Linux系统启动流程和内核管理详细介绍

概述

- 博客主要包含CentOS5 和 CentOS6的启动流程介绍、相关的系统服务管理、Grub启 动引导管理、自定义满足基本使用需求的Linux系统、Centos系统启动故障排错、源码 编译安装linux内核、BusyBox 介绍、Centos 7启动流程介绍、Centos 7 Unit介绍 Centos 7 服务管理和查看、Centos7启动排错、破解centos口令、修复grub引导。

一.Linux组成

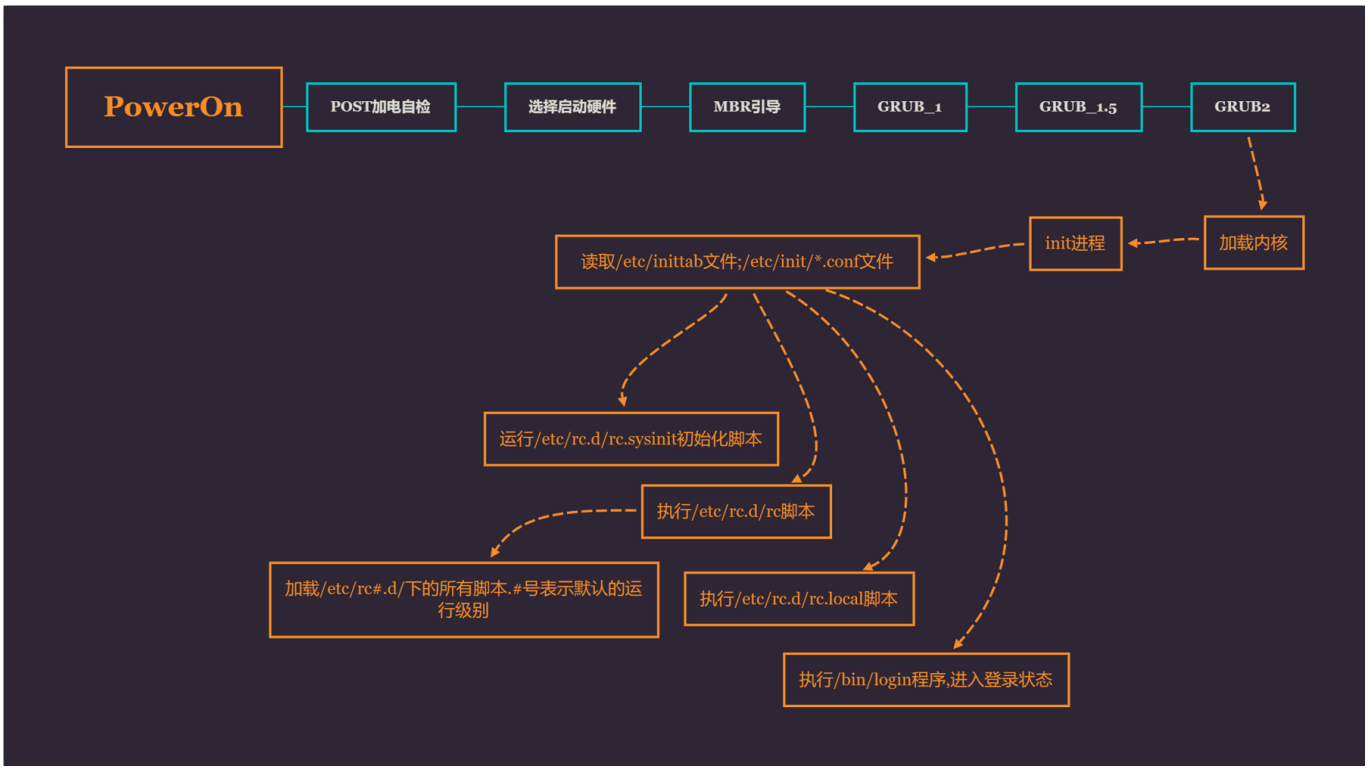
- Linux: kernel+rootfs
  - kernel: 进程管理、内存管理、网络管理、驱动程序、文件系统、安全功能

