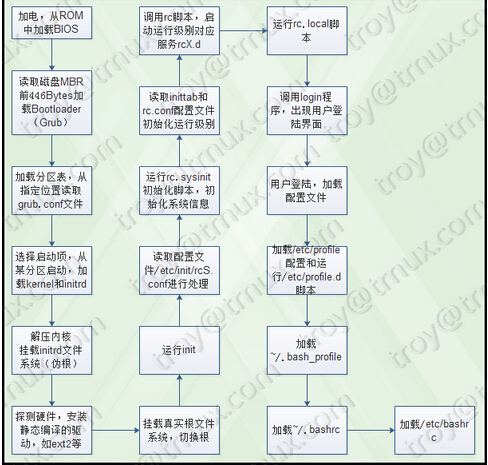
**1、此文章摘自：23期脱产班二麻师兄网站。**

**linux启动过程：**

1. 加电自检，加载BIOS
2. 读取磁盘MBR，加载GRUB stage1
3. 根据GURB选择启动菜单，根据所选文件系统，进入grub stage 1\_5阶段
4. 从相应的文件系统中读取grub文件，进入stage2
5. 读取内核，解压缩内核，加载内核
6. 读取ramfs，解压缩ramfs，加载ramfs虚拟文件系统
7. 安装必要驱动，挂载根文件系统，切换根文件系统
8. 启动init进程
9. init读取rcS.conf配置文件：
10. 首先执行rc.sysinit脚本，初始化系统（包括主机名，网络，重新挂载根文件系统为rw，加载内核模块，挂载fstab上的分区，设置selinux等）
11. 其次根据inittab中的运行级别，加载rc.conf
12. 执行/etc/rc.d/rc脚本，传入参数$RUNLEVEL
13. 启动/etc/rc$RUNLEVEL.d下的脚本（rc.local也在，次序最后）
14. 执行终端登录程序（/bin/login）



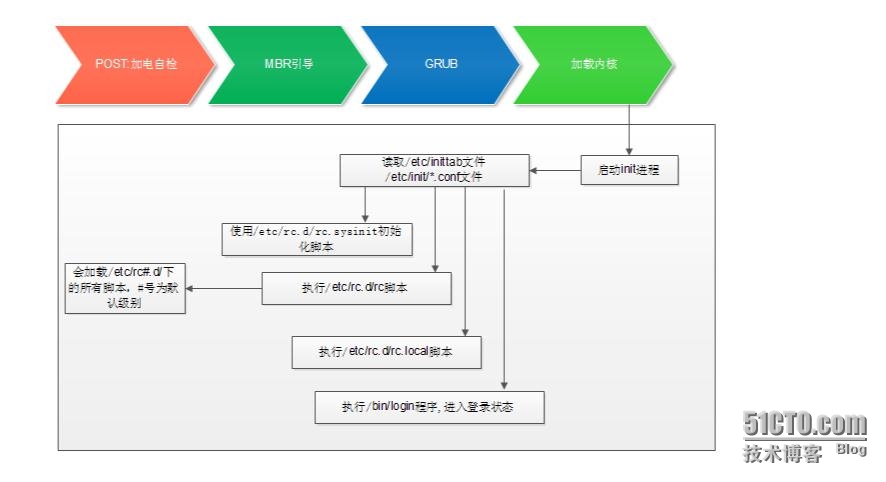
**2、本文摘自51cto一位师兄的博文**

我们在使用Linux操作系统的时候，我们只需按下电源键，等待，然后输入账户和密码就可以使用Linux操作系统了。那么在按下电源到输入账号和密码之前，操作系统都做了些什么？下面就来讲述在这段时间发生的动作。

    下面以CentOS6系列为例来讲解Linux的启动流程。

一、Linux启动流程详解

    下图就是启动流程的大致过程：

[](http://s3.51cto.com/wyfs02/M02/41/1D/wKiom1PQxc6SML8nAAFcDmK4XC0648.jpg)

二、POST加电自检

    POST(PowerOnSelfTest)首先对每一个设备进行检查。完成后会寻找存有引导记录的设备，找到后读入操作系统引导记录，然后将系统控制权交给引导记录，并由引导记录来完成系统的顺利启动。

