Linux 云计算集群架构师

学神 IT 教育：从零基础到实战，从入门到精通！

联系方式：

学神 IT 教育官方网站: http://www.xuegod.cn

Linux 云计算架构师进阶学习群 QQ 群: 1072932914

学习顾问：小语老师　　学习顾问：边边老师　　学神微信公众号

微信扫码添加学习顾问微信，同时扫码关注学神公众号了解最新行业动态，获取更多学习资料及答疑就业服务！

第十七章　Linux 系统启动原理及故障排除

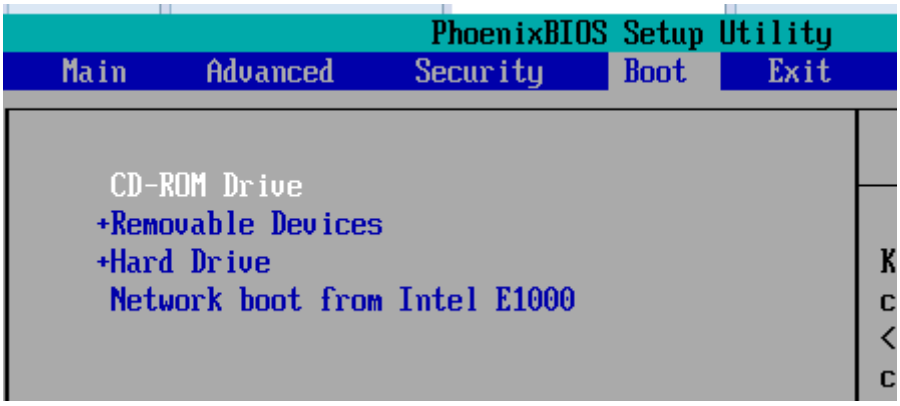
本节所讲内容：

17.1　centos6 系统启动过程及相关配置文件
17.2　centos8 系统启动过程及相关配置文件
17.3　实战-加密 grub 防止黑客通过 reboot 系统破解 root 密码
17.4　实战-通过 liveCD 进入救援模式-重装 grub 修复损坏的系统

17.1　centos6 系统启动过程及相关配置文件

17.1.1　centos6 系统启动过程（了解）

1. 加载 BIOS（传统 BIOS） 的硬件信息，根据设定取得第一个可开机引导设置，如：光驱，硬盘，网络，USB。(shift++调整顺序)



2. 如果是硬盘为第一引导设备，读取硬盘中 MBR 主引导扇区中的 boot Loader 就是 grub 引导

GRUB（GRand Unified Bootloader 简称“GRUB”）是一个来自 GNU 项目的多操作系统启动程序。

MBR 的硬盘的 0 柱面、0 磁头、1 扇区称为主引导扇区（也叫主引导记录 MBR）。它由三个部分组成，主引导程序、硬盘分区表 DPT（Disk Partition table）和硬盘有效标志（55AA）。

为什么 MBR 分区表，只能分 4 个主分区?

注：磁盘默认一个扇区大小为：512 字节。MBR 由以下 3 部分组成：

第一部分是：主引导程序（boot loader）占 446 个字节。主引导程序，它负责从活动分区中装载，并运行系统引导程序。

第二部分是 Partition table 区（分区表），即 DPT，占 64 个字节，硬盘中分区有多少以及每一分区的大小都记在其中。每个分区表项长 16 个字节，16 个字节来记录一个分区的信息，16*4=64 字节。为分区项 1、分区项 2、分区项 3、分区项 4。64 个字节，4 个分区表项，每个表项占用 16 个字节，（这就是为什么 MBR 分区体系只能分成 4 个区【我们平时看到的分区一般可以从 26 个字母中选取任意多个当做分区标识（多于 4 个），这是因为那些分区是逻辑分区，这里的 4 个分区指的是主分区和扩展分区的数目，而逻辑分区是在扩展分区中划分出来的，也叫做二级、三级扩展分区。】

第三部分是 MBR 有效标识位，占 2 个字节，固定为 55AA。如果这个标志位 0x55AA，就认为这个是 MBR。55AA 作为可引导扇区的结束标志位。

所以：16*4+446+2=512

```
[root@xuegod63 ~]# dd if=/dev/sda of=mbr bs=1 count=512
[root@xuegod63 ~]# hexdump -C mbr
```

#hexdump 主要用来查看二进制文件的 16 进制编码

-C：输出规范的十六进制和 ASCII 码

```
000001f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 55 aa  |..............U.|
```

3. 依据 主引导程序（grub） 的设定加载 Kernel 到内存中运行 ， Kernel 会开始侦测硬件并加载驱动程序； grub 主引导加载 Kernel 后，就会自动退出，把硬件的控制权交给 Kernel

4. 在硬件驱动成功后，Kernel 会主动执行 init 进程，而 init 进程会读取/etc/inittab 配置取得运行级别（runlevel） 信息；

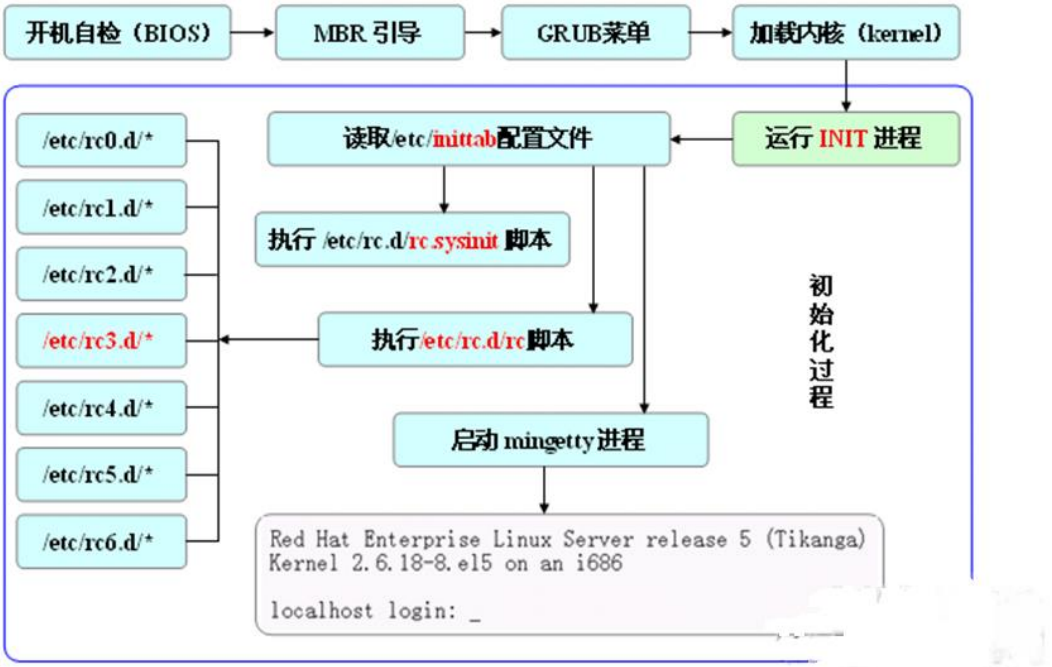
5. init 执行 /etc/rc.d/rc.sysinit 脚本文件来准备软件执行的作业环境（如网络、时区等、初始化环境）；

6. init 执行 run-level 下各个服务并启动（script 方式）；

7. init 执行开机后自动运行脚本 /etc/rc.d/rc.local 文件；

8. init 执行虚拟终端机控制程序 mingetty 来启动 login 程序，最后就等待用户登入啦；

如图：



17.1.2 centos6 启动相关的配置文件

[root@xuegod64 Desktop]# vim /boot/grub/grub.conf

default=0 设定默认启动菜单项，当系统中有多个内核时，0 表示默认加载第 1 个，1 表示第 2 个内核

timeout=5 菜单项等待选项时间为 5 秒

splashimage=(hd0,0)/grub/splash.xpm.gz 指明菜单背景图片路径

hiddenmenu 隐藏菜单

title CentOS (2.6.32-358.6.1.el6.x86_64) 定义菜单项

root (hd0,0)

(hd0,0) grub 查找 kernel 文件所在设备分区，grub 的根

kernel /vmlinuz-2.6.32-358.6.1.el6.x86_64 ro root=/dev/vg_have/lv_root rd_NO_LUKS LANG=en_US.UTF-8 SYSFONT=latarcyrheb-sun16 crashkernel=auto rhgb

quiet　　启动的内核

initrd /initramfs-2.6.32-358.6.1.el6.x86_64.img　　内核匹配的 ramfs 文件

修改系统启动级别：

[root@xuegod64 Desktop]# vim /etc/inittab

Default runlevel. The runlevels used are:

#　0 - halt (Do NOT set initdefault to this)

#　1 - Single user mode

#　2 - Multiuser, without NFS (The same as 3, if you do not have networking)

#　3 - Full multiuser mode

#　4 - unused

#　5 - X11

#　6 - reboot (Do NOT set initdefault to this)

#

id:3:initdefault:　　#这里决定系统默认启动的级别

/etc/rc.d/rc.sysinit　shell 脚本 作用：系统初始化：　像：主机名 和/etc/fstab 都在这里指定了，完成了包括 mount 分区 激活 swap 加载 modules 等重要的工作. 准备软件执行的作业环境（如网络、时区等、初始化环境）；

启动对应级别下的服务如：　init 3 级别

/etc/rc.d/rc3.d/（这里的程序/服务 S 开头的全部开机执行；K 开头的表示开机不执行，表明了关机时顺序）

rcn.d (n 为 1 到 6) 是对应于不同的 runlevel 下起不同的服务. 这些目录下都是一些符号连接，连接到/etc/rc.d/init.d 下的一些文件.以 S 开头的软连接表示 on 启动，以 K 开头的表示为 off 关闭. 第一个字母后面的数值是一个优先级.

