

## 第 7 章 FreeBSD 启动过程

(翻译中出现的任何问题或错误，请广大读者及时反馈给我：freebsdhandbook@163.com)

### 7.1 概要

启动一个电脑和加载操作系统的过程被叫做“bootstrap process”或简单地叫“booting”。FreeBSD 的启动过程提供了许多弹性来适应实际的变化，允许你选择启动在同一台电脑上安装的不同操作系统，或是同一操作系统的不同版本。

这章将比较细致地对你可能会涉及到的配置选项和如何定制 FreeBSD 的启动过程作一个描述。这包括内核启动前会发生的每一件事情，探测设备，启动 `init`。如果你不十分有把握的话，当你的屏幕的文本颜色由白变灰的时候，你就可以看到系统检测信息了。

阅读完这章，你将了解到：

- FreeBSD 的 bootstrap 系统有哪些组件，它们如何互相影响。
- 在启动 FreeBSD 时你可以给组件哪些选项来控制启动过程。

**注意：**这章将只描述运行在 Intel x86 系统上的启动过程。

### 7.2 启动过程中的问题

打开电脑和启动操作系统似乎会引起一个两难的选择。根据定义，电脑在操作系统启动之前是不知道如何做事的。这包括从磁盘运行程序。所以如果电脑没有操作系统就不能运行程序，那操作系统是如何启动的呢？

在 x86 系统中，主要由基本输入输出系统 (BIOS) 来负责加载操作系统。BIOS 首先检查磁盘的主引导区 (MBR)，它是在磁盘的一个比较特殊的地方。BIOS 加载和运行 MBR，而 MBR 能完成加载操作系统以外的任务。

如果你只有一个操作系统安装在电脑上，那标准的 MBR 将起作用。这个 MBR 会搜索磁盘上的启动 slice，然后在这个 slice 上运行代码来加载引导操作系统的余下的部分程序。

如果你在磁盘上安装了多个操作系统，你可以安装一个不同的 MBR，MBR 可以显示一个不同操作系统的列表，允许你选择一个启动。FreeBSD 使用这样的 MBR，其他操作系统也会提供其它的 MBR。

FreeBSD 的 bootstrap 系统的其余部分被分成三个阶段。第一个阶段是运行 MBR，它只知道把电脑带入一个特殊的状态，然后运行第二阶段。第二阶段要执行的程序要多一点。第三阶段就完成加载操作系统的任务。这个工作被分成三个阶段是因为 PC 的标准限制了程序的大小。这一系列串起来的任务允许 FreeBSD 提供一个比较灵活的加载程序。

内核启动后，就会探测设备并对它们进行初始化，一旦内核启动过程完成，内核就把控制权交给用户处理进程 `init`，它可以确定磁盘是否处于可用状态。`init` 接着就启动用户级资源配置来加载文件系统，设置网卡来激活网络，接着就启动所有的通常在 FreeBSD 一运行就启动的进程。

### 7.3 MBR 启动步骤 1、2 和 3

#### 7.3.1 MBR, `/boot/boot0`

FreeBSD MBR 是驻留在 `/boot/boot0`。这是 MBR 的一个拷贝，因为真正的 MBR 必须被放置在磁盘的一个特殊部分，在 FreeBSD 区域的外面。`boot0` 非常简单，在主引导区的程序可能只有 512 个字节。如果你已经安装了 FreeBSD 的 MBR，而且还安装了多个操作系统，你将会在启动时看到一个比较熟悉的画面。

#### 例 7-1. `boot0` 画面

F1 DOS

F2 FreeBSD

F3 Linux

F4 ??

