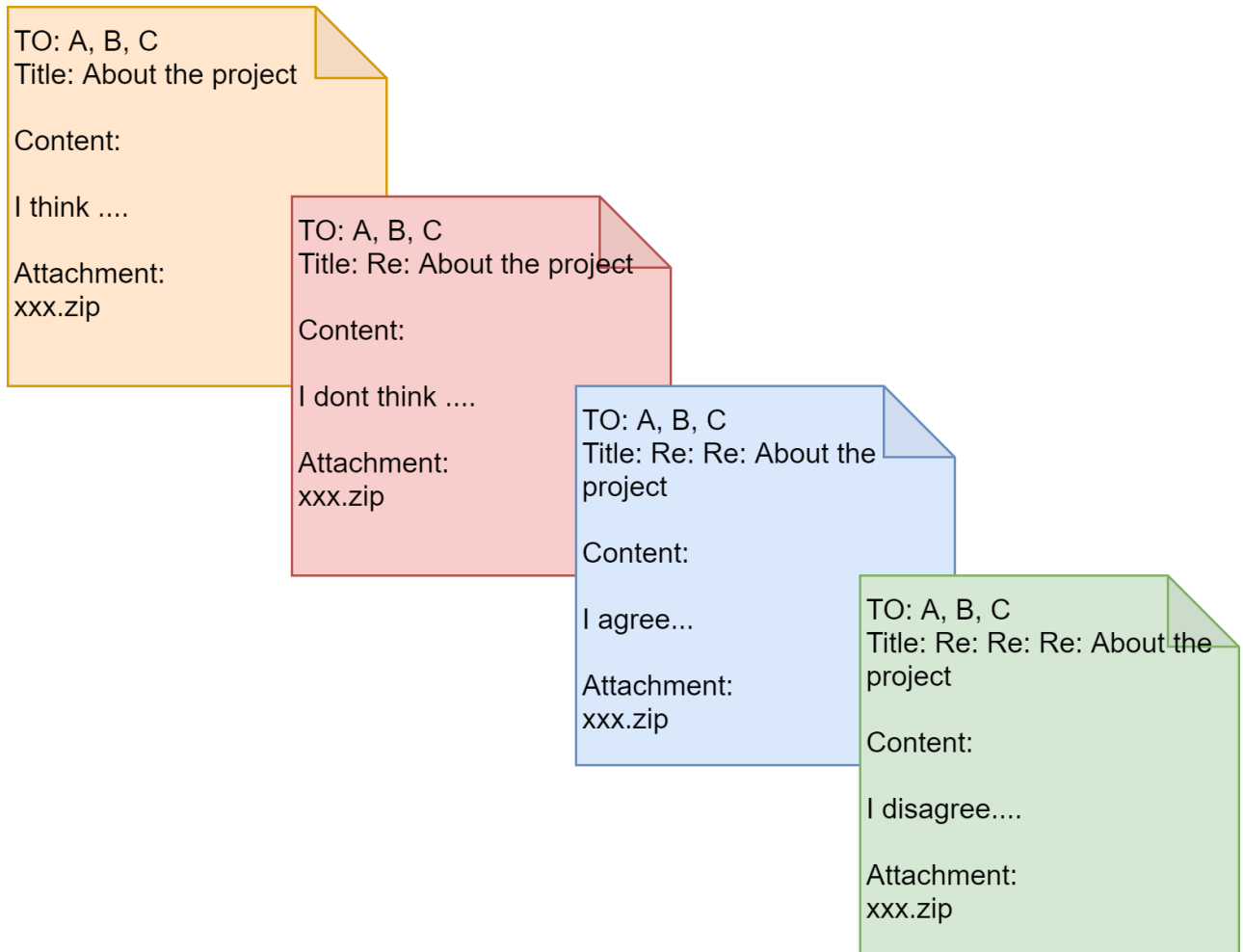
最最传统的模型就是软件开发的瀑布模型。所谓的瀑布模型，其实就是将整个软件开发过程分成多个阶段，往往是上一个阶段完全做完，才将输出结果交给下一个阶段。这种模型类似进程间通信的管道模型。

