# 系统启动和内核管理

## 内容概述

- **CentOS 6 之前版本的启动流程**
- **服务管理**
- **Grub管理**
- **启动排错**
- **CentOS 7 以后版本启动流程**
- **Unit 介绍**
- **服务管理和查看**
- **启动排错**
- **破解口令**
- **修复Grub2**
- **内核优化**

# 1 CentOS 6 的启动管理

## 1.1 Linux 组成

- kernel 实现进程管理、内存管理、网络管理、驱动程序、文件系统、安全功能等功能
- rootfs 包括程序和 glibc 库

  程序：二进制执行文件

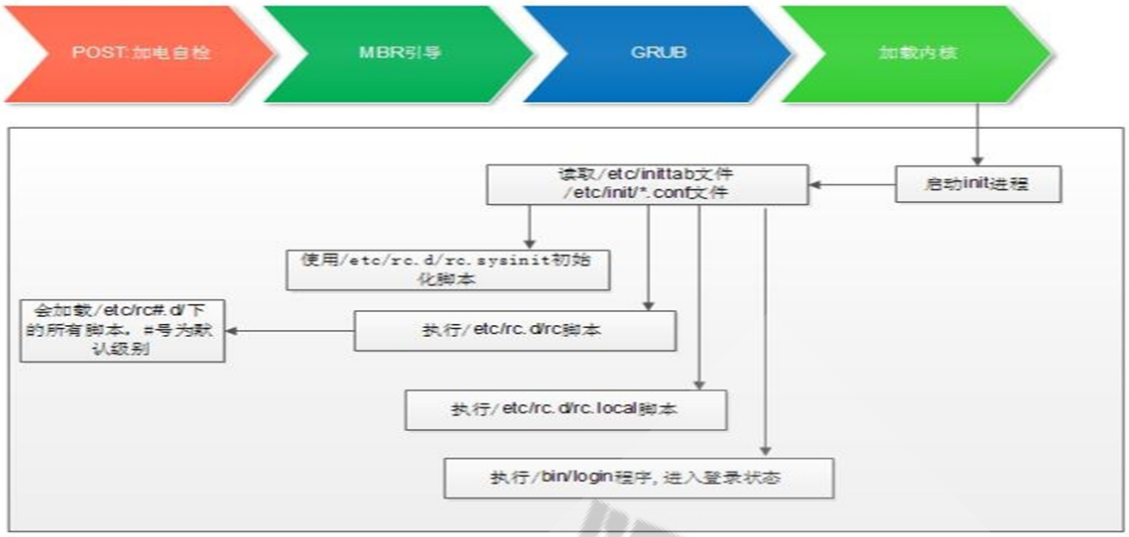
  库：函数集合, function, 调用接口（头文件负责描述）

## 1.2 内核设计流派

- 宏内核(monolithic kernel)：又称单内核和强内核，Unix，Linux
  把所有系统服务都放到内核里，所有功能集成于同一个程序，分层实现不同功能，系统庞大复杂，Linux其实在单内核内核实现了模块化，也就相当于吸收了微内核的优点
- 微内核(micro kernel)：Windows，Solaris，HarmonyOS
  简化内核功能，在内核之外的用户态尽可能多地实现系统服务，同时加入相互之间的安全保护，每种功能使用一个单独子系统实现，将内核功能移到用户空间，性能差

## 1.3 CentOS 6启动流程

## 1.3.1 CentOS 6 启动流程




1. 加载BIOS的硬件信息，获取第一个启动设备
2. 读取第一个启动设备MBR的引导加载程序(grub)的启动信息
3. 加载核心操作系统的核心信息，核心开始解压缩，并尝试驱动所有的硬件设备
4. 核心执行init程序，并获取默认的运行信息
5. init程序执行/etc/rc.d/rc.sysinit文件，重新挂载根文件系统
6. 启动核心的外挂模块
7. init执行运行的各个批处理文件(scripts)
8. init执行/etc/rc.d/rc.local
9. 执行/bin/login程序，等待用户登录
10. 登录之后开始以Shell控制主机

## 1.3.1 硬件启动POST

POST：Power-On-Self-Test，加电自检，是BIOS功能的一个主要部分。负责完成对CPU、主板、内存、硬盘子系统、显示子系统、串并行接口、键盘等硬件情况的检测

主板的ROM：BIOS，Basic Input and Output System，保存着有关计算机系统最重要的基本输入输出程序，系统信息设置、开机加电自检程序和系统启动自举程序等

主板的RAM：CMOS互补金属氧化物半导体，保存各项参数的设定，按次序查找引导设备，第一个有引导程序的设备为本次启动设备

## 1.3.2 启动加载器 bootloader

### 1.3.2.1 grub 功能和组成

bootloader: 引导加载器，引导程序

- Windows: ntloader，仅是启动OS
- Linux：功能丰富，提供菜单，允许用户选择要启动系统或不同的内核版本；把用户选定的内核装载到内存中的特定空间中，解压、展开，并把系统控制权移交给内核

Linux的bootloader

- LILO：LInux LOader，早期的bootloader，功能单一
- GRUB: GRand Unified Bootloader，CentOS 5,6 GRUB 0.97: GRUB Legacy， CentOS 7 以后使用GRUB 2.02

GRUB 启动阶段

- primary boot loader :

  1st stage：MBR的前446个字节

  1.5 stage：MBR 之后的扇区，让stage1中的bootloader能识别stage2所在的分区上的文件系统

- secondary boot loader ： 2nd stage，分区文件/boot/grub/

### 1.3.2.2 CentOS 6 grub 安装

安装 grub的两种方法：

(1) grub-install 安装grub stage1和stage1_5到/dev/DISK磁盘上，并复制GRUB相关文件到 DIR/boot目录下

```
grub-install --root-directory=DIR /dev/DISK
```

(2) grub命令

```
#grub
grub> root (hd#,#)
grub> setup (hd#)
```

范例: 利用grub命令修复grub

```
[root@centos6 ~]#dd if=/dev/zero of=/dev/sda bs=1 count=446
446+0 records in
446+0 records out
446 bytes (446 B) copied, 0.00070171 s, 636 kB/s

[root@centos6 ~]#grub
Probing devices to guess BIOS drives. This may take a long time.
    GNU GRUB  version 0.97  (640K lower / 3072K upper memory)
 [ Minimal BASH-like line editing is supported.  For the first word, TAB
   lists possible command completions.  Anywhere else TAB lists the possible
   completions of a device/filename.]
grub> root (hd0,0)
root (hd0,0)
 Filesystem type is ext2fs, partition type 0x83
grub> setup (hd0)
setup (hd0)
 Checking if "/boot/grub/stage1" exists... no
 Checking if "/grub/stage1" exists... yes
 Checking if "/grub/stage2" exists... yes
 Checking if "/grub/e2fs_stage1_5" exists... yes
 Running "embed /grub/e2fs_stage1_5 (hd0)"...  27 sectors are embedded.
succeeded
 Running "install /grub/stage1 (hd0) (hd0)1+27 p (hd0,0)/grub/stage2
/grub/grub.conf"... succeeded
Done.
grub> quit
[root@centos6 ~]#
```