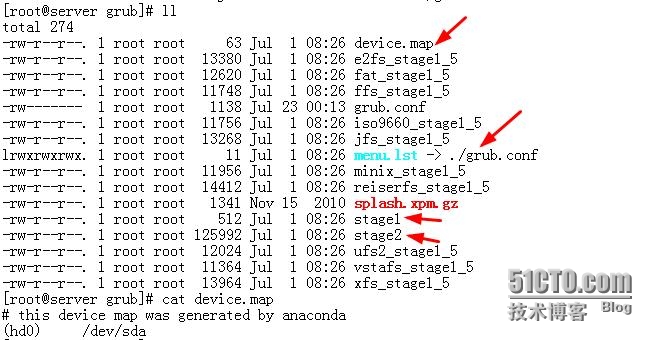
三、MBR引导

    MBR(Master Boot Record) MBR记录一般是在磁盘 0 磁道 1 扇区，共512个字节。前446个字节是BootLoder，后 4\*16 的 64 个字节是存放分区信息的，最后 2 个字节是校验信息，一般是 55AA。

四、GRUB（GRand Unified Bootloader）、加载内核

    就是MBR中的前 446 个字节，是BooTLoader的一种，它的作用是要选择要启动的内核。

1、GRUB程序的组成：

[](http://s3.51cto.com/wyfs02/M01/41/21/wKioL1PQ4efw0SE_AAJbTpZuDV4737.jpg)

    主要是由device.map，menulst，stage1,stage2,以及一系列的stage1\_5组成。对于这些部分我的理解是这样：

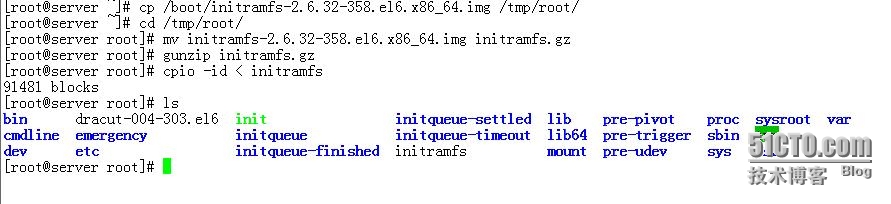
* device.map：存放的是内核文件的根分区
* menu.lst：是grub.conf的链接文件，但是这个名字我觉得更与它的功能接近，就是菜单列表。文件里设置了可以选择的内核菜单。存放于stage2中。
* stage：用于grub引导程序过大，所以分2段引导，第一段存放在MBR中，第二段存放于内核文件系统中，第一段引导完成后可以找到第二段。    但是，第二段是存放于内核文件系统中的，此时还没有格式化文件系统，如何可以访问到第二段的 menu.lst 呢？？就需要借助于中间层 stage1\_5，有它来协助 stage1 段来访问 stage2 段。stage1\_5通常位于 stage1 字段后的 63 个扇区。  由于stage2 在内存中存放可以使用的文件系统不确定，所以这就是有多个 stage1\_5 的原因了。

2、grub.conf 文件参数意义

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | default=1    # 默认启动的内核title, 0 表示是第一个  timeout=5    # 默认等待时间  splashimage=(hd0,0)/grub/splash.xpm.gz    # 指定菜单的背景图片的路径。为xpm格式，采用gzip压缩，只能为14bits色  hiddenmenu    # 隐藏菜单  title CentOS (2.6.32-358.el6.x86\_64)    # 标题名，用户可自定义      root (hd0,0)    # 指定 grub 的根位置      # 指定 kernel 文件的位置，还要指出 root（系统启动后） 的位置，挂载方式 ro，这项很关键。      # 加载后会启动 init 进程。      kernel /vmlinuz-2.6.32-358.el6.x86\_64 ro root=/dev/mapper/vg0-root rd\_NO\_LUKS rd\_NO\_DM LANG=en\_US.UTF-8 rd\_LVM\_LV=vg0/swap rd\_NO\_MD SYSFONT=latarcyrheb-sun16 crashkernel=auto rd\_LVM\_LV=vg0/root  KEYBOARDTYPE=pc KEYTABLE=us rhgb crashkernel=auto quiet rhgb quiet      # 在内核启动过程中装载根文件系统时有用      initrd /initramfs-2.6.32-358.el6.x86\_64.img |

    initramfs-2.6.32-358.el6.x86\_64.img 的本质是？？？？

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | # initramfs 是以 gzip 压缩的 cpio 格式的文件。内核启动时将他作为一个临时的根文件系统。  # grub 的 stage2 将initrd加载到内存里，让后将其中的内容释放到内容中，  # 内核便去执行init脚本，这时内核将控制权交给了init文件处理。    # init 它也主要是加载各种存储介质相关的设备驱动程序。当所需的驱动程序加载完后，  # 会创建一个根设备，然后将根文件系统rootfs以只读的方式挂载。    # 这一步结束后，释放未使用的内存，转换到真正的根文件系统上面去，同时运行/sbin/init程序，  # 执行系统的1号进程。此后系统的控制权就全权交给/sbin/init进程了。 |

[](http://s3.51cto.com/wyfs02/M02/41/21/wKioL1PQ7u_RXEQAAAHfgGw63dE269.jpg)

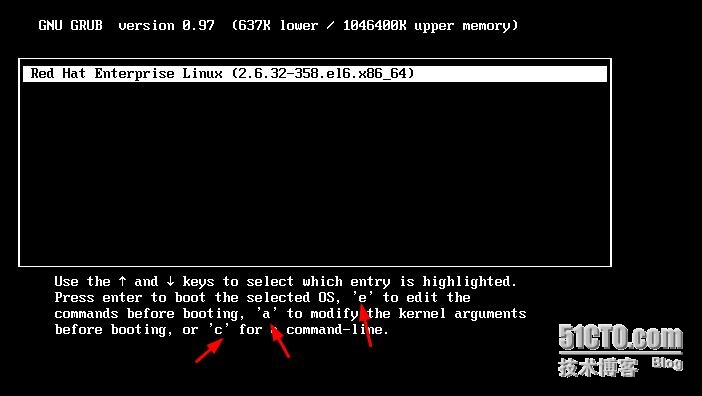
    如何制作 initramfs 呢？

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | # 可以使用 dircut 命令，用法如下：  # 只有在 initramfs 文件损毁的时候会使用到    [root@server ~]# dracut /mnt/initramfs-`uname -r`.img `uname -r`  [root@server ~]# du -sh /mnt/initramfs-2.6.32-358.el6.x86\_64.img  16M /mnt/initramfs-2.6.32-358.el6.x86\_64.img  [root@server ~]# du -sh /boot/initramfs-2.6.32-358.el6.x86\_64.img  16M /boot/initramfs-2.6.32-358.el6.x86\_64.img    # 详细介绍，可使用这个链接：https://zh.opensuse.org/index.php?title=SDB:Dracut&variant=zh-cn#.E5.90.AF.E7.94.A8\_dracut\_.E7.94.9F.E6.88.90.E7.9A.84\_initramfs |

3、grub的功能

    1）提供菜单，并提供交互式接口

        e: 进入编辑模式

[](http://s3.51cto.com/wyfs02/M02/41/22/wKiom1PQ8uLRTFq2AAFFyUCSOSo430.jpg)

    2）选择要启动的内核或系统

        允许传递引导参数给内核

        选择界面可隐藏

|  |  |
| --- | --- |
| 1  2 | # 可以自启动是通过 grub 像内核传递参数。  # 应用之一是：修改 root 密码（忘记密码），使用 e 选项，传递单用户指令。 |

    3）为编辑功能提供保护机制

        启用内核文件

        选择运行指定的内核得先输入密码

        传递参数

        使用e命令得先输入密码

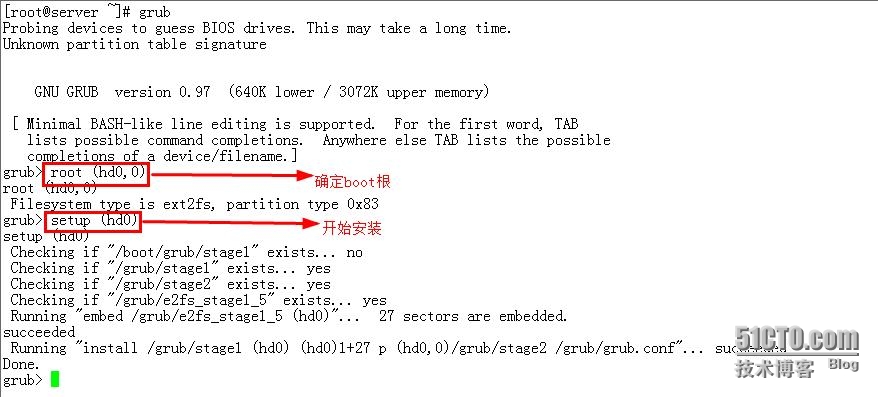
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | [root@server root]# grub-md5-crypt  Password:  Retype password:  $1$Mp7Rp1$l2wzOILNUqpUhAR7zXvQb0  # 上面是生成的加密字符串  # 然后将信息加入到 grub.conf 文件中，格式如下：  password --md5 $1$Mp7Rp1$l2wzOILNUqpUhAR7zXvQb0    # 当然加入 grub.conf 文件的位置不同，加密效果也不一样。  # 加入到 title 之前的话，会加密整个菜单。  # 加入到 title 指内的话，会加密对应的操作系统的入口。 |

4、安装grub的方式

    1）使用grub-install命令

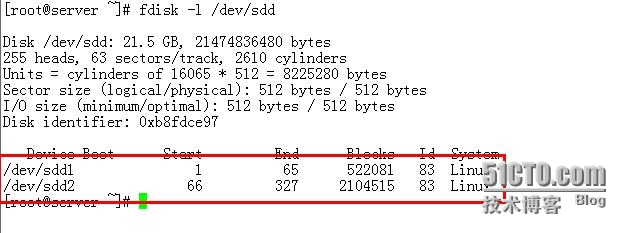
        # grub-install [--root-directory=/path/to/somewhere] DEVICE

    2）grub交互是命令

[](http://s3.51cto.com/wyfs02/M00/41/22/wKiom1PQ_HiCHYTqAALwLfHE0OI236.jpg)

    在这里我们自制一个精简grub,当然启动的时候只启动一个bash就可以。示例是在 VMware 虚拟机下完成的。

第一步：准备好块新磁盘，分 2 个区出来

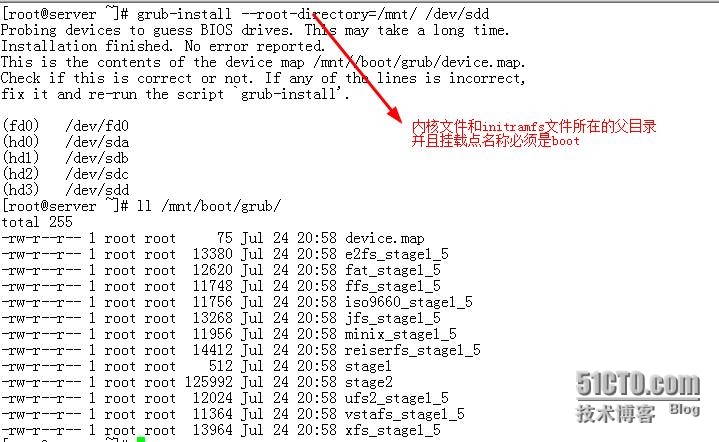
[](http://s3.51cto.com/wyfs02/M02/41/22/wKiom1PRAY_CbjOxAAFQGeXx0nE979.jpg)

第二步：模拟启动时的boot目录，和根目录，并挂载。如下

|  |  |
| --- | --- |
| 1  2  3 | [root@server ~]# df -H | tail -n 2  /dev/sdd1              518M    11M   481M   3% /mnt/boot  /dev/sdd2              2.2G    71M   2.0G   4% /mnt/sysroot |

第三步：制作 grub，并生成 grub 的配置文件

    制作grub

[](http://s3.51cto.com/wyfs02/M00/41/22/wKiom1PRA4njlzRSAANnZuPPwug409.jpg)

