第6章 init

本章的主题是init进程，它是一个由内核启动的用户级进程。init有许多重要的职责，比如启动getty（以便用户能够登录），实施运行级和照顾孤儿进程等。本章将讨论如何配置 init，如何利用不同的运行级。

6.1 init的重要作用

init是Linux系统操作中绝不可少的程序之一，但你大多数时间都可以忽略它。一个编得好的Linux系统中，都会附带适用于多数系统的init，而且在这些系统上，根本不用为init操心。通常，只有在你挂上串行终端、拨入（注意非拨出） modem或打算更改默认的运行级时，才有必要去关心init。

内核自行启动（已经被载入内存，开始运行，并已初始化所有的设备驱动程序和数据结构等）之后，就通过启动一个用户级程序 init的方式，完成了自己的引导进程。所以， init始终是第一个进程（其进程编号始终为1）。

内核会在过去曾使用过init的几个地方查找它，但它的正确位置（对 Linux系统来说）是/sbin/init。如果内核找不到init，它就会试着运行/bin/sh，如果运行失败，系统的启动也会失败。

init启动时，先执行大量的管理任务，比如检查文件系统，清除 /tmp，启动各种服务以及针对用户将能够登录的每台终端和虚拟控制台，启动 getty（有关详情，参见第7章），完成启动进程。

系统正常启动之后， init在用户已经注销登录后，针对每个终端重新启动 getty（以便下一名用户的登录）。init还收养“孤儿”（独立的）进程：进程启动一个子进程并先其子进程死掉，这个子进程立即就成了init的子进程。由于各种技术方面的原因，这一点相当重要，因为它有助于我们进一步了解进程列表和进程树图表（ init本身是不能死的。即使用SIGKILL也不能杀死它）。有几个 init的变体是可以用的。许多 Linux版本都采用 sysvinit（Miquel van Smoorenburg编写的），这个程序基于System V init设计。Unix的BSD版本中另有一个init。两者间的主要区别在于运行级： System V有运行级， BSD则没有（至少过去没有）。这点区别不太重要。我们下面只谈谈sysvinit。

6.2 通过init启动getty：/etc/inittab文件

系统启动时， init开始读取/etc/inittab配置文件。系统运行期间，如果系统发出一个 HUP信号“ kill -HUP 1 as root ”，它将再次读取这个配置文件。这个特性令其不必要启动系统，更改init配置。

/etc/inittab文件有点复杂。我们将从配置几个简单的 getty行开始。位于/etc/inittab的这些行由四个用冒号定界的字段组成：

```
id:runlevels:action:process
```

下面是对这四个字段的说明。此外， /etc/inittab中可包含空行和以“ # ”号开头的行；这些行是可被忽略。

1. id

这个字段定义文件内的行。对 getty行来说，它指定的是它运行的终端（设备文件名内/dev/tty后面的字符）。对其他行来说，它没什么作用（只是一个长度限制），但它应该是独一无二的。

2. runlevels

应该为代码行考虑运行级别。运行级别应该以单一的数位来指定，不带定界符（关于运行级别的详情，参见下一小节）。

3. action

代码行应该采取的行动，例如respawn用于在退出时，再次运行下一个字段中的命令。

4. process

准备运行的命令。

为了在第一个虚拟终端上启动 getty（/dev/tty1}），所有普通多用户运行级别（ 2～5）内，都应该写入这样一行：

```
1:2345:respawn:/sbin/getty 9600 tty1
```

第一个字段指这一行用于 /dev/tty1。第二个字段指该行的运行级别是 2、3、4、5。第三个字段是说这个命令应该在其退出之后，再执行一次（以便另外的用户能够登录、注销，然后再次登录）。最后一个字段是命令，这个命令将在第一个虚拟终端上运行 getty（不同版本的getty的运行是不一样的。查看你的手册页并保证其是完全正确的）。

如果你想在系统中增添终端或拨入 modem，最好在/etc/inittab内多增添几行，一行对应一个终端或一条拨入线路。关于这方面的详情，可参考 init、inittab和getty手册页。

如果命令在启动时就失败了，而且 init被配置为重启，它就会使用大量的系统资源： init启动它，不行，再重启，再不行，再重启，如此“ 纠缠不休 ”，最后耗完所有的系统资源为止。为了防止出现这种情况， init将对重启命令的频率进行跟踪，如果频率过快，它将在重启之前，延迟5min。

