一个新进程的诞生（一）先整体看一下

Original 闪客 低并发编程 2022-02-09 16:30

收录于合集
#操作系统源码 43 #一个新进程的诞生 8

本系列作为 你管这破玩意叫操作系统源码 的第三大部分，讲述了操作系统第一个进程从无到有的诞生过程，这一部分你将看到内核态与用户态的转换、进程调度的上帝视角、系统调用的全链路、fork 函数的深度剖析。

不要听到这些陌生的名词就害怕，跟着我一点一点了解他们的全貌，你会发现，这些概念竟然如此活灵活现，如此顺其自然且合理地出现在操作系统的启动过程中。

本篇章作为一个全新的篇章，需要前置篇章的知识体系支撑。

第一部分 进入内核前的苦力活
第二部分 大战前期的初始化工作

当然，没读过的也问题不大，我都会在文章里做说明，如果你觉得有困惑，就去我告诉你的相应章节回顾就好了，放宽心。

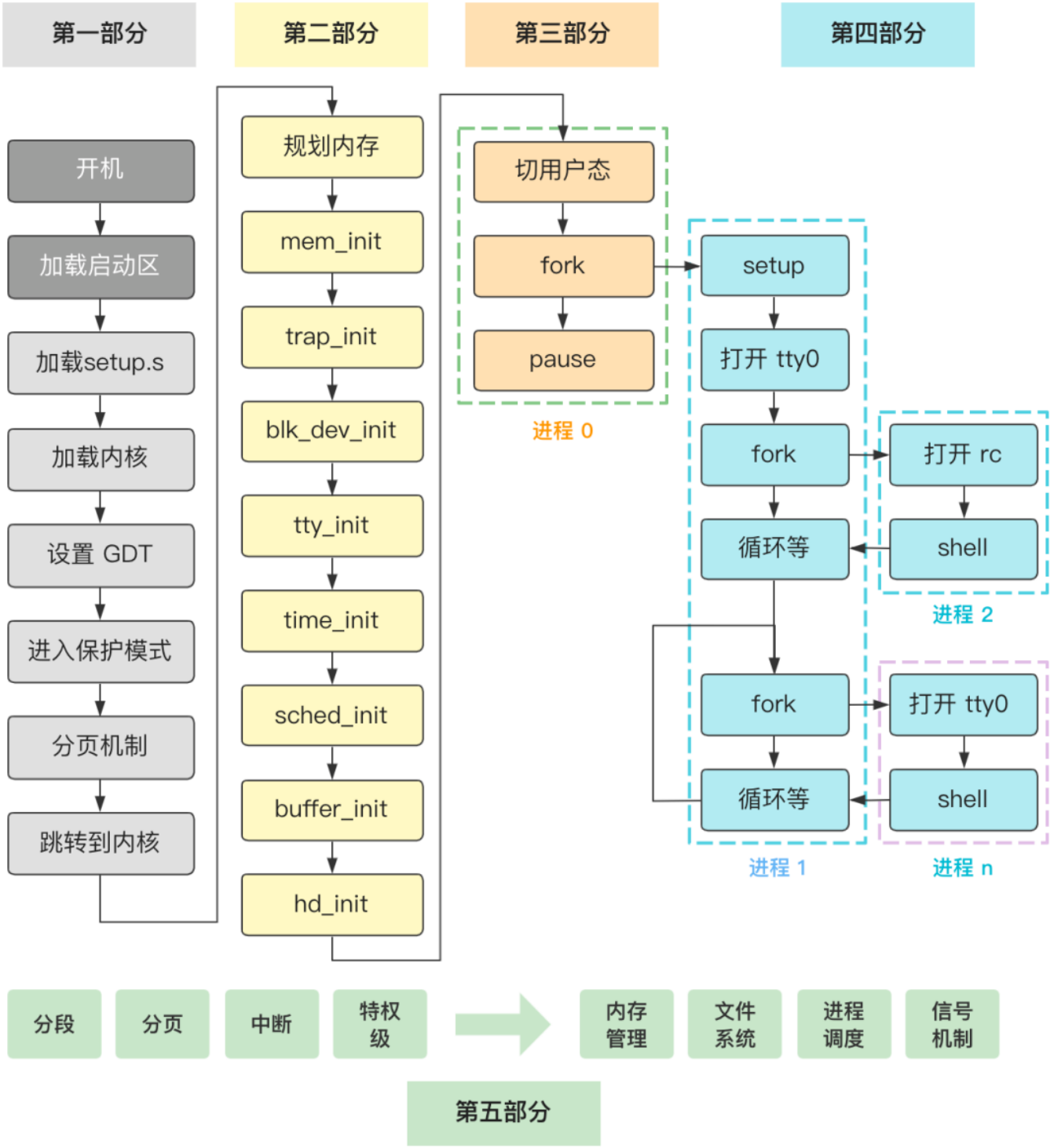
话不多说，我们开始今天的内容吧！

------- 正文开始 -------

本文为整个第三部分的第一回内容。

一个新进程的诞生，从操作系统的源码角度来说，其实就两行代码。而关于创建进程的重点，其实就一行代码，就是大名鼎鼎的 fork 函数。

我们先来看一张图，看看操作系统从开机到怠速都做了些什么事情。



你看，第一部分和第二部分，为我们这个第三部分做了充足的铺垫工作。

第一部分 进入内核前的苦力活
第二部分 大战前期的初始化工作

到了第三部分，简单说就是从内核态切换到用户态，然后通过 fork 创建出一个新的进程，再之后老进程进入死循环。

```