[root@centos6 ~]# ls /etc/rc.d

init.d　rc　rc0.d　rc1.d　rc2.d　rc3.d　rc4.d　rc5.d　rc6.d　rc.local　rc.sysinit

init.d 里面放的是脚本

[root@centos6 ~]# ls /etc/rc.d/rc5.d

rc5.d 图形模式

[root@centos6 ~]# ls /etc/rc.d/rc0.d

rc0.d 就是关机模式，里面只启动 2 个软连接

[root@centos6 ~]# ls /etc/rc.d/rc0.d

rc6.d 重启模式，只启动 2 个软连接

[root@xuegod63 ~]# ll /etc/rc.d/rc5.d/ | grep network

lrwxrwxrwx. 1 root root 17 Dec 18　2012 S10network -> ../init.d/network　#表示 network 是第 10 个启动的服务。 所以 init 是顺序启动系统，需要一个一个服务启动成功，再执行下一步操作，启动系统比较慢。而 centos8 中的 systemd 第一个启动，然后 systemd 可以并行启动多个服务，启动比较快。

例：

```
[root@xuegod63 rc3.d]# vim /etc/rc.d/init.d/network
#! /bin/bash
#
# network          Bring up/down networking
#
# chkconfig: 2345 10 90
```

看有 chkconfig 的那一行，2345 表示在 runlevel 2 3 4 5 下被启动，10 是为此服务的启动顺序，90 为关机时关闭此服务的顺序。

```
[root@centos6 ~]# ll /etc/rc.d/rc2.d | grep network
[root@centos6 ~]# ll /etc/rc.d/rc3.d | grep network
[root@centos6 ~]# ll /etc/rc.d/rc4.d | grep network
[root@centos6 ~]# ll /etc/rc.d/rc5.d | grep network
[root@centos6 ~]# ll /etc/rc.d/rc0.d | grep network
```

centos6 下默认使用 chkconfig 查看服务在那个级别启动

```
[root@xuegod63 ~]# chkconfig --list
[root@xuegod63 ~]# chkconfig --list  | grep network
network         0:off   1:off   2:on 3:on 4:on 5:on 6:off
[root@xuegod63 ~]# ll /etc/rc.d/rc3.d/ | grep network
lrwxrwxrwx. 1 root root 17 Dec 18  2012 S10network -> ../init.d/network  #开机顺序
[root@xuegod63 ~]# chkconfig network off
[root@xuegod63 ~]# ll /etc/rc.d/rc3.d/ | grep network
lrwxrwxrwx  1 root root 17 May 23 21:17 K90network -> ../init.d/network
#只显示 K90 关机顺序了
[root@xuegod64 rc3.d]# chkconfig --list network
network         0:off   1:off   2:off   3:off   4:off   5:off   6:off
```

代表开机也不启动

所有服务都运行成功后，设置开机自动执行某个命令： /etc/rc.local

```
[root@xuegod64 rc3.d]# vim /etc/rc.local
[root@xuegod64 rc3.d]# ll !$
ll /etc/rc.local
lrwxrwxrwx. 1 root root 13 Dec 18  2012 /etc/rc.local -> rc.d/rc.local
[root@xuegod64 rc3.d]# ll /etc/rc.d/rc.local
-rwxr-xr-x. 1 root root 240 Feb  5 21:17 /etc/rc.d/rc.local
```

```
[root@localhost ~]# ps aux | grep ming
root      1234  0.0  0.0   2008   496 tty2     Ss+  17:46   0:00 /sbin/mingetty /dev/tty2
root      1236  0.0  0.0   2008   504 tty3     Ss+  17:46   0:00 /sbin/mingetty /dev/tty3
root      1238  0.0  0.0   2008   500 tty4     Ss+  17:46   0:00 /sbin/mingetty /dev/tty4
root      1240  0.0  0.0   2008   500 tty5     Ss+  17:46   0:00 /sbin/mingetty /dev/tty5
root      1242  0.0  0.0   2008   504 tty6     Ss+  17:46   0:00 /sbin/mingetty /dev/tty6
root      1364  0.0  0.0   5984   752 pts/0    R+   19:05   0:00 grep ming
```

可以通过 ctl +alt +F2 来切终端，就是靠 mingetty 来调用的 tty2,3,4,5,6 字符设备文件

17.2 centos8 系统启动过程及相关配置文件

17.2.1 centos8 系统启动过程

Centos8 启动流程：

1.在按下电源之后，BIOS 会加载硬件信息并对硬件进行自检。

2.在自检完成之后，会读取由 BIOS 设置的第一个可启动设备，此时可以读入 MBR 引导记录中的主引导程序（boot loader）。Boot Loader 就是在操作系统内核运行之前运行的一段程序,GRUB 是其中一种的引导程序的名称。

3.主引导程序可以指定哪个内核（Kernel）文件来进行启动，并将被指定的内核加载到内存当中运行。

4.在系统启动完成之后 Linux 才会调用外部程序开始准备软件执行的环境。并加载所有操作系统运行所需要的软件程序。

系统启动流程如果简单看的话就是以上这些步骤了，但是“简单的”学习并不是我们的目的，所以…

1.加载 BIOS 或 UEFI 初始化，开机自检

2. 选择启动设备

3. 引导装载程序，centos6 是 grub，centos7 和 8 是 grub2

4. 加载装载程序的配置文件：/etc/grub.d/ /etc/default/grub /boot/grub2/grub.cfg

5. 加载内核选项

6. 加载 initramfs 初始化伪文件系统

7. 内核初始化，centos8 使用 systemd 代替 init

8. 执行 initrd.target 所有单元，包括挂载 /etc/fstab

9. 从 initramfs 根文件系统切换到磁盘根目录

10. systemd 执行默认 target 配置，配置文件/etc/systemd/system/default.target

11. systemd 执行 sysinit.target 初始化系统及 basic.target 准备操作系统

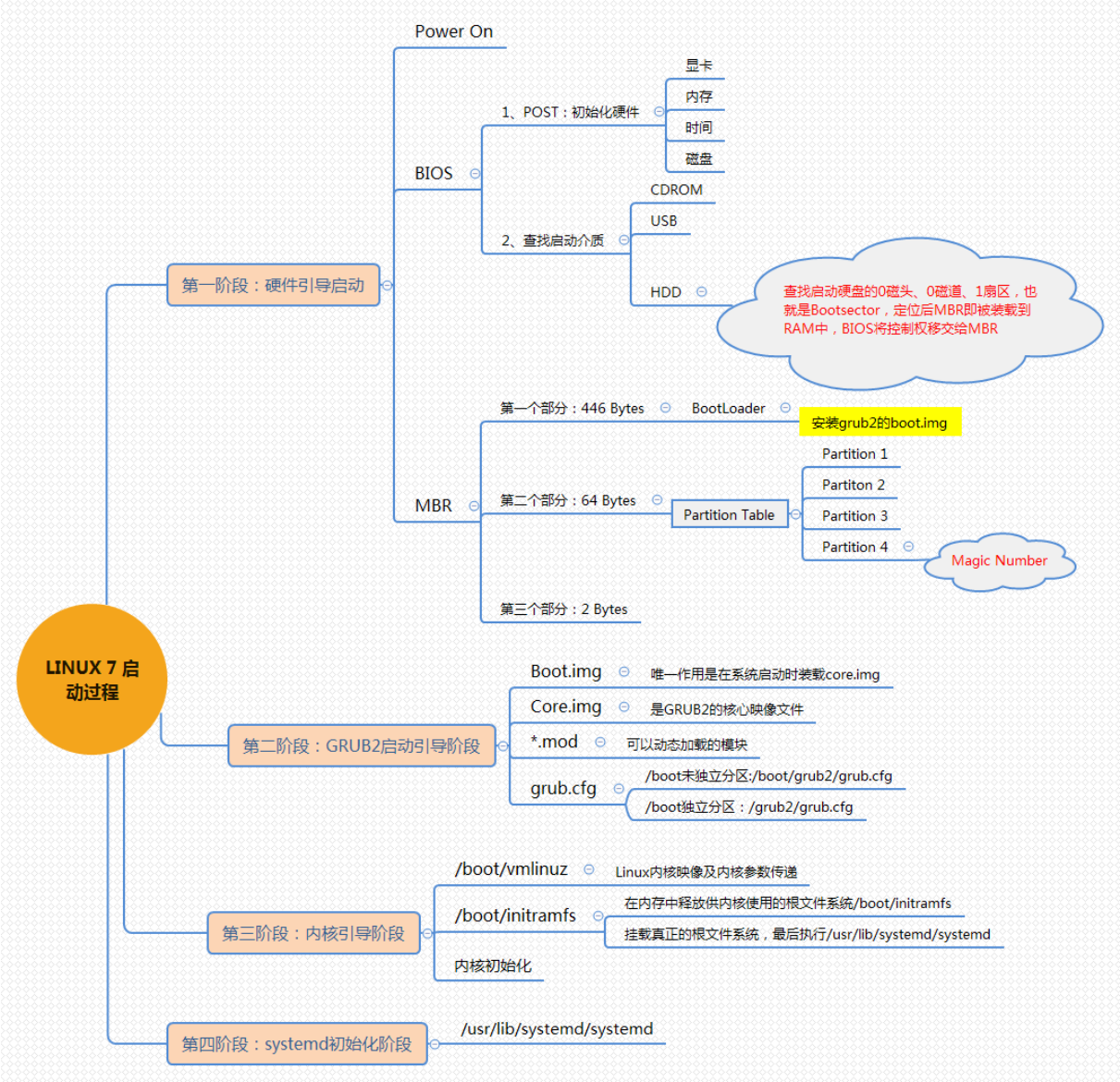
12. systemd 启动 mulit-user.target 下的本机与服务器服务