F5 Drive 1

Default: F2

其它操作系统，如 windows 95，会用自己的 MBR 来改写已存在的 MBR。如果碰到这样的事，或你想替换 FreeBSD 的 MBR，你可以使用下面的命令：

```
# fdisk -B -b /boot/boot0 device
```

这儿的 *device* 是你用来启动的设备，如第一个 IDE 磁盘 ad0，在第二个 IDE 控制器上的第一个磁盘，在第一个 SCSI 磁盘 da0 等等。

### 7.3.2 步骤 1, /boot/boot1, 步骤 2, /boot/boot2

Boot1 可以在引导分区的引导扇区上找到，它无论是在 boot0 上，或是在 MBR 上的其它程序都能找到这样的程序继续启动进程。

Boot1 是非常简单的，它也可能只有 512 字节，只是表明了 FreeBSD 的存储有磁盘分区信息的磁盘标签 (Disklabel)，找到它之后，就会执行 boot2。

Boot2 稍微有点复杂，它记录着 FreeBSD 的文件系统以便于在它上面找到文件，另外它也提供了一个选择可运行的内核或引导程序的简单接口。Loader 还要复杂一点，它提供了一个易于使用的启动配置信息，通常 boot2 之后就是运行它了，但以前它可以直接运行内核。

#### 例 7-2. boot2 画面

```
>> FreeBSD/i386 B00T

Default: 0: ad(0,a)/kernel

boot:
```

如果你需要替换已安装的 boot1 和 boot2，可以使用 disklabel。

```
# disklabel -B disklice
```

*disklice* 是启动系统的磁盘和 slice 的地方，如 ad0s1，在第一个 IDE 磁盘的第一个 slice 上。

危险的模式：如果你使用了刚才的磁盘名称，如 ad0，在 disklabel 命令中，你将创建一个危险的磁盘，没有 slices。这可能不是你想要做的，所以确定你在键入 RETURN 之前，

你已反复检查了命令 `disklabel`。

### 7.3.3 步骤 3, /boot/loader

引导程序 loader 是三步中的最后一步，它可能在 `/boot/loader`。Loader 有一个非常友好的配置方法，使用一个易于使用的内建命令，通过一个强大的接口来备份。

#### 7.3.3.1 Loader 的执行过程

初始化过程中，引导程序探测到一个控制台和一些磁盘，计算出从哪个磁盘启动。因此，它可以设置成可变化的，然后解释程序就开始启动，命令就会被解释执行。

最后，默认情况下启动程序停顿 10 秒钟（当然你可以按任意键继续），然后启动内核。如果进程被打断了，用户可以使用命令来调整参数，卸载或装载模块，最后启动或重新启动。一个更深入的技术讨论，你可以阅读 loader 的联机手册。

#### 7.3.3.2 Loader 内建命令

这些命令集包括：

`autoboot seconds`

在规定的时间内不被打断的话，继续启动内核。如果显示一个倒计时，默认的时间是 10 秒钟。

`boot [-options] [kernel name]`

直接配合所给的参数加载内核。

`boot-conf`

在启动时，使用自动的变量配置模块，这只在你先用 `unload` 时才有意义，并改变一些变量，通常是 `kernel`。

`help [topic]`

显示来自 `/boot/loader.help` 的求助信息，如果所给的主题(topic)是 `index`，那就显示所有的主题列表。

`include filename ...`

执行所给的文件，这文件将被读入并一行一行地执行，一旦有错误发生就直接停止这个 `include` 命令。

`load [-t type] filename`

加载 `kernel`，`kernel` 模块或原先指定的文件类型，紧接着文件名。任何跟在文件之后的参数都将会传给这个文件来执行。

`ls [-l] [path]`

列出在给定路径中的文件。如果没有指定路径，将显示 `root` 目录的文件列表。如果有附加 `-l` 参数，那么将一起显示文件大小。

`lsdev [-v]`

列出所有可以加载 `module` 的设备，如果指定 `-v` 参数，那么会列出更详细的信息

`lsmod [-v]`

显示已被加载的 `module`，如果有指定 `-v` 参数，那么更详细的信息会一起列出。

`more filename`

显示所指定的文件内容，并在每 `LINES`（环境变量）暂停。

`reboot`

直接重新启动机器。

`set variable, set variable=value`

设置 loader 的环境变量。

`unload`

卸载所有被加载的 `module`。

### 7.3.3.3 Loader 举例

这里就是一些 loader 使用的实际例子。

1, 在单用户模式下启动你的普通内核 :

```
boot -s
```

2, 卸载你的普通内核和模块, 然后仅仅引导你的旧 (或另一个) 内核 :

```
unload
```

```
load kernel.old
```

3, 你可以使用 `kernel.GENERIC`, 这是安装光盘上的通用 kernel, 或是 `kernel.old`, 这是你上一个安装的 kernel (如果你有升级或重新配置你自己的 kernel 的话)。