范例：grub的第1阶段故障无法启动,进行修复

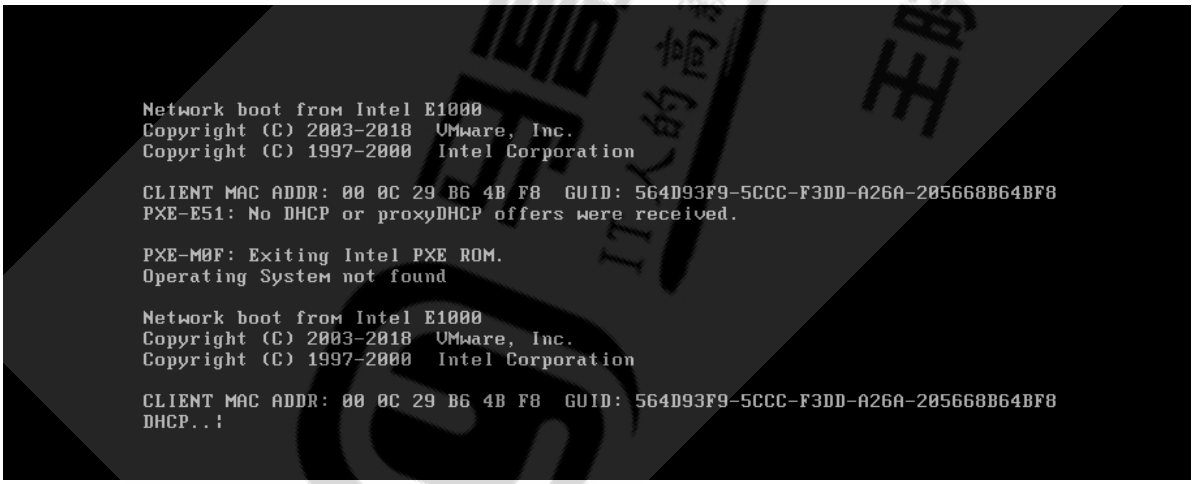
```
[root@centos6 grub]#hexdump  -C -n 512 /dev/sda
00000000  eb 48 90 10 8e d0 bc 00  b0 b8 00 00 8e d8 8e c0  |.H..............|
```

```
00000010  fb be 00 7c bf 00 06 b9  00 02 f3 a4 ea 21 06 00  |...|.........!..|
00000020  00 be be 07 38 04 75 0b  83 c6 10 81 fe fe 07 75  |....8.u........u|
00000030  f3 eb 16 b4 02 b0 01 bb  00 7c b2 80 8a 74 03 02  |.........|...t..|
00000040  80 00 00 80 b0 0c 05 00  00 08 fa 90 90 f6 c2 80  |................|
00000050  75 02 b2 80 ea 59 7c 00  00 31 c0 8e d8 8e d0 bc  |u....Y|..1......|
00000060  00 20 fb a0 40 7c 3c ff  74 02 88 c2 52 f6 c2 80  |. ..@|<.t...R...|
00000070  74 54 b4 41 bb aa 55 cd  13 5a 52 72 49 81 fb 55  |tT.A..U..ZRrI..U|
00000080  aa 75 43 a0 41 7c 84 c0  75 05 83 e1 01 74 37 66  |.uC.A|..u....t7f|
00000090  8b 4c 10 be 05 7c c6 44  ff 01 66 8b 1e 44 7c c7  |.L...|.D..f..D|.|
000000a0  04 10 00 c7 44 02 01 00  66 89 5c 08 c7 44 06 00  |....D...f.\..D..|
000000b0  70 66 31 c0 89 44 04 66  89 44 0c b4 42 cd 13 72  |pf1..D.f.D..B..r|
000000c0  05 bb 00 70 eb 7d b4 08  cd 13 73 0a f6 c2 80 0f  |...p.}....s.....|
000000d0  84 f0 00 e9 8d 00 be 05  7c c6 44 ff 00 66 31 c0  |........|.D..f1.|
000000e0  88 f0 40 66 89 44 04 31  d2 88 ca c1 e2 02 88 e8  |..@f.D.1........|
000000f0  88 f4 40 89 44 08 31 c0  88 d0 c0 e8 02 66 89 04  |..@.D.1......f..|
00000100  66 a1 44 7c 66 31 d2 66  f7 34 88 54 0a 66 31 d2  |f.D|f1.f.4.T.f1.|
00000110  66 f7 74 04 88 54 0b 89  44 0c 3b 44 08 7d 3c 8a  |f.t..T..D.;D.}<.|
00000120  54 0d c0 e2 06 8a 4c 0a  fe c1 08 d1 8a 6c 0c 5a  |T.....L.......l.Z|
00000130  8a 74 0b bb 00 70 8e c3  31 db b8 01 02 cd 13 72  |.t...p..1......r|
00000140  2a 8c c3 8e 06 48 7c 60  1e b9 00 01 8e db 31 f6  |*....H|`......1.|
00000150  31 ff fc f3 a5 1f 61 ff  26 42 7c be 7f 7d e8 40  |1.....a.&B|..}.@|
00000160  00 eb 0e be 84 7d e8 38  00 eb 06 be 8e 7d e8 30  |.....}.8.....}.0|
00000170  00 be 93 7d e8 2a 00 eb  fe 47 52 55 42 20 00 47  |...}.*...GRUB .G|
00000180  65 6f 6d 00 48 61 72 64  20 44 69 73 6b 00 52 65  |eom.Hard Disk.Re|
00000190  61 64 00 20 45 72 72 6f  72 00 bb 01 00 b4 0e cd  |ad. Error.......|
000001a0  10 ac 3c 00 75 f4 c3 00  00 00 00 00 00 00 00 00  |..<.u...........|
000001b0  00 00 00 00 00 00 00 00  e0 96 01 00 00 00 80 20  |............... |
000001c0  21 00 83 aa 28 82 00 08  00 00 00 00 20 00 00 aa  |!...(....... ...|
000001d0  29 82 83 fe ff ff 00 08  20 00 00 80 1a 06 00 fe  |)....... .......|
000001e0  ff ff 83 fe ff ff 00 88  3a 06 00 00 71 02 00 fe  |........:...q...|
000001f0  ff ff 05 fe ff ff 00 88  ab 08 00 78 54 10 55 aa  |...........xT.U.|
00000200
```

#破坏grub第1阶段

```
[root@centos6 grub]#dd if=/dev/zero of=/dev/sda bs=1 count=446
446+0 records in
446+0 records out
446 bytes (446 B) copied, 0.000871905 s, 512 kB/s
[root@centos6 grub]#hexdump  -C -n 512 /dev/sda
00000000  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000001b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 80 20  |............... |
000001c0  21 00 83 aa 28 82 00 08  00 00 00 00 20 00 00 aa  |!...(....... ...|
000001d0  29 82 83 fe ff ff 00 08  20 00 00 80 1a 06 00 fe  |)....... .......|
000001e0  ff ff 83 fe ff ff 00 88  3a 06 00 00 71 02 00 fe  |........:...q...|
000001f0  ff ff 05 fe ff ff 00 88  ab 08 00 78 54 10 55 aa  |...........xT.U.|
00000200
[root@centos6 grub]#hexdump  -C -n 512 /dev/sda -v
00000000  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

```
000000a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000110  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000120  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000130  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000140  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000150  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000160  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000170  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000180  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000190  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000001a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000001b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 80 20  |............... |
000001c0  21 00 83 aa 28 82 00 08  00 00 00 00 20 00 00 aa  |!...(....... ...|
000001d0  29 82 83 fe ff ff 00 08  20 00 00 80 1a 06 00 fe  |)....... .......|
000001e0  ff ff 83 fe ff ff 00 88  3a 06 00 00 71 02 00 fe  |........:...q...|
000001f0  ff ff 05 fe ff ff 00 88  ab 08 00 78 54 10 55 aa  |...........xT.U.|
00000200
[root@centos6 grub]#reboot
#无法启动，出现下面提示
```




```
光盘启动，进入rescue模式
#chroot /mnt/sysimage
#grub-install /dev/sda
#sync
#exit
#exit
```

范例：修复grub的第1.5 阶段故障

```
[root@centos6 ~]#dd if=/dev/zero of=/dev/sda bs=512 count=25 seek=1
[root@centos6 ~]#reboot
#无法启动，显示下面界面
```

```
光盘启动，进入rescue模式
#chroot /mnt/sysimage
#grub-install /dev/sda
#sync
#按 ctrl+alt+delete 三个键重启动
```

### 1.3.2.3 grub legacy 管理

配置文件：/boot/grub/grub.conf <-- /etc/grub.conf
stage2及内核等通常放置于一个基本磁盘分区

grub legacy 功用：
(1) 提供启动菜单、并提供交互式接口
　a：内核参数
　e：编辑模式，用于编辑菜单
　c：命令模式，交互式接口
(2) 加载用户选择的内核或操作系统
　允许传递参数给内核
　可隐藏启动菜单
(3) 为菜单提供了保护机制
　为编辑启动菜单进行认证
　为启用内核或操作系统进行认证

grub的命令行接口

```
help: 获取帮助列表
help KEYWORD: 详细帮助信息
find (hd#,#)/PATH/TO/SOMEFILE:
root (hd#,#)
kernel /PATH/TO/KERNEL_FILE: 设定本次启动的内核文件；额外还可添加许多内核支持使用的cmdline参数
                    例如：max_loop=100 selinux=0 init=/path/to/init
initrd /PATH/TO/INITRAMFS_FILE: 设定为选定的内核提供额外文件的ramdisk
boot: 引导启动选定的内核
```

cat /proc/cmdline 内核参数

内核参数文档:

```
/usr/share/doc/kernel-doc-2.6.32/Documentation/kernel-parameters.txt
```

grub legacy识别硬盘设备

```
(hd#,#)
hd#: 磁盘编号，用数字表示；从0开始编号
#: 分区编号，用数字表示； 从0开始编号

示例：
(hd0,0) 第一块硬盘，第一个分区
```

手动在grub命令行接口启动系统

```
grub> root (hd#,#)
grub> kernel /vmlinuz-VERSION-RELEASE ro root=/dev/DEVICE
grub> initrd /initramfs-VERSION-RELEASE.img
grub> boot
```

grub legacy配置文件：/boot/grub/grub.conf

```
default=#:  #设定默认启动的菜单项；落单项(title)编号从0开始
timeout=#:   #指定菜单项等待选项选择的时长
splashimage=(hd#,#)/PATH/XPM_FILE：#菜单背景图片文件路径
password [--md5| --encrypt] STRING: #启动菜单编辑认证
hiddenmenu：#隐藏菜单
title TITLE：#定义菜单项“标题”，可出现多次
root (hd#,#)：#查找stage2及kernel文件所在设备分区；为grub的根
kernel /PATH/TO/VMLINUZ_FILE [PARAMETERS]：#启动的内核
initrd /PATH/TO/INITRAMFS_FILE: #内核匹配的ramfs文件
password [--md5|--encrypted ] STRING: #启动选定的内核或操作系统时进行认证
```

grub加密生成grub口令

```
grub-md5-crypt
grub-crypt
```

破解root口令：

```
(1) 编辑grub菜单(选定要编辑的title，而后使用a 或 e 命令)
(2) 在选定的kernel后附加1，s，S，single 都可以进入单用户模式
(3) 在kernel所在行，键入“b”命令
```

范例: 给grub 添加密码,防止破解root密码

```
[root@centos6 ~]#grub-crypt
Password:
Retype password:
$6$RedtvBeODOsM8yKq$yKwmmnHSDb9wDRUuZbC3H1ZNwIlf/Mh88MXa3JzXloXyy0hXIxFwLIoMdgmY
FfkWXxkP.vW3ypIla4P5zUKuT.