> 内核具体功能解释

- rootfs:程序和glibc
- 库：函数集合, function, 调用接口（头文件负责描述）
- 程序：二进制执行文件
- 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

二.CentOS6大致启动流程

- 大致启动流程如下




```
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
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
```

三.系统启动流程

BIOS

- 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
- 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
- RAM：CMOS互补金属氧化物半导体，保存各项参数的设定
- 启动时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
- 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

```
LILO: LInux LOader
GRUB: GRand Unified Bootloader
    GRUB 0.X: GRUB Legacy,  GRUB2
```

- MBR:硬盘的前512字节(第一个扇区)
  - 前446字节为:bootloader
  - 中间64字节:分区表
  - 最后2字节:55AA标记位
- GRUB
  - primary boot loader : 1st stage，1.5 stage
  - secondary boot loader ：2nd stage，分区文件
- kernel 初始化
  - 探测可识别到的所有硬件设备
  - 加载硬件驱动程序（借助于ramdisk加载驱动）
  - 以只读方式挂载根文件系统
  - 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
CentOS 5 /boot/initrd-VERSION-release.img
CentOS 6,7 /boot/initramfs-VERSION-release.img
```

- 模块文件：/lib/modules/VERSION-release
- ramdisk:核内中的特性之一：使用缓冲和缓存来加速对磁盘上的文件访问，并加载相应 的硬件驱动:ramdisk --> ramfs 提高速度

```
CentOS 5 -- > initrd.img
工具程序：mkinitrd
CentOS 6，7 --> initramfs.img
工具程序：mkinitrd，dracut
```

ramdisk管理

- ramdisk文件的制作： (1) mkinitrd命令 为当前正在使用的内核重新制作ramdisk文件 mkinitrd /boot/initramfs-$(uname -r).img $(uname -r) (2) dracut命令
  为当前正在使用的内核重新制作ramdisk文件 dracut /boot/initramfs-$(uname -r).img $(uname -r)

系统启动流程

- 系统启动及初始化： 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
sbin/init  CentOS6之前
```

- 运行级别：为系统运行或维护等目的而设定0-6：7个级别

```
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
默认级别：3，5
切换级别：init #
查看级别：
    runlevel
    who -r
```

- init初始化基本步骤

```
init读取其初始化文件：/etc/inittab
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
- 配置文件：/etc/inittab
- 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
```

- 示例：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
```

- CentOS6 /etc/inittab和相关文件
- /etc/inittab

```
    设置系统默认的运行级别
    id:3:initdefault:
```

- 破解CentOS 6 的root口令

```
/etc/init/control-alt-delete.conf
/etc/init/tty.conf
/etc/init/start-ttys.conf
/etc/init/rc.conf
/etc/init/prefdm.conf
```

- /etc/rc.d/rc.sysinit: 系统初始化脚本,主要做以下事务

```
(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
```

- 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
  - 作用类似下面的代码

```
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
```

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

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

- 删除： `chkconfig --del name`
- 修改指定的链接类型 `chkconfig [--level levels] name <on|off|reset>`
  - --level LLLL: 指定要设置的级别；省略时表示2345
- xinetd管理的服务
  - service 命令：手动管理服务 `service aervicename start|stop|restart service --status-all`

- 瞬态（Transient）服务被xinetd进程所管理
- 进入的请求首先被xinetd代理
- 配置文件：/etc/xinetd.conf、/etc/xinetd.d/ 与libwrap.so文件链接
- 用chkconfig控制的服务：`chkconfig tftp on`
- 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,可以根据情况,进行自定义修改

