

# 第1章 Linux系统启动流程

## 1.CentOS7启动流程说明

文字简化说明：

- 1 1. BIOS开机自检
- 2 2. 读取MBR信息选择启动设备
- 3 3. 加载grub菜单，选择操作系统
- 4 4. 加载内核及驱动程序
- 5 5. 启动systemd程序加载必要文件，以下操作并行执行
  - 6 1) 执行initrd.target (/usr/lib/systemd/system/initrd.target)
  - 7 包含挂载/etc/fstab文件中的文件系统。
  - 8 2) systemd执行默认的target配置。
  - 9 3) systemd执行sysinit.target，初始化系统及加载basic.target准备启动系统。
  - 10 4) systemd启动multi-user.target (生产工作模式)下的服务程序，即开机自启动的程序，程序目录为/etc/systemd/system和/usr/lib/systemd/system。
  - 11 5) systemd执行multi-user.target下的/etc/rc.d/rc.local内容。
  - 12 6) systemd执行multi-user.target下的gtty.target及登录服务。
  - 13 7) systemd执行graphical所需要的服务(如果安装了图形桌面功能)。

### 第1步：BIOS开机自检

- 1 简单来说，BIOS是主板上的一块芯片，负责主板通电后各部件的自检，设置和保存，一切正常后才能启动操作系统。其记录了电脑最基本的信息，是软件与硬件打交道的最基础的桥梁，没有他，电脑就不能正常工作。

### 第2步：读取MBR信息

- 1 当正确检查完所有硬件信息后，计算机就会根据BIOS里的设置去读取相应的启动系统里的硬件设备。
- 2 如果预先设定了从硬盘启动加载系统，那么BIOS就会读取硬盘的MBR（即0磁道0柱面1扇区的前446字节）。
- 3 计算机读取BIOS所指定的磁盘MBR信息之后，就会将其读入到内存中。被读入到内存中执行的其实就是Boot Loader（引导加载程序），对应于Linux系统，就是加载Grub信息。

MBR包含的内容：

- 1 0磁道0扇区：512bytes
- 2 446 bytes: boot loader启动相关
- 3 64 bytes: 分区表
- 4 2 bytes: 55AA

查看MBR分区：

- 1 hexdump -C -n 512 /dev/sda

## 第3步：加载Grub菜单

- 1 引导加载程序(**Boot Loader**)是计算机在加载操作系统内核之前运行的一段小程序。这段小程序可以初始化硬件设备、建立内存空间的映射图，从而将系统的软硬件环境加载到一个适合的状态，以便为最终调用操作系统内核做好准备。**Linux**主流使用**Grub**做为引导菜单。**CentOS7**使用**Grub2**作为引导程序。

提示:

- 1 **grub.conf**的知识其实在企业**Linux**运维中用途不是很大，这里讲解**Grub**的目的是为了解**Linux**系统的整个启动流程做铺垫，因为在运维工作中，极少会在线处理问题，出了问题也是直接切换服务了，之后再来慢慢研究是修复还是重装。

## 第4步：加载Kernel内核及驱动程序

- 1 根据**Grub**设定的内核映像所在的路径，系统会读取内存映像，并进行解压缩操作。完成解压缩内核之后，屏幕会输出“OK, booting the kernel”的信息。其实就是根据**grub.conf**中的如下设置加载内核及相关参数

xfs的驱动程序:

- 1 `df -T`
- 2 `modinfo xfs`
- 3 `ll -h /lib/modules/3.10.0-957.el7.x86_64/kernel/fs/xfs/xfs.ko.xz`

可引导的内核在哪里?

- 1 `[root@linux ~]# ll /boot/`
- 2 总用量 94132
- 3 `-rw-r--r--. 1 root root 151918 11月 9 2018 config-3.10.0-957.el7.x86_64`
- 4 `drwxr-xr-x. 5 root root 79 4月 6 19:42 grub2`
- 5 `-rw-----. 1 root root 57270580 4月 6 19:41 initramfs-0-rescue-3c29b9dc81a34220b666aff272f3851c.img`
- 6 `-rw-----. 1 root root 21810980 4月 6 19:40 initramfs-3.10.0-957.el7.x86_64.img`
- 7 `-rw-r--r--. 1 root root 314036 11月 9 2018 symvers-3.10.0-957.el7.x86_64.gz`
- 8 `-rw-----. 1 root root 3543471 11月 9 2018 System.map-3.10.0-957.el7.x86_64`
- 9 `-rwxr-xr-x. 1 root root 6639904 4月 6 19:41 vmlinuz-0-rescue-3c29b9dc81a34220b666aff272f3851c`
- 10 `-rwxr-xr-x. 1 root root 6639904 11月 9 2018 vmlinuz-3.10.0-957.el7.x86_64`

vmlinuz和initramfs作用:

