# 第11回 | 整个操作系统就 20 几行代码

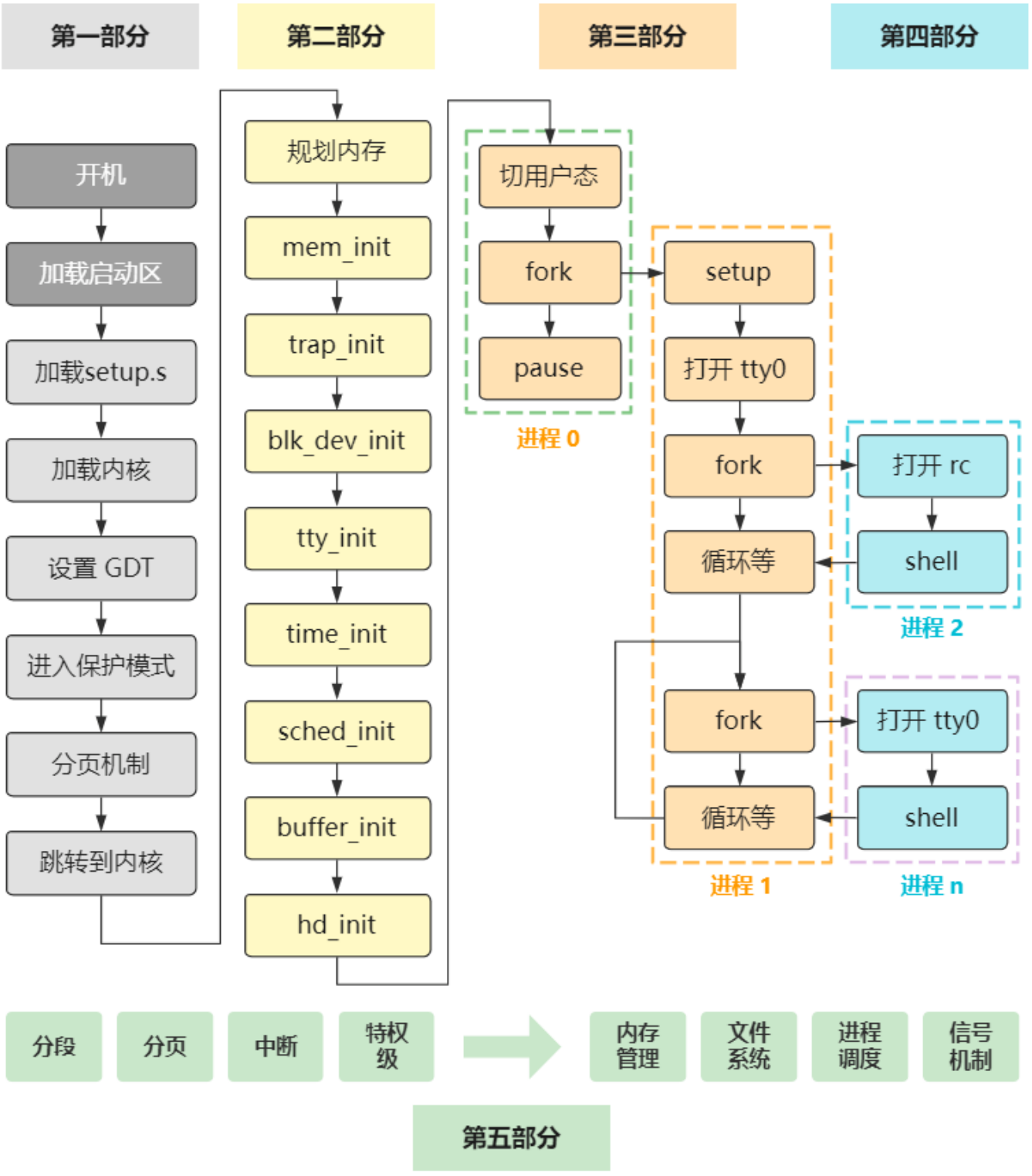
Original 闪客 低并发编程 2021-12-19 16:30

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

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 -------**

第二部分正式开始啦！

在第一部分，用了总共十回的篇章，把进入 main 方法前的苦力工作都完成了，我们的程序终于跳到第一个由 c 语言写的，也是操作系统的全部代码骨架的地方，就是 main.c 文件里的 main 方法。

```
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()) {
        init();
    }

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

数一数看，总共也就 20 几行代码。

但这的确是操作系统启动流程的全部秘密了，我用空格将这个代码分成了几个部分。

第一部分是一些参数的取值和计算。

```
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;
    ...