```
1:2345:respawn:/usr/sbin/mingetty tty1
2:2345:respawn:/usr/sbin/mingetty tty2
...
6:2345:respawn:/usr/sbin/mingetty tty6
```

- mingetty会自动调用login程序 `x:5:respawn:/etc/X11/prefdm -nodaemon`
- 启动过程总结：`/sbin/init --> (/etc/inittab) --> 设置默认运行级别 --> 运行系统初始脚本、完成系统初始化 --> (关闭对应下需要关闭的服务)启动 需要启动服务 --> 设置登录终端`
- 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

四.grub legacy

CentOS 6启动流程

`POST --> Boot Sequence(BIOS) --> Boot Loader --> Kernel(ramdisk) --> rootfs --> switchroot --> /sbin/init -->(/etc/inittab, /etc/init/*.conf) --> 设定默认运行级别 --> 系统初始化脚本rc.sysinit --> 关闭或启动对应级别的服务 --> 启动终端`

- 参看：比较详细的启动过程图解

grub: GRand Unified Bootloader

```
grub 0.97: grub legacy
grub 2.x: grub2
grub legacy:
    stage1: mbr
    stage1_5: 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
    stage2: 磁盘分区(/boot/grub/)
```

grub安装

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

- 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
- 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：设定本次启动时用到的内核文件；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` 查看当前内核所使用参数
- 内核参数文档:/usr/share/doc/kernel-doc-2.6.32/Documentation/kernel-parameters.txt
- 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
(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

- 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] 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
  - 启动系统时，设置其运行级别1
- 进入单用户模式：

```
(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
(3) 在kernel所在行，键入“b”命令
```

五.自制linux系统

1.分区并创建文件系统

```
fdisk /dev/sdb
分两个必要的分区
/dev/sdb1对应/boot    /dev/sdb2对应根 /
```

```
mkfs.ext4 /dev/sdb1
mkfs.ext4 /dev/sdb2
```

2.挂载boot

```
mkdir /mnt/boot  子目录必须为boot
mount /dev/sdb1 /mnt/boot
```

3.安装grub

```
grub-install --root-directory=/mnt /dev/sdb
```

4.恢复内核和initramfs文件

```
cp /boot/vmlinuz-2.6.32-642.el6.x86_64  /mnt/boot/
cp /boot/initramfs-2.6.32-642.el6.x86_64.img  /mnt/boot
```

5.建立grub.conf

```
vim /mnt/boot/grub/grub.conf
title wanglinux
root (hd0,0)
kernel /vmlinuz-2.6.32-642.el6.x86_64 root=/dev/sda2  selinux=0
init=/bin/bash
initrd /initramfs-2.6.32-642.el6.x86_64.img
```

6.chroot /mnt/sysroot

7.创建一级目录

```
mkdir /mnt/sysroot
mount  /dev/sdb2   /mnt/sysroot
mkdir -pv
/mnt/sysroot/{etc,lib,lib64,bin,sbin,tmp,var,usr,sys,proc,opt,home,root,boot,
dev,mnt,media}
```

8.复制bash和相关库文件

9.复制相关命令及相关库文件

- 如：ifconfig,insmod,ping,mount,ls,cat,df,lsblk,blkid等

/proc目录

- /proc目录包含内核自己内部状态信息及统计信息，可配置参数等通过proc伪 文件系统加以输出
    - 帮助：man proc
    - 参数：只读：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 (1) sysctl命令用于查看或设定此目录中诸多参数 sysctl -w path.to.parameter=VALUE sysctl -w kernel.hostname=mail.magedu.com (2) 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 echo "websrv" > /proc/sys/kernel/hostname

sysctl命令查看当前生效的内核参数

- 默认配置文件：/etc/sysctl.conf (1) 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) 查看所有生效参数 `sysctl -a`
- 常用的几个参数