[root@centos6 ~]#vim /boot/grub/grub.conf
default=0
timeout=5
password --encrypt
$6$RedtvBeODOsM8yKq$yKwmmnHSDb9wDRUuZbC3H1ZNwIlf/Mh88MXa3JzXloXyy0hXIxFwLIoMdgmY
FfkWXxkP.vW3ypIla4P5zUKuT.
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title CentOS 6 (2.6.32-754.el6.x86_64)
```

范例：生成grub启动背景图片

```
[root@centos6 ~]#convert -resize 640x480 -colors 14 winner.png splash.xpm
[root@centos6 ~]#more splash.xpm
#生成splash.xpm.gz
[root@centos6 ~]#gzip   splash.xpm
[root@centos6 ~]#mv splash.xpm.gz /boot/grub
```

## 1.3.3 加载 kernel

kernel 自身初始化过程

1. 探测可识别到的所有硬件设备
2. 加载硬件驱动程序（借助于ramdisk加载驱动）
3. 以只读方式挂载根文件系统
4. 运行用户空间的第一个应用程序：/sbin/init

Linux内核特点：

- 支持模块化：.ko（内核对象），如：文件系统，硬件驱动，网络协议等
- 支持内核模块的动态装载和卸载

内核组成部分：

- 核心文件：/boot/vmlinuz-VERSION-release
  ramdisk：辅助的伪根系统，加载相应的硬件驱动，ramdisk --> ramfs 提高速度
  CentOS 5 /boot/initrd-VERSION-release.img
  CentOS 6 以后版本 /boot/initramfs-VERSION-release.img
- 模块文件：/lib/modules/VERSION-release

范例：误删除内核文件/boot/vmlinuz-2.6.32-754.el6.x86_64无法启动，故障恢复

```
[root@centos6 ~]#rm -f /boot/vmlinuz-2.6.32-754.el6.x86_64
[root@centos6 ~]#reboot

#进入rescue模式
#chroot /mnt/sysimage
#mount /dev/sr0 /mnt/
#cp /mnt/isolinux/vmlinuz /boot/vmlinuz-2.6.32-754.el6.x86_64
#sync
#exit
#reboot
```

ramdisk文件的制作：

- mkinitrd命令

```
mkinitrd /boot/initramfs-$(uname -r).img $(uname -r)
```

- dracut命令

```
dracut /boot/initramfs-$(uname -r).img $(uname -r)
```

范例：误删除/boot/initramfs-2.6.32-754.el6.x86_64.img无法启动，故障恢复

```
[root@centos6 ~]#rm -f /boot/initramfs-2.6.32-754.el6.x86_64.img
[root@centos6 ~]#reboot
#进入rescue模式
#chroot /mnt/sysimage
#mkinitrd /boot/initramfs-$(uname -r).img $(uname -r)
#sync
#exit
#exit
#reboot
```

## 1.3.4 init初始化

POST --> BootSequence (BIOS) --> Bootloader(MBR) --> kernel(ramdisk) --> rootfs(只读) --> init（systemd）

init程序的类型：

SysV: init, CentOS 5之前

配置文件：/etc/inittab

Upstart: init,CentOS 6

配置文件：/etc/inittab, /etc/init/*.conf

Systemd：systemd, CentOS 7

配置文件：/usr/lib/systemd/system

/etc/systemd/system

### 1.3.4.1 运行级别

运行级别：为系统运行或维护等目的而设定；0-6：7个级别，一般使用3, 5做为默认级别

```
0：关机
1：单用户模式(root自动登录)，single，维护模式
2：多用户模式，启动网络功能，但不会启动NFS；维护模式
3：多用户模式，正常模式；文本界面
4：预留级别；可同3级别
5：多用户模式，正常模式；图形界面
6：重启
```

切换级别：

```
init #
```

查看级别：

```
runlevel
who -r
```

定义运行级别

```
/etc/inittab
```

CentOS 5 的inittab文件还定义以下内容

```
初始运行级别(RUN LEVEL)
系统初始化脚本
对应运行级别的脚本目录
捕获某个关键字顺序
定义UPS电源终端/恢复脚本
在虚拟控制台生成getty
在运行级别5初始化X
```

CentOS 5 的inittab文件每一行格式：

```
id:runlevel:action:process

id：是惟一标识该项的字符序列
runlevels：  定义了操作所使用的运行级别
action：  指定了要执行的特定操作
          wait：切换至此级别运行一次
          respawn：此process终止，就重新启动之
          initdefault：设定默认运行级别；process省略
          sysinit：设定系统初始化方式
process：定义了要执行的进程
```

范例：CentOS 5 的inittab文件

```
id:5:initdefault:
si::sysinit:/etc/rc.d/rc.sysinit
l0:0:wait:/etc/rc.d/rc 0
```

```
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
x:5:respawn:/etc/X11/prefdm -nodaemon
```

CentOS 6 /etc/inittab和相关文件

CentOS 6 init程序为 upstart, 其配置文件/etc/inittab, /etc/init/*.conf，配置文件的语法 遵循 upstart 配置文件语法格式，和CentOS5不同

```
/etc/inittab 设置系统默认的运行级别
/etc/init/control-alt-delete.conf
/etc/init/tty.conf
/etc/init/start-ttys.conf
/etc/init/rc.conf
/etc/init/prefdm.conf
```

### 1.3.4.2 初始化脚本 sysinit

```
/etc/rc.d/rc.sysinit
```

```
[root@centos6 ~]#cat /etc/init/rcS.conf
```

系统初始化脚本功能

```
(1) 设置主机名
(2) 设置欢迎信息
(3) 激活udev和selinux
(4) 挂载/etc/fstab文件中定义的文件系统
(5) 检测根文件系统，并以读写方式重新挂载根文件系统
(6) 设置系统时钟
(7) 激活swap设备
(8) 根据/etc/sysctl.conf文件设置内核参数
(9) 激活lvm及software raid设备
(10)加载额外设备的驱动程序
(11)清理操作
```

### 1.3.4.3 服务管理

```
[root@centos6 ~]#cat /etc/init/rc.conf
# rc - System V runlevel compatibility
#
# This task runs the old sysv-rc runlevel scripts.  It
# is usually started by the telinit compatibility wrapper.
#
# Do not edit this file directly. If you want to change the behaviour,
# please create a file rc.override and put your changes there.

start on runlevel [0123456]

stop on runlevel [!$RUNLEVEL]

task

export RUNLEVEL
console output
exec /etc/rc.d/rc $RUNLEVEL
```

service 命令：手动管理服务

```
service 服务 start|stop|restart
service --status-all
```

/etc/rc.d/rc 控制服务脚本的开机自动运行

```
for srv in /etc/rc.d/rcN.d/K*; do
    $srv stop
done
for srv in /etc/rc.d/rcN.d/S*; do
    $srv start
done
```

说明：rc N --> 意味着读取/etc/rc.d/rcN.d/
K: *K##*：##运行次序；数字越小，越先运行；数字越小的服务，通常为依赖到别的服务
S: *S##*：##运行次序；数字越小，越先运行；数字越小的服务，通常为被依赖到的服务

配置服务开机启动

- chkconfig命令
- ntsysv命令

chkconfig 命令管理服务

```
#查看服务在所有级别的启动或关闭设定情形：
chkconfig [--list] [name]

#添加服务
SysV的服务脚本放置于/etc/rc.d/init.d (/etc/init.d)
#!/bin/bash
chkconfig: LLLL nn nn  #LLLL 表示初始在哪个级别下启动，-表示都不启动
description : 描述信息

chkconfig --add name
```

```
#删除服务
chkconfig --del name

#修改指定的运行级别
chkconfig [--level levels] name <on|off|reset>
说明：--level LLLL：指定要设置的级别；省略时表示2345
```

范例：自定义服务脚本

```
[root@centos6 ~]#cat /etc/init.d/testsrv
#!/bin/bash
# chkconfig: - 96 3
# description: This is test service script

. /etc/init.d/functions

start(){
    [ -e /var/lock/subsys/testsrv ] && exit || touch /var/lock/subsys/testsrv
    echo $PATH
    action "Starting testsrv"
    sleep 3
}
stop(){
    [ -e   /var/lock/subsys/testsrv ] && rm /var/lock/subsys/testsrv || exit
    action "Stopping testsrv"
}

status(){
    [ -e /var/lock/subsys/testsrv ] && echo "testsrv is running..." || echo
"testsrv is stopped"
}

case $1 in
start)
    start
    ;;
stop)
    stop
    ;;
restart)
    stop
    start
    ;;
status)
    status
    ;;
*)
    echo $"Usage: $0 {start|stop|status|restart}"
    exit 2
esac


[root@centos6 ~]#chkconfig --add testsrv
[root@centos6 ~]#chkconfig --list testsrv
testsrv         0:off   1:off   2:off   3:off   4:off   5:off   6:off
```

```
[root@centos6 ~]#service  testsrv start
/sbin:/usr/sbin:/bin:/usr/bin
Starting testsrv                                           [  OK  ]
[root@centos6 ~]#chkconfig --del testsrv
```

#### 1.3.4.4 非独立服务

服务分为独立服务和非独立服务

瞬态（Transient）服务被超级守护进程 xinetd 进程所管理，也称为非独立服务

进入的请求首先被xinetd代理

配置文件：

```
/etc/xinetd.conf
/etc/xinetd.d/<service>
```

用chkconfig控制非独立服务开机启动

示例：

```
chkconfig  tftp  on
```

范例: CentOS6 开启 telnet服务

```
[root@centos6 ~]#yum -y install telnet-server
[root@centos6 ~]#chkconfig telnet  on
[root@centos6 ~]#service xinetd start
```

#### 1.3.4.5 开机启动文件 rc.local