}
```

包括**根设备 ROOT_DEV**，之前在汇编语言中获取的各个设备的**参数信息 drive_info**，以及通过计算得到的**内存边界**
main_memory_start
main_memory_end
buffer_memory_start
buffer_memory_end

从哪获得之前的设备参数信息呢？如果你前面看了，那一定还记得这个表，都是由 setup.s 这个汇编程序调用 BIOS 中断获取的各个设备的信息，并保存在约定好的内存地址 0x90000 处，现在这不就来取了么，我就不赘述了。

| 内存地址 | 长度(字节) | 名称 |
| --- | --- | --- |
| 0x90000 | 2 | 光标位置 |
| 0x90002 | 2 | 扩展内存数 |
| 0x90004 | 2 | 显示页面 |
| 0x90006 | 1 | 显示模式 |
| 0x90007 | 1 | 字符列数 |

| | | |
|---|---|---|
| 0x90008 | 2 | 未知 |
| 0x9000A | 1 | 显示内存 |
| 0x9000B | 1 | 显示状态 |
| 0x9000C | 2 | 显卡特性参数 |
| 0x9000E | 1 | 屏幕行数 |
| 0x9000F | 1 | 屏幕列数 |
| 0x90080 | 16 | 硬盘1参数表 |
| 0x90090 | 16 | 硬盘2参数表 |
| 0x901FC | 2 | 根设备号 |

第二部分是各种初始化 **init** 操作。

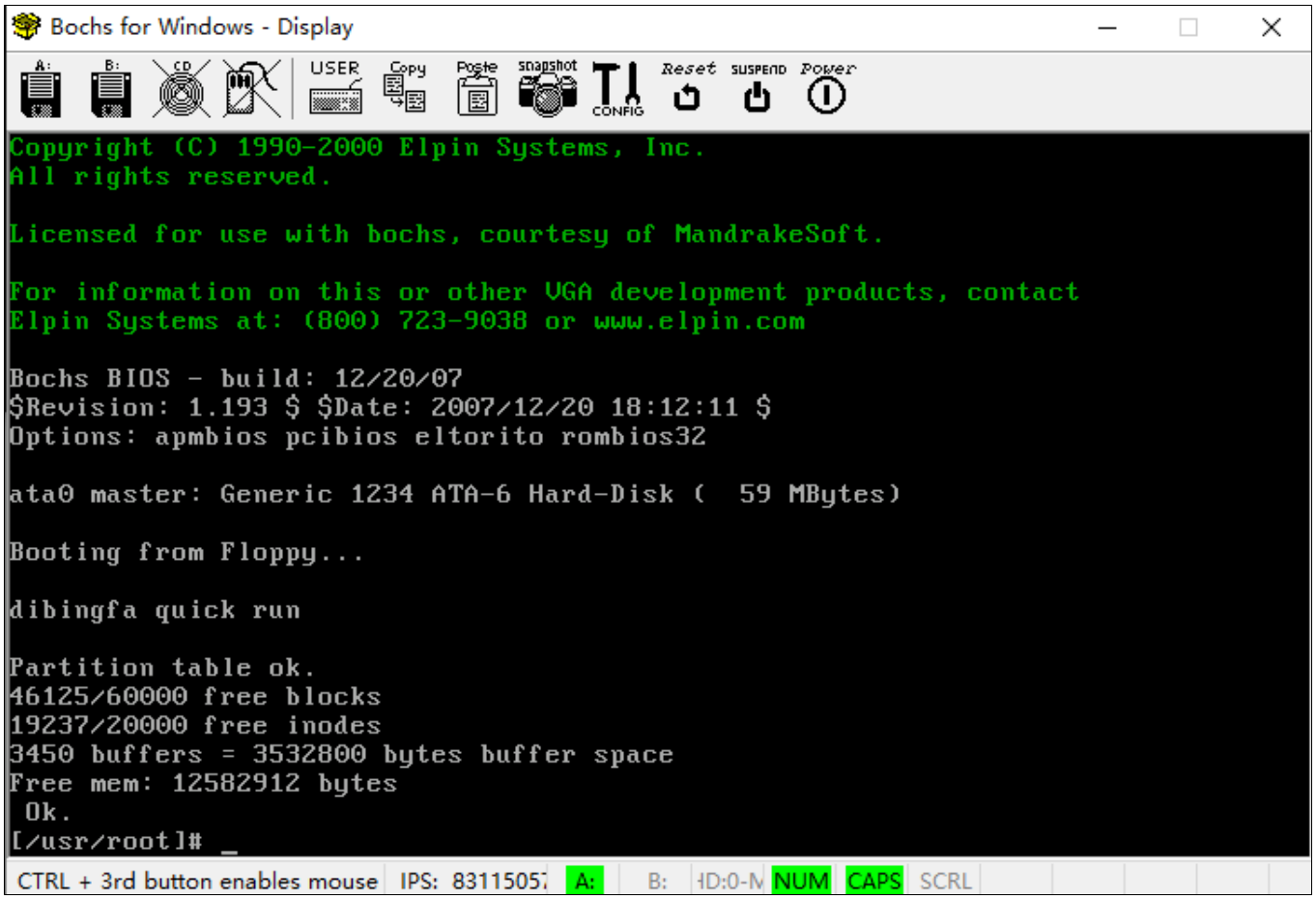
```
void main(void) {
    ...
    mem_init(main_memory_start,memory_end);
    trap_init();
    blk_dev_init();
    chr_dev_init();
    tty_init();
    time_init();
    sched_init();
    buffer_init(buffer_memory_end);
    hd_init();
    floppy_init();
    ...
}
```