```
net.ipv4.ip_forward
net.ipv4.icmp_echo_ignore_all
vm.drop_caches
fs.file-max = 1020000
```

/sys目录

- sysfs：为用户使用的伪文件系统，输出内核识别出的各硬件设备的相关属 性信息，也有内核对硬件特性的设定信息；有些参数是可以修改的，用于调整硬件 工作特性
- udev通过此路径下输出的信息动态为各设备创建所需要设备文件，udev是 运行用户空间程序
- 专用工具：udevadmin, hotplug
- udev为设备创建设备文件时，会读取其事先定义好的规则文件，一般在 /etc/udev/rules.d及/usr/lib/udev/rules.d目录下

六.内核编译

- linux单内核体系设计、但充分借鉴了微内核设计体系的优点，为内核引入模块 化机制
- 内核组成部分：

```
kernel：内核核心，一般为bzImage，通常在/boot目录下
    名称为 vmlinuz-VERSION-RELEASE
kernel object：内核对象，一般放置于
    /lib/modules/VERSION-RELEASE/
[ ]: N
[M]: M
[*]: Y
辅助文件：ramdisk
         initrd
         initramfs
```

- 内核版本

```
查看运行中的内核版本：
uname命令：
    uname - print system information
    uname [OPTION]...
        -n: 显示节点名称
        -r: 显示VERSION-RELEASE
        -a:显示所有信息
```

内核模块命令

- lsmod命令：

```
显示由核心已经装载的内核模块
显示的内容来自于: /proc/modules文件
```

- modinfo命令：

```
显示内核模块的详细描述信息
    modinfo [ -k kernel ]  [ modulename|filename... ]
    -n：只显示模块文件路径
    -p：显示模块参数
    -a：作者
    -d：描述
示例：lsmod |grep xfs
      modinfo  xfs
```

内核模块管理

- modprobe命令：

```
装载或卸载内核模块
    modprobe [ -C config-file ]  [ modulename ]  [ module parame-ters... ]
    modprobe [ -r ] modulename…
```

- 配置文件：/etc/modprobe.conf, /etc/modprobe.d/*.conf
- depmod命令：内核模块依赖关系文件及系统信息映射文件的生成工具
- 装载或卸载内核模块：

```
insmod命令：指定模块文件，不自动解决依赖模块
insmod [ filename ]  [ module options... ]
insmod `modinfo –n exportfs`
```

```
lnsmod `modinfo -n xfs`
rmmod命令：卸载模块
rmmod [ modulename ]
rmmod xfs
rmmod exportfs
```

编译内核

1.前提：

```
(1) 准备好开发环境
(2) 获取目标主机上硬件设备的相关信息
(3) 获取目标主机系统功能的相关信息
    例如:需要启用相应的文件系统
(4) 获取内核源代码包
     www.kernel.org
```

2.开发环境准备

```
包组
    Development Tools
目标主机硬件设备相关信息
    CPU:
    cat /proc/cpuinfo
    x86info -a
    lscpu
PCI设备:
    lspci
        -v
        -vv
    lsusb
        -v
        -vv
块设备
    lsblk
了解全部硬件设备信息
    hal-device: CentOS 6
```

3.步骤

```
安装开发包组
下载源码文件
.config：准备文本配置文件
make menuconfig：配置内核选项
make [-j #]  或者用两步实现： make -j # bzImage ; make -j # modules
make modules_install：安装模块
```

```
make install ：安装内核相关文件
    安装bzImage为 /boot/vmlinuz-VERSION-RELEASE
    生成initramfs文件
    编辑grub的配置文件
```

- 编译安装内核示例

```
yum install gcc ncurses-devel flex  bison openssl-devel elfutils-libelf devel
tar xf linux-5.2.9.tar.xz -C /usr/src
cd /usr/src
ln -sv linux-5.2.9 linux
cd /usr/src/linux
cp /boot/config-$(uname -r)   ./.config
make help
make menuconfig
make -j 2  或者 make -j 2 bzImage ; make -j 2 modules
make modules_install
make install
reboot
```

编译内核两大步

- 1.配置内核选项