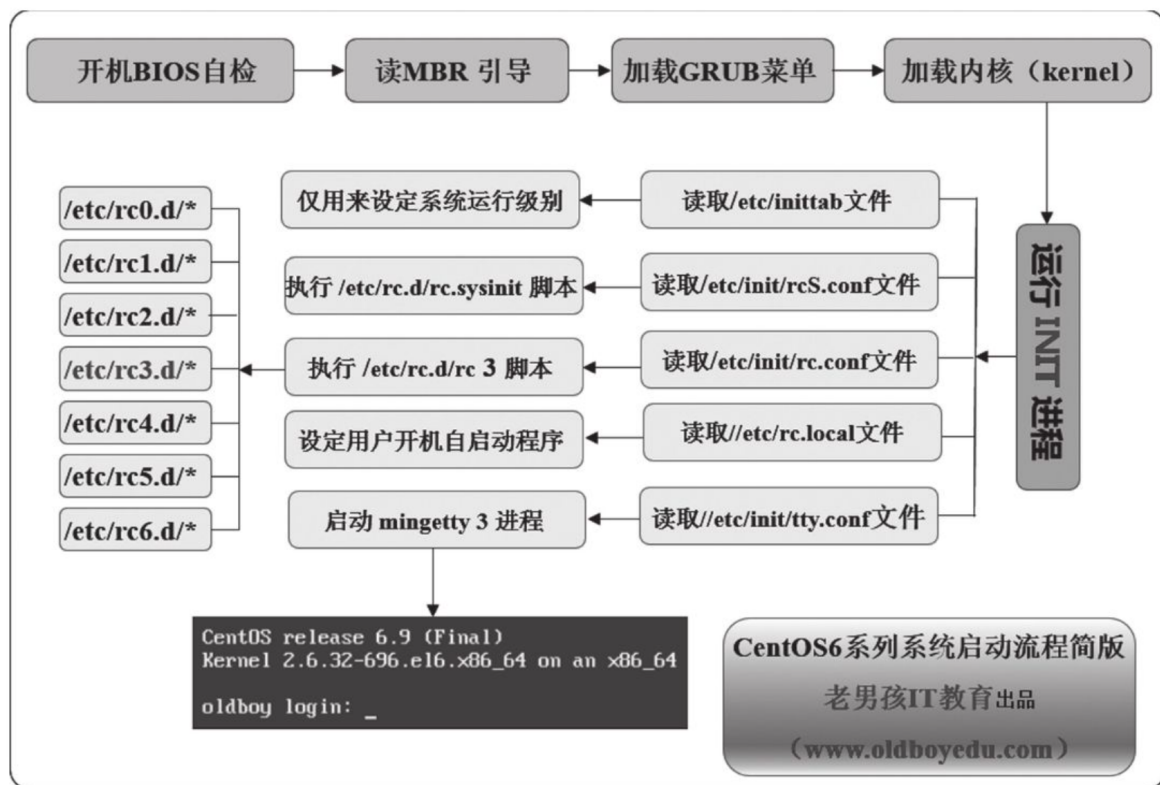
- 1 `vmlinuz: VM=Virtual Memory z=gzip`
- 2 `vmlinuz`是可引导可压缩的内核,主要作用是用来管理进程,内存,文件,驱动,网络等。
- 3
- 4 `initrd.img: initial ramdisk,`
- 5 主要作用是用于加载驱动模块,通常步骤是先启动内核,然后内核挂载`initrd.img`,并挂载各种模块和驱动,最重要的是文件系统的驱动模块,有了它才能挂载根文件系统,最后可以运行**Linux**第一个应用程序`init`或`systemd`。

## 第5步：启动systemd程序加载必要文件

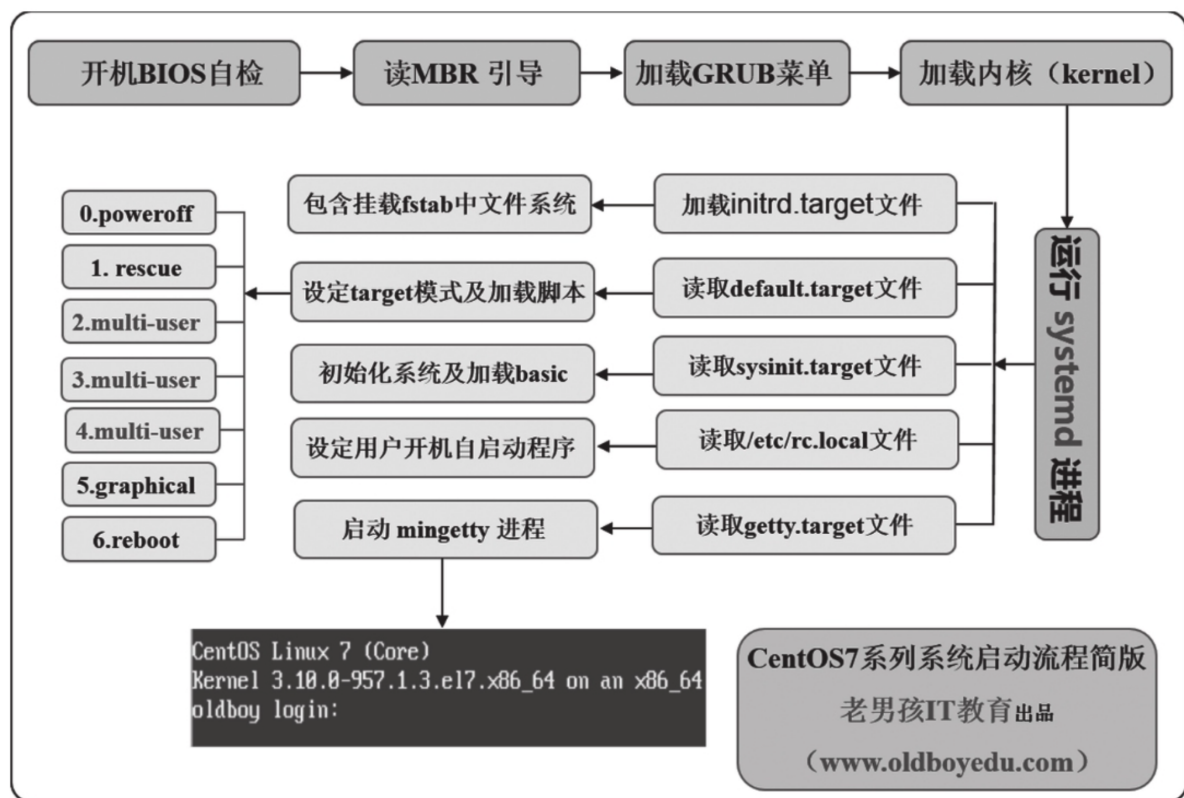
- 1 CentOS7和CentOS6的启动流程绝大部分还是相同的，但也有一些小区别，例如，CentOS6下第一个启动的init进程被改为了systemd（并行启动模式），下面重点说一下CentOS7加载systemd进程后的启动流程，即从CentOS6启动流程的第五步开始讲起，前四步与CentOS6启动流程的描述一致。
- 2
- 3 1) 执行initrd.target (/usr/lib/systemd/system/initrd.target)
- 4 包含挂载/etc/fstab文件中的文件系统。
- 5 2) systemd执行默认的target配置。
- 6 3) systemd执行sysinit.target，初始化系统及加载basic.target准备启动系统。
- 7 4) systemd启动multi-user.target（生产工作模式）下的服务程序，即开机自启动的程序，程序目录为/etc/systemd/system和/usr/lib/systemd/system。
- 8 5) systemd执行multi-user.target下的/etc/rc.d/rc.local内容。
- 9 6) systemd执行multi-user.target下的gtyy.target及登录服务。
- 10 7) systemd执行graphical所需要的服务(如果安装了图形桌面功能)。