所谓的管道，就是在两个进程之间建立一条单向的通道，其实是一段缓存，它会将前一个命令的输出，作为后一个命令的输入。



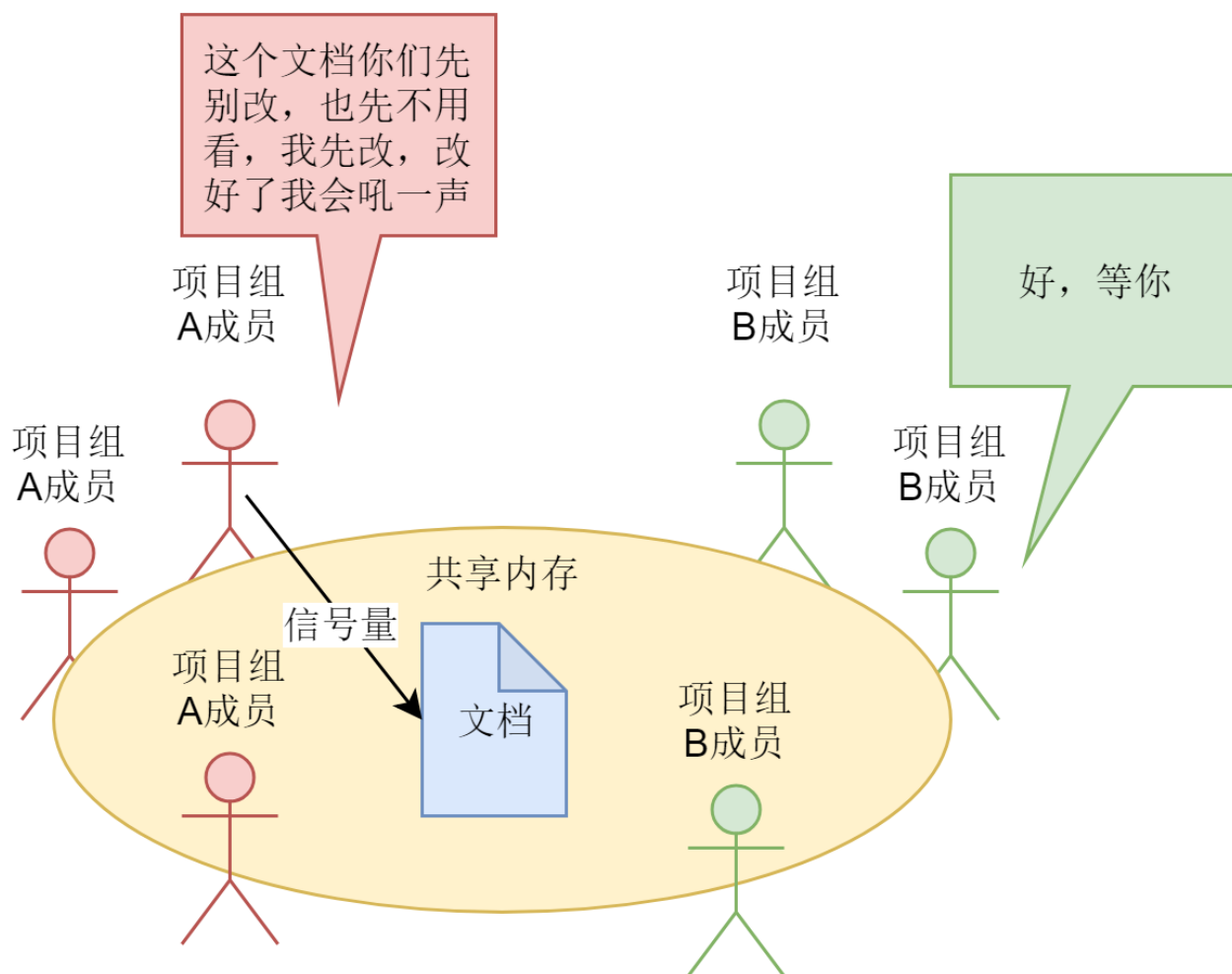
张昭说，瀑布模型的开发流程效率比较低，现在大部分公司都不使用这种开发模式了，因为团队之间无法频繁地沟通。而且，管道的使用模式，也不适合进程间频繁的交换数据。

于是，他们还得想其他的办法。是不是可以借鉴传统外企的沟通方式——邮件呢？邮件有一定的格式，例如抬头、正文、附件等。发送邮件可以建立收件人列表，所有在这个列表中的人，都可以反复地在此邮件基础上回复，达到频繁沟通的目的。这个啊，就是消息队列模型。



和管道将信息一股脑儿地从一个进程，倒给另一个进程不同，消息队列有点儿像邮件，发送数据时，会分成一个一个独立的数据单元，也就是消息体，每个消息体都是固定大小的存储块，在字节流上不连续。

有了消息这种模型，两个进程之间的通信就像咱们平时发邮件一样，你来一封，我回一封，可以频繁沟通了。



但是有时候，项目组之间的沟通需要特别紧密，而且要分享一些比较大的数据。如果使用邮件，就发现，一方面邮件的来去不及时；另外一方面，附件大小也有限制，所以，这个时候，我们经常采取的方式就是，把两个项目组在需要合作的期间，拉到一个会议室进行合作开发，这样大家可以直接交流文档呀，架构图呀，直接在白板上画或者直接扔给对方，就可以直接看到。