13. systemd 执行 multi-user.target 下的/etc/rc.d/rc.local

14. systemd 执行 multi-user.target 下的 getty.target 及登录服务

15. systemd 执行 graphical 需要的服务

centos7/8 启动过程：



ll -h /boot/grub2/i386-pc/core.img　　img 称为镜像文件

[root@xuegod83 ~]# find /boot -name *img*

/boot/grub2/i386-pc/core.img

/boot/grub2/i386-pc/boot.img

17.2.2　Systemd 运行原理-了解一下

Systemd 概述：systemd 即为 system daemon [ˈdi:mən] 守护进程，是 linux 下的一种 init 软件,开发目标是系统服务间的依赖关系，并依此实现系统初始化时服务的并行启动，同时达到降低 Shell 的系统开销的效果，最终代替现在常用的 System V 与 BSD 风格 init 程序。

与多数发行版使用的 System V 风格 init 相比，systemd 采用了以下新技术：（1）采用 Socket 激活式与总线激活式服务，以提高相互依赖的各服务的并行运行性能；（2）用 Cgroups（资源调优和分配后面讲 docker 会讲到）代替 PID 来追踪进程，以此即使是两次 fork（分叉）之后生成的守护进程也不会脱离 systemd 的控制。（fork 进程生成子进程，子进程又生成子进程）

unit 对象：unit 表示不同类型的 systemd 对象（就是服务器启动脚本），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（套接字(Socket)，就是对网络中不同主机上的应用进程之间进行双向通信的端点的抽象名，一个 ip 和一个端口就是一个套接字)、保存的系统快照以及其它与 init 相关的信息

Systemd 配置文件：

•/usr/lib/systemd/system/ #这个目录存储每个服务的启动脚本，类似于之前 6 系统的 /etc/init.d/

ls /usr/lib/systemd/system/*Network*

•/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/rc.d/rcN.d/Sxx 类的功能，比上面目录优先运行

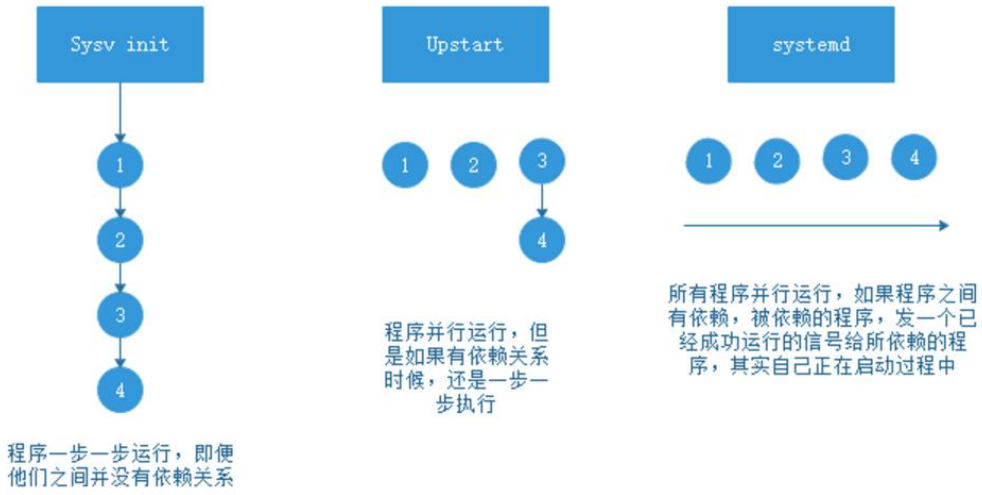
注意： 对于新创建的 unit 文件，或者修改了的 unit 文件，要通知 systemd 重载此配置文件,而后可以选择重启

[root@xuegod63 ~]# systemctl daemon-reload

重新加载，意为重读，有些服务的配置修改了，但是你不想重启服务，那么用这个就可以。

总结：centos5-6-7 3 个系统版本启动过程：

CentOS 5: SysV init ； CentOS 6: Upstart ； CentOS 7/8: Systemd



17.2.3 管理系统服务

命令： systemctl COMMAND name.service

CentOS7/8

启动：systemctl start name.service

停止：systemctl stop name.service

重启：systemctl restart name.service

状态：systemctl status name.service

重新加载配置文件：systemctl reload 或 restart name.service

CentOS6

启动：service name start

停止：service name stop
重启：service name restart
状态：service name status

CentOS6

设定某服务开机自启	chkconfig name on
设定某服务开机禁止启动	chkconfig name off
查看所有服务的开机自启状态	chkconfig --list
用来列出该服务在哪些运行级别下启用和禁用	chkconfig sshd --list

chkconfig 命令的对应关系

CentOS7/8

设定某服务开机自启	systemctl enable name.service
设定某服务开机禁止启动	systemctl disable name.service
查看所有服务的开机自启状态	systemctl list-unit-files --type service
查看服务是否开机自启	systemctl is-enabled name.service

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:允许开机启动
- disabled:禁止开机启动
- static: 表示该服务与其他服务相关联,不能单独设置该服务的启动状态

17.2.4 运行级别

centos6 下 Linux 运行级别 0-6 的各自含义
0： 关机模式
1：单用户模式 ，一用于破解 root 密码
2：无网络，支持的多用户模式
3：有网络支持的多用户模式（一般叫字符界面，工作中最长使用的模式）
4：保留，未使用
5：有网络支持，支持图形界面，支持的多用户模式（图形界面）
6：重新引导系统，及重启

可以在不同级别下，设置服务是否随系统启动运行。在 centOS7/8 上运行级别的含义已经和之前不同了，已由.target 来代替运行级别，我们可以称 target 为目标态，我们可以通过 target 定制更符合我们工作的运行环境。

[root@xuegod63 ~]# ls /usr/lib/systemd/system/*.target

#查看我们的机器上有多少个 target

```
/usr/lib/systemd/system/remote-fs-pre.target
/usr/lib/systemd/system/remote-fs.target
/usr/lib/systemd/system/rescue.target
/usr/lib/systemd/system/rpcbind.target
/usr/lib/systemd/system/runlevel0.target
/usr/lib/systemd/system/runlevel1.target
/usr/lib/systemd/system/runlevel2.target
/usr/lib/systemd/system/runlevel3.target
/usr/lib/systemd/system/runlevel4.target
/usr/lib/systemd/system/runlevel5.target
/usr/lib/systemd/system/runlevel6.target
/usr/lib/systemd/system/shutdown.target
/usr/lib/systemd/system/sigpwr.target
```

[root@xuegod63 ~]# ll /usr/lib/systemd/system/*.target | grep runlevel

```
lrwxrwxrwx 1 root root  15 Apr 27 22:17 /usr/lib/systemd/system/runlevel0.target -> poweroff.target
lrwxrwxrwx 1 root root  13 Apr 27 22:17 /usr/lib/systemd/system/runlevel1.target -> rescue.target
lrwxrwxrwx 1 root root  17 Apr 27 22:17 /usr/lib/systemd/system/runlevel2.target -> multi-user.target
lrwxrwxrwx 1 root root  17 Apr 27 22:17 /usr/lib/systemd/system/runlevel3.target -> multi-user.target
lrwxrwxrwx 1 root root  17 Apr 27 22:17 /usr/lib/systemd/system/runlevel4.target -> multi-user.target
lrwxrwxrwx 1 root root  16 Apr 27 22:17 /usr/lib/systemd/system/runlevel5.target -> graphical.target
lrwxrwxrwx 1 root root  13 Apr 27 22:17 /usr/lib/systemd/system/runlevel6.target -> reboot.target
```

注： 发现在 runlevel2-4 都是调用 multi-user.target 这个 unit。所以在 centos7/8 上 runlevel2-4 是一个意思

[root@xuegod63 ~]# systemctl list-unit-files --type target #查看所有 target 的状态

[root@xuegod63 ~]# systemctl list-dependencies runlevel3.target

#查看 3 级别 Unit 的所有依赖。Unit 之间存在依赖关系：A 依赖于 B，就意味着 Systemd 在启动 A 的时候，同时会去启动 B。也可以理解为 3 运行级别下都开启哪些服务，绿色的就是开启的。

在 centOS7/8 上所谓的目标态，其实就是由各种指定的服务和基础 target 组合而成的。

总结：centos6 和 7 运行级别的变化

6		7-8	
init		systemd	
Traditional runlevel	New target name		Symbolically linked to...
Runlevel 0	runlevel0.target	->	poweroff.target
Runlevel 1	runlevel1.target	->	rescue.target
Runlevel 2	runlevel2.target	->	multi-user.target
Runlevel 3	runlevel3.target	->	multi-user.target
Runlevel 4	runlevel4.target	->	multi-user.target
Runlevel 5	runlevel5.target	->	graphical.target
Runlevel 6	runlevel6.target	->	reboot.target

Init 0 → systemctl poweroff 关机

Init 1 → systemctl isolate rescue.target 单用户

Init 3 → systemctl isolate multi-user.target 字符界面
Init 5 → systemctl isolate graphical.target 图形化
Init 6 → systemctl reboot 重启

17.2.5 运行级别的切换

1、在 centOS6 上，我们切换级别使用 init，在 centOS7 上虽然也能使用，但是调用的不再是原来的程序了。centos7 使用 systemctl isolate name.target 来切换 target。 # isolate ['aɪsəleɪt] 分离，隔离