## 2.CentOS6以及CentOS7启动流程图对比

CentOS6启动流程：



CentOS7启动流程：



### 3.分析启动流程耗时

运行systemd-analyze blame命令可以打印出启动过程的详细流程

```

1 [root@linux ~]# systemd-analyze blame
2     5.043s network.service
3     2.609s dev-mapper-centos\x2droot.device
4     1.694s lvm2-pvscan@8:2.service
5     1.669s lvm2-monitor.service
6     1.059s mysqld.service
7     843ms tuned.service
8     204ms auditd.service
9     171ms polkit.service
10    164ms rsyslog.service
11    128ms chronyd.service
12    102ms systemd-logind.service
13     94ms rhel-dmesg.service
14     93ms systemd-vconsole-setup.service
15     77ms sysstat.service
16     77ms systemd-user-sessions.service
17     62ms boot.mount
18     61ms systemd-udev-trigger.service
19     58ms rhel-import-state.service
20     50ms rhel-readonly.service
21     36ms plymouth-quit-wait.service
22     35ms plymouth-start.service
23     34ms systemd-udevd.service
24     27ms plymouth-read-write.service
25     26ms systemd-tmpfiles-setup-dev.service
26     25ms systemd-journald.service
27     24ms sshd.service
28     23ms plymouth-quit.service
29     22ms systemd-tmpfiles-setup.service
30     19ms systemd-journal-flush.service
  
```

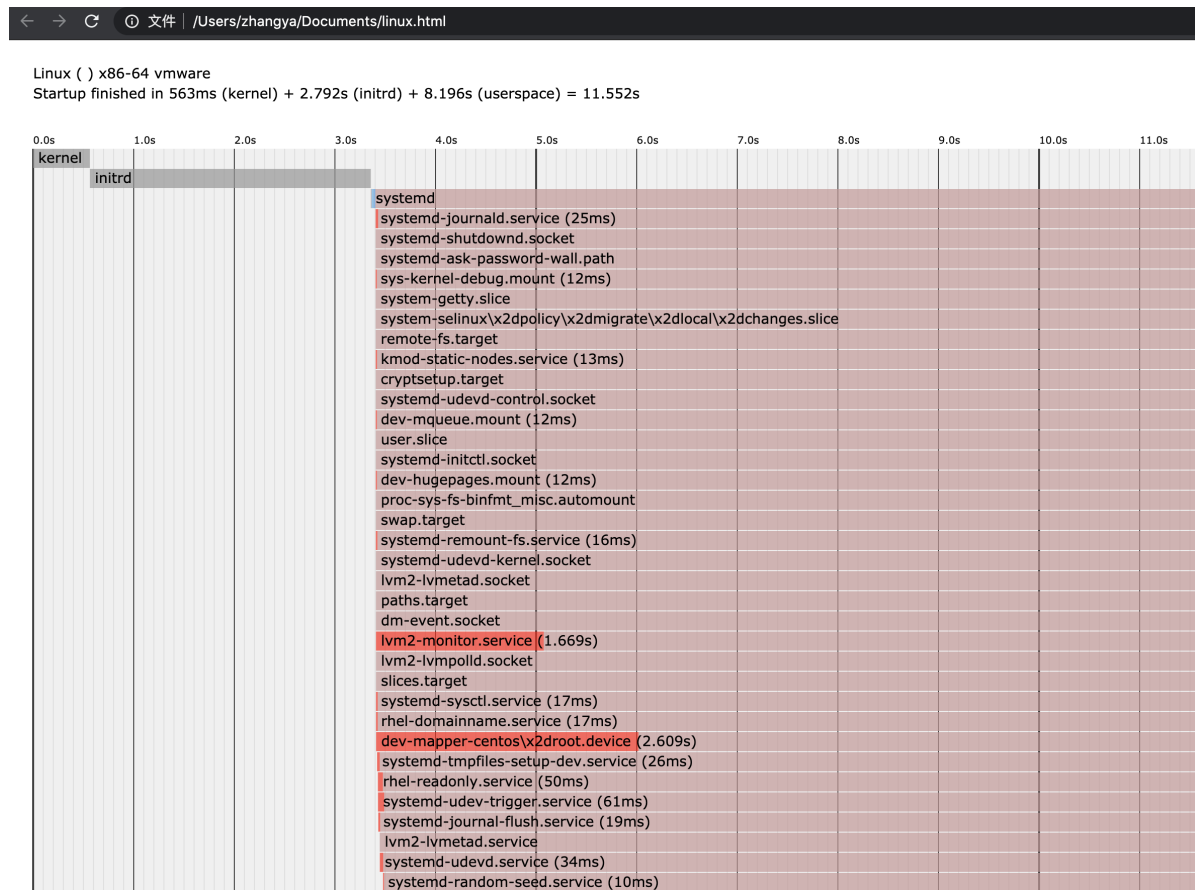
```

31      17ms rhel-domainname.service
32      17ms systemd-sysctl.service
33      16ms systemd-remount-fs.service
34      15ms systemd-rfkill@rfkill0.service
35      13ms systemd-fsck-root.service
36      13ms kmod-static-nodes.service
37      12ms sys-kernel-debug.mount
38      12ms dev-mqueue.mount
39      12ms dev-hugepages.mount
40      10ms systemd-update-utmp.service
41      10ms systemd-random-seed.service
42      9ms systemd-tmpfiles-clean.service
43      5ms systemd-update-utmp-runlevel.service
44      5ms sys-kernel-config.mount