```
支持“更新”模式进行配置：make help
(a) make config：基于命令行以遍历的方式配置内核中可配置的每个选项
(b) make menuconfig：基于curses的文本窗口界面
(c) make gconfig：基于GTK (GNOME）环境窗口界面
(d) make xconfig：基于QT(KDE)环境的窗口界面
支持“全新配置”模式进行配置
(a) make defconfig：基于内核为目标平台提供的“默认”配置进行配置
(b) make allyesconfig：所有选项均回答为“yes“
(c) make allnoconfig：所有选项均回答为“no“
```

- 2.编译

```
全编译:make [-j #]
编译内核的一部分功能：
(a) 只编译某子目录中的相关代码
    cd /usr/src/linux
    make dir/
(b) 只编译一个特定的模块
    cd /usr/src/linux
    make dir/file.ko
    示例：只为e1000编译驱动：
        make drivers/net/ethernet/intel/e1000/e1000.ko
```

- 交叉编译内核:编译的目标平台与当前平台不相同 `make ARCH=arch_name`
- 要获取特定目标平台的使用帮助 `make ARCH=arch_name help`
- 示例： `make ARCH=arm help`
- 在已经执行过编译操作的内核源码树做重新编译需要事先清理操作：

```
make clean：清理大多数编译生成的文件，但会保留config文件等
make mrproper：清理所有编译生成的文件、config及某些备份文件
make distclean：mrproper、清理patches以及编辑器备份文件
```

卸载内核

```
删除/lib/modules/目录下不需要的内核库文件
删除/usr/src/linux/目录下不需要的内核源码
删除/boot目录下启动的内核和内核映像文件
更改grub的配置文件，删除不需要的内核启动列表
    centos7:vim /boot/grub2/grub.cfg
           :/menuentry
    centos8:
          rm -f /boot/loader/entries/7e3e9120767340a8bd946a83d7c3b84d-$(uname -
r)-80.el8.x86_64.conf
```

Busybox介绍

- Busybox 最初是由 Bruce Perens 在 1996 年为 Debian GNU/Linux 安装盘编 写的。其目标是在一张软盘(存储空间只有1MB多)上创建一个GNU/Linux 系统， 可以用作安装盘和急救盘
- Busybox 是一个开源项目，遵循GPL v2协议。Busybox将众多的UNIX命令集 合进一个很小的可执行程序中，可以用来替代GNU fileutils、shellutils等工具 集。Busybox中各种命令与相应的GNU工具相比，所能提供的选项比较少，但 是也足够一般的应用了。Busybox主要用于嵌入式系统
- Busybox 是一个集成了三百多个最常用Linux命令和工具的软件。BusyBox 包 含了一些简单的工具，例如ls、cat和echo等等，还包含了一些更大、更复杂的 工具，例grep、find、mount以及telnet。有些人将 BusyBox 称为 Linux 工具 里的瑞士军刀。简单的说BusyBox就好像是个大工具箱，它集成压缩了 Linux 的许多工具和命令，也包含了 Android 系统的自带的shell
- 定制小型的Linux操作系统：linux内核+busybox
- 官方网站：https://busybox.net/
- Busybox使用
    - busybox 的编译过程与Linux内核的编译类似
    - busybox的使用有三种方式：

```
busybox后直接跟命令，如 busybox ls
直接将busybox重命名，如 cp busybox tar
创建符号链接，如 ln -s busybox rm
```

- busybox的安装
- 使用创建软连接的方式使用busybox最为可取，但为busybox中每个命令都创建一个 软链接，相当费事，busybox提供自动方法：busybox编译成功后，执行make install 则会产生一个_install目录，其中包含了busybox及每个命令的软链接
- 编译Busybox

