第12回 | 管理内存前先划分出三个边界值

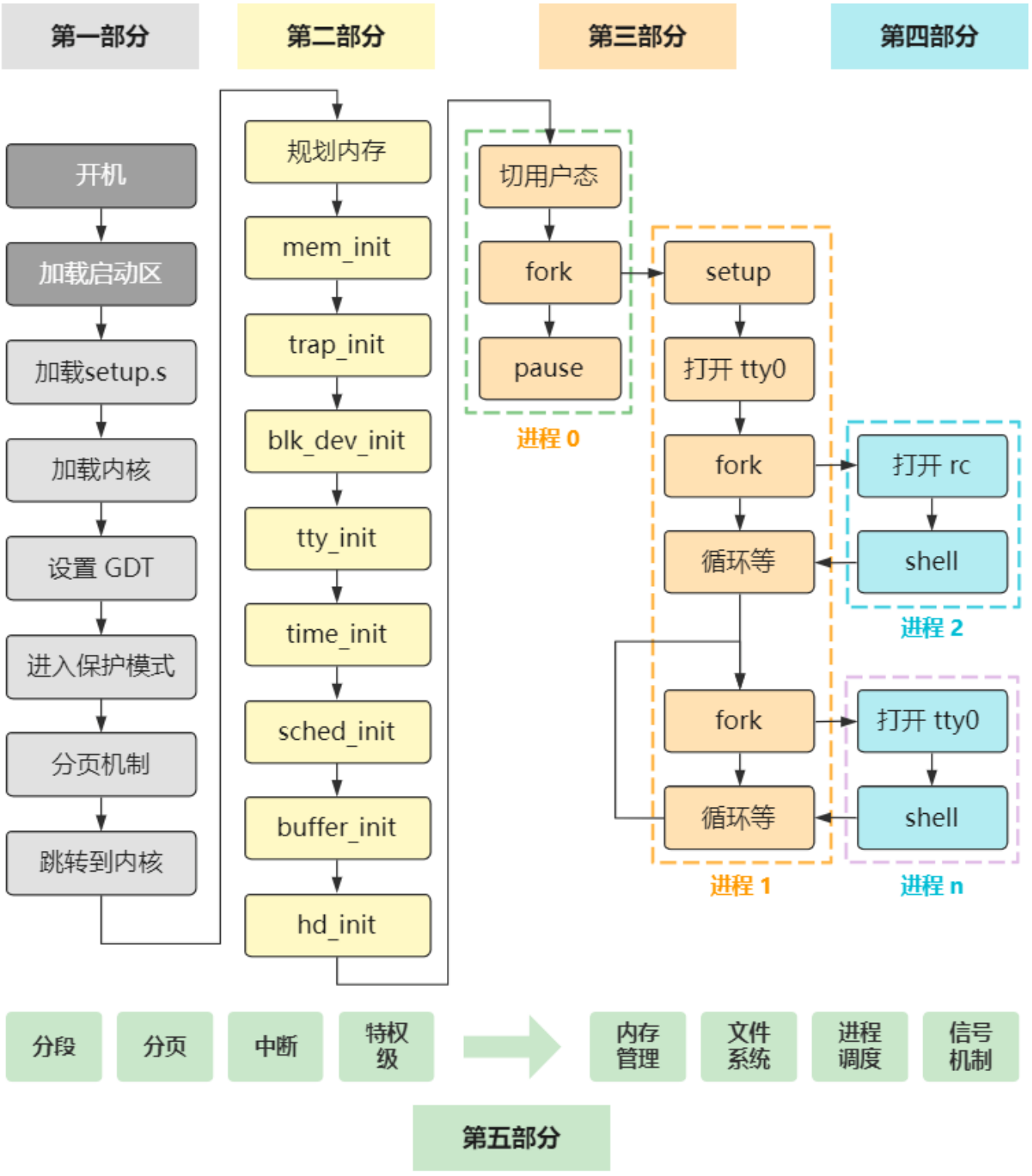
Original 闪客 低并发编程 2021-12-22 16:30

收录于合集
#操作系统源码

43个

新读者看这里，老读者直接跳过。

本系列会以一个读小说的心态，从开机启动后的代码执行顺序，带着大家阅读和赏析 Linux 0.11 全部核心代码，了解操作系统的技术细节和设计思想。



你会跟着我一起，看着一个操作系统从啥都没有开始，一步一步最终实现它复杂又精巧的设计，读完这个系列后希望你能发出感叹，原来操作系统源码就是这破玩意。

以下是**已发布文章**的列表，详细了解本系列可以先从开篇词看起。

开篇词

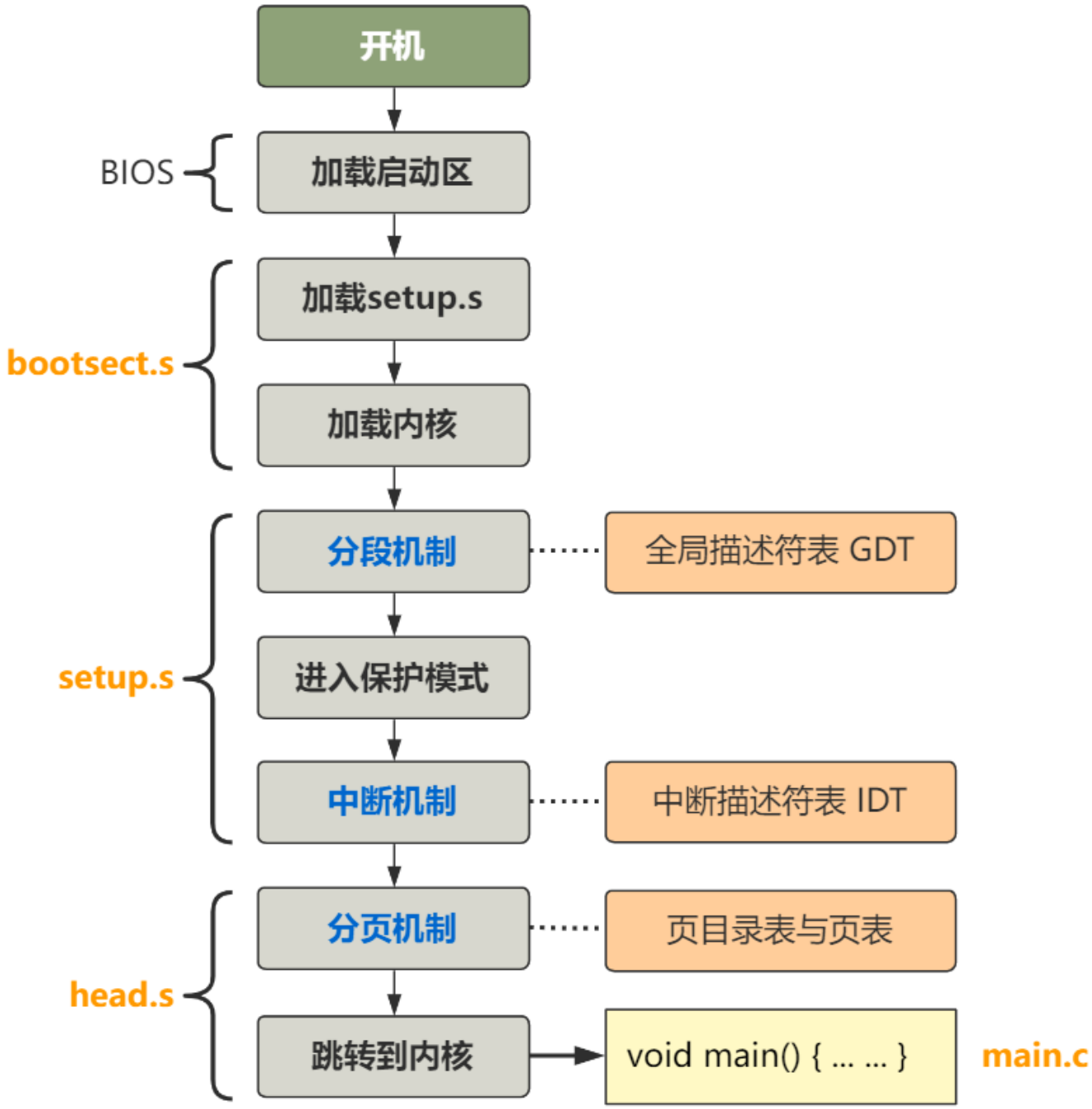
第一部分 进入内核前的苦力活

第二部分 大战前期的初始化工作

本系列的 GitHub 地址如下（文末阅读原文可直接跳转）
https://github.com/sunym1993/flash-linux0.11-talk

------- 正文开始 -------

书接上回，上回书咱们回顾了一下 main.c 函数之前我们做的全部工作，给进入 main 函数做了一个充分的准备。



那今天我们就话不多说，从 main 函数的第一行代码开始读。

还是把 main 的全部代码都先写出来，很少。

```
void main(void) {

    ROOT_DEV = ORIG_ROOT_DEV;
    drive_info = DRIVE_INFO;
    memory_end = (1<<20) + (EXT_MEM_K<<10);
    memory_end &= 0xfffff000;
    if (memory_end > 16*1024*1024)
        memory_end = 16*1024*1024;
    if (memory_end > 12*1024*1024)
        buffer_memory_end = 4*1024*1024;