```

我们也可以将其生成网页查看

```
1 | systemd-analyze plot > linux.html
```



## 第2章 Linux运行级别

### 1.什么是系统运行级别

- 1 不同的运行级别就是指系统运行在不同功能的级别。
- 2 比如打游戏有新手模式，高级模式，炼狱模式一样，不同级别分别对应不同的使用场景。

### 2.系统运行级别分类

1	启动级别	systemd名称	功能说明
2	0	poweroff.target	关机操作
3	1	rescue.target	单用户模式，救援模式
4	2	multi-user.target	
5	3	multi-user.target	多用户模式，文本界面
6	4	multi-user.target	
7	5	graphical.target	多用户模式，图形界面
8	6	reboot.target	重启操作

上面的运行级别名称从哪里得来的呢？通过查看系统命令可以得出结论

```

1 [root@linux ~]# ll /usr/lib/systemd/system/runlevel[0-9].target
2 lrwxrwxrwx. 1 root root 15 12月 13 11:01
  /usr/lib/systemd/system/runlevel0.target -> poweroff.target
3 lrwxrwxrwx. 1 root root 13 12月 13 11:01
  /usr/lib/systemd/system/runlevel1.target -> rescue.target
4 lrwxrwxrwx. 1 root root 17 12月 13 11:01
  /usr/lib/systemd/system/runlevel2.target -> multi-user.target
5 lrwxrwxrwx. 1 root root 17 12月 13 11:01
  /usr/lib/systemd/system/runlevel3.target -> multi-user.target
6 lrwxrwxrwx. 1 root root 17 12月 13 11:01
  /usr/lib/systemd/system/runlevel4.target -> multi-user.target
7 lrwxrwxrwx. 1 root root 16 12月 13 11:01
  /usr/lib/systemd/system/runlevel5.target -> graphical.target
8 lrwxrwxrwx. 1 root root 13 12月 13 11:01
  /usr/lib/systemd/system/runlevel6.target -> reboot.target

```

## 3.运行级别常用命令

运行级别的说明文件路径：

```
1 cat /etc/inittab
```

查看当前运行级别：

```
1 systemctl get-default
```

切换运行级别：

```
1 init N
```

设置默认启动级别：最常用的两个级别是3和5

```
1 systemctl set-default TARGET.target
```

## 4.练习

自己试试如何将图形化的Linux切换到命令行级别以及更改默认启动运行级别为命令行界面

# 第3章 Linux Systemd

## 1.什么是systemd

- 1 CentOS7之前的系统都是使用init进程作为系统启动后的第一个进程，但是init有两个缺点：
- 2 1.启动时间长，因为init的进程是串行的，只有前一个启动完毕后一个进程才启动。
- 3 2.启动脚本复杂，以前的系统初始化需要加载很多脚本，依赖关系复杂，靠脚本自己处理。
- 4
- 5 而systemd的启动则是并行运行的，而且服务的启动配置文件统一语法，所以管理起来更方便。

## 2.systemd的优势

- 1 1.较新的系统都已经使用systemd来管理服务，例如Debian9,Centos7,Ubunut16等
- 2 2.系统引导时实现服务的并行启动，效率更高，启动更快
- 3 3.自动解决依赖关系
- 4 4.服务的启动配置文件统一语法，管理起来更方便

## 3.systemd常用管理命令

管理服务运行相关命令：

- |   |                           |             |
|---|---------------------------|-------------|
| 1 | systemctl start nginx     | #启动服务       |
| 2 | systemctl stop nginx      | #停止服务       |
| 3 | systemctl restart nginx   | #重启服务       |
| 4 | systemctl reload nginx    | #重新加载服务配置文件 |
| 5 | systemctl status nginx    | #查看服务的运行状态  |
| 6 | systemctl is-active nginx | #查看服务是否正在运行 |