```
/etc/rc.local
/etc/rc.d/rc.local
```

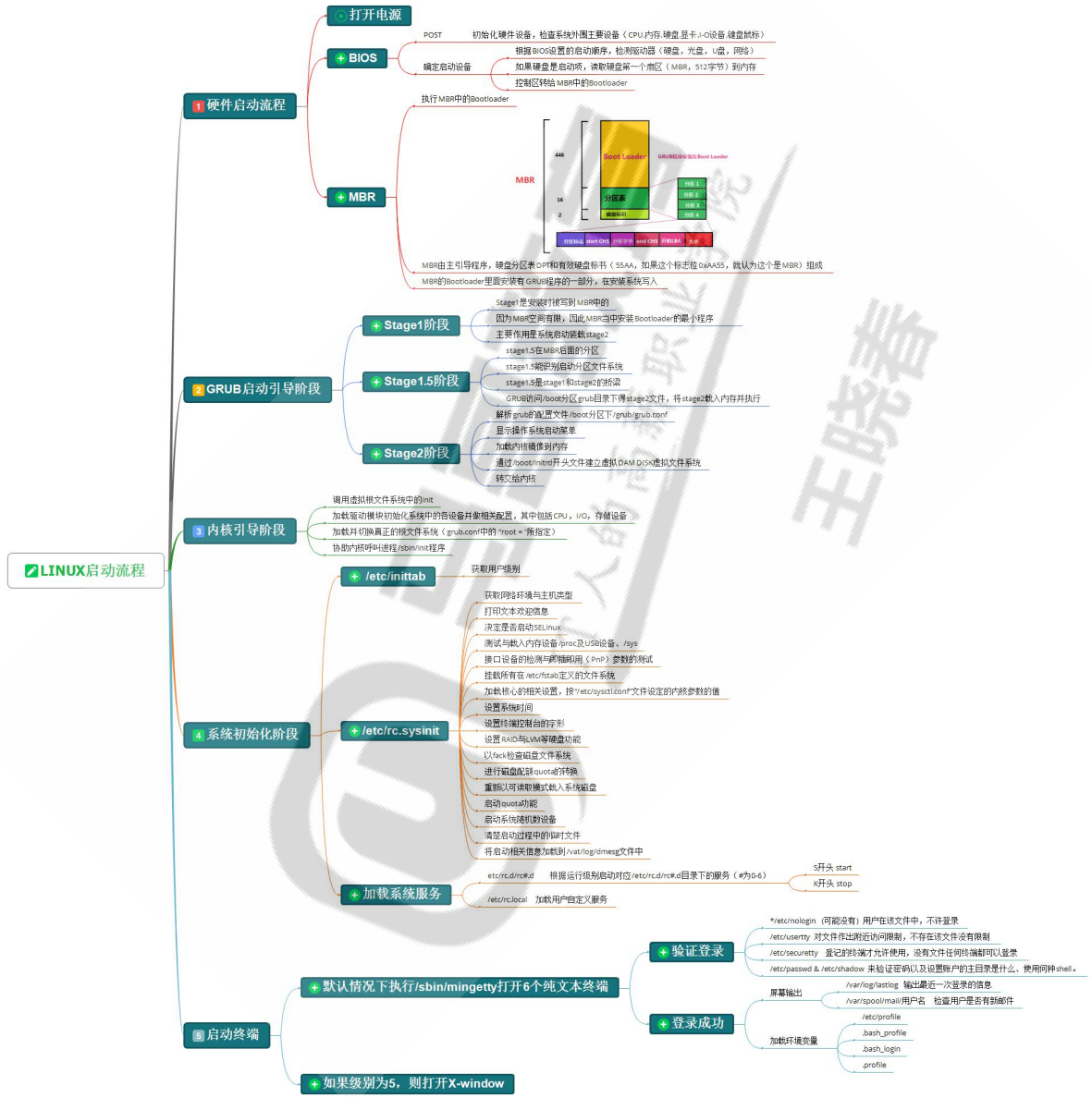
注意：正常级别下，最后启动一个服务S99local没有链接至/etc/rc.d/init.d一个服务脚本，而是指向了/etc/rc.d/rc.local脚本
不便或不需写为服务脚本放置于/etc/rc.d/init.d/目录，且又想开机时自动运行的命令，可直接放置于/etc/rc.d/rc.local文件中
/etc/rc.d/rc.local在指定运行级别脚本后运行

**注意: 默认Ubuntu 无 /etc/rc.local 文件,可以创建此脚本文件并添加执行权限,rc.local的首行必须有shebang机制**

## 1.3.5 CentOS 6 启动过程总结

```
POST
boot loader
vmlinux(initramfs.img)
roofs
/sbin/init
/etc/inittab 设置默认运行级别
/etc/rc.d/rc.sysinit 运行系统初始脚本完成系统初始化
/etc/rc#.d/Sxxxx 启动需要启动服务关闭对应下需要关闭的服务
/etc/rc.d/rc.local
设置登录终端
```

参看：http://s4.51cto.com/wyfs02/M02/87/20/wKiom1fVBELjXsvaAAUkuL83t2Q304.jpg




# 1.4 启动过程的故障排错

## 1.4.1 实战案例

故障: 删除 /sbin/init 无法启动

恢复过程

方法1: 从同一个版本的另的主机复制init文件

```
光盘启动进入secure mode
ifconfig eth0 10.0.0.6/24
scp 10.0.0.16:/sbin/init /mnt/sysimages/sbin/
```

方法2:

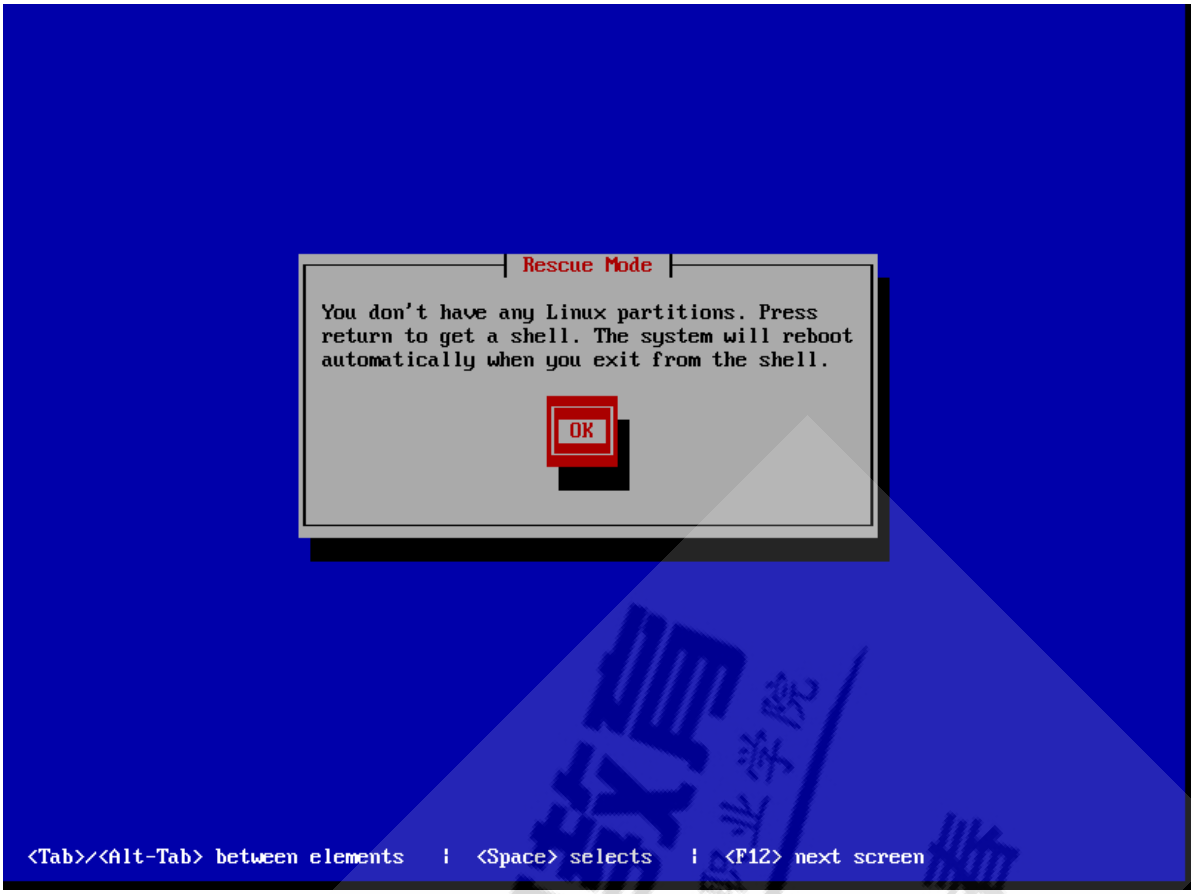
```
1 先进入grub菜单，在kernel参数后加 selinux=0 init=/bin/bash

2 mount -o remount,rw /

3 mount /dev/sr0 /mnt/

方法1
4 rpm2cpio /mnt/Packages/upstart.xxx.rpm | cpio -idv ./sbin/init
5 mv ./sbin/init /sbin/

方法2
4 rpm -ivh /mnt/Packages/upstart.xxx.rpm --force
```

## 1.4.2 实战案例

故障：rm -rf /boot/* 和 /etc/fstab 进行恢复

恢复过程

1. 用光盘进入 rescue mode，找到/ 所在分区并恢复/etc/fstab