例 1：在 centos6/7 下切换到字符界面：
[root@xuegod63 ~]# init 3 #切换到字符界面
[root@xuegod63 ~]# init 5 #切换到图形界面
例 2：centos7 切换到字符界面
[root@xuegod63 ~]# systemctl isolate multi-user.target
或：
[root@xuegod63 ~]# systemctl isolate runlevel3.target

2、centos7 设置默认系统默认启动级别
systemctl set-default name.target 来修改我们的目标态。
我们看一下我们的默认目标态究竟为何物。
[root@xuegod63 ~]# ll /etc/systemd/system/default.target

```
[root@xuegod63 ~]#  ll /etc/systemd/system/default.target
lrwxrwxrwx. 1 root root 36 9月  19 2017 /etc/systemd/system/default.target -> /lib/s
ystemd/system/graphical.target
```

注：它其实就是创建了一个软链接到指定的 target 上去了
例 1：默认系统启动使用 3 级别字符界面
[root@xuegod63 ~]# systemctl set-default multi-user.target
Removed symlink /etc/systemd/system/default.target.
Created symlink from /etc/systemd/system/default.target to /usr/lib/systemd/system/multi-user.target.
[root@xuegod63 ~]# ll /etc/systemd/system/default.target #查看链接
lrwxrwxrwx 1 root root 41 5 月 23 19:08 /etc/systemd/system/default.target -> /usr/lib/systemd/system/multi-user.target
例 2：默认系统启动使用 5 级别图形界面
[root@xuegod63 ~]# systemctl set-default graphical.target
[root@xuegod63 ~]# systemctl get-default 查看默认启动级别
系统默认启动就是启动这个软连接，你改变了软连接的指向，就等于改变了启动级别

17.2.6 grub2 和 grub 区别（了解）

在 centOS6 上，我们的 grub 文件是/boot/grub/grub.conf
在 centOS7/8 使用 grub2，配置文件改成/boot/grub2/grub.cfg 了，但是功能还是大致一样的都是用于加载内核的，不过在 centOS7/8 上设置默认启动项发生了一些变化。
如果我们的系统中有两个内核？怎么改变默认启动的内核顺序？
例 1: centos8 修改内核启动顺序

[root@xuegod63 ~]# vim /etc/default/grub

GRUB_TIMEOUT=5　#开机时 grub 默认 5 秒后启动内核

GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"

改：GRUB_DEFAULT= saved

为：GRUB_DEFAULT= 1　#这里我们改成 1，0 代表第一个内核，1 代表第二个，以此类推。

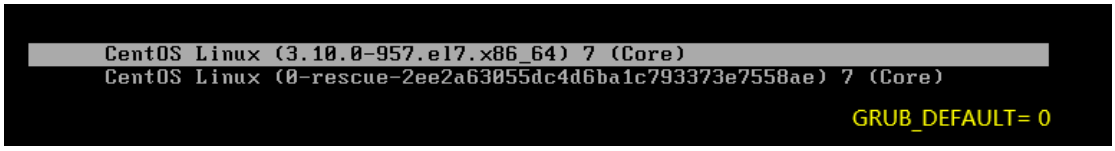
UB_DISABLE_SUBMENU=true

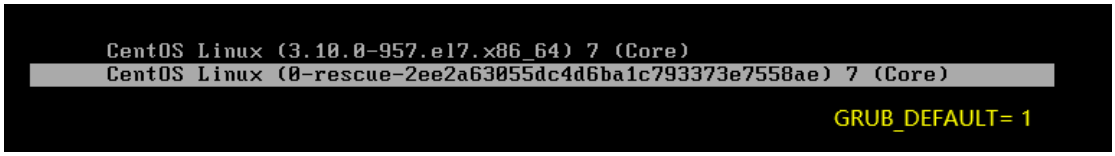
GRUB_TERMINAL_OUTPUT="console"

GRUB_CMDLINE_LINUX="crashkernel=auto rhgb quiet net.ifnames=0"

GRUB_DISABLE_RECOVERY="true"

[root@xuegod63 ~]# grub2-mkconfig -o /boot/grub2/grub.cfg　#修改完成后，并没有立即生效，使用此命令来生成 grub.cfg 文件，我们在下次启动的时候就会默认选择新的默认内核。





[root@xuegod63 ~]# uname -r　　#查当前系统内核

3.10.0-693.2.2.el7.x86_64

[root@xuegod63 ~]# reboot

[root@xuegod63 ~]# uname　-r　　#重启成功后，　发现加载的内核变了

3.10.0-693.el7.x86_64

例 2: centos6 修改内核启动顺序-了解

[root@xuegod63 ~]# vim /boot/grub/grub.conf

改：10 default=0

为：10 default=1

[root@xuegod63 ~]# reboot

17.3　实战-加密 grub 防止黑客通过单用户系统破解 root 密码

实战场景：如何防止别人恶意通过 reboot 系统破解 root 密码，进入系统窃取数据?

给 grub 加密，不让别人通过 grub 进入单用户。

17.3.1　基于 centos6 进行 grub 加密（了解）

[root@xuegod63 ~]# grub-md5-crypt

Password: 123456

Retype password: 123456

1oaqo5$3d/cmTosm68jTw6o1wCu31

[root@localhost init]# vim /boot/grub/grub.conf

#boot=/dev/sda

default=0

timeout=5

splashimage=(hd0,0)/grub/splash.xpm.gz

hiddenmenu 此列下进行添加

password --md5 1oaqo5$3d/cmTosm68jTw6o1wCu31

title Red Hat Enterprise Linux (2.6.32-220.el6.x86_64)

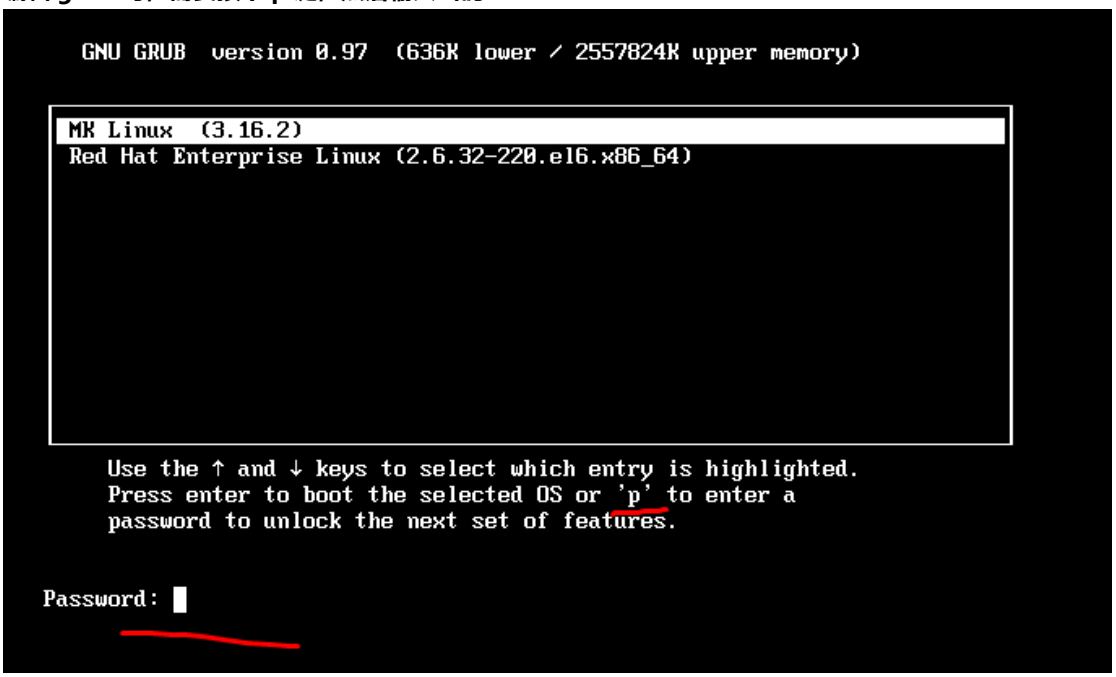
root (hd0,0)

如图：

```
#boot=/dev/sda
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
password --md5 $1$oaqo5$3d/cmTosm68jTw6o1wCu31
title Red Hat Enterprise Linux (2.6.32-220.el6.x86_64)
```

重启测试：

编辑 grub 时，需要按下 p 键，然后输入密码：123456



17.3.2 基于 centos7/8 进行 grub 加密

生成密码

[root@xuegod63 ~]# grub2-mkpasswd-pbkdf2

输入口令： 123456

Reenter password: 123456

PBKDF2 hash of your password is

grub.pbkdf2.sha512.10000.8F355BAB512AFB7B8C990A1FEB887B8F2F3F1C54467E9B9F0535F2268E1FFC5F4E8D33F7633D7FBEC25B2039C6D8B3226A90528D4883AB9B99E391A4965D069F.DDE992693BE2C09FFEEC1149120B6B84DBAB933DE6CF7BFF718E1DDC858AB73EE32

CFF45EB7F06AC45AA6792E91C4CD09E2B445FC288C47E79F537DBBABAD756

[root@xuegod63 ~]# vim /etc/grub.d/00_header　　#在最后后面添加如下内容，注 mk 这个用户名可以换成自己的用户名

cat <<EOF

set superusers='yum'

password_pbkdf2　yum　　和下面密码在一行

grub.pbkdf2.sha512.10000.8F355BAB512AFB7B8C990A1FEB887B8F2F3F1C54467E9B9F053
5F2268E1FFC5F4E8D33F7633D7FBEC25B2039C6D8B3226A90528D4883AB9B99E391A4965
D069F.DDE992693BE2C09FFEEC1149120B6B84DBAB933DE6CF7BFF718E1DDC858AB73EE32
CFF45EB7F06AC45AA6792E91C4CD09E2B445FC288C47E79F537DBBABAD756