服务运行状态说明：

- |   |   |
|---|---|
| 1 | #当我们使用system status查看一个服务的运行状态时，有以下几种情况 |
| 2 | loaded #配置文件已经被加载                       |
| 3 | active (running) #服务正在运行                |
| 4 | inactive (dead) #服务没有在运行                |
| 5 | enabled #服务被设定为开机自启动                    |
| 6 | disabled #服务被设定为开机不自启动                  |
| 7 | static #服务开机不自启动，但是可以被其他服务调用启动          |

管理服务开机启动相关命令：

- |   |                            |                     |
|---|----------------------------|---------------------|
| 1 | systemctl enable nginx     | #设置服务开机自启动          |
| 2 | systemctl disable nginx    | #设置服务开机不自启动         |
| 3 | systemctl is-enabled nginx | #查看服务是否开机自启动        |
| 4 | systemctl list-unit-files  | #查看所有服务的开机自启动状态     |
| 5 | systemctl daemon-reload    | #重新载入更改的systemd配置文件 |

## 4.systemd文件格式

systemd文件路径说明：

- |   |                          |                            |
|---|--------------------------|----------------------------|
| 1 | /etc/systemd/system/     | #系统自带的启动文件                 |
| 2 | /usr/lib/systemd/system/ | #一般使用软件包安装的软件的启动配置文件在这个目录下 |

systemd文件格式说明：

- |   |  |                             |
|---|--|-----------------------------|
| 1 | <code>systemd unit</code> 文件通常是由三部分组成的 |                             |
| 2 | <code>[Unit]</code>                    | #一般定义的是通用选项，比如描述信息，依赖关系等    |
| 3 | <code>[Service]</code>                 | #特定服务的类型，具体的启动关闭重启选项都在此部分配置 |
| 4 | <code>[Install]</code>                 | #定义由开机自启和不开机自启命令时实现的选项      |

Unit段常用说明：

- |   |                            |                               |
|---|----------------------------|-------------------------------|
| 1 | <code>Description</code>   | #描述信息                         |
| 2 | <code>Documentation</code> | #说明文档的在线地址                    |
| 3 | <code>After</code>         | #定义启动顺序，表示当前配置的服务应该晚与哪些服务之后启动 |
| 4 | <code>wants</code>         | #依赖其他的服务                      |

Service段常用说明：

- |   |                            |                          |
|---|----------------------------|--------------------------|
| 1 | <code>Type</code>          | #定义服务类型                  |
| 2 | <code>forking</code>       | #需要父进程启动子进程的服务类型为forking |
| 3 | <code>PIDFile</code>       | #定义PID文件路径               |
| 4 | <code>ExecStart</code>     | #指定启动服务命令绝对路径            |
| 5 | <code>ExecReload</code>    | #指定重新加载服务的配置文件的命令绝对路径    |
| 6 | <code>ExecStop</code>      | #指定停止服务命令绝对路径            |
| 7 | <code>ExecStartPre</code>  | #在启动之前运行的命令              |
| 8 | <code>ExecStartPost</code> | #在启动之后运行的命令              |

Install段常用说明：

- |   |                       |            |
|---|-----------------------|------------|
| 1 | <code>wantedBy</code> | #哪些服务需要被依赖 |
|---|-----------------------|------------|

举例：nginx的systemd文件解释：

- |    |   |                          |
|----|---|--------------------------|
| 1  | <code>[Unit]</code>   |                          |
| 2  | <code>Description=nginx - high performance web server</code>                          | #说明信息                    |
| 3  | <code>Documentation=http://nginx.org/en/docs/</code>                                  | #帮助说明                    |
| 4  | <code>After=network-online.target remote-fs.target nss-lookup.target</code>           | #需要依赖<br>这些服务，在这些服务启动后启动 |
| 5  | <code>wants=network-online.target</code>  | #需要<br>依赖的服务             |
| 6  |   |                          |
| 7  | <code>[Service]</code>  |                          |
| 8  | <code>Type=forking</code>   | #启动类型为forking            |
| 9  | <code>PIDFile=/var/run/nginx.pid</code>   | #pid文件的绝对路径              |
| 10 | <code>ExecStart=/usr/sbin/nginx -c /etc/nginx/nginx.conf</code>                       | #启动命令的绝对路径               |
| 11 | <code>ExecReload=/bin/sh -c "/bin/kill -s HUP \$(/bin/cat /var/run/nginx.pid)"</code> | #<br>重新加载配置的绝对路径         |
| 12 | <code>ExecStop=/bin/sh -c "/bin/kill -s TERM \$(/bin/cat /var/run/nginx.pid)"</code>  | #<br>停止服务命令的绝对路径         |
| 13 |   |                          |
| 14 | <code>[Install]</code>  |                          |
| 15 | <code>wantedBy=multi-user.target</code>   | #需要依赖的服务                 |

## 5.自定义nginx的systemd启动文件

编译安装nginx过程略。



```
1 #1.修改nginx配置文件的pid为自定义路径
2 [root@linux ~]# grep pid /opt/nginx/conf/nginx.conf
3 pid          /opt/nginx/pid/nginx.pid;
4
5 #2.写入自定义systemd配置
6 cat > /usr/lib/systemd/system/linux6.service << 'EOF'
7 [Unit]
8 Description=DIY nginx
9 Documentation=https://www.cnblogs.com/alaska/
10 After=network-online.target remote-fs.target nss-lookup.target
11 Wants=network-online.target
12
13 [Service]
14 Type=forking
15 PIDFile=/opt/nginx/pid/nginx.pid
16 ExecStart=/opt/nginx/sbin/nginx -c /opt/nginx/conf/nginx.conf
17 ExecReload=/bin/sh -c "/bin/kill -s HUP $(/bin/cat /opt/nginx/pid/nginx.pid)"
18 ExecStop=/bin/sh -c "/bin/kill -s TERM $(/bin/cat /opt/nginx/pid/nginx.pid)"
19 ExecStartPre=/bin/sh -c "/usr/bin/chown -R www:www /opt/nginx/"
20
21 [Install]
22 WantedBy=multi-user.target
23 EOF
24
25 #3.重新加载systemd配置
26 systemctl daemon-reload
27
28 #4.启动我们自定义的服务
29 systemctl start linux6
30 systemctl status linux6
```