包括**内存初始化 mem_init**，**中断初始化 trap_init**、**进程调度初始化 sched_init** 等等。我们知道学操作系统知识的时候，其实就分成这么几块来学的，看来在操作系统源码上看，也确实是这么划分的，那我们之后照着源码慢慢品，就好了。

第三部分是**切换到用户态模式**，并在一个新的进程中做一个最终的初始化 init。

```
void main(void) {
    ...
    sti();
    move_to_user_mode();
    if (!fork()) {
        init();
    }
    ...
}
```

这个 init 函数里会创建出一个进程，设置终端的标准 IO，并且再创建出一个执行 shell 程序的进程用来接受用户的命令，到这里其实就出现了我们熟悉的画面（下面是 bochs 启动 Linux 0.11 后的画面）。

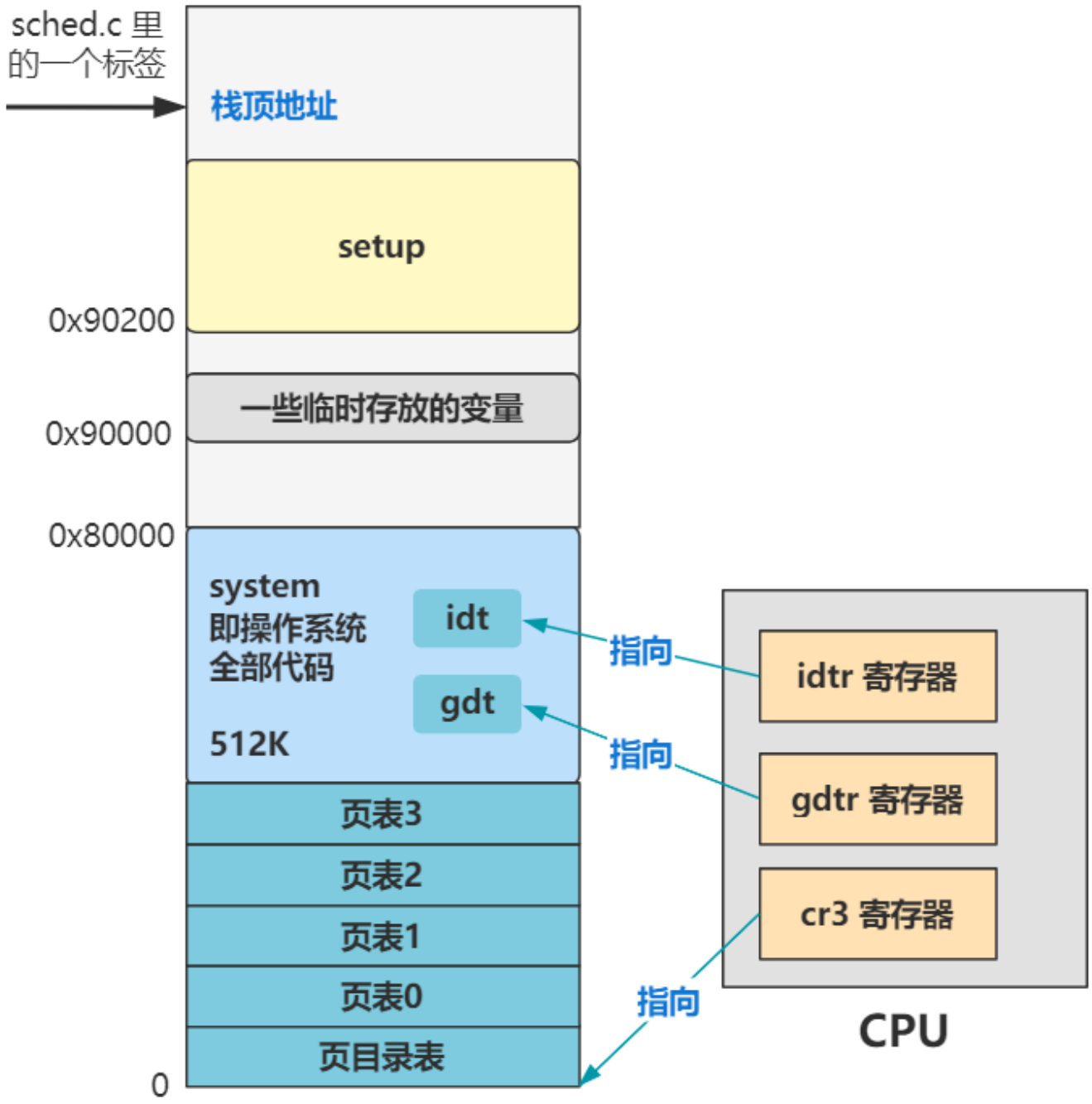



第四部分是个**死循环**，如果没有任何任务可以运行，操作系统会一直陷入这个死循环无法自拔。