EOF

如下图：

```
cat <<EOF
set superusers='yum'
password_pbkdf2 yum grub.pbkdf2.sha512.10000.CA12B
6F51700939941212 48DB2DA68464D58D04DBD4798BE2346.4E
01CA9B35FEED75D9A24A321C20F313E7F7047D3A2C6020AB15
EOF
"/etc/grub.d/00_header" 372L, 9296C
```

```
cat <<EOF
set superusers='yum'
password_pbkdf2 yum     不能换行
grub.pbkdf2.sha512.10000.CA12B7597F9A9ADFA414F556C713134B3
2DA68464D58D04DBD4798BE2346.4ECDA6F0922A42B55FDEB7D1D71469
321C20F313E7F7047D3A2C6020AB15
EOF
```

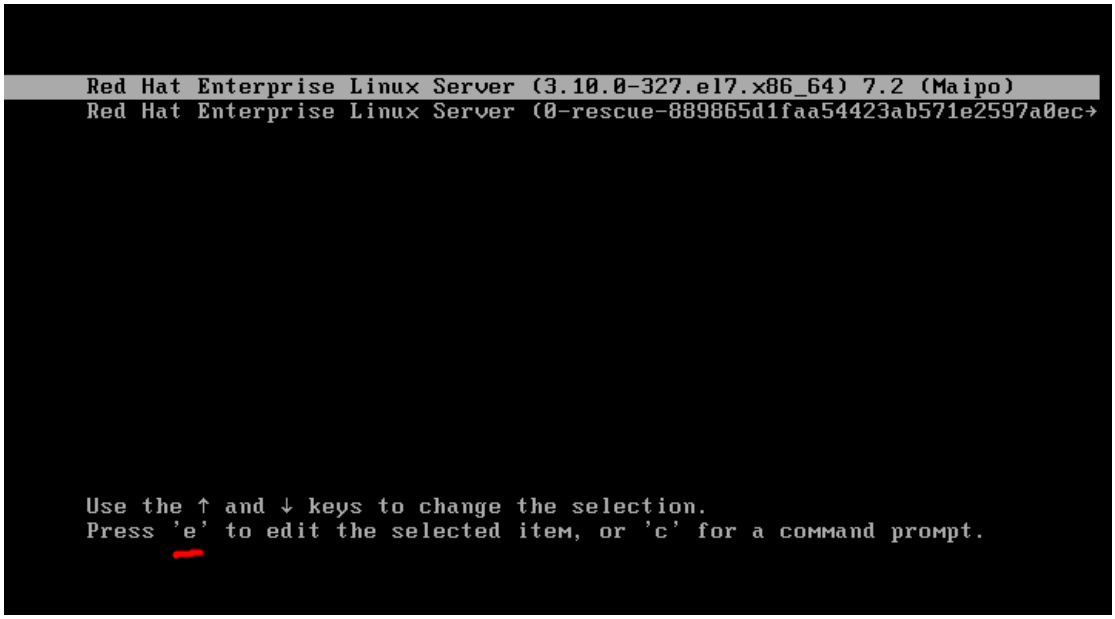
```
cat <<EOF
set superusers='yum'
password_pbkdf2 yum grub.pbkdf2.sha512.10000.CA12B7
6F51700939941212 48DB2DA68464D58D04DBD4798BE2346.4EC
01CA9B35FEED75D9A24A321C20F313E7F7047D3A2C6020AB15
EOF    这里不能有空格
```

```
[root@xuegod64 ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
Generating grub configuration file ...
/etc/grub.d/00_header: line 372: warning: here-document at line 369 delimited by end-of-file (wanted `eof')
done
```

有空格会报上面的错误

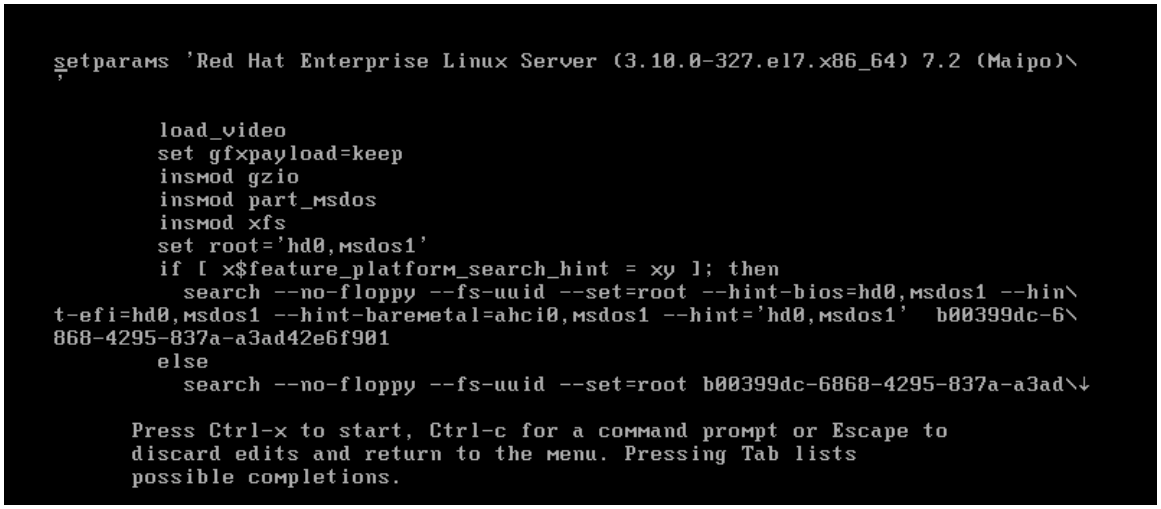
[root@xuegod63 ~]# grub2-mkconfig -o /boot/grub2/grub.cfg　#更新 grub 信息

重启验证：



输入用户名和密码

看到可以进入 GRUB 菜单，就证明你加密成功了



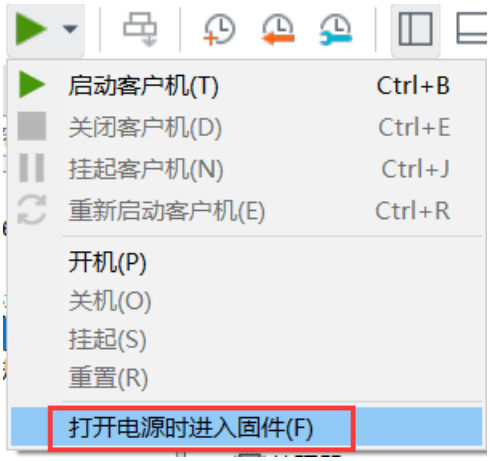
按 ctrl-x 开始启动

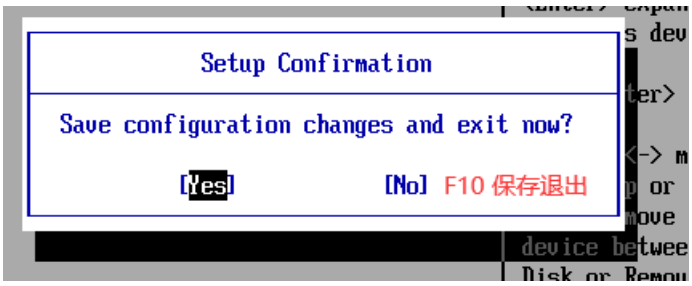
17.4 实战-通过 liveCD 进入救模式-重装 grub 修复损坏的系统

实战：使用系统光盘进入救援模式拯救坏掉的系统

实战场景：当系统坏了，进不去了，还需要把里面的数据复制出来，怎么办?

