一个简单的内核的探索

根据wikipedia给出的解释来看，所谓操作系统：

An operating system (OS) is system software that manages computer hardware and software resources and provides common services for computer programs. All computer programs, excluding firmware, require an operating system to function.

所以可以说操作系统是直接与底层硬件打交道的一个“程序”，因此对它的研究也会让我们对计算机的运行，组成，有着更加深刻的了解与认识。

一般来说，操作系统具有这样的功能：

* 进程管理（Processing management）
* 内存管理（Memory management）
* 文件系统（File system）
* 网络通信（Networking）
* 安全机制（Security）
* 用户界面（User interface）
* 驱动程序（Device drivers）

这些功能中，对于我们学习有帮助的，主要是进程管理，内存管理，文件系统。这是一个操作系统比较核心的内容，现代的操作系统课程也会把重点放在这些地方，特别是进程之间的转换，锁死等内容，更是很多操作系统课程最开始必讲的内容。

但对于我们来说，这么一个简单的内核，只能是用最基本的代码，实现最基本的功能，相比起MIT的xv6和Stanford的Pintos我们差得远了。我们的重点主要在于底层结构的探索，在掌握了计算机组成原理的情况下，明白它是如何从加电到实现这些功能的。

关于内存管理，我们做一个比较详细的实现，而为了运行一些硬件，我们也必须设计几个驱动程序，在最后我们会做一个简单的进程切换。

其它的，像是文件系统，也是操作系统的重点，但由于我们使用的是1.44MB的软盘来装我们的操作系统，实在没有多余空间实现文件的管理了（事实上，我们最后编译出来的内核几乎已经把它占满了）。至于网络通信，我还没开始学习，而安全机制也不在我近期学习计划中，用户界面对于我就更是个ostentatious的事物了。还是老老实实从基础开始吧：

作为一个操作系统，它最大的特点，就是能够直接在裸机上运行，但这并不意味着操作系统就是计算机启动的第一道工序了。实际上，计算机启动过程中，是包含如下几个部分的（以下基于32位操作系统来叙述）：

当按下电源的时候，会发送一个重置信号给CPU 🡪CPU将寄存器设为预设值🡪然后开始执行指令🡪第一条执行的指令地址是0xFFFF0🡪这个地址指向BIOS🡪BIOS运行POST（Power-On Self Record）🡪 检测之后，没有问题，跳到MBR(Master Boot Record)

前面的部分都是计算机的标准，我们无法改变，也不需要了解深入，直接从BIOS看起。

BIOS是一个程序员，或者说是电脑工作者，在电脑启动后第一个接触的部分了。所谓BIOS，指的是在ROM里面存储的一个小程序，它可以设定计算机的一些硬件状态（也就是CMOS参数），以及设定我们的启动顺序等。并在这些设定完成后，将操作转移到MBR。

等等，那为什么这个地址指向BIOS呢？这就需要明白PAS（物理地址空间），所谓Physical Address Space的内容了。所谓的物理地址空间，定义了所有可以获取的地址，这不仅仅包括了物理内存（RAM），还包括了硬件设备，甚至有些地址处什么也没有，但这样的地址也是存在的。但无论怎样的地址，都是可以通过指针来加以获取的。

首先，BIOS如何确定这一个分区是否可以引导呢？这个是通过一个魔数(Magic number，在计算机里经常用到的一个术语，往往是一些特定的数字，用以提供必要的信息)来判断的。如果一个存储设备的第一个扇区中偏移510和511字节处的byte是0x55和0xAA，那么这个分区就是引导分区。

好，现在，BIOS找到了这个内容了，找到之后，BIOS会将这个扇区，也就是512字节的内容，加载到内存0x0000:0x7c00处，然后开始执行我们的代码。（关于这个地址，参考<https://www.glamenv-septzen.net/en/view/6>）

那么问题来了，这第一步加载的内容实在太少了，很难把我们的操作系统的关键说出来，一个解决办法是写一个引导程序，通过这个引导程序引导我们的系统。但如果要实现一个好一点的引导程序，比如支持multiboot的引导程序，这个小的扇区还是太小了。所以这里又引入了双阶段的引导，从这个扇区引导到我们的引导程序，再由引导程序选择要引导的系统。现在主要的BootLoader都是这样实现的，比如接下来要说的GRUB(这个有一定争议的地方在于，GRUB实现引导可以看成三阶段引导)。

这里我实现的这个小的内核，就是用了GRUB来做引导，因为写一个BootLoader并不容易，而且更多的是需要一些文档规范的知识，对操作系统的研究帮助不大。

所以从这里开始，就直接进入了具体的实现。

首先，我们需要做的是，要有一个可以引导的磁盘，这里采用1.44MB的软盘作为我们的磁盘。这个磁盘里面需要装一个GRUB，并且设定好它的扇区。(之所以使用软盘，也是因为)

这就是我们写系统的第一个步骤了。

但这里我并不准备具体实现它，而是采用JamesM先生已经实现好的这个小磁盘。

具体实现的话，链接如下：

<https://www.gnu.org/software/grub/manual/html_node/Making-a-GRUB-bootable-CD_002dROM.html>

我们从GRUB加载的程序开始来实现。主要的部分用C语言，但也有很多部分用汇编来写。

http://www.engineersgarage.com/tutorials/how-computer-pc-boots-up