```
fdisk -l
mkdir /mnt/rootdir
mount /dev/sdaN /mnt/rootdir
ls /mnt/rootdir
mount /dev/sda2  /mnt/rootdir

vim /mnt/rootdir/etc/fstab
/dev/sda1 /boot ext4 defaults 0 0
/dev/sda2  /    ext4 defaults 0 0
/dev/sda3  /data ext4 defaults 0 0
/dev/sda5  swap swap defaults 0 0

reboot
```

2. rescue mode 恢复内核和initrd 文件
   /dev/sda2 --> /mnt/sysimage

```
chroot /mnt/sysimage
mount /dev/sr0 /mnt/

#方法1
rpm -ivh /mnt/Packages/kernel.xxxx.rpm --force

#方法2
cp  /mnt/isolinux/vmlinuz  /boot/
mkinitrd  /boot/initramfs.img  `uname -r`
```

3. 修复 grub

```
grub-install /dev/sda
vim /boot/grub/grub.conf 方法2
[root@centos6 ~]#cat /boot/grub/grub.conf
default=0
timeout=5
title centos
kernel /vmlinuz root=/dev/sda2
initrd /initramfs.img
```

4. reboot

# 2 systemd和启动流程

## 2.1 systemd 特性

Systemd：从 CentOS 7 版本之后开始用 systemd 实现init进程，系统启动和服务器守护进程管理器，负责在系统启动或运行时，激活系统资源，服务器进程和其它进程

**Systemd新特性**

- 系统引导时实现服务并行启动
- 按需启动守护进程
- 自动化的服务依赖关系管理
- 同时采用socket式与D-Bus总线式激活服务
- socket与服务程序分离
- 向后兼容sysv init脚本
- 使用systemctl 命令管理，systemctl命令固定不变，不可扩展，非由systemd启动的服务，systemctl无法与之通信和控制
- 系统状态快照

systemd 核心概念：unit

unit表示不同类型的systemd对象，通过配置文件进行标识和配置；文件中主要包含了系统服务、监听socket、保存的系统快照以及其它与init相关的信息

Unit类型：

```
#查看unit类型
[root@centos8 ~]#systemctl -t help
Available unit types:
service
socket
target
device
mount
automount
swap
timer
path
slice
scope
```

- service unit: 文件扩展名为.service, 用于定义系统服务
- Socket unit: .socket, 定义进程间通信用的socket文件，也可在系统启动时，延迟启动服务，实现按需启动

- Target unit: 文件扩展名为.target，用于模拟实现运行级别
- Device unit: .device, 用于定义内核识别的设备
- Mount unit: .mount, 定义文件系统挂载点
- Snapshot unit: .snapshot, 管理系统快照
- Swap unit: .swap, 用于标识swap设备
- Automount unit: .automount，文件系统的自动挂载点
- Path unit: .path，用于定义文件系统中的一个文件或目录使用,常用于当文件系统变化时，延迟激活服务，如：spool 目录

unit的配置文件

```
/usr/lib/systemd/system      #每个服务最主要的启动脚本设置，类似于之前的/etc/init.d/
/lib/systemd/system          #ubutun的对应目录,兼容于CentOS7,8和Ubuntu

/run/systemd/system          #系统执行过程中所产生的服务脚本，比上面目录优先运行
/etc/systemd/system          #管理员建立的执行脚本，类似于/etc/rcN.d/Sxx的功能，比上面目录优先运行
```

## 2.2 systemctl管理系统服务service unit

命令：

```
systemctl COMMAND name.service
```

```
#启动：相当于service name start
systemctl start name.service

#停止：相当于service name stop
systemctl stop name.service

#重启：相当于service name restart
systemctl restart name.service

#查看状态：相当于service name status
systemctl status name.service

#禁止自动和手动启动：
systemctl mask name.service

#取消禁止
systemctl unmask name.service

#查看某服务当前激活与否的状态：
systemctl is-active name.service

#查看service文件内容
systemctl cat sshd

#查看所有已经激活的服务：
systemctl list-units --type|-t service

#查看所有服务：
systemctl list-units --type service --all|-a

#设定某服务开机自启，相当于chkconfig name on
```

```
systemctl enable name.service

#设定某服务开机禁止启动：相当于chkconfig name off
systemctl disable name.service

#查看所有服务的开机自启状态，相当于chkconfig --list
systemctl list-unit-files --type service

#用来列出该服务在哪些运行级别下启用和禁用：chkconfig -list name
ls /etc/systemd/system/*.wants/name.service

#查看服务是否开机自启：
systemctl is-enabled name.service

#列出失败的服务
systemctl --failed --type=service

#开机并立即启动或停止
systemctl  enable --now postfix
systemctl  disable  --now  postfix


#查看服务的依赖关系：
systemctl list-dependencies name.service

#杀掉进程：
systemctl kill unitname
```

服务状态

```
#显示状态
systemctl list-unit-files --type service --all
```

- loaded　　Unit配置文件已处理
- active(running) 一次或多次持续处理的运行
- active(exited)　成功完成一次性的配置
- active(waiting)　运行中，等待一个事件
- inactive　不运行
- enabled 开机启动
- disabled 开机不启动
- static　开机不启动，但可被另一个启用的服务激活
- indirect　重定向到别处

范例：systemctl 命令示例

```
#显示所有单元状态
systemctl 或 systemctl list-units
#只显示服务单元的状态
systemctl --type=service
#显示sshd服务单元
systemctl -l status sshd.service
#验证sshd服务当前是否活动
systemctl is-active sshd
#启动，停止和重启sshd服务
```

```
systemctl start sshd.service
systemctl stop sshd.service
systemctl restart sshd.service

#重新加载配置
systemctl reload sshd.service
#列出活动状态的所有服务单元
systemctl list-units --type=service
#列出所有服务单元
systemctl list-units --type=service --all
#查看服务单元的启用和禁用状态
systemctl list-unit-files  --type=service

#列出依赖的单元
systemctl list-dependencies sshd
验证sshd服务是否开机启动
systemctl is-enabled sshd
禁用network，使之不能自动启动,但手动可以
systemctl disable network

#启用network
systemctl enable  network

#禁用network，使之不能手动或自动启动
systemctl mask network

#启用network
systemctl unmask network
```

## 2.3 service unit文件格式

/etc/systemd/system：系统管理员和用户使用
/usr/lib/systemd/system：发行版打包者使用

帮助参考：

systemd.directives（7），systemd.unit(5),systemd.service(5), systemd.socket(5), systemd.target(5),systemd.exec(5)

参考链接

```
http://www.jinbuguo.com/systemd/systemd.service.html
```

unit 格式说明：

- 以 “#” 开头的行后面的内容会被认为是注释
- 相关布尔值，1、yes、on、true 都是开启，0、no、off、false 都是关闭
- 时间单位默认是秒，所以要用毫秒（ms）分钟（m）等须显式说明

service unit file文件通常由三部分组成：

- [Unit]：定义与Unit类型无关的通用选项；用于提供unit的描述信息、unit行为及依赖关系等
- [Service]：与特定类型相关的专用选项；此处为Service类型
- [Install]：定义由“systemctl enable”以及"systemctl disable“命令在实现服务启用或禁用时用到的一些选项

Unit段的常用选项：

- Description：描述信息
- After：定义unit的启动次序，表示当前unit应该晚于哪些unit启动，其功能与Before相反
- Requires：依赖到的其它units，强依赖，被依赖的units无法激活时，当前unit也无法激活
- Wants：依赖到的其它units，弱依赖
- Conflicts：定义units间的冲突关系

```
[root@centos8 ~]#head -n 5 /lib/systemd/system/postfix.service
[Unit]
Description=Postfix Mail Transport Agent
After=syslog.target network.target
Conflicts=sendmail.service exim.service
```

Service段的常用选项：

- Type：定义影响ExecStart及相关参数的功能的unit进程启动类型
  - simple：默认值，这个daemon主要由ExecStart接的指令串来启动，启动后常驻于内存中
  - forking：由ExecStart启动的程序透过spawns延伸出其他子程序来作为此daemon的主要服务。原生父程序在启动结束后就会终止
  - oneshot：与simple类似，不过这个程序在工作完毕后就结束了，不会常驻在内存中
  - dbus：与simple类似，但这个daemon必须要在取得一个D-Bus的名称后，才会继续运作.因此通常也要同时设定BusNname= 才行
  - notify：在启动完成后会发送一个通知消息。还需要配合 NotifyAccess 来让 Systemd 接收消息
  - idle：与simple类似，要执行这个daemon必须要所有的工作都顺利执行完毕后才会执行。这类的daemon通常是开机到最后才执行即可的服务
- EnvironmentFile：环境配置文件
- ExecStart：指明启动unit要运行命令或脚本的绝对路径
- ExecStartPre： ExecStart前运行
- ExecStartPost： ExecStart后运行
- ExecStop：指明停止unit要运行的命令或脚本
- Restart：当设定Restart=1 时，则当次daemon服务意外终止后，会再次自动启动此服务
- RestartSec: 设置在重启服务( `Restart=` )前暂停多长时间。 默认值是100毫秒(100ms)。 如果未指定时间单位，那么将视为以秒为单位。 例如设为"20"等价于设为"20s"。
- PrivateTmp：设定为yes时，会在生成/tmp/systemd-private-UUID-NAME.service-XXXXX/tmp/目录

Install段的常用选项：

- Alias：别名，可使用systemctl command Alias.service
- RequiredBy：被哪些units所依赖，强依赖
- WantedBy：被哪些units所依赖，弱依赖
- Also：安装本服务的时候还要安装别的相关服务

注意：对于新创建的unit文件，或者修改了的unit文件，要通知systemd重载此配置文件,而后可以选择重启