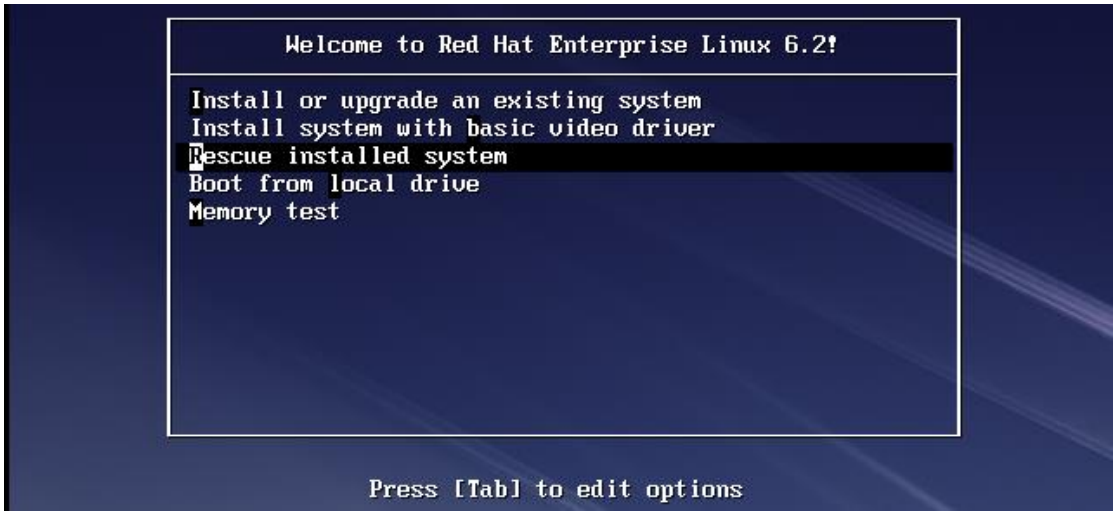
可以进入救援模式拷贝数据

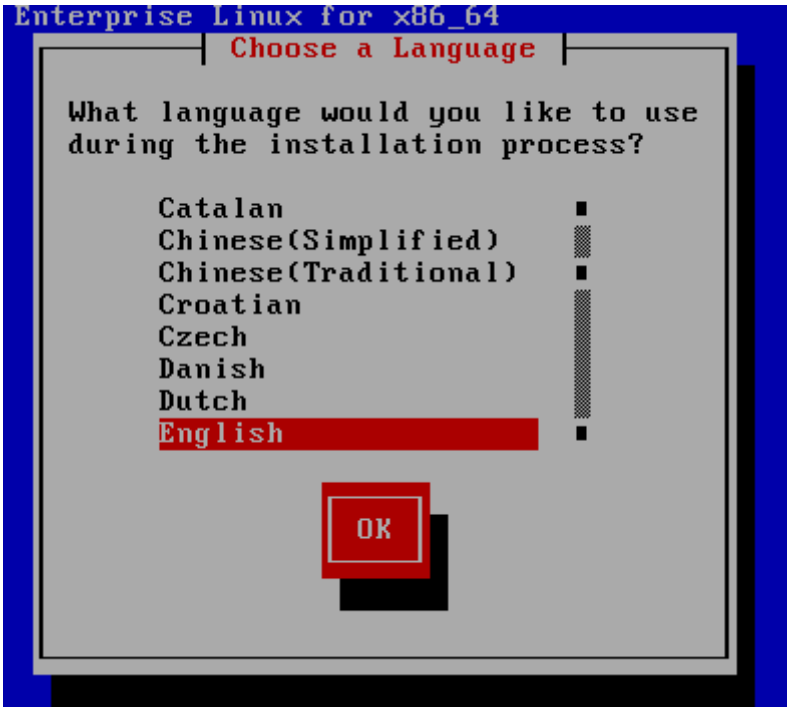


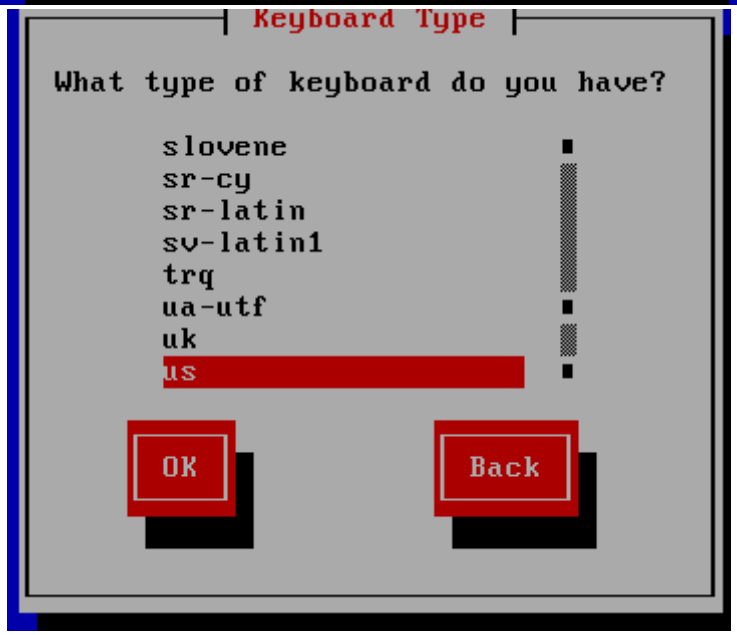


17.4.1 基于 6 版本系统进入救援模式

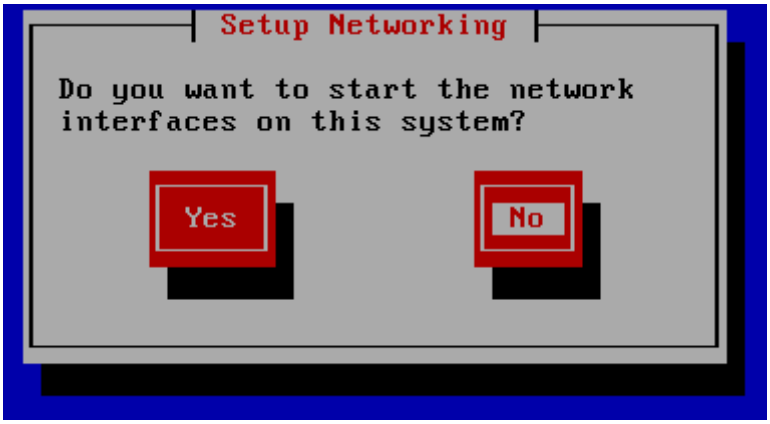
修改 BIOS 启动顺序，直接以光盘引导系统

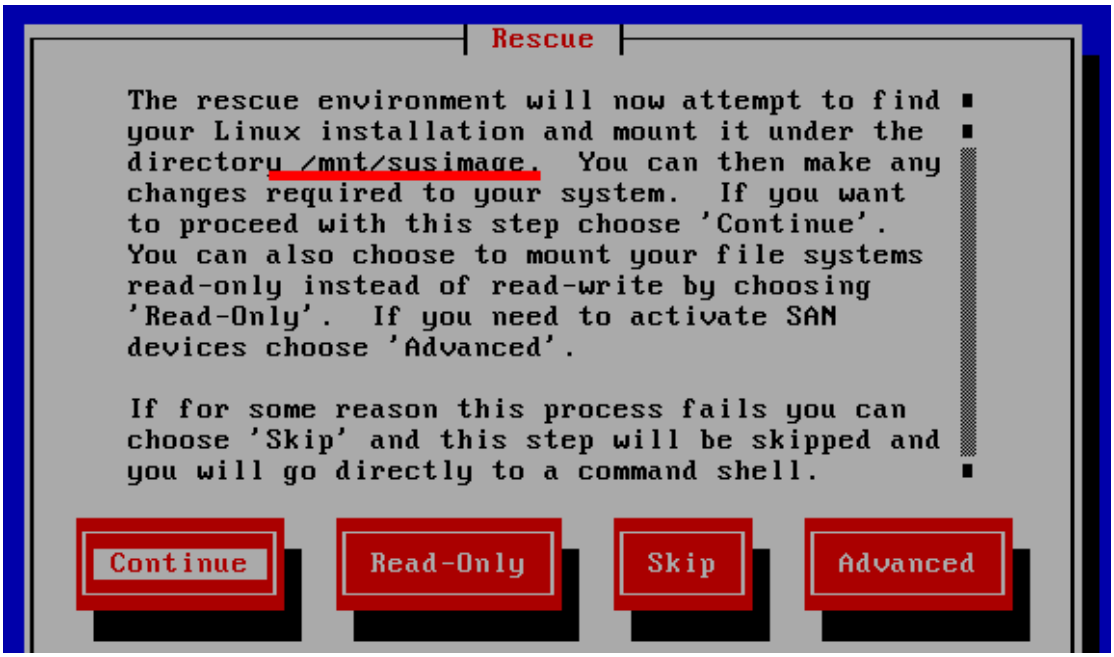


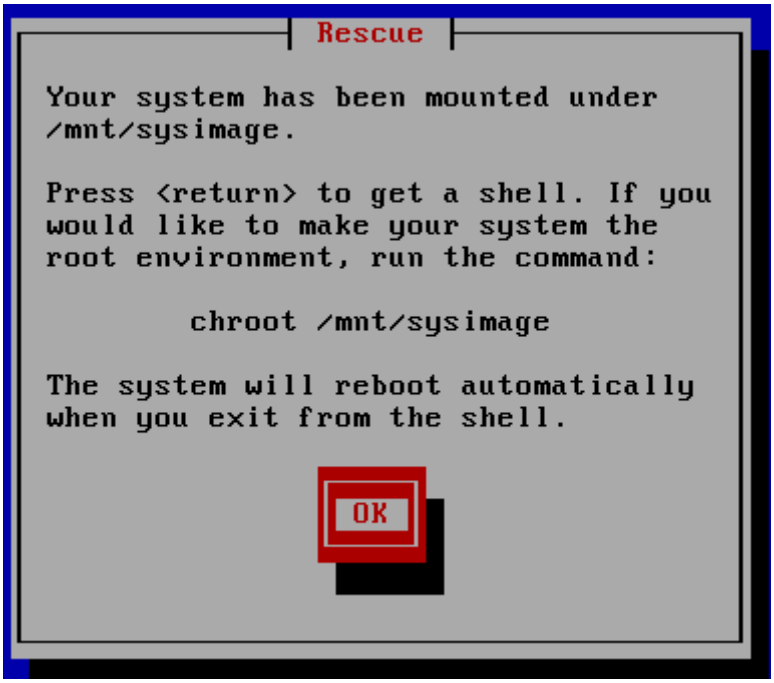


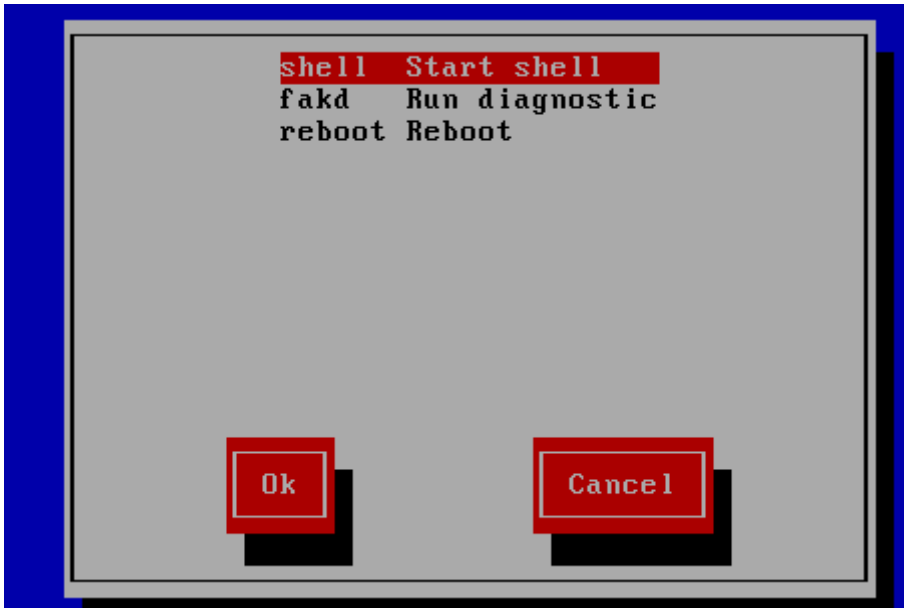


学神 IT 教育官方 QQ 群：1072932914 或唐老师 QQ：3340273106 领取更多资料





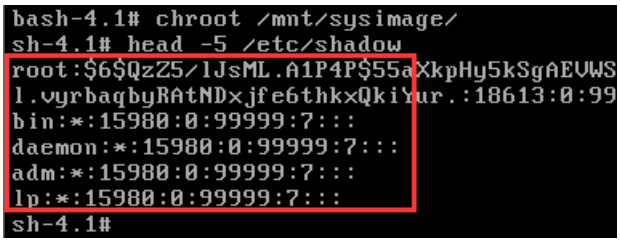





```
bash-4.1# pwd
/
bash-4.1# ls
bin  etc       init  lib_old  modules  proc  sbin     sys  usr      var
dev  firmware  lib   mnt      oldtmp   root  selinux  tmp  usr_old
bash-4.1# cat /etc/shadow
root::14438:0:99999:7:::
install::14438:0:99999:7:::
bash-4.1# ls /mnt/sysimage/
bin   dev  home  lost+found  mnt  proc  sbin     srv  tmp  var
boot  etc  lib   media       opt  root  selinux  sys  usr
bash-4.1#
```

ramfs ： 内存文件系统

chroot /mnt/sysimage # 切换文件系统根



此时我们就可以把数据 copy 出来了。

Exit

Reboot

17.4.2 实战-当 MBR 引导记录损坏后-重装 grub 进行修复

使用场景： 修复 MBR，主要出现在安装双系统时，后安装的系统把原来系统的 MBR 删除了，需要修复。

第一步：在 centOS8 下破坏硬盘的前 446 字节：

MBR 的硬盘的 0 柱面、0 磁头、1 扇区称为主引导扇区（也叫主引导记录 MBR）。它由三个部分组成，主引导程序、硬盘分区表 DPT（Disk Partition table）和硬盘有效标志（55AA）。

注：磁盘默认一个扇区大小为：512 字节。MBR 由以下 3 部分组成：

第一部分是：主引导程序（boot loader）占 446 个字节。主引导程序，它负责从活动分区中装载，并运行系统引导程序。

[root@CT731 ~]#dd if=/dev/zero of=/dev/sda bs=1 count=446 （第十一章）