```
yum install   gcc gcc-c++ glibc glibc-devel pcre pcre-devel openssl openssl-devel
systemd-devel zlib-devel glibc-static ncurses-devel
wget https://busybox.net/downloads/busybox-1.30.1.tar.bz2
tar xvf busybox-1.31.0.tar.bz2
cd busybox-1.31.0/
make menuconfig 按下面选择，把busybox编译也静态二进制、不用共享库
   Settings -->Build Options -->[*] Build BusyBox as a static binary (no shared
libs)
make && make install   如果出错，执行make clean后，重新执行上面命令
mkdir /mnt/sysroot/
cp -a _install/* /mnt/sysroot/
```

练习

1、破解root口令，并为grub设置保护功能 2、破坏本机grub stage1，而后在救援模式下修复之 3、删除vmlinuz和initramfs文件后无法启动,两种方法恢复之 4、增加新硬盘，在其上制作能单独运行kernel和bash的系统 5、在U盘上定制linux和busybox，使其可启动系统，并具有网络功能 6、删除/etc/fstab和/boot目录的所有文件，并恢复之 7、编译安装kernel，启用支持ntfs文件系统功能

> 答案:Linux_作死实验

七.systemd服务笔记

- POST --> Boot Sequence --> Bootloader --> kernel + initramfs(initrd) --> rootfs --> /sbin/init
- init:

```
    CentOS 5 SysV init
    CentOS 6 Upstart
    CentOS 7 Systemd
```

- Systemd：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(相对于centos6及以前版本)

```
1.系统引导时实现服务并行启动
2.按需启动守护进程
3.自动化的服务依赖关系管理
4.同时采用socket式与D-Bus总线式激活服务
5.系统状态快照
```

- 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
- 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/`
  `/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
- Unit类型

```
systemctl -t unitname 查看unit类型
service unit: 文件扩展名为.service，用于定义系统服务
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
Socket unit: .socket，用于标识进程间通信用的socket文件，也可在系统启 动时，延迟启动服
务，实现按需启动
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
```

- systemd关键特性

```
基于socket的激活机制：socket与服务程序分离
基于d-bus的激活机制：
基于device的激活机制：
基于path的激活机制：
系统快照：保存各unit的当前状态信息于持久存储设备中
向后兼容sysv init脚本
```

- 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管理服务

管理系统服务

- CentOS 7: service unit 能兼容早期的服务脚本
- 命令：systemctl COMMAND name.service

```
启动：service name start ==> systemctl start name.service
停止：service name stop ==> systemctl stop name.service
重启：service name restart ==> systemctl restart name.service
状态：service name status ==> systemctl status name.service
```

- 条件式重启：已启动才重启，否则不做操作 `service name condrestart ==> systemctl try-restart name.service`
- 重载或重启服务：先加载，再启动 `systemctl reload-or-restart name.service`
- 重载或条件式重启服务： `systemctl reload-or-try-restart name.service`
- 禁止自动和手动启动： `systemctl mask name.service`
- 取消禁止： `systemctl unmask name.service`

服务查看

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

chkconfig命令的对应关系：

- 设定某服务开机自启： `chkconfig name on ==> systemctl enable name.service`
- 设定某服务开机禁止启动： `chkconfig name off ==> systemctl disable name.service`
- 查看所有服务的开机自启状态： `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 sshd –list ==> ls /etc/systemd/system/*.wants/sshd.service`
- 查看服务是否开机自启： `systemctl is-enabled name.service`
- 查看服务的依赖关系： `systemctl list-dependencies name.service`
- 杀掉进程： `systemctl kill unitname`

服务状态说明

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

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`
- 列出失败的服务 `systemctl --failed --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`

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
- 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
- 帮助：sytemd.units(5),systemd.service(5), systemd.socket(5), systemd.target(5),systemd.exec(5)
- 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
- 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
- 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`

服务Unit文件示例

- 1.vim /etc/systemd/system/bak.service