```
systemctl  daemon-reload
```

范例：查看service文件

```
[root@rocky8 ~]#systemctl cat sshd.service
```

```
# /usr/lib/systemd/system/sshd.service
[Unit]
Description=OpenSSH server daemon
Documentation=man:sshd(8) man:sshd_config(5)
After=network.target sshd-keygen.target
Wants=sshd-keygen.target

[Service]
Type=notify
EnvironmentFile=-/etc/crypto-policies/back-ends/opensshserver.config
EnvironmentFile=-/etc/sysconfig/sshd
ExecStart=/usr/sbin/sshd -D $OPTIONS $CRYPTO_POLICY
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=42s

[Install]
WantedBy=multi-user.target
```

范例:

```
[root@centos8 ~]#head -n 5 /lib/systemd/system/postfix.service
[Unit]
Description=Postfix Mail Transport Agent
After=syslog.target network.target
Conflicts=sendmail.service exim.service
```

范例: 自定义service的unit文件

```
[root@centos8 ~]#vim /lib/systemd/system/hello.service
[Unit]
Description=Hello World
[Service]
TimeoutStartSec=0
ExecStart=/bin/sh -c "while true; do echo Hello World; sleep 1; done"
ExecStop=/bin/kill sh
[Install]
WantedBy=multi-user.target

[root@centos8 ~]#systemctl daemon-reload
[root@centos8 ~]#systemctl enable --now   hello
[root@centos8 ~]#systemctl status hello
● hello.service - Hello World
   Loaded: loaded (/usr/lib/systemd/system/hello.service; enabled; vendor preset:
disabled)
   Active: active (running) since Thu 2020-06-11 10:02:05 CST; 3min 16s ago
 Main PID: 661 (sh)
    Tasks: 2 (limit: 4895)
   Memory: 1.0M
   CGroup: /system.slice/hello.service
           ├─ 661 /bin/sh -c while true; do echo Hello World; sleep 1; done
           └─1535 sleep 1

Jun 11 10:05:12 centos8.wangxiaochun.com sh[661]: Hello World
Jun 11 10:05:13 centos8.wangxiaochun.com sh[661]: Hello World
```

```
Jun 11 10:05:14 centos8.wangxiaochun.com sh[661]: Hello World
Jun 11 10:05:15 centos8.wangxiaochun.com sh[661]: Hello World
Jun 11 10:05:16 centos8.wangxiaochun.com sh[661]: Hello World
Jun 11 10:05:17 centos8.wangxiaochun.com sh[661]: Hello World
Jun 11 10:05:18 centos8.wangxiaochun.com sh[661]: Hello World
Jun 11 10:05:19 centos8.wangxiaochun.com sh[661]: Hello World
Jun 11 10:05:20 centos8.wangxiaochun.com sh[661]: Hello World
Jun 11 10:05:21 centos8.wangxiaochun.com sh[661]: Hello World

[root@centos8 ~]#tail /var/log/messages   -f
Sep  2 09:54:32 centos8 sh[26543]: Hello World
Sep  2 09:54:33 centos8 sh[26543]: Hello World
Sep  2 09:54:34 centos8 sh[26543]: Hello World
Sep  2 09:54:35 centos8 sh[26543]: Hello World
Sep  2 09:54:36 centos8 sh[26543]: Hello World
Sep  2 09:54:37 centos8 sh[26543]: Hello World
Sep  2 09:54:38 centos8 sh[26543]: Hello World
Sep  2 09:54:39 centos8 sh[26543]: Hello World
Sep  2 09:54:40 centos8 systemd[1]: Stopping Hello World...
Sep  2 09:54:40 centos8 systemd[1]: Stopped Hello World.

#上面serivce文件也可支持ubuntu18,04
[root@ubuntu1804 ~]#tail /var/log/syslog
Apr 23 09:49:31 ubuntu1804 systemd[1]: Started Terminate Plymouth Boot Screen.
Apr 23 09:49:31 ubuntu1804 systemd[1]: Reached target Multi-User System.
Apr 23 09:49:31 ubuntu1804 systemd[1]: Reached target Graphical Interface.
Apr 23 09:49:31 ubuntu1804 systemd[1]: Starting Update UTMP about System
Runlevel Changes...
Apr 23 09:49:31 ubuntu1804 systemd[1]: Started Update UTMP about System Runlevel
Changes.
Apr 23 09:49:31 ubuntu1804 systemd[1]: Startup finished in 3.623s (kernel) +
12.818s (userspace) = 16.441s.
Apr 23 09:49:52 ubuntu1804 systemd-timesyncd[688]: Synchronized to time server
91.189.89.198:123 (ntp.ubuntu.com).
Apr 23 09:50:43 ubuntu1804 systemd[1]: Reloading.
Apr 23 09:50:44 ubuntu1804 systemd[1]: Started Hello World.
Apr 23 09:50:44 ubuntu1804 sh[1202]: Hello World
```

范例：服务Unit文件

```
[Unit]
Description=The Nginx HTTP Server daemon  # 描述信息
After=network.target remote-fs.target nss-lookup.target # 指定启动nginx之前需要其他
的其他服务，如network.target等

[Service]
# Type为服务类型，仅启动一个主进程的服务为simple，需要启动若干子进程的服务为forking
Type=forking

# 设置执行systemctl start nginx后需要启动的具体命令
ExecStart=/usr/local/nginx/sbin/nginx

# 设置执行systemctl reload nginx后需要执行的具体命令
ExecReload=/usr/local/nginx/sbin/nginx -s reload

# 设置执行systemctl stop nginx后需要执行的具体命令
ExecStop=/bin/kill -s QUIT ${MAINPID}
```

```
[Install]
# 设置在什么模式下被安装，设置开机启动的时候需要
WantedBy=multi-user.target
```

范例：服务Unit文件

```
vim /usr/lib/systemd/system/tomcat.service
[Unit]
Description=java tomcat project
After=syslog.target network.target

[Service]
Type=forking
EnvironmentFile=/usr/local/tomcat/conf/tomcat.conf
ExecStart=/usr/local/tomcat/bin/startup.sh
ExecStop=/usr/local/tomcat/bin/shutdown.sh
PrivateTmp=true
User=tomcat

[Install]
WantedBy=multi-user.target
```

范例：服务Unit文件

```
vim /etc/systemd/system/bak.service
[Unit]
Description=backup  /etc
Requires=atd.service

[Service]
Type=simple
ExecStart=/bin/bash -c "echo /data/bak.sh | at now"

[Install]
WantedBy=multi-user.target

systemctl daemon-reload
systemctl start bak
```

范例：Ubuntu实现开机自动运行程序

```
[root@ubuntu1804 ~]#vim  /etc/rc.local
[root@ubuntu1804 ~]#cat /etc/rc.local
#!/bin/bash
echo -e '\E[31;1mstarting test service\E[0m'
sleep 10
[root@ubuntu1804 ~]#chmod +x  /etc/rc.local
[root@ubuntu1804 ~]#reboot
```

## 2.4 运行级别

target units：相当于CentOS 6之前的runlevel ,unit配置文件：.target

```
ls /usr/lib/systemd/system/*.target
systemctl list-unit-files --type target  --all
```

和运行级别对应关系

```
0  ==> runlevel0.target, poweroff.target
1  ==> runlevel1.target, rescue.target
2  ==> runlevel2.target, multi-user.target
3  ==> runlevel3.target, multi-user.target
4  ==> runlevel4.target, multi-user.target
5  ==> runlevel5.target, graphical.target
6  ==> runlevel6.target, reboot.target
```

查看依赖性：

```
systemctl list-dependencies graphical.target
```

级别切换：相当于 init N

```
systemctl isolate name.target
```

进入默认target

```
systemctl  default
```

范例：

```
#切换至字符模式
systemctl isolate multi-user.target
```

注意：只有/lib/systemd/system/*.target文件中AllowIsolate=yes 才能切换(修改文件需执行systemctl daemon-reload才能生效)

获取默认运行级别： 相当于查看 /etc/inittab

```
systemctl get-default
```

修改默认级别：相当于修改 /etc/inittab

```
systemctl set-default name.target
```

范例：

```
[root@centos8 ~]#systemctl set-default multi-user.target
[root@centos8 ~]#ls -l /etc/systemd/system/default.target
lrwxrwxrwx. 1 root root 37 Nov  7 19:32 /etc/systemd/system/default.target ->
/lib/systemd/system/multi-user.target
```

切换至紧急救援模式：

```
systemctl rescue
```

切换至emergency模式：

```
systemctl emergency
```

说明：rescue.target 比emergency 支持更多的功能，例如日志等

传统命令init，poweroff，halt，reboot都成为systemctl的软链接

```
#关机
systemctl halt、systemctl poweroff

#重启：
systemctl reboot

#挂起：
systemctl suspend

#休眠：
systemctl hibernate

#休眠并挂起：
systemctl hybrid-sleep
```

范例：禁用ctrl+alt+delete 重启快捷键

```
[root@centos8 ~]#ls -l  /lib/systemd/system/ctrl-alt-del.target
lrwxrwxrwx. 1 root root 13 May 23  2019 /lib/systemd/system/ctrl-alt-del.target
-> reboot.target
[root@centos8 ~]#systemctl mask ctrl-alt-del.target
Created symlink /etc/systemd/system/ctrl-alt-del.target → /dev/null.
[root@centos8 ~]#init q
[root@centos8 ~]#systemctl daemon-reload
```

## 2.5 设置内核参数

设置内核参数，只影响当次启动

启动时，到启动菜单，按e键，找到在linux 开头的行后添加systemd.unit=desired.target

比如：