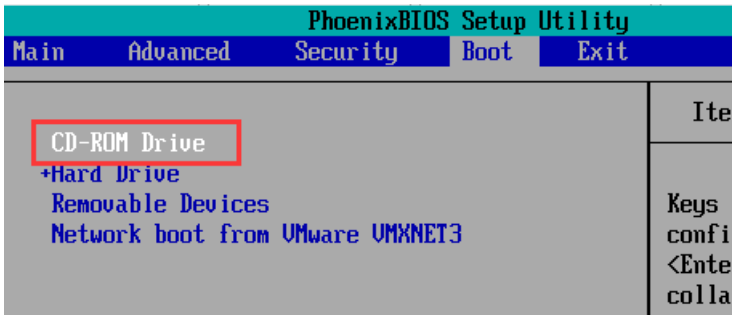
if：输入文件名；of：输出文件名；bs=1 一次一字节；count=446 复制 446 次

446+0 records in

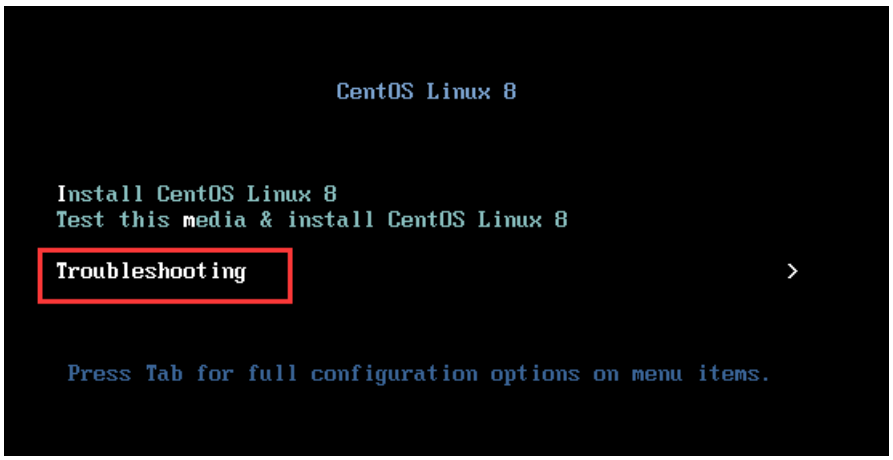
446+0 records out

446 bytes (446 B) copied,0.000758682 s,588 kB/s

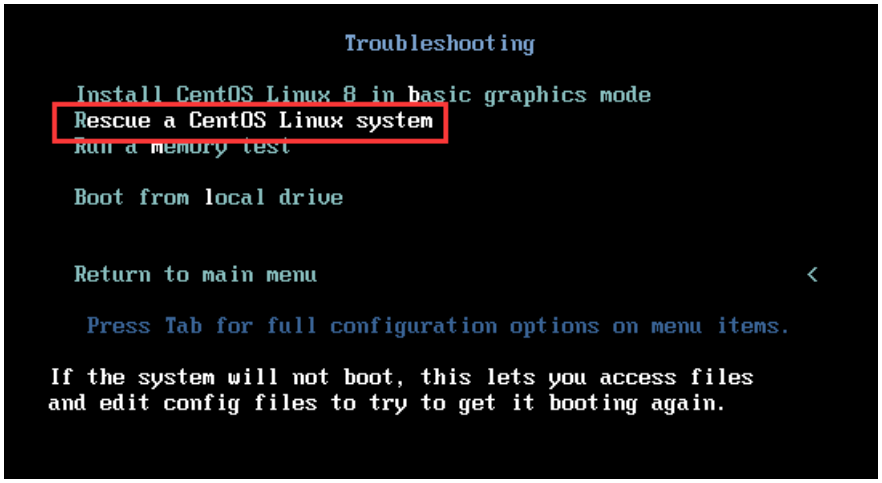
第二步：将 centos8 系统光盘挂载到虚拟机光驱上，，重启计算机，修改 BIOS 引导顺序，让光盘启动。



进入启动的界面 如果直接进入这个页面，说明是 bios 启动顺序是光盘优先，或者本来是硬盘优先，但是因为硬盘 mbr 前 446 个字节 boot loader 主引导程序找不到了，继续顺序启动 bios 设置的下一个设备



上面有三项，我们选择第三项进入 troubleshooting 故障排除界面 ，进入第三项后，点击第二项，进入救援模式的 centos 的系统



然后我们进入如下模式，选择 1，继续进行，接下来，我们就会进入到一个 shell 模式中，进行系统修复：

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

Please make a selection from the above: 1
================================================================================
================================================================================
Rescue Shell

Your system has been mounted under /mnt/sysimage.

If you would like to make the root of your system the root of the active system,
run the command:

        chroot /mnt/sysimage

When finished, please exit from the shell and your system will reboot.
Please press ENTER to get a shell:
sh-4.4# chroot /mnt/sysimage
bash-4.4# grub2-install /dev/sda
Installing for i386-pc platform.
Installation finished. No error reported.
bash-4.4# exit
exit
sh-4.4# reboot
```

先退一下，再重启，修复完成

使用场景：修复 MBR 主要出现在安装双系统时，后安装的系统把原系统的 MBR 删除了，需要修复