6.3 运行级别

运行级别是指init和整个系统的状态，该系统定义了对哪些系统服务进行操作。运行级别是按照编号来识别的，如表 6-1所示。

表6-1 运行级别号

0	使系统停止
1	单用户模式（用于特殊管理）
-2	普通操作（用户自定义）
6	重启

有的系统管理员利用运行级别来定义正在使用的自系统，比如是否正在运行 X，网络是否可操作等等。其他管理员则在不更改运行级别的情况下，让所有的子系统单独运行或启动和终止它们，因为对控制他们的系统而言，运行级别太粗略了。具体情况需要你自行决定，但最容易的方式是照你的Linux版本中所说的去做。

运行级别的配置是在/etc/inittab行内进行的，如下所示：

```
12:2:wait:/etc/init.d/rc 2
```

第一个字段是一个任意指定的标签，第二个字段表示这一行适用于运行级别 2。第三个字段表示进入运行级别时， init应该运行第四个字段内的命令一次，而且 init应该等待该命令结束。/etc/init.d/rc命令运行启动和终止输入以便进入运行级别 2时所需的任何命令。

第四个字段中的命令执行设置运行级别时的一切“ 杂活 ”。它启动已经没有运行的服务，终止不应该再在新运行级别内运行的服务。根据 Linux版本的不同，采用的具体命令也不同，而且运行级别的配置也是有差别的。

init启动时，它会在/etc/inittab内查找一个代码行，这一行指定了默认的运行级别：

```
id:2:initdefault:
```

你可以要求init在启动时，进入非默认运行级别，这是通过为内核指定一个“ single ”或“ emergency ” 命令行参数来实现的。比如说，内核命令行参数的指定可通过 LILO来执行。这样一来，你就可以选择单用户模式了（即运行级别 1）。

系统正在运行时， telinit命令可更改运行级别。运行级别发生变化时， init就会从/etc/inittab运行相应的命令。

6.4 /etc/inittab中的特殊配置

/etc/inittab中，有几个特殊的特性，允许 init重新激活特殊事件。这些特殊特性都是用第三个字段中的特殊关键字标记出来的。比如：

1. powerwait

允许init在电源被切断时，关闭系统。其前提是具有 UPS和监视UPS并通知init电源已被切断的软件。

2. ctrlaltdel

允许init在用户于控制台键盘上按下 Ctrl+Alt+Del组合键时，重新启动系统。注意，如果该系统放在一个公共场所，系统管理员可将 Ctrl+Alt+Del组合键配置为别的行为，比如忽略等。

3. sysinit

系统启动时准备运行的命令。比如说，这个命令将清除 /tmp。

上面列出的特殊关键字尚不完整。其他的关键字及其使用详情，可参考你的 inittab手册页。

6.5 在单用户模式下引导

一个重要的运行级别就是单用户模式（运行级别 1），该模式中，只有一个系统管理员使用特定的机器，而且尽可能少地运行系统服务，其中包含登录。单用户模式对少数管理任务（比如在/usr分区上运行fsck）而言，是很有必要的，因为这需要卸载分区，但这是不可能的，除非所有的服务系统已被杀死。

一个正在运行的系统可以进入单用户模式，具体做法是利用 init，请求运行级别 1。内核启动时，在内核命令行指定 single或emergency关键字，就可进入运行级别 1了。内核同时也为init指定命令行， init从关键字得知自己不应该采用默认的运行级别（内核命令行的输入方式

和你启动系统的方式有关）。

有时，以单用户模式进行启动是必要的，这样一来，用户在装入分区之前，或至少在装入分散的/usr分区之前，能手工运行 fsck（在分散的文件系统上，任何活动都可能使其更为分散，所以应该尽可能地运行 fsck）。

如果自动化的 fsck在启动时失败了，启动脚本 init的运行将自动进入单用户模式。这样做是为了防止系统使用不连贯的文件系统，这个文件系统是 fsck不能自动修复的。文件系统不连贯的现象极为少见，而且通常会导致硬盘的不连贯或实验性的内核释放，但最好能做到防患于未然。

由于安全上的考虑，在单用户模式下，启动外壳脚本之前，配置得当的系统会要求用户提供root密码。否则，它会简单地为 LILO输入合适的一行代码，以 root的身份登录（当然，如果/etc/passwd已经由于文件系统的问题而不连贯了，就不适合这里的原则了，为对付这种情况，你最好随时准备一张启动盘）。