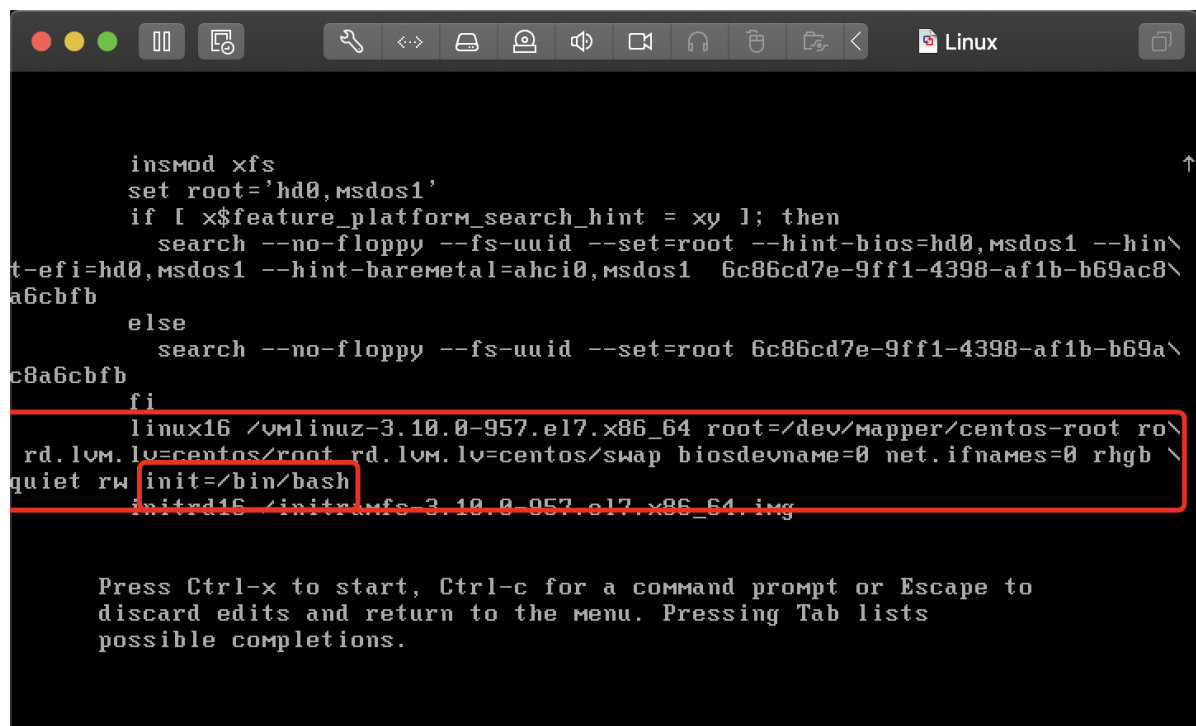
## 第4章 Linux单用户模式

### 1.什么是Linux单用户模式

- 1 单用户模式的最主要应用场景就是当我们忘记系统root密码的时候，就可以使用单用户模式找回。
- 2 如果谁都能随随便便的进入单用户模式修改root密码，那就风险太大了，所有单用户模式有前提条件：
- 3 1.进入单用户模式必须重启服务器，所以需要有重启服务器的权限
- 4 2.单用户模式是启动服务进程之前，所以需要有能实际接触服务器的条件才行，如果是物理机那就表示人必须在机器旁边，接键盘显示器才能操作。

### 2.实战：找回忘记的Linux密码

#### 方法1: 添加rw init=/bin/bash



```
insmod xfs
set root='hd0,msdos1'
if [ x${feature_platform_search_hint} = xy ]; then
    search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hin\
t-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 6c86cd7e-9ff1-4398-af1b-b69ac8\
a6cbfb
else
    search --no-floppy --fs-uuid --set=root 6c86cd7e-9ff1-4398-af1b-b69a\
c8a6cbfb
fi
linux16 /vmlinuz-3.10.0-957.el7.x86_64 root=/dev/mapper/centos-root ro\
rd.lvm.lv=centos/root rd.lvm.lv=centos/swap biosdevname=0 net.ifnames=0 rhgb \
quiet rw init=/bin/bash
initrd16 /initramfs-3.10.0-957.el7.x86_64.img

Press Ctrl-x to start, Ctrl-c for a command prompt or Escape to
discard edits and return to the menu. Pressing Tab lists
possible completions.
```

- 1 1. 开机启动时在启动菜单界面选择按e进入grub编辑模式
- 2 2. 在linux开头的行最后添加以下内容
- 3 rw init=/bin/bash
- 4
- 5 3. 启动后重新挂载
- 6 mount -o rw,remount /
- 7
- 8 4. 重新修改root密码
- 9 passwd root

详细步骤:

- 1 <https://mp.weixin.qq.com/s/1f-zCq8j4gL3xiB3AV9K3A>

## 方法2: 添加rd.break

```
insmod xfs
set root='hd0,msdos1'
if [ x$feature_platform_search_hint = xy ]; then
    search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hin\
t-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 --hint='hd0,msdos1' 6c86cd7e-9\
ff1-4398-af1b-b69ac8a6cbfb
else
    search --no-floppy --fs-uuid --set=root 6c86cd7e-9ff1-4398-af1b-b69a\
c8a6cbfb
fi
linux16 /vmlinuz-3.10.0-957.el7.x86_64 root=/dev/mapper/centos-root ro\
rd.lvm.lv=centos/root rd.lvm.lv=centos/swap biosdevname=0 net.ifnames=0 rhgb \
quiet LANG=zh_CN.UTF-8 rd.break
initrd16 /initramfs-3.10.0-957.el7.x86_64.img

Press Ctrl-x to start, Ctrl-c for a command prompt or Escape to
discard edits and return to the menu. Pressing Tab lists
possible completions.
```

操作步骤如下：

1. 开机启动时在启动菜单界面选择按e进入grub编辑模式
2. 在linux开头的行最后添加rd.break
3. 按ctrl+x保存退出重启
4. 进入单用户模式后输入以下命令
- 5 mount -o remount,rw /sysroot/
- 6 chroot /sysroot/
- 7 passwd
- 8 exit
- 9 reboot

## 第5章 Linux救援模式

### 1.什么是Linux救援模式

- 1 救援模式就是当我们的系统出现故障启动失败后，使用系统光盘镜像重新加载然后进入系统进行修复的一种模式。

### 2.实战Linux启动故障修复

注意：做启动修复实验前建议先快照一下

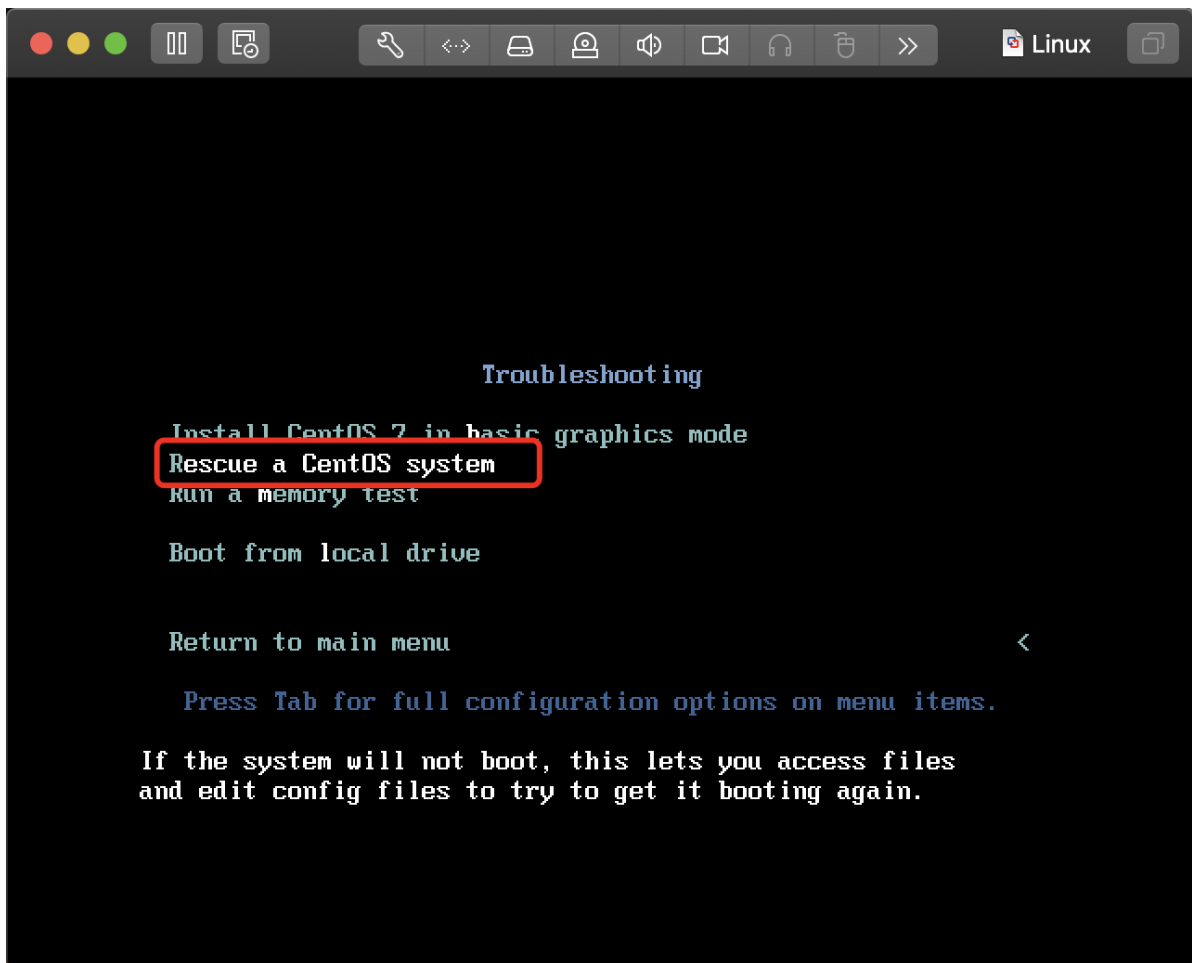
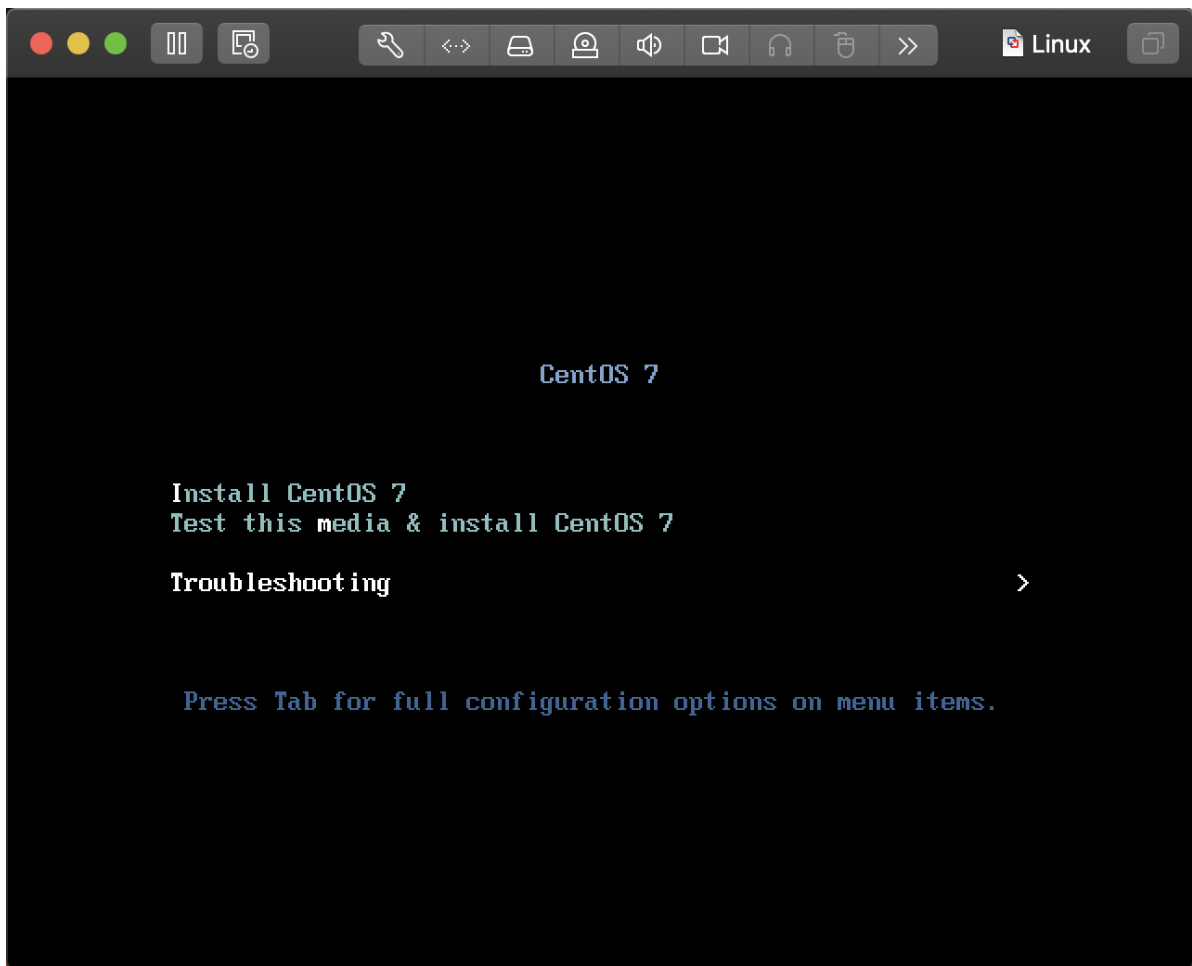
模拟场景1: 误删除/boot所有文件