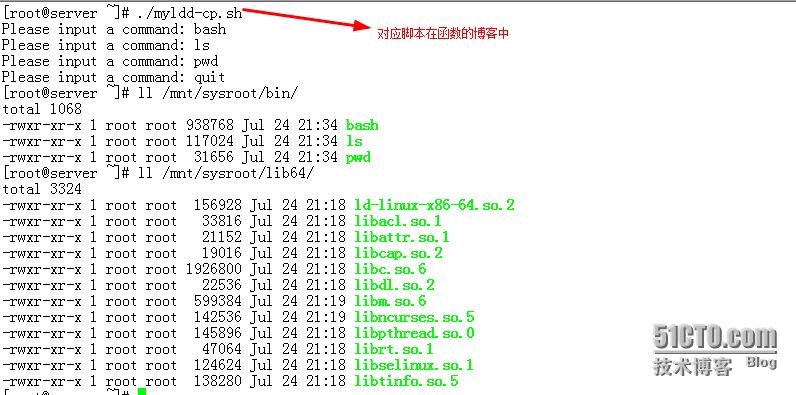
    拷贝内核文件和 initramfs 文件到 /mnt/boot 目录下，并生成 grub.conf 文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | [root@server ~]# cp /boot/vmlinuz-2.6.32-358.el6.x86\_64 /mnt/boot/vmlinuz  [root@server ~]# cp /boot/initramfs-2.6.32-358.el6.x86\_64.img /mnt/boot/initramfs.img  [root@server ~]# ll /mnt/boot/  total 20106  drwxr-xr-x 2 root root     1024 Jul 24 20:58 grub  -rw-r--r-- 1 root root 16542269 Jul 24 21:06 initramfs.img  -rwxr-xr-x 1 root root  4043888 Jul 24 21:05 vmlinuz  # 生成的 grub.conf 文件  [root@server ~]# cat /mnt/boot/grub/grub.conf  default=0  timeout=10    title Mini Linux      root (hd0,0)      kernel /vmlinuz ro root=/dev/sda2 selinux=0 init=/bin/bash      initrd /initramfs.img |

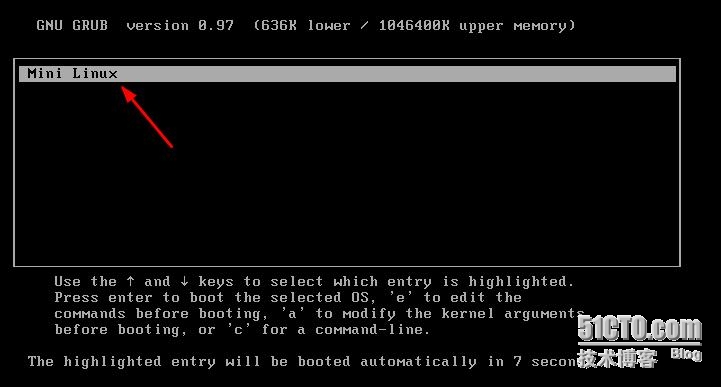
 第四步: 生成根文件系统

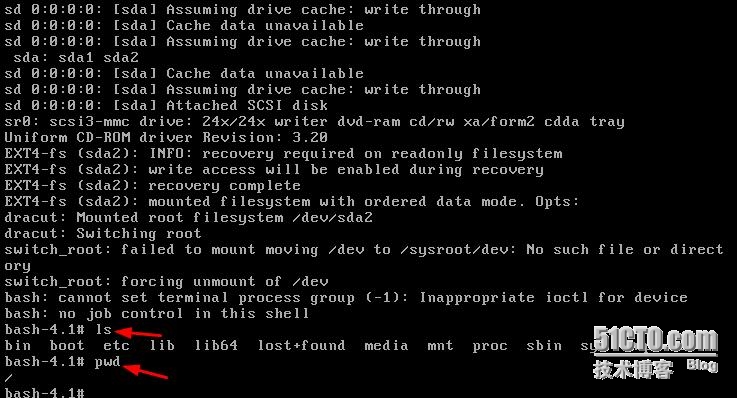
[](http://s3.51cto.com/wyfs02/M02/41/22/wKiom1PRBtWiGC9QAAJM8RoL0uA295.jpg)

第五步：拷贝使用命令和库到对应根目录下

[](http://s3.51cto.com/wyfs02/M00/41/22/wKioL1PRDKij5REwAAM4ghH8fUw006.jpg)

    将刚才的磁盘镜像作为另一个虚拟机的启动盘。

[](http://s3.51cto.com/wyfs02/M01/41/22/wKioL1PRD2KC4J8-AAFGpnnDVLM992.jpg)

[](http://s3.51cto.com/wyfs02/M01/41/22/wKioL1PRD3LSg0DtAALqgLLfnXg176.jpg)