    else if (memory_end > 6*1024*1024)
        buffer_memory_end = 2*1024*1024;
    else
        buffer_memory_end = 1*1024*1024;
    main_memory_start = buffer_memory_end;

    mem_init(main_memory_start,memory_end);
    trap_init();
    blk_dev_init();
    chr_dev_init();
    tty_init();
    time_init();
    sched_init();
    buffer_init(buffer_memory_end);
    hd_init();
    floppy_init();

    sti();
    move_to_user_mode();
    if (!fork()) {      /* we count on this going ok */
        init();
    }

    for(;;) pause();
}
```

我们今天就看这第一小段。

首先，ROOT_DEV 为系统的根文件设备号，drive_info 为之前 setup.s 程序获取并存储在内存 0x90000 处的设备信息，我们先不管这俩，等之后用到了再说。

我们看后面这一坨很影响整体画风的一段代码。

```
void main(void) {
    ...
    memory_end = (1<<20) + (EXT_MEM_K<<10);
    memory_end &= 0xfffff000;
    if (memory_end > 16*1024*1024)
        memory_end = 16*1024*1024;
    if (memory_end > 12*1024*1024)
        buffer_memory_end = 4*1024*1024;
    else if (memory_end > 6*1024*1024)
        buffer_memory_end = 2*1024*1024;
    else
        buffer_memory_end = 1*1024*1024;
    main_memory_start = buffer_memory_end;
    ...
}
```

这一坨代码和后面规规整整的 xxx_init 平级的位置，要是我们这么写代码，肯定被老板批评，被同事鄙视了。但 Linus 写的，就是经典，学就完事了。

这一坨代码虽然很乱，但仔细看就知道它只是为了计算出三个变量罢了。

main_memory_start
memory_end
buffer_memory_end

而观察最后一行代码发现，其实两个变量是相等的，所以其实仅仅计算出了两个变量。

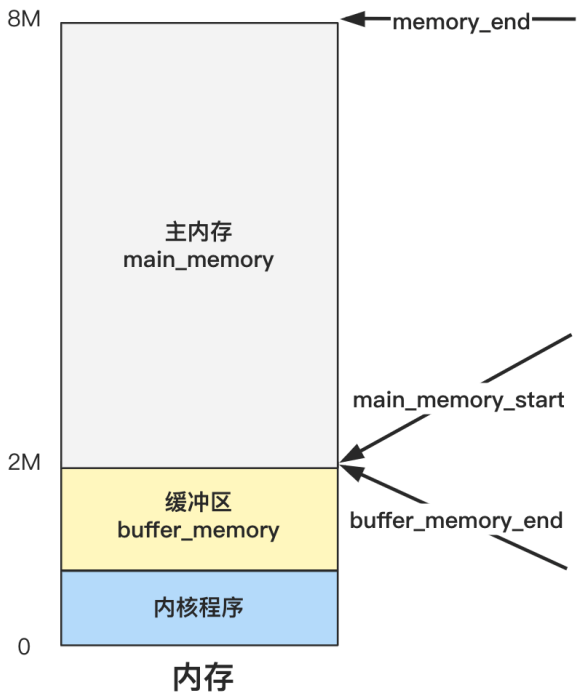
main_memory_start
memory_end

然后再具体分析这个逻辑，其实就是一堆 if else 判断而已，判断的标准都是 memory_end 也就是内存最大值的大小，而这个内存最大值由第一行代码可以看出，是等于 1M + 扩展内存大小。

那 ok 了，**其实就只是针对不同的内存大小，设置不同的边界值罢了**，为了理解它，我们完全没必要考虑这么周全，就假设总内存一共就 **8M** 大小吧。

那么如果内存为 8M 大小，**memory_end** 就是
8 * 1024 * 1024
也就只会走倒数第二个分支，那么 **buffer_memory_end** 就为
2 * 1024 * 1024
那么 **main_memory_start** 也为
2 * 1024 * 1024

那这些值有什么用呢？一张图就给你说明白了。



你看，其实就是定了三个箭头所指向的地址的三个边界变量，具体主内存区是如何管理和分配的，要看下面代码的功劳。

```
void main(void) {
    ...
    mem_init(main_memory_start, memory_end);
    ...
}
```