可以看出来，共享会议室这种模型，类似进程间通信的共享内存模型。前面咱们讲内存管理的时候，知道每个进程都有自己独立的虚拟内存空间，不同的进程的虚拟内存空间映射到不同的物理内存中去。这个进程访问 A 地址和另一个进程访问 A 地址，其实访问的是不同的物理内存地址，对于数据的增删查改互不影响。

但是，咱们是不是可以变通一下，拿出一块虚拟地址空间来，映射到相同的物理内存中。这样这个进程写入的东西，另外一个进程马上就能看到了，都不需要拷贝来拷贝去，传来传去。



马哥说：“共享内存也有问题呀。如果两个进程使用同一个共享内存，大家都往里面写东西，很有可能就冲突了。例如两个进程都同时写一个地址，那先写的那个进程会发现内容被别人覆盖了。”

张昭说：“当然，和共享内存配合的，有另一种保护机制，使得同一个共享的资源，同时只能被一个进程访问叫信号量。”

信号量其实是一个计数器，主要用于实现进程间的互斥与同步，而不是用于存储进程间通信数据。

我们可以将信号量初始化为一个数值，来代表某种资源的总体数量。对于信号量来讲，会定义两种原子操作，一个是 P 操作，我们称为申请资源操作。这个操作会申请将信号量的数值减去 N，表示这些数量被他申请使用了，其他人不能用了。另一个是 V 操作，我们称为归还资源操作，这个操作会申请将信号量加上 M，表示这些数量已经还给信号量了，其他人可以使用了。

例如，你有 100 元钱，就可以将信号量设置为 100。其中 A 向你借 80 元，就会调用 P 操作，申请减去 80。如果同时 B 向你借 50 元，但是 B 的 P 操作比 A 晚，那就没有办法，只好等待 A 归还钱的时候，B 的 P 操作才能成功。之后，A 调用 V 操作，申请加上 30 元，也就是还给你 30 元，这个时候信号量有 50 元了，这时候 B 的 P 操作才能成功，才能借走这 50 元。

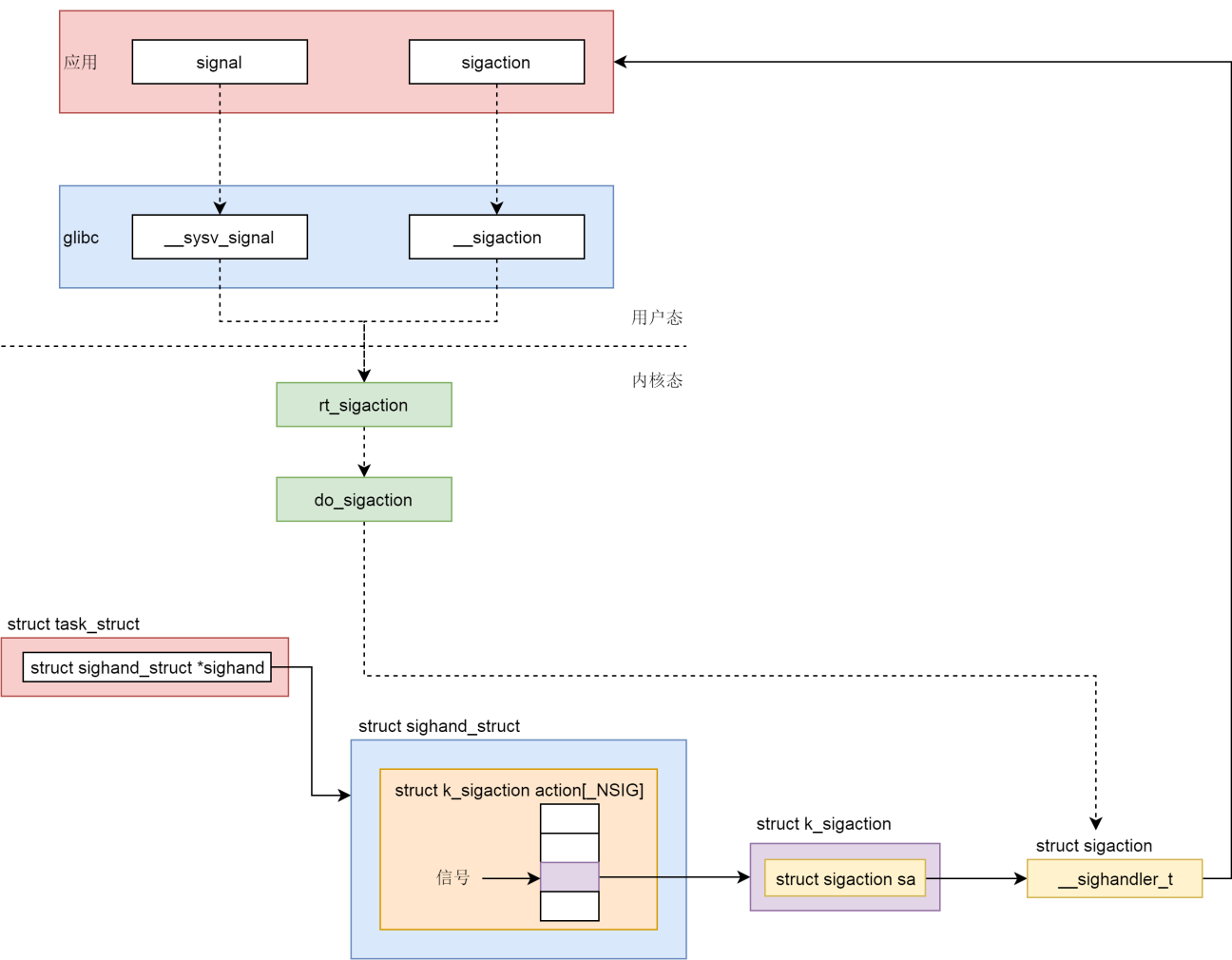
所谓原子操作（Atomic Operation），就是任何一块钱，都只能通过 P 操作借给一个人，不能同时借给两个人。也就是说，当 A 的 P 操作（借 80）和 B 的 P 操作（借 50），几乎同时到达的时候，不能因为大家都看到账户里有 100 就都成功，必须分个先来后到。