五、启动 init 进程

    init 进程是系统启动的第一个进程。其他进程都是有此进程控制fork出来的，也就是说，init是其他进程的“祖先”，PID=1。当然，PID=0进程我们称为空闲进程是属于内核的。

    init启动后所做的事情：

1、读取/etc/inittab来读取默认级别 假设：读取到的默认级别是 3

    /etc/init/\*.conf

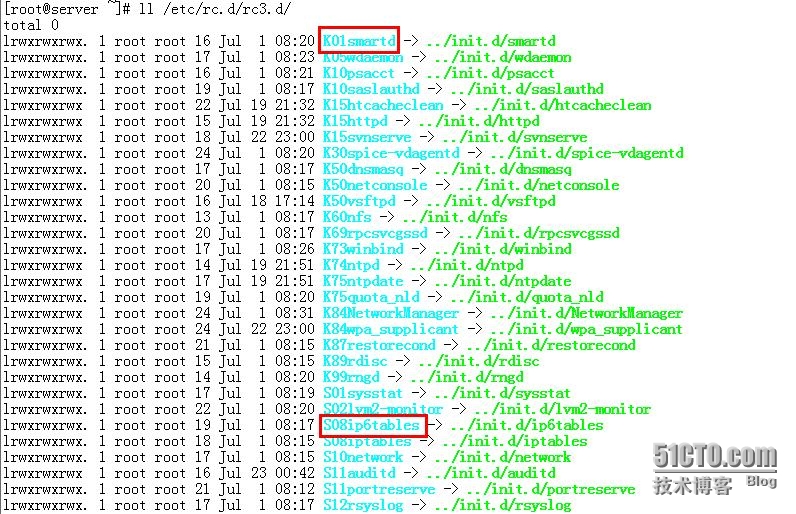
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | # 启动级别：（runlevel）  # 0：关机  # 1：单用户模式  # 2：多用户模式，不支持 NFS 功能  # 3：完全多用户模式，支持文本接口  # 4：预留模式  # 5：完全多用户模式，支持图形模式  # 6：关机    # /etc/inittab文件格式如下，这是遵循了CentOS5的风格。  id:3:initdefault: |

2、执行初始化系统脚本 /etc/rc.d/rc.sysinit 来初始化脚本

    这个大家有机会可以读读，脚本主要的作用如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | # 获取网络环境与主机类型。首先会读取网络环境设置文件"/etc/sysconfig/network"，获取主机名称与默认网关等网络环境。  # 打印文本欢迎信息。  # 挂载/etc/fstab文件中定义的其它文件系统。  # 激活swap。  # 检测根文件系统，并以读写方式重新挂载  # 决定是否启动SELinux和udev。  # 接口设备的检测与即插即用（pnp）参数的测试。  # 用户自定义模块的加载。用户可以再"/etc/sysconfig/modules/\*.modules"加入自定义的模块，此时会加载到系统中。  # 加载核心的相关设置。按"/etc/sysctl.conf"这个文件的设置值配置功能。  # 设置系统时间（clock）。  # 设置终端的控制台的字形。  # 设置raid及LVM等硬盘功能。  # 以方式查看检验磁盘文件系统。  # 进行磁盘配额quota的转换。  # 重新以读取模式载入系统磁盘。  # 启动quota功能。  # 启动系统随机数设备（产生随机数功能）。  # 清楚启动过程中的临时文件。  # 将启动信息加载到"/var/log/dmesg"文件中。 |

3、然后执行 /etc/rc.d/rc 脚本

    此脚本运行的时候此脚本的时候，会读取/etc/rc.d/rc3.d/下的所有脚本，来控制系统启动的时候启动或者关闭那些服务。[](http://s3.51cto.com/wyfs02/M01/41/44/wKiom1PSA-mDbCDSAAZ9kiqbBP4645.jpg)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | # 这里面的脚本大致分为 2 类，以 S 开头的和以 K 开头的脚本。  # S 代表开机时启动的，K 表示开机时不启动。    # 每个文件的命名格式，例如：S01sysstat  S 开机启动 01 启动次序  sysstat 启动脚本名  # 启动的定义是在每个脚本中定义的：例如  #     # chkconfig: -     85         15  #            默认级别 启动次序 不启动次序 |

    这些脚本是否开机自启动，可由 chkconfig 命令来管理。

|  |  |
| --- | --- |
| 1  2  3  4  5 | # 常用选项：  # chkconfig --add SRV\_SCRIPT  可以将自己写的一些脚本，让此命令统一管理。例如：chkconfig --add httpd  # chkconfig --del SRV\_SCRIPT   将服务删除，意思就是服务不由chkconfig这种机制来管理  # chkconfig SRV\_SCRIPT {on|off}    设置服务开机是否自启动，默认管理的级别是2345.  # --level 2345 等，指定默认级别，可以与其他选项一起使用 |

4、执行/etc/rc.d/rc.local脚本，此脚本是启动过程中最后启动的一个脚本。

    最后会执行 /bin/login 登录用户。至此系统启动过程完成。