void main(void) {
    // 第二部分的内容，各种初始化工作
    ...
    // 第三部分的内容，一个新进程的诞生
    move_to_user_mode();
    if (!fork()) {
        // 新进程里干了啥，是第四部分的内容
        init();
    }
    // 死循环，操作系统怠速状态
    for(;;) pause();
}
```

至于 fork 出来的新进程做了什么事，就是 init 函数里的故事里，这个不在第三部分的讨论范畴。

所以你看，一共就两行代码，顶多再算上最后一行的死循环，三行，就把创建新进程这个事搞定了。再加上新进程里要做的 init 函数，一共四行代码，就走到了 main 函数的结尾，也就标志着操作系统启动完毕！

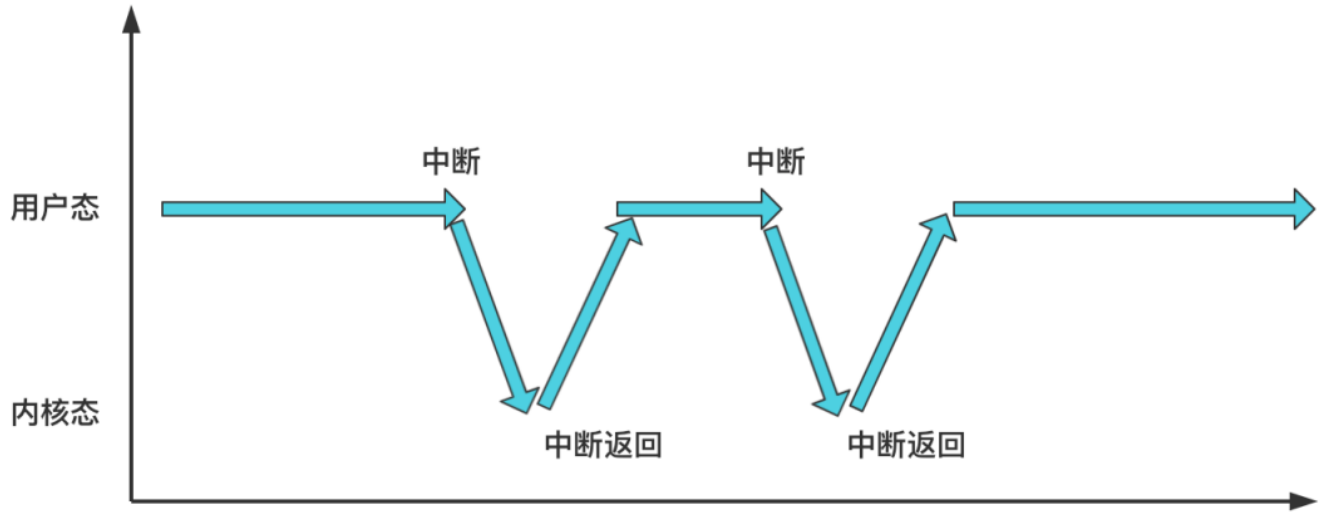
但就是这没有多少个字母的四行代码，是整个操作系统的精髓所在，也是最难的四行代码。理解了它们，你就会有原来操作系统就是这破玩意的感叹了 ~

今天我们就总览一下这四句，很轻松。

第一句是 move_to_user_mode

直译过来即可，就是转变为用户态模式。因为 Linux 将操作系统特权级分为用户态与内核态两种，之前都处于内核态，现在要先转变为用户态，仅此而已。

一旦转变为了用户态，那么之后的代码将一直处于用户态的模式，除非发生了中断，比如用户发出了系统调用的中断指令，那么此时将会从用户态陷入内核态，不过当中断处理程序执行完之后，又会通过中断返回指令从内核态回到用户态。



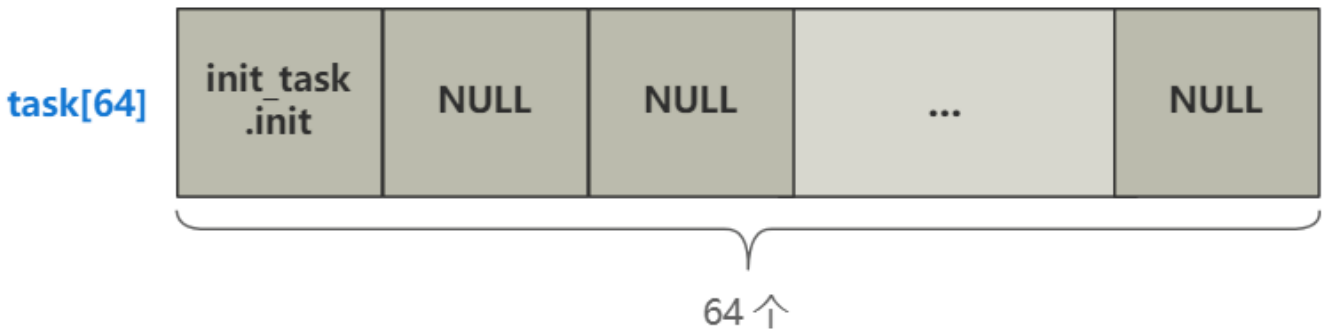
整个过程被操作系统的机制拿捏的死死的，始终让用户进程处于用户态运行，必要的时候陷入一下内核态，但很快就会被返回而再次回到用户态，是不是非常无奈？

第二句是 fork

这是创建一个新进程的意思，而且所有用户进程想要创建新的进程，都需要调用这个函数。

原来操作系统只有一个执行流，就是我们一直看过来的所有代码，就是进程 0，只不过我们并没有意识到它也是一个进程。调用完 fork 之后，现在又多了一个进程，叫做进程 1。

当然，更准确的说法是，我们一路看过来的代码能够被我们自信地称作进程 0 的确切时刻，是我们在 第18回 | 进程调度初始化 sched_init 里为当前执行流添加了一个进程管理结构到 task 数组里，同时开启了定时器以及时钟中断的那一个时刻。



因为此时时钟中断到来之后，就可以执行到我们的进程调度程序，进程调度程序才会去这个 task 数组里挑选合适的进程进行切换。所以此时，我们当前执行的代码，才真正有了一个进程的身份，才勉强得到了一个可以被称为进程 0 的资格，毕竟还没有其他进程参与竞争。

如果你觉得这些话很困惑，就对了，在理解了整个这一块的细节之后，尤其是对于进程调度这种被人赋予了好多虚头巴脑的名词的地方，你会豁然开朗的。

第三句是 init

只有进程 1 会走到这个分支来执行。这里的代码可太多了，它本身需要完成如加载根文件系统的任务，同时这个方法将又会创建出一个新的进程 2，在进程 2 里又会加载与用户交互的 shell 程序，此时操作系统就正式成为了用户可用的一个状态了。

当然，当你知道了新进程诞生的过程之后，进程 2 的创建，就和进程 1 的创建一样了，在后面的章节中你将不会再困惑创建新进程的过程，减轻了学习负担。所以这一部分，又是作为下一部分的重要基础，环环相扣。