而缓冲区是如何管理和分配的，就要看

```
void main(void) {
    ...
    buffer_init(buffer_memory_end);
    ...
}
```

是如何折腾的了。

那我们今天就不背着这两个负担了，仅仅需要知道这三个参数的计算，以及后面是为谁效力的，就好啦，是不是很轻松？后面我们再讲，如何利用这三个参数，来做到内存的管理。

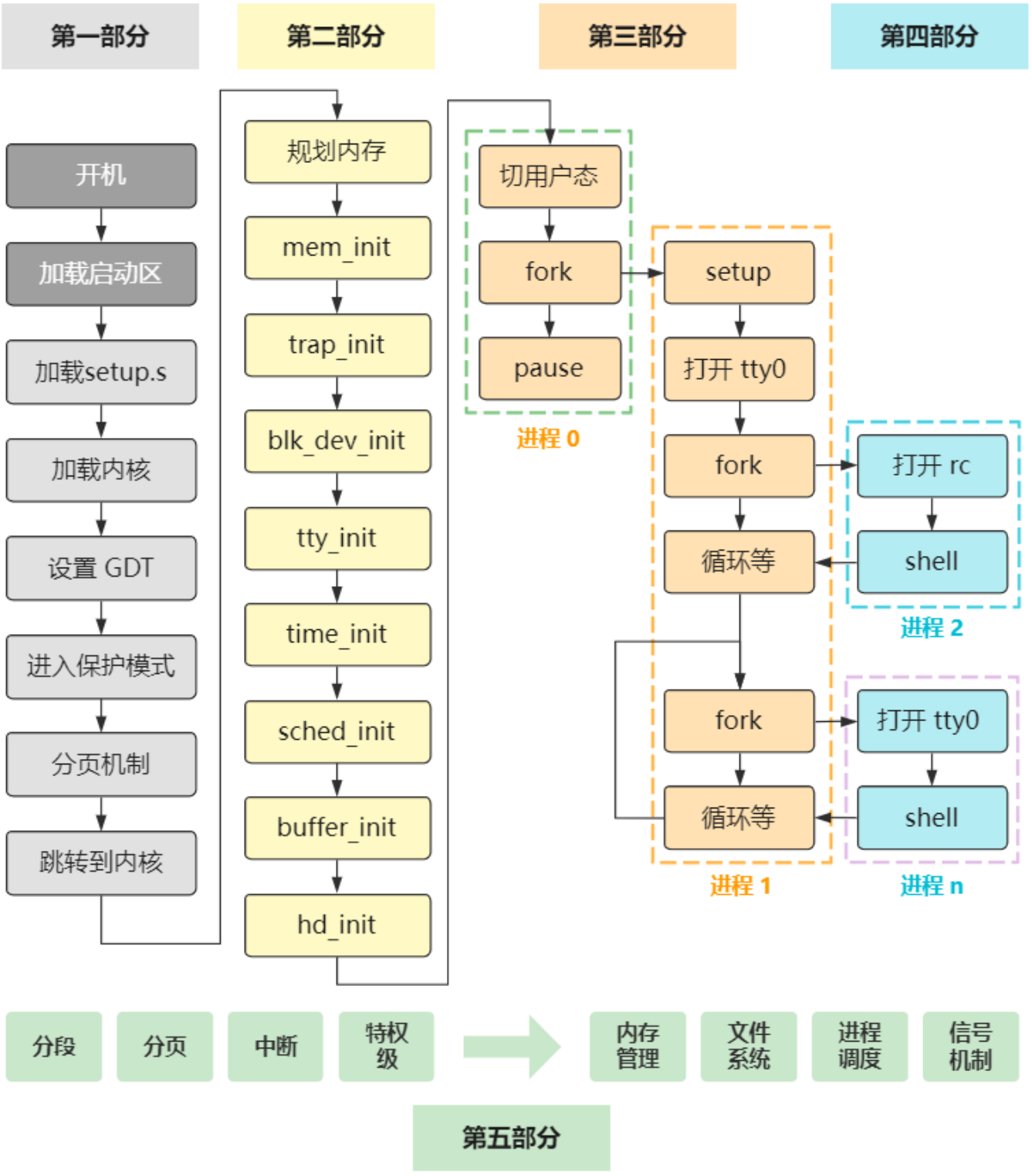
预知后事如何，且听下会分解。

------- 关于本系列 -------

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。

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11回 | 整个操作系统就 20 几行代码

下一篇
操作系统就用一张大表管理内存?

Read more