```
systemd.unit=emergency.target
systemd.unit=rescue.target
```

## 2.6 CentOS 7之后版本引导顺序

1. UEFi或BIOS初始化，运行POST开机自检
2. 选择启动设备
3. 引导装载程序, centos7是grub2,加载装载程序的配置文件：
   /etc/grub.d/
   /etc/default/grub
   /boot/grub2/grub.cfg
4. 加载initramfs驱动模块
5. 加载内核选项

6. 内核初始化，centos7使用systemd代替init
7. 执行initrd.target所有单元，包括挂载/etc/fstab
8. 从initramfs根文件系统切换到磁盘根目录
9. systemd执行默认target配置，配置文件/etc/systemd/system/default.target
10. systemd执行sysinit.target初始化系统及basic.target准备操作系统
11. systemd启动multi-user.target下的本机与服务器服务
12. systemd执行multi-user.target下的/etc/rc.d/rc.local
13. Systemd执行multi-user.target下的getty.target及登录服务
14. systemd执行graphical需要的服务

通过systemd-analyze 工具可以了解启动的详细过程

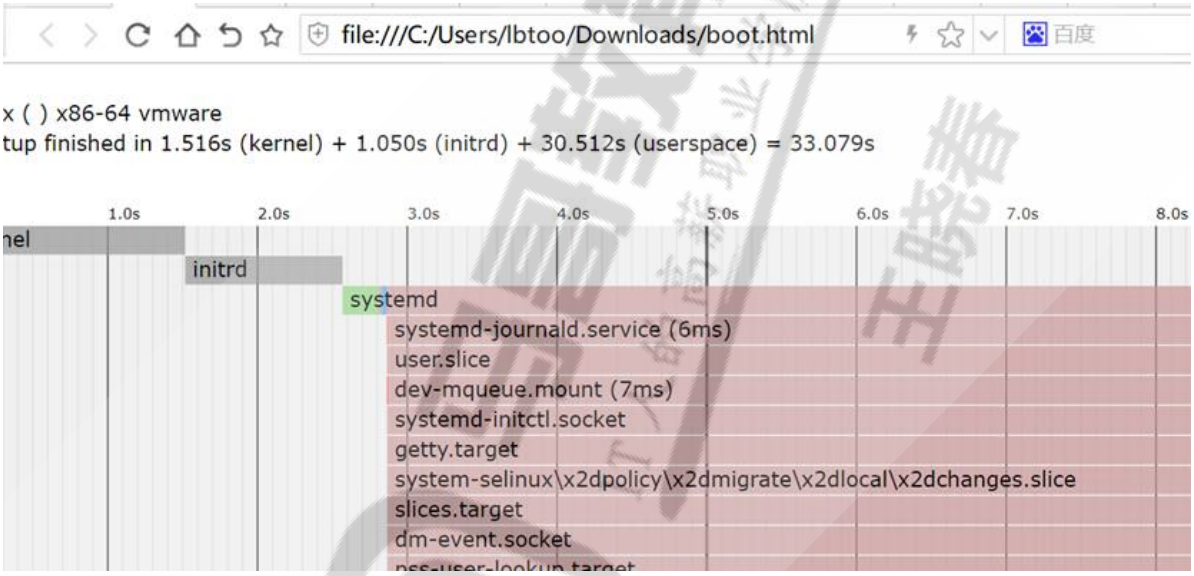
范例：

```
[root@centos8 ~]#systemd-analyze blame
          1.862s kdump.service
          1.047s tuned.service
           666ms dracut-initqueue.service
           523ms auditd.service
           379ms initrd-switch-root.service
           314ms sssd.service
           302ms systemd-rfkill.service
           219ms NetworkManager-wait-online.service
           211ms polkit.service
           178ms systemd-udev-trigger.service
           159ms dracut-pre-pivot.service
           138ms autofs.service
           129ms systemd-vconsole-setup.service
           109ms NetworkManager.service
           104ms sysroot.mount
            96ms boot.mount
            80ms initrd-parse-etc.service
            63ms dracut-cmdline.service
            55ms systemd-fsck@dev-disk-by\x2duuid-
3527552a\x2d2f80\x2d4110\x2d9c37\x2d4aae9e500fd6.service
            50ms systemd-udevd.service
            50ms user@0.service
            44ms systemd-logind.service
            43ms systemd-journald.service
            41ms data.mount
            40ms rsyslog.service
            39ms sshd.service
            36ms systemd-tmpfiles-setup-dev.service
            35ms systemd-user-sessions.service
            35ms import-state.service
            33ms systemd-tmpfiles-setup.service
            32ms plymouth-quit-wait.service
            31ms systemd-sysctl.service
            30ms plymouth-quit.service
            30ms nis-domainname.service
            29ms systemd-remount-fs.service
            28ms dev-disk-by\x2duuid-
8363289d\x2d138e\x2d4e4a\x2dabaf\x2d6e028babc924.swap
            28ms dracut-pre-udev.service
            25ms dev-hugepages.mount
            23ms kmod-static-nodes.service
            23ms plymouth-start.service
```

```
          21ms sys-kernel-debug.mount
          20ms systemd-journal-flush.service
          20ms plymouth-read-write.service
          19ms systemd-random-seed.service
          15ms initrd-cleanup.service
          15ms systemd-fsck-root.service
          15ms plymouth-switch-root.service
          11ms initrd-udevadm-cleanup-db.service
           9ms systemd-update-utmp.service
           9ms dracut-shutdown.service
           8ms systemd-update-utmp-runlevel.service
           5ms dev-mqueue.mount
           2ms sys-kernel-config.mount
```

范例：生成网页

```
systemd-analyze plot > boot.html
```




## 2.7 破解 CentOS 7和8的 root 密码

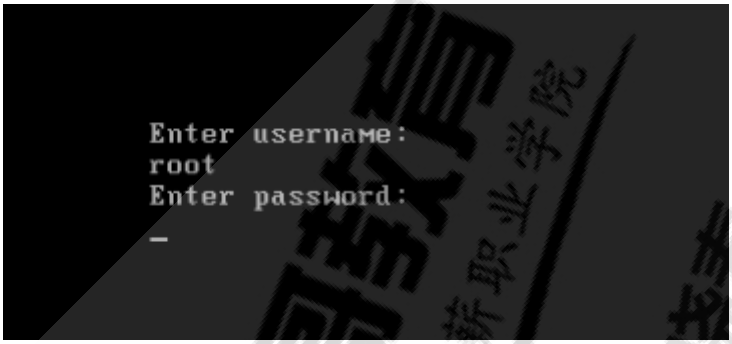
方法一

```
启动时任意键暂停启动
按e键进入编辑模式
将光标移动linux 开始的行，添加内核参数 rd.break
按ctrl-x启动
mount -o remount,rw  /sysroot
chroot /sysroot
passwd root

#如果SELinux是启用的,才需要执行下面操作,如查没有启动,不需要执行
touch /.autorelabel

exit
reboot
```

方法二

```
#此方式也适用于ubuntu18.04
启动时任意键暂停启动
按e键进入编辑模式
将光标移动linux 开始的行，改为 rw init=/sysroot/bin/sh
按ctrl-x启动
chroot /sysroot
passwd root

#如果SELinux是启用的,才需要执行下面操作,如查没有启动,不需要执行
touch /.autorelabel

exit
reboot
```

## 2.8 实现 GRUB2 安全




```
#添加grub密码
[root@centos8 ~]#grub2-setpassword
Enter password:
Confirm password:
[root@centos8 ~]#ls -l /boot/grub2/
total 32
drwxr-xr-x 2 root root  4096 Jan 19 15:17 fonts
-rw-r--r-- 1 root root  5101 Jan 19 15:18 grub.cfg
-rw-r--r-- 1 root root  1024 Jan 19 15:18 grubenv
drwxr-xr-x 2 root root 12288 Jan 19 15:17 i386-pc
-rw------- 1 root root   298 Jan 19 18:20 user.cfg
[root@centos8 ~]#ls -l /boot/grub2/user.cfg
-rw------- 1 root root 298 Jan 19 18:20 /boot/grub2/user.cfg
[root@centos8 ~]#cat /boot/grub2/user.cfg
GRUB2_PASSWORD=grub.pbkdf2.sha512.10000.60AAA29A65F4DC77E8861EF25BDE2034C9B30CE1
E07EE688D7F30460E7E87E7356B0893A6DFFB250B27D2EB9D3ED3E9207199C494D7882E2E8C772C8
2E2DDB7A.5E42FD69FA04293DECD68F077E83875A8E4572A7FBB89BA9F161B15EAFE54FBA963FE5D
52E1676494482339623180 3E5118DA1D9CAF3EB73C175A7D7A3682A90