我们的教育，往往是强调知识点。但我认为，整个知识都是成体系的，没有哪个地方可以单点立起整个的理解大厦，必须一环扣一环。幸运的是，每一环都是十分简单且纯粹的。

第四句是 pause

当没有任何可运行的进程时，操作系统会悬停在这里，达到怠速状态。没啥好说的，我一直强调，操作系统就是由中断驱动的一个死循环。

一共四句话，切换到用户态，创建新进程，初始化，然后悬停怠速。

乍一看，是不是特别简单？是的，不过当你展开每一段代码的细节后你会发现，一个庞大的世界让你无从下手。但当你把全部细节都捋顺了之后你又会发现，不过如此。

欲知后事如何，且听下回分解。

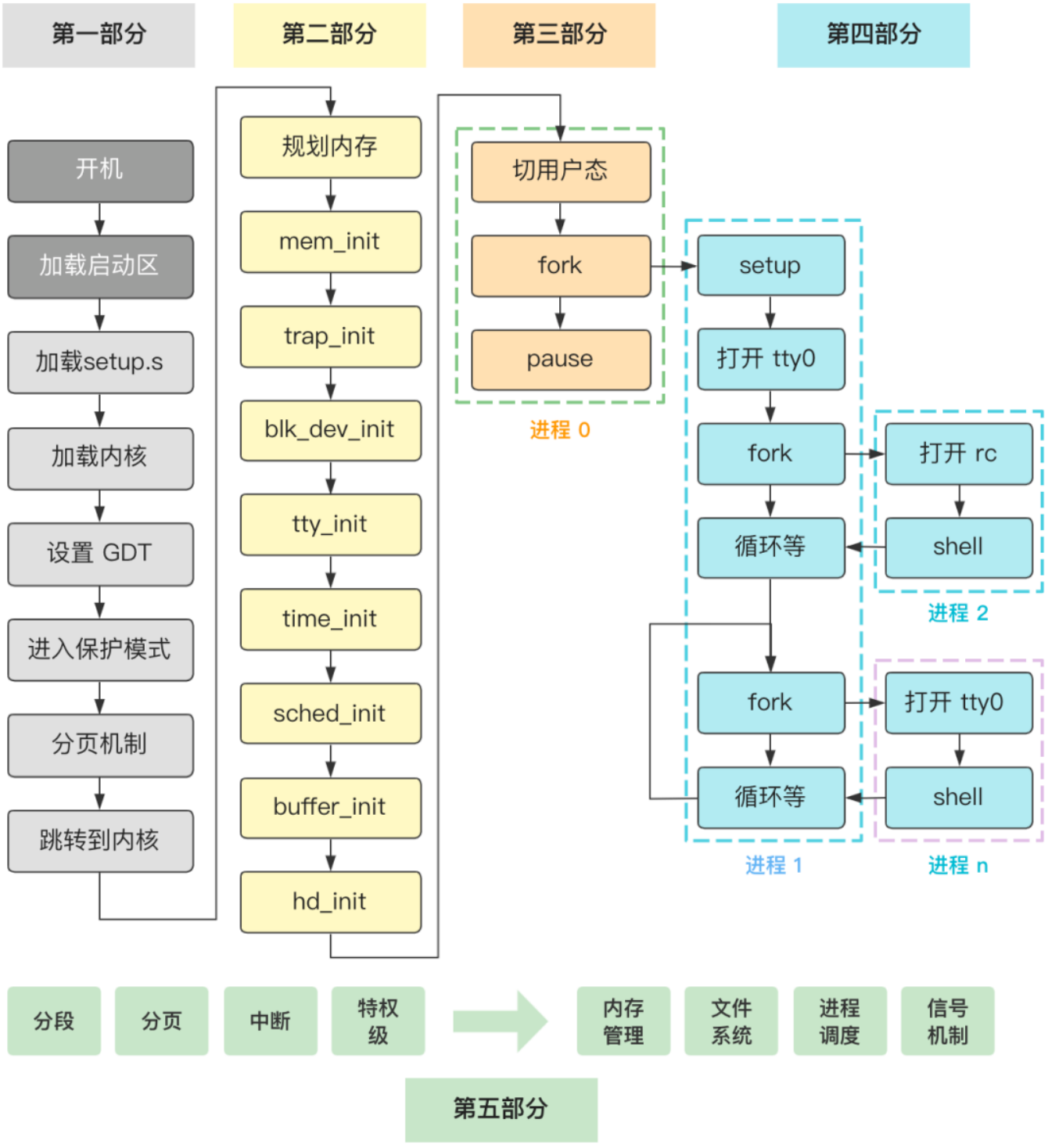
------- 关于本系列的完整内容 -------

本系列的开篇词看这

闪客新系列！你管这破玩意叫操作系统源码

本系列的扩展资料看这（也可点击**阅读原文**），这里有很多有趣的资料、答疑、互动参与项目，持续更新中，希望有你的参与。
https://github.com/sunym1993/flash-linux0.11-talk

本系列全局视角



最后，祝大家都能追更到系列结束，只要你敢持续追更，并且把每一回的内容搞懂，我就敢让你在系列结束后说一句，我对 Linux 0.11 很熟悉。

公众号更新系列文章不易，阅读量越来越低，希望大家多多传播，不方便的话点个小小的**在看**我也会很开心，我相信星火燎原的力量，谢谢大家咯。

另外，本系列**完全免费**，希望大家能多多传播给同样喜欢的人，同时给我的 GitHub 项目点个 star，就在**阅读原文**处，这些就足够让我坚持写下去了！我们下回见。

低并发编程

战略上藐视技术，战术上重视技术

175篇原创内容

Official Account

收录于合集 #操作系统源码 43

上一篇

第二部分完结撒花！大战前期的初始化工作

下一篇

一个新进程的诞生（二）从内核态到用户态

Read more