```
[Unit]
Description=backup  /etc
Requires=atd.service
[Service]
Type=simple
ExecStart=/bin/bash -c "echo /testdir/bak.sh|at now"
[Install]
WantedBy=multi-user.target
```

- 2.systemctl daemon-reload
- 3.systemctl start bak
- vim /etc/systemd/system/tomcat.service

```
[Unit]
Description=java tomcat project
After=tomcat.service
[Service]
Type=forking
User=users
Group=users
PIDFile=/usr/local/tomcat/tomcat.pid
ExecStart=/usr/local/tomcat/bin/startup.sh
ExecReload=ExecStop=/usr/local/tomcat/bin/shutdown.sh
PrivateTmp=true
[Install]
WantedBy=multi-user.target
```

运行级别

- target units： unit配置文件：.target `ls /usr/lib/systemd/system/*.target systemctl list-unit-files --type target --all`
- 运行级别：

```
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：`init N ==> systemctl isolate name.target`
    - eg:`systemctl isolate multi-user.target`
- 注：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**)**
- 查看target： `runlevel who -r systemctl list-units --type target`1
- 获取默认运行级别： `/etc/inittab ==> systemctl get-default`
- 修改默认级别：

```
/etc/inittab ==> systemctl set-default name.target
systemctl set-default multi-user.target
ls –l /etc/systemd/system/default.target
```

- 切换至紧急救援模式： `systemctl rescue`
- 切换至emergency模式： `systemctl emergency`
- 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
```

八.centos7启动及排错

- CentOS 7 引导顺序

```
UEFi或BIOS初始化，运行POST开机自检
选择启动设备
引导装载程序，centos7是grub2
加载程序的配置文件：
    /etc/grub.d/
    /etc/default/grub
    /boot/grub2/grub.cfg
加载initramfs驱动模块
加载内核选项
内核初始化，centos7使用systemd代替init
执行initrd.target所有单元，包括挂载/etc/fstab
从initramfs根文件系统切换到磁盘根目录
systemd执行默认target配置，配置文件/etc/systemd/system/default.target
systemd执行sysinit.target初始化系统及basic.target准备操作系统
systemd启动multi-user.target下的本机与服务器服务
systemd执行multi-user.target下的/etc/rc.d/rc.local
Systemd执行multi-user.target下的getty.target及登录服务
systemd执行graphical需要的服务
```

设置内核参数，只影响当次启动

- 启动时，在linux16行后添加systemd.unit=desired.target `systemd.unit=emergency.target` `systemd.unit=rescue.target`
- rescue.target比emergency支持更多的功能，例如日志等
- systemctl default 进入默认target
- 启动排错
- 文件系统损坏
    - 先尝试自动修复，失败则进入emergency shell，提示用户修复
- 在/etc/fstab不存在对应的设备和UUID

    - 等一段时间，如不可用，进入emergency shell

- 在/etc/fstab不存在对应挂载点
    - systemd尝试创建挂载点，否则提示进入emergency shell.
- 在/etc/fstab不正确的挂载选项
    - 提示进入emergency shell

破解CentOS7的root口令方法一

```
1.启动时任意键暂停启动
2.按e键进入编辑模式
3.将光标移动linux16开始的行，添加内核参数rd.break
4.按ctrl-x启动
5.mount -o remount,rw  /sysroot
6.chroot /sysroot
7.passwd root
8.touch /.autorelabel
9.exit
10.reboot
```

破解CentOS7的root口令方法二

```
1.启动时任意键暂停启动
2.按e键进入编辑模式
3.将光标移动linux16开始的行，改为rw init=/sysroot/bin/sh
4.按ctrl-x启动
5.hchroot /sysroot
6.passwd root
7.touch /.autorelabel
8.exit
9.reboot
```

修复GRUB2

```
GRUB“the Grand Unified Bootloader”
    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
调整默认启动内核
```

```
vim /etc/default/grub
GRUB_DEFAULT=0
```