**注意:** 照着下列的步骤可以配合原先的 `module` 来加载其它的 kernel :

```
unload
```

```
set kernel = "kernel.old"
```

```
boot-conf
```

4, 加载内核配置的 `script` 文件 (这是一个自动的 `script` 文件, 用来执行你在 kernel 启动阶段所要执行的命令) :

```
load -t userconfig_script
```

```
/boot/kernel.conf
```

## 7.4 启动时内核的调节

一旦 kernel 通过 loader (一般来说) 或 boot2 (略过执行 loader), kernel 将会检查它的启动标记, 如果有, 就开始按照标记做一些必要的调节。

### 7.4.1 内核启动标记

这儿是一些启动标记 :

a

在内核初始化期间, 询问要使用哪一个设备作为 root 文件系统

C

从 CDROM 启动

C

运行 UserConfig, 启动时的内核配置

S

从单用户模式启动

V

在内核启动期间更详细的信息

### 7.5 Init: 过程控制初始化

内核一旦启动完成, 它就把控制权转交给用户层命令 `init`, 它就在 `/sbin/init` 中, 在 `loader` 中, 程序路径可以通过 `init_path` 指定。

#### 7.5.1 自动重新启动

这个程序会确定系统将要使用的文件系统是存在的。如果不存在, 那么 `fsck` 就不能正常地被执行去修复磁盘驱动器, 接着 `init` 将把系统切换到单用户模式, 系统管理者就可以在这时候直接处理这个错误。

#### 7.5.2 单用户模式

这个模式能够通过自动启动顺序来延伸, 或用户启动时通过加上 `-s` 选项, 或在 `loader` 中设置 `boot_single`。它也可以不重新启动通过呼叫 `shutdown` 来达到, 或在多用户模式下加上 `-h` 选项。如果系统控制台 `console` 被设置成 `insecure`, 那在开始单用户模式之前, 系统就要求键入 `root` 密码。

例如 7-3. 在 `/etc/ttys` 中设置一个 `console` 为 `insecure`

```
# name  getty                type  status  comments
#
# This entry needed for asking password when init goes to single-user mode
# If you want to be asked for password, change "secure" to "insecure" here
console none                unknown off insecure
```

**注意:** 一个 `insecure` 的 `console` 代表你的 `console` 的安全等级是 `insecure` 的, 并且确定如果有人要进入单用户模式, 就要输入 `root` 密码, 请注意, `insecure` 不代表你的

console 是 insecure 的 ,而是 ,如果你要多一点的安全防护 ,请选择 insecure ,而不是 secure。

### 7.5.3 多用户模式

如果 `init` 正确地找到了你的文件系统 ,或结束了单用户模式 ,系统就会进入多用户模式 ,并开始系统的资源配置。系统将先执行默认的配置文件 `/etc/default/rc.conf` ,和系统的详细配置 `/etc/rc.conf` ,接着按照 `/etc/fstab` 来加载文件系统 ,再激活网络服务 ,和其它的系统守护程序 (daemon) ,最后 ,执行部分程序的起始 `script` 文件。`rc` 是个参考资源设置系统的好地方 ,同样 ,直接查阅那些 `scripts` 也是个好方法。

### 7.6 关机程序

使用 `shutdown` 可以控制系统进行关机 , `init` 将会执行 `/etc/rc.shutdown` 这个 `script` 文件 ,接着输出终止 (terminate) 信号给所有的程序 ,这时如果有无法终止的程序 ,那么就使用 `kill` 来杀死信号。