#清空grub密码
[root@centos8 ~]#cat /dev/null > /boot/grub2/user.cfg
#或者
[root@centos8 ~]#rm -f /boot/grub2/user.cfg
```

## 2.9 修复 GRUB2

GRUB2：CentOS 7，8及ubuntu1804都使用

引导提示时可以使用命令行界面，可从文件系统引导

主要配置文件：/boot/grub2/grub.cfg

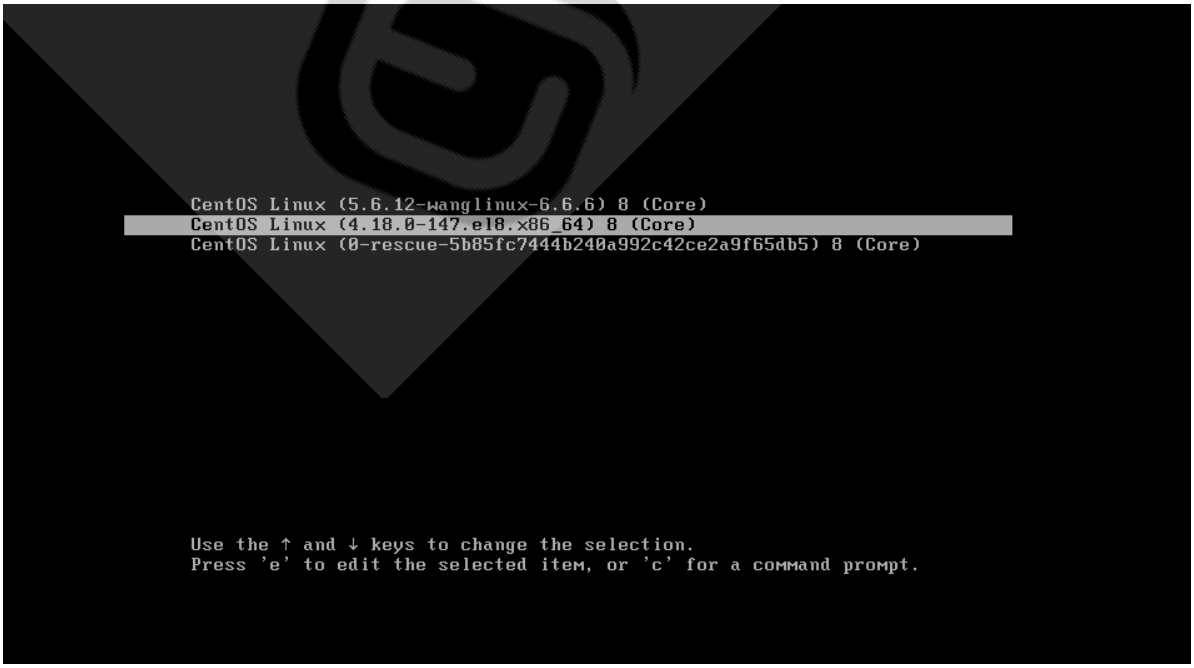
修复配置文件：grub2-mkconfig > /boot/grub2/grub.cfg

修复grub

```
grub2-install /dev/sda  #BIOS环境
grub2-install           #UEFI环境
```

范例：调整默认启动内核

```
[root@centos8 ~]#cat /boot/grub2/grubenv
# GRUB Environment Block
saved_entry=5b85fc7444b240a992c42ce2a9f65db5-5.6.12-wanglinux-6.6.6
kernelopts=root=UUID=f7f53add-b184-4ddc-8d2c-5263b84d1e15 ro crashkernel=auto
resume=UUID=eebe3bc7-6d52-4ad9-86aa-916f1a123fd4 rhgb quiet net.ifnames=0
boot_success=0
################################################################################
################################################################################
################################################################################
################################################################################
################################################################################
################################################################################
################################################################################
################################################################################
################################################################################
##########################################[root@centos8 ~]#cd
[root@centos8 ~]#ls /boot/loader/entries/
5b85fc7444b240a992c42ce2a9f65db5-0-rescue.conf
5b85fc7444b240a992c42ce2a9f65db5-4.18.0-147.el8.x86_64.conf
5b85fc7444b240a992c42ce2a9f65db5-5.6.12-wanglinux-6.6.6.conf

#以下命令是修改 /boot/grub2/grubenv 实现
[root@centos8 ~]#grub2-set-default  1
```




# 2.10 故障排错实战案例

## 2.10.1 实战案例1：CentOS 7,8 破坏MBR后进行恢复

```
dd if=/dev/zero of=/dev/sda bs=1 count=446
光盘进入救援模式
grub2-install --root-directory=/mnt/sysimage /dev/sda
```

## 2.10.2 实战案例2：CentOS 7,8 删除 /boot/grub2/ 所有内容进行恢复


```
#光盘进入救援模式
chroot /mnt/sysimage
grub2-install  /dev/sda
grub2-mkconfig -o /boot/grub2/grub.cfg
```

## 2.10.3 实战案例3：CentOS 7,8 删除 /boot/ 下所有文件后进行恢复

```
1 光盘救援模式下安装grub2
特别说明：Centos8 必须先修复grub，再安装kernel,否则安装kernel-core时会提示grub出错
#centos8
chroot /mnt/sysroot

#centos7
chroot /mnt/sysimage

#安装grub
grub2-install /dev/sda  #BIOS环境
grub2-install           #UEFI环境

2安装Kernel
#CentOS 7
mount /dev/sr0 /mnt
rpm -ivh /mnt/Packages/kernel-3.10.0-1062.el7.x86_64.rpm --force

#CentOS 8
mount /dev/sr0 /mnt
rpm -ivh /mnt/BaseOS/Packages/kernel-core-4.18.0-147.el8.x86_64.rpm --force

3修复grub配置文件
#生成grub.cfg文件
grub2-mkconfig -o /boot/grub2/grub.cfg

4 退出重启
sync
sync
exit
exit
```

# 3 /proc 目录和内核参数管理

/proc目录：内核把自己内部状态信息及统计信息，以及可配置参数通过proc伪文件系统加以输出

帮助：man proc

内核参数：

- 只读：只用于输出信息
- 可写：可接受用户指定“新值”来实现对内核某功能或特性的配置

/proc/sys 设置

> sysctl是一个允许改变正在运行中的Linux系统的接口，修改的是针对整个系统的内核参数。sysctl的修改是立即且临时的（重启后失效）。也可以通过修改sysctl.conf配置文件，达到永久生效。

- sysctl 命令用于查看或设定此目录中诸多参数

  ```
  sysctl -w path.to.parameter=VALUE
  ```

- 默认配置文件：/etc/sysctl.conf 及以下文件

  ```
  /run/sysctl.d/*.conf
  /etc/sysctl.d/*.conf
  /usr/local/lib/sysctl.d/*.conf
  /usr/lib/sysctl.d/*.conf
  /lib/sysctl.d/*.conf
  /etc/sysctl.conf
  ```

范例：

```
sysctl -w kernel.hostname=mail.magedu.com
```

- echo命令通过重定向方式也可以修改大多数参数的值

  ```
  echo "VALUE" > /proc/sys/path/to/parameter
  ```

  范例：

  ```
  echo “websrv”  > /proc/sys/kernel/hostname
  ```

sysctl命令：

(1) 临时设置某参数

```
sysctl -w parameter=VALUE
```

(2) 通过读取配置文件设置参数

```
sysctl -p [/path/to/conf_file]
```

(3) 查看指定参数当前值

```
sysctl  [/path/to/conf_file]
```

(4) 查看所有生效参数

```
sysctl -a
```

常用的内核参数:

```
net.ipv4.ip_forward
net.ipv4.icmp_echo_ignore_all
net.ipv4.ip_nonlocal_bind        #允许应用程序可以监听本地不存在的IP
vm.drop_caches
fs.file-max = 1020000            #全局打开文件的最大数

vm.overcommit_memory = 0
#0表示内核将检查是否有足够可用内存供应用进程使用；如果有足够的可用内存，内存申请允许；否则内存申请失败，并把错误返回给应用进程。
#1表示内核允许分配所有的物理内存，而不管当前的内存状态如何。
#2表示内核允许分配超过所有物理内存和交换空间总和的内存。

vm.swappiness = 10               #设置内存还剩余10%空闲空间时，就会使用swap空间，默认值为30，值越大表示越倾向于使用swap

#禁用IPv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
```

范例:

```
[root@centos8 ~]#cat /proc/sys/net/ipv4/icmp_echo_ignore_all
0
[root@centos8 ~]#vim /etc/sysctl.d/test.conf
[root@centos8 ~]#cat /etc/sysctl.d/test.conf
net.ipv4.icmp_echo_ignore_all=1
[root@centos8 ~]#sysctl  -p /etc/sysctl.d/test.conf
net.ipv4.icmp_echo_ignore_all = 1
[root@centos8 ~]#cat /proc/sys/net/ipv4/icmp_echo_ignore_all
1
```

范例: 清除缓存

```
[root@centos8 ~]#man proc | grep -A 3 drop_caches
       /proc/sys/vm/drop_caches (since Linux 2.6.16)
              Writing  to  this  file causes the kernel to drop clean caches,
dentries, and inodes from memory, causing that memory to become free.  This can
be useful
              for memory management testing and performing reproducible
filesystem benchmarks.  Because writing to this file causes the benefits of
caching to be lost,
              it can degrade overall system performance.
--
                  echo 1 > /proc/sys/vm/drop_caches

              To free dentries and inodes, use:
```

```
                echo 2 > /proc/sys/vm/drop_caches

            To free pagecache, dentries and inodes, use:

                echo 3 > /proc/sys/vm/drop_caches

            Because writing to this file is a nondestructive operation and
dirty objects are not freeable, the user should run sync(1) first.


[root@centos8 ~]#cp /dev/sda /data/f1.img
^C
[root@centos8 ~]#ll -h /data/f1.img
-rw-r----- 1 root root 1.3G Aug 31 16:02 /data/f1.img
[root@centos8 ~]#free -h
              total        used        free      shared  buff/cache   available
Mem:          952Mi       115Mi        72Mi       0.0Ki       764Mi       695Mi
Swap:         2.0Gi        71Mi       1.9Gi
[root@centos8 ~]#echo 3 > /proc/sys/vm/drop_caches
[root@centos8 ~]#free -h
              total        used        free      shared  buff/cache   available
Mem:          952Mi       115Mi       787Mi       0.0Ki        48Mi       742Mi
Swap:         2.0Gi        71Mi       1.9Gi
```