马哥说：“有了上面的这些机制，基本常规状态下的工作模式，对应到咱们平时的工作交接，收发邮件、联合开发等。我还想到，如果发生了异常怎么办？例如出现线上系统故障，这个时候，什么流程都来不及了，不可能发邮件，也来不及开会，所有的架构师、开发、运维都要被通知紧急出动。所以，7 乘 24 小时不间断执行的系统都需要有告警系统，一旦出事情，就要通知到人，哪怕是半夜，也要电话叫起来，处理故障。是不是应该还有一种异常情况下的工作模式。”

张昭说：“当然应该有，我们可以建立像操作系统里面的信号机制。信号没有特别复杂的数据结构，就是用一个代号一样的数字。Linux 提供了几十种信号，分别代表不同的意义。信

号之间依靠它们的值来区分。这就像咱们看警匪片，对于紧急的行动，都是说，‘1 号作战任务’ 开始执行，警察就开始行动了。情况紧急，不能啰里啰嗦了。”

信号可以在任何时候发送给某一进程，进程需要为这个信号配置信号处理函数。当某个信号发生的时候，就默认执行这个函数就可以了。这就相当于咱们运维一个系统应急手册，当遇到什么情况，做什么事情，都事先准备好，出了事情照着做就可以了。



这些项目组合作的流程设计合理，因而推行起来十分顺畅，现在接个千万级别的项目没有任何问题，根据交易量估值市值，起码有十个亿。

马哥有些小激动，原来自己身价这么高了，是不是也能上个市啥的，实现亿万富翁的梦想呢？于是马哥找了一些投资人聊了聊，投资人说，要想冲一把上市，还差点劲，目前的项目虽然大，但是想象力不够丰富。

那接下来，马哥如何做才能满足市场的想象力，最终成功上市呢？预知后事，且听下回分解。

# 趣谈 Linux 操作系统


像故事一样的操作系统入门课

刘超

网易杭州研究院

云计算技术部首席架构师



新版升级：点击「 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 64 | 知识串讲：用一个创业故事串起操作系统原理（三）

## 精选留言 (5)

 写留言



安排

2019-08-26

最后这几篇总结真是太好了，如果全程跟下来，再结合这几篇总结收货会很大，极客买的最值得课程，没有之一。👍



👍 3



莫名

2019-08-26

知识串讲很赞👍



leslie

2019-08-26

对于今天课程的串讲：简述一下我的学习感受，今天的内容其实是通过管理；把操作系统、计算机组成原理、程序接口，串联了消息体系；课程的内容直接把近期的直接整体串

联起来了。

老师今天的课明显不是学习1遍就能明白的掌握的：要重学多遍的知识。老师的专题其实非常非常好-虽然学习的确实蛮辛苦，不过收获还是蛮大，这个系列其实是把系统相关...  
展开 ▾



**EidLeung**

2019-08-26

从网络协议到操作系统，听着超哥的故事学习，爽歪歪！



**小龙的城堡**

2019-08-26

浅显易懂，再结合之前的代码，就搞懂了 😊