```
*** Warning -- SELinux targeted policy relabel is required.
*** Relabeling could take a very long time, depending on file
*** system size and speed of hard drives.
```

SELinux 从禁用状态转变为启用时，重启系统会花费一段时间重新标记文件的上下文。

17.4.3 实战-在 centOS8 下误删除 grub 文件进行修复

第一步：删除 grub2

[root@xuegod63 ~]# rm -rf /boot/grub2

第二步，重启计算机

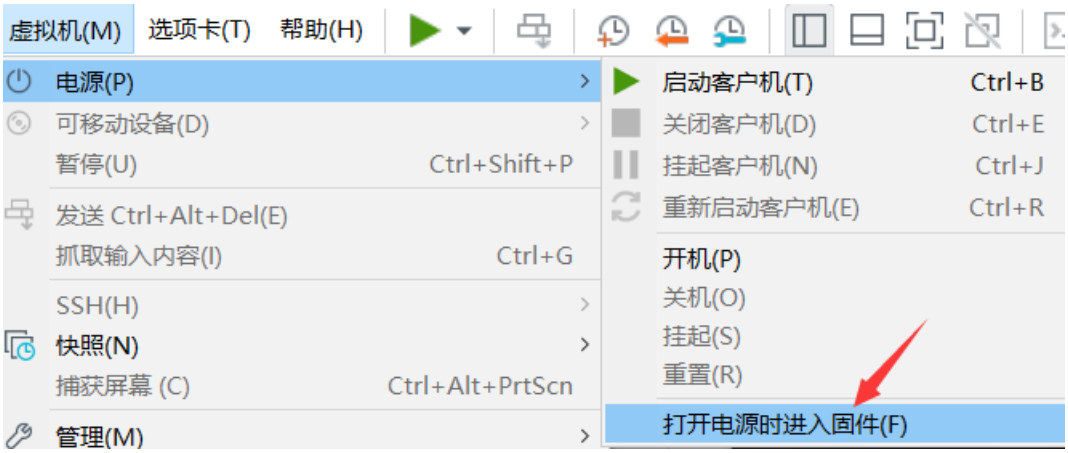
[root@xuegod63 ~]# reboot

进入如下界面：

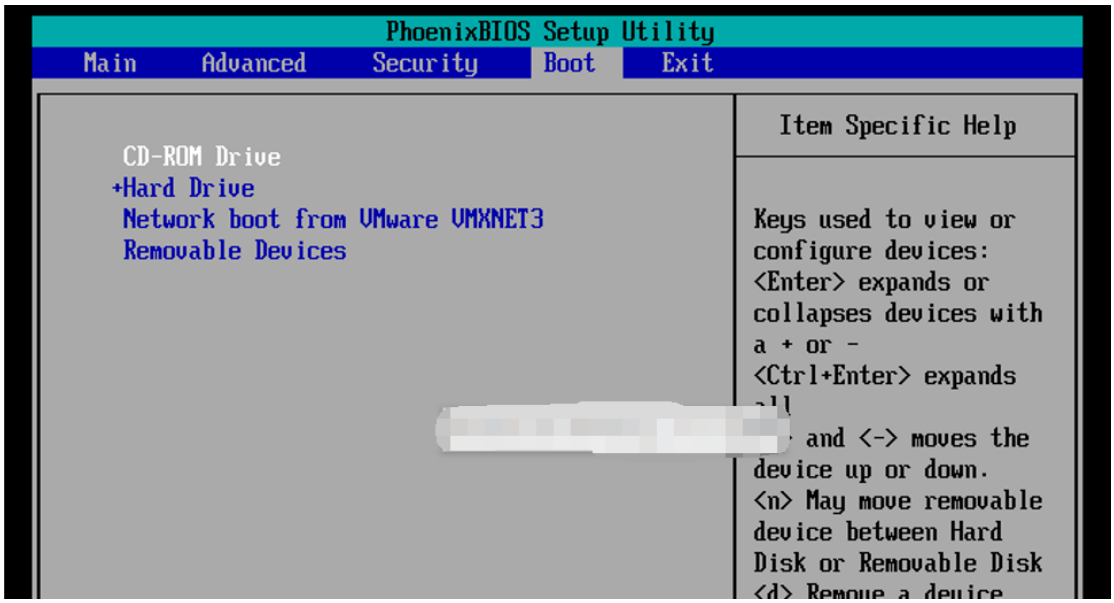
```
.
error: file '/grub2/i386-pc/normal.mod' not found.
Entering rescue mode...
grub rescue> _
```

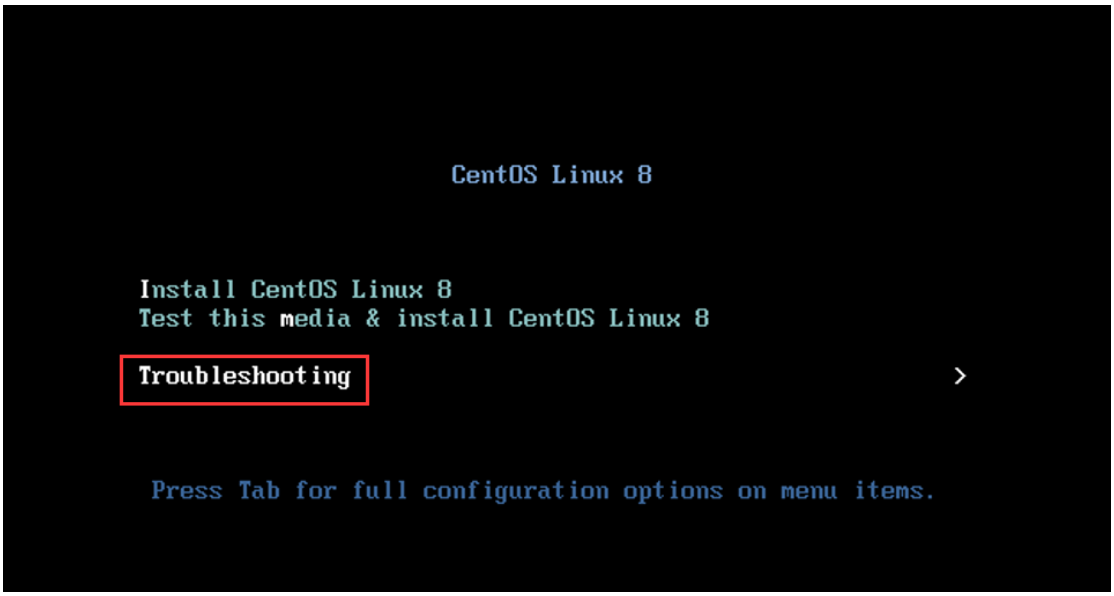
现在开始解决 grub

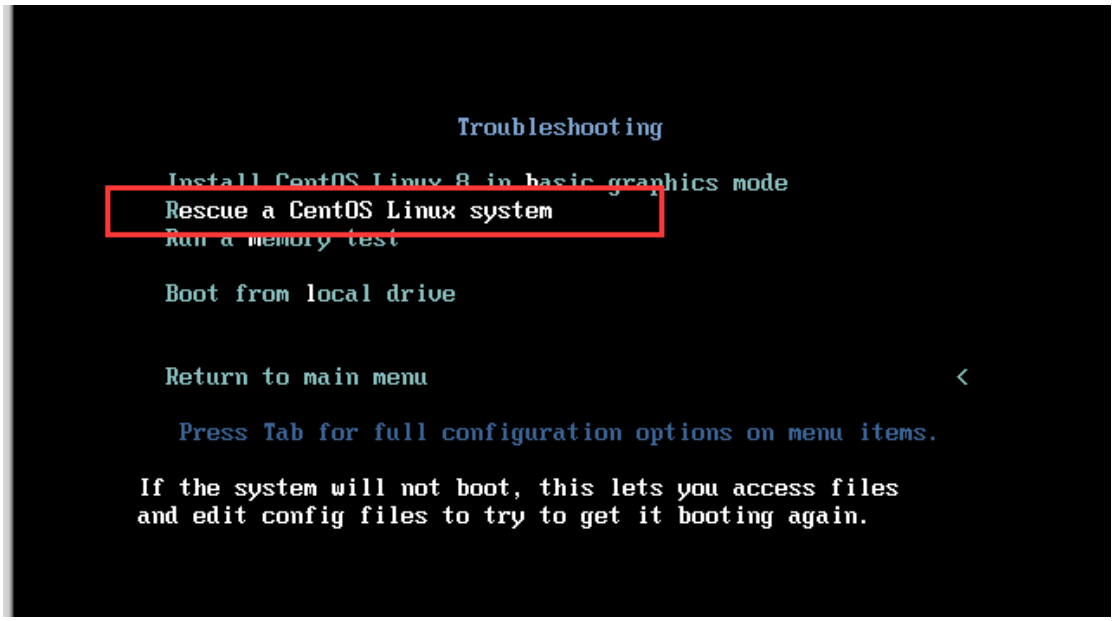
关闭客户机



重启系统，按 Esc，进入光盘救援模式，选择第三项，进入光盘救援（前提是挂载光盘）

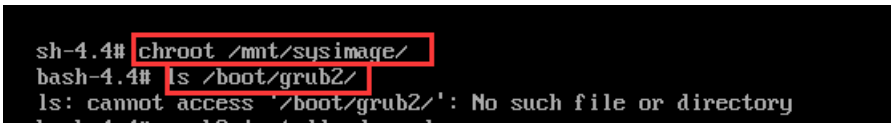






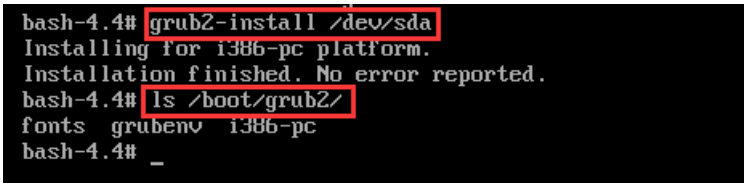
使用 live cd 进入救援模式后：1 回车

第一步：切根

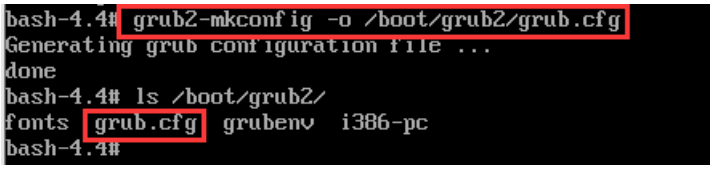


查看/boot/grub2 目录还不存在

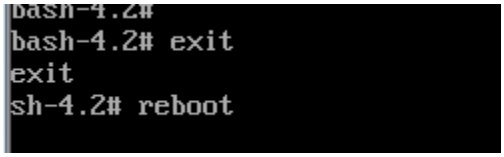
执行 grub2-install /dev/sda 后查看，/boot/grub2 已经恢复 ，但是缺少 grub.cfg 文件



接下来，我们需要生成配置文件：



exit 然后，重启电脑：



修改 BIOS 引导，让硬盘做第一引导

总结: