**读Kernel感悟-Linux内核启动-内核的生成**

2010年11月03日 22:02:00 [iteye\_20954](https://me.csdn.net/iteye_20954) 阅读数：2

这段时间在看《**[Linux内核](http://www.top-e.org/jiaoshi/class/?62.html" \t "_blank)**源代码情景分析》，顺便写了一些感悟。读内核源代码是一件很有意思的事。它像一条线，把操作系统，编译原理，C语言，数据结构与算法，计算机体系结构等等计算机的基础课程串起来。

我看内核源代码是用lxr+glimpse（不一定要自己架，可以直接访问校内外的lxr网站）的。如果在windows下也可以用sourceinsight。以下的当前路径为内核源代码路径，通常为/usr/src/linux。内核版本为2.6.13，平台为x86

好，让我们开始**[Linux内核](http://www.top-e.org/jiaoshi/class/?62.html" \t "_blank)**之旅。

我们的出发点是在CPU加电的一刹那，系统处于16位实地址模式下，终点是内核开始运行start\_kernel()，系统处于32位页式寻址的保护模式下。那时内核映象bzImage已经解压完毕，运行于内核态。系统中已经有了一个叫swapper的0号进程，有自己的内核堆栈，情况就相对好理解得多。（尽管与用户态程序相比，还要多操心不少事，包括对硬件的直接操作，内核态各种数据结构的初始化，对页表的操作等等）。不过，不妨先做些准备动作。

首先，什么是内核？目前，只知道编译内核后，产生一个叫bzImage的压缩内核映象。它不同于任何普通的可执行程序。我们甚至不知道它从哪里开始执行。只知道把它往/boot/下一放，往bootloader的配置文件(例如grub的menu.lst)中写上相关信息，机子就顺利启动了。因此，我对它的生成过程产生了浓厚兴趣。于是，我查看了相关资料，最直接的资料来自于arch/i386/boot/下的Makefile。从Makefile中可以知道。bzImage的产生过程是这样的：不过我不满足于此。于是，我想到了去看arch/i386/boot/下的Makefile。从arch/i386/boot/Makefile和arch/i386/boot/compressed/Makefile中可以看出（具体过程省略，）

1.先生成vmlinux.这是一个elf可执行文件

2.然后objcopy成arch/i386/boot/compressed/vmlinux.bin，去掉了原elf文件中的一些无用的section等信息。

3.gzip后压缩为arch/i386/boot/compressed/vmlinux.bin.gz

4.把压缩文件作为数据段链接成arch/i386/boot/compressed/piggy.o

5.链接：arch/i386/boot/compressed/vmlinux=head.o+misc.o+piggy.o

其中head.o和misc.o是用来解压缩的。

6.objcopy成arch/i386/boot/vmlinux.bin，去掉了原elf文件中的一些无用的section等信息。

7.用arch/i386/boot/tools/build.c工具拼接**bzImage=bootsect+setup+vmlinux.bin**

过程好复杂。

这里要介绍一下objcopy命令，它的作用是把一个object文件转化为另一种格式的文件。在这里，objcopy的作用就是去掉原来elf文件中的elfheader和一些无用的section信息。为什么要这么做呢？因为elf文件中的elfheader和一些section的作用是告诉elfloader如何载入elf可执行文件。但是，linux内核作为一种特殊的elf文件，需要特殊的辅助程序去装载它。往往它的装载地址是固定的。这时，为了保证通用性而存在的elfheader和一些section对内核的装载就没有意义了。加上为了使内核尽可能小，所以干脆把这些信息去掉。

我们可以看一下vmlinux和arch/i386/boot/compressed/vmlinux。用file命令查看，它们也是elf可执行文件。只是没有main函数而已

参考：

Documentation/kbuild/makefiles.txt

Documentation/kbuild/modules.txt