**主要内容：**

**1. 开机过程详细说明**

**2. 启动过程几个主要文件简介**

**Linux系统的启动流程：**

**POST(**CPU, Memory**)-->BIOS-->Boot Squence启动次序-->MBR主引导记录(**三段446字节**)-->Boot Loader**(代码)**-->Kernel**(初始化)**-->initrd**(内核与init程序不在同一分区导致了这一步。先有鸡还是先有蛋？这是一个压缩文件，一个独立的文件系统)**-->/sbin/init(/etc/inittab)-->login**

**一：详细过程说明：**

1. BIOS：系统首先由POST(PowerOnSelfTest,上电自检)程序来对内部各个设备进行检查；自检后，就首先按照系统CMOS设置中保存的启动顺序搜寻软硬盘驱动器及CD—ROM、网络服务器等有效地启动驱动器，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。

     硬盘主引导记录MBR（Master Boot Record）：位于硬盘0磁道0柱面1扇区，该扇区共512bytes，其中MBR占446bytes ；MBR所做的唯一的事情就是装载第二引导装载程序。

    分区表DPT（Disk Partition Table）占64bytes；

    硬盘有效标志（Magic Number）占2bytes；

    2. 引导扇区的前446字节，其中定义如何启动本硬盘上的系统（根据分区表找到对应分区上的内核）；而对于Linux，一般多用Grub引导，由于grub相对较大，所以分为两段式的进行引导，第一段存储于硬盘MBR中，第二段放置于操作系统内核所在的分区上。Grub根据MBR中第一段找到第二段，继续引导，第二段中放置的有GRUB菜单等信息，可以让用户选择需要继续引导启动的系统；并且菜单中指定的有内核及RamDisk信息；

    3. 根据用户选择将对应的内核读到内存，解压展开；然后内核开始初始化；初始化完成后需要读取根分区（根是一切的起点），这时候如果系统不是普通磁盘，是scsi或是raid形式时，就需要先加载相关的文件系统驱动来驱动该磁盘设备，从而读取根分区（鸡和蛋问题）；这时候给内核提供了一个minilinux，即initrd，其中含有内核所需的一些基本模块驱动，该linux只在内存中运行。内核启动时展开该initrd来加载相应的驱动，在该驱动的补充之下从而挂载上根分区；

    4. 然后运行根分区脚本/sbin/init 来初始化系统；这个客户自行程序运行会读取初始化配置文件inittab：在其中顺序定义并运行的有1.默认的运行级别 2.默认的系统服务初始化脚本sysinit位置 3.各种运行级别；系统会根据默认的运行级别，来对应执行相应级别下的脚本，该处脚本是链接文件，链接到init.d中相对应的文件，真正运行的是init.d里的脚本）。

 注：

  rc N;表示用rc脚本去运行rc N.d目录下的脚本；rc脚本就是去执行所需级别脚本的功能脚本；

 目录下的文件均为脚本链接文件，指向/etc/rc.d/rcN.d/目录，并且命名时以S或K开头，后面跟上0-99的数字；S代表启动时执行；K代表关闭时执行；01-99代表启动或关闭的级别（数字越小越优先）

    5. 初始化结束前执行最后一个文件：/etc/rc.d/rc.local，系统会读取该脚本中的所有命令并执行一遍；但是该脚本只在启动时执行一次，系统关闭时不能执行，所以不要为了实现开机启动而将某些服务写入这个脚本，那样会造成服务关机时的非正常关闭；

   注：

 内核：模块化设计，大部分设备模块是在需要时加载驱动，并且大部分模块的驱动放置于根分区上。

############################

/boot/vmlinuz-version 内核所在分区

/ init程序在sbin下，init程序与内核在不同的文件系统上：

/boot:一般不能放在逻辑卷上 LVM，如果根做成了逻辑卷，那么boot分区必须单独做分区。如果根是普通分区就可以直接挂在根上。

**initrd: RAMDISK 实质就是将内存当做磁盘运行，实际就是一个小linux**

init **/boot/initrd####** 6.2之前是gz压缩文件，之后从gz压缩变为xz压缩 解压缩后变为一个asc2 归档文件，再用cpio解压缩

驱动： 根分区(设备驱动, 文件系统模块)

###########################

启动级别：

**0: 关机 只启动shutdown**

**1: 单用户模式**

**2：正常模式，nfs服务是关闭的；**

**3：多用户模式，文本模式**

**4：保留**

**5：多用户模式，图形界面**

**6: 重启**

查看当前运行级别：runlevel who -r

############################

Kernel启动时完成的任务：

硬件探测

硬件驱动初始化(initrd: RAMDISK)

挂载根文件系统(根切换initrd-->rootfs)

启动init进程(之后就是用户空间的引导完成)

######################

**系统初始化脚本完成的任务：**

激活udev和selinux；

设置内核参数，根据/etc/sysctl.conf

设置系统时钟

设置键映射

启用交换分区

设置主机名称

检测根文件系统，重新以读写方式挂载

激活软RIAD和LVM

启用磁盘配额

挂载额外的文件系统,/etc/fstab

清除一些过期的文件锁和PID文件等

#############################

**二：启动过程中的几个主要文件及其作用：**

**init(/etc/inittab):**

**设定默认运行级别**

**执行系统初始化脚本(/etc/rc.d/rc.sysinit)**

**运行某级别相应的脚本(/etc/rc.d/rcN.d)**

K

S

|  |  |
| --- | --- |
| 文件名称  （按照加载次序列出） | 作用 |
| **/etc/inittab** | **定义在进入或切换各个级别时系统需要执行的动作**  **init在初始化系统时需要读取其中配置** |
| **/etc/rc.d/rc.sysinit** | **由init进程调用执行**  **完成下面的初始化工作：**  **1. 获取网络环境及主机类型**  **2. 测试与载入内存设备/proc及USB设备/sys**  **3. 决定是否启动SELinux**  **4. 接口设备的检测与即插即用(PNP)参数的测试**  **5. 用户自定义模块的加载**  **6. 加载核心的相关设置**  **7. 设置系统时间**  **8. 设置中断控制台(console)的字形**  **9. 设置RAID与LVM等硬盘功能**  **10. 以fsak检验磁盘文件系统**  **11. 进行磁盘配额quota的转换（非必要）**  **12. 重新以可读取模式载入系统磁盘**  **13. 启动quota的功能**  **14. 启动随机数设备**  **15. 清除启动过程中生成的临时文件**  **16. 将启动相关信息加载到/var/log/message文件中** |
| **/ettc/rc.d/rc** | **由init进程调用执行**  **根据制定的运行级别，加载或终止相应的系统服务** |
| **/etc/rc.local** | **由rc脚本调用执行**  **保存用户定义的徐开机后自动执行的命令** |

**inittab文件说明：**  
  
    格式：  id：runlevels：action ：process

|  |  |
| --- | --- |
| id | 用于在inittab文件中唯一标识一**条记录,没有特别的意义** |
| runlevels | 用于指定记录能在哪些级别下运行(可以有多个，表示在相应的运行级均需要运行；也可以为空，为空时表示0~6都要运行) |
| action | 用于指定记录将执行的动作类型 |
| process | 用于设置启动进程所执行的动作命令  （字段中进程可以是任意的守候进程、可执行脚本或程序） |

  inittab文件中每一记录都从新的一行开始，所以每个记录项最多可有512个字符；  
  
  运行级(runlevels)就是操作系统当前正在运行的功能级别。这个级别从1到6;  
  
  当运行级别改变，并且正在运行的程序并没有在新的运行级别中指定需要运行，那么init会先发送一个SIGTERM 信号终止，然后是SIGKILL. 运行级别发生变化时，init就会从/etc/inittab运行相应的命令

**文件中有效的action值：**

ACTION：常用动作

initdefault: 设定默认运行级别

sysinit: 系统初始化

wait: 等待进入其相应级别以后执行后面的命令

respawn: 首先执行一次PROCESS，当PROCESS终止时会自动重新启动

|  |  |
| --- | --- |
| **有效的action值** | **respawn：**如果process字段指定的进程不存在，则启动该进程，init不等待处理结束，而是继续扫描inittab文件中的后续进程，当这样的进程终止时，init会重新启动它，如果这样的进程已存在，则什么也不做。 |
| **wait：**启动process字段指定的进程，必须等到执行结束才去处理inittab中的下一记录项。 |
| **once:**启动process字段指定的进程，不等待处理结束就去处理下一记录项。当这样的进程终止时，也不再重新启动它，在进入新的运行级别时，如果这样的进程仍在运行，init也不重新启动它。 |
| **boot：**只有在系统启动时，init才处理这样的记录项，启动相应进程，并不等待处理结束就去处理下一个记录项。当这样的进程终止时，系统也不重启它。 |
| **bootwait**：系统启动后，当第一次从单用户模式进入多用户模式时处理这样的记录项，init启动这样的进程，并且等待它的处理结束，然后再进行下一个记录项的处理，当这样的进程终止时，系统也不重启它 |
| **off**：如果指定的进程正在运行，init就给它发SIGTERM警告信号，在向它发出信号SIGKILL强制其结束之前等待5秒，如果这样的进程不存在，则忽略这一项。 |
| **powerfail：**当init接到断电的信号(SIGPWR)时，处理指定的进程。当然前提是有U P S和监视UPS并通知init电源已被切断的软件。RHlinux默认没有列出该选项。 |
| **powerwait** 当init接到断电的信号(SIGPWR)时，处理指定的进程，但init不会等待正在运行的进程结束，并且等到处理结束才去检查其他的记录项。 |
| **sysinit:** 指定的进程在访问控制台之前执行，这样的记录项仅用于对某些设备的初始化，目的是为了使init在这样的设备上向用户提问有关运行级别的问题，init需要等待进程运行结束后才继续。 |
| **initdefault:**指定一个默认的运行级别，只有当init一开始被调用时才扫描这一项，如果runlevel字段指定了多个运行级别，其中最大的数字 是默认的运行级别，如果runlevel字段是空的，init认为字段是0123456，于是进入级别6，这样便陷入了一个循环，如果inittab文件中没 有包含initdefault的记录项，则在系统启动时请求用户为它指定一个初始运行级别 |
| **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。我是设置成打印一句警告的话了（防止其他人恶意重启系统）：监视到特定的键盘组合键被按下时采取的动作，现在还不完善。 |

启动的服务不同；

启动脚本都在：但是否启动是 下面的决定：

/etc/rc.d/init.d/

/etc/rc.d/rcN.d这里才分别定义 创建的是指向上面的链接 不过有的是K开头的有的是S开头的

K 开头的stop

S 开头的start

init说到底就是一个位于,/sbin/init的可执行文件罢了-->**配置文件/etc/inittab**

这个配置文件第一行就决定了默认启动级别：

id:3:initdefault:

chkconfig SERVICE on|off

2345

chkconfig --level LEVEL SERVICE on|off

chkconfig --list [SERVICE]

**ntsysv 也可以控制启动**