```
void main(void) {
    ...
    for(;;) pause();
}
```

OK，不要细品每一句话，我们本回就是要你有个整体印象，之后会细细讲这里的每一个部分。

这里再放上目前的内存布局图。

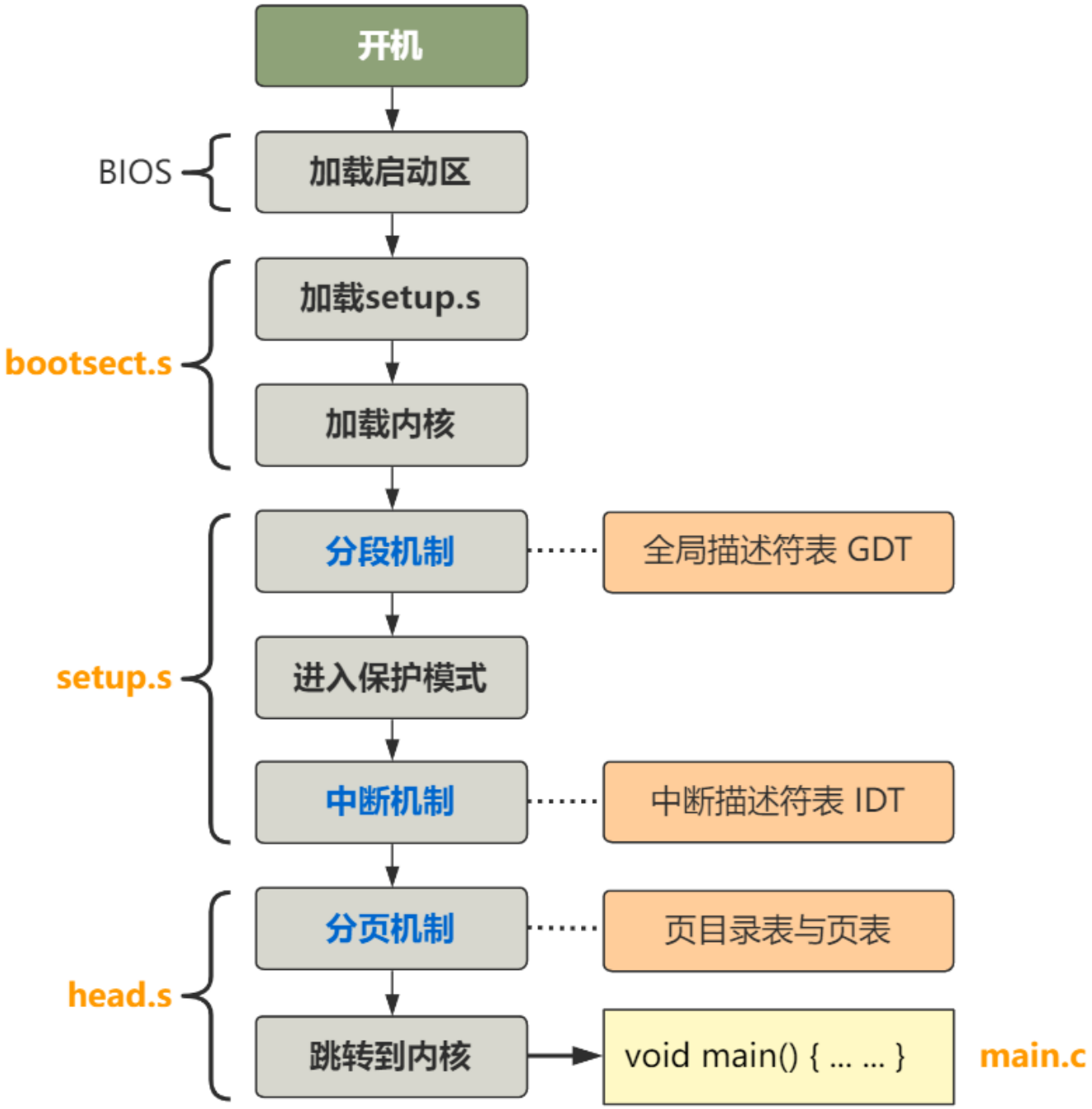



这个图大家一定要牢记在心，操作系统说白了就是在内存中放置各种的数据结构，来实现“管理”的功能。

所以之后我们的学习过程，主心骨其实就是看看，操作系统在经过一番折腾后，又在内存中建立了什么数据结构，而这些数据结构后面又是如何用到的。

比如进程管理，就是在内存中建立好多复杂的数据结构用来记录进程的信息，再配合上进程调度的小算法，完成了进程这个强大的功能。

为了让大家目前心里有个底，我们把前面的工作再再再再在这里做一个回顾，用一张图表示就是：




看到了吧，我们已经把 boot 文件夹下的三个汇编文件的全部代码都一行一行品读过了，其主要功能就是三张表的设置：全局描述符表、中断描述符表、页表。同时还设置了各种段寄存器，栈顶指针。并且，还为后续的程序提供了设备信息，保存在 0x90000 处往后的几个位置上。

最后，一个华丽的跳转，将程序跳转到了 main.c 文件里的 main 函数中。

所以，本讲就是让大家深呼吸，把之前的准备工作再消化消化。如果第一部分全部认真看过的同学，必定觉得这一回是废话。

如果你不这样觉得，那就得再回去重新梳理一边咯，如果有不会的，赶紧查资料搞懂它，因为之后要打一系列的硬仗了！根基不稳，地动山摇！

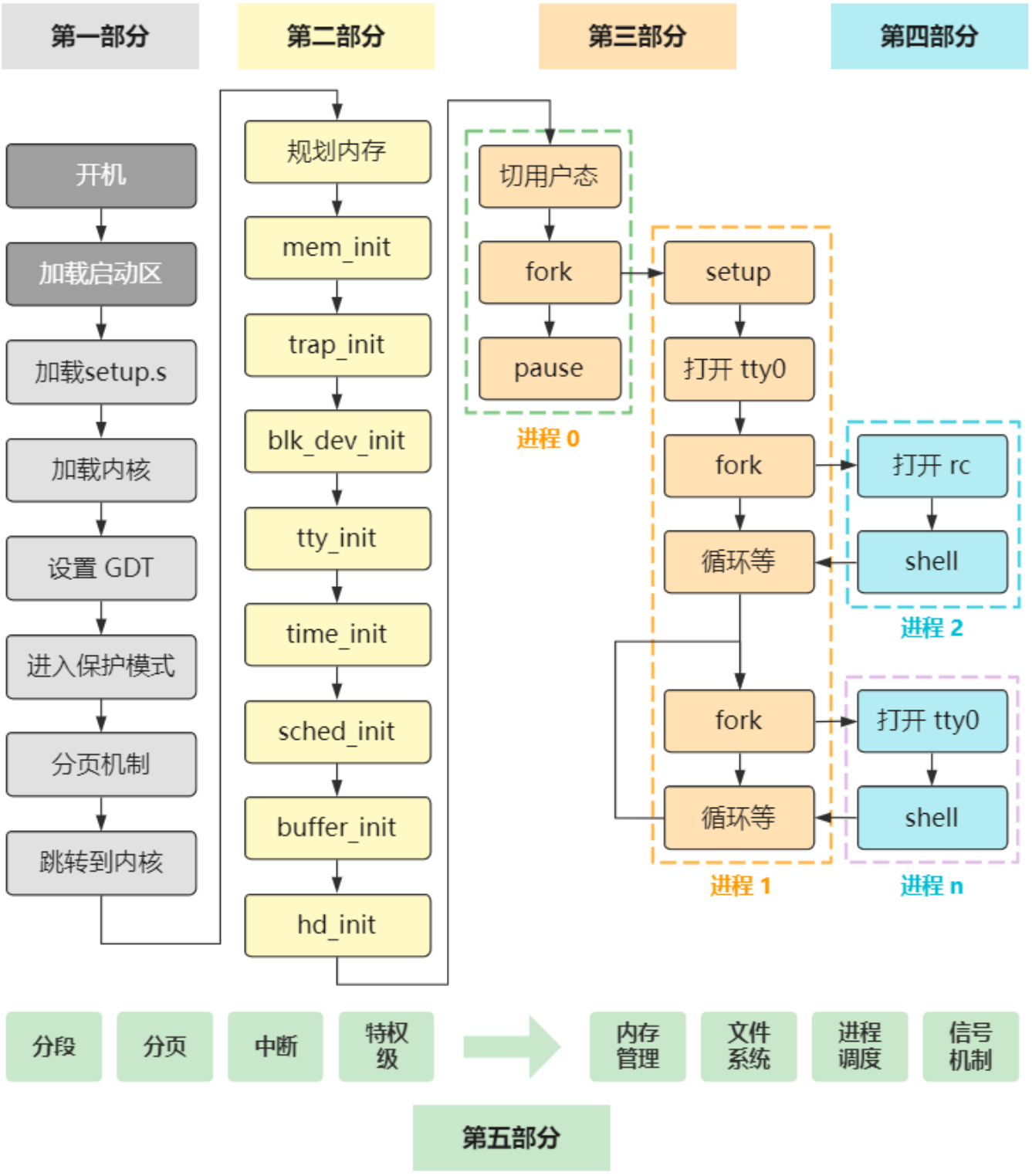
预知后事如何，且听下回分解。

**------- 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
第一部分完结 进入内核前的苦力活

下一篇
第12回 | 管理内存前先划分出三个边界值

Modified on 2021-12-19

Read more

People who liked this content also liked

10.重构改善既有代码的设计(2)—简化条件逻辑
尹先文

四行代码创建复杂（无限级）树
DotNet开源大全

设计原型与代码的实现
探探设计