- 1 rm -rf /boot/\*
- 2 reboot

恢复步骤：

第1步：选择光盘启动

第2步：选择救援模式



第3步：输入1继续救援模式

```
Starting installer, one moment...
anaconda 21.48.22.147-1 for CentOS 7 started.
* installation log files are stored in /tmp during the installation
* shell is available on TTY2
* if the graphical installation interface fails to start, try again with the
  inst.text bootoption to start text installation
* when reporting a bug add logs from /tmp as separate text/plain attachments
=====
Rescue

The rescue environment will now attempt to find your Linux installation and
mount it under the directory : /mnt/sysimage. You can then make any changes
required to your system. Choose '1' to proceed with this step.
You can choose to mount your file systems read-only instead of read-write by
choosing '2'.
If for some reason this process does not work choose '3' to skip directly to a
shell.

1) Continue
2) Read-only mount
3) Skip to shell
4) Quit (Reboot)

Please make a selection from the above:

[anaconda] 1:main* 2:shell 3:log 4:storage-log 5:program-log Switch tab: Alt+Tab | Help: F1
```

#### 第4步：修复启动文件

```
1 #1.切换到
2 chroot /mnt/sysimage
3
4 #2.将光盘镜像挂在到临时目录
5 mount /dev/sr0 /mnt
6
7 #3.安装grub2到指定磁盘grub2-install /dev/sda
8
9 #4.从挂载的光盘镜像上安装Linux内核
10 rpm -ivh /mnt/Packages/kernel-3.10.0-1062.el7.x86_64.rpm --
11 force
12
13 #5.重新生成grub2的配置文件
14 grub2-mkconfig -o /boot/grub2/grub.cfg
15
16 #6.退出救援模式
17 exit
18 exit
```