

麒麟信安服务器操作系统 3.5.3 管理员手册

文档标识：KYJS-KY3.5.3-Server-SAM-V1.0

文档版本：V1.0

发布日期：2024 年 08 月 27 日



变更记录

版本	修订时间	修订人	修订类型	修订章节	修订内容
V1.0	2024.08.27	潘晨博	A	全部	生成全文

注 1：修订类型分为 A-ADDED，M-MODIFIED，D-DELETED

注 2：对该文件内容增加、删除或修改均需填写此记录，详细记载变更信息，以保证其可追溯性

目 录

1 目的	1
2 适用范围	1
3 查看系统信息	1
3.1 查看版本信息	1
3.1.1 查看 cpu 信息	1
3.1.2 查看内存信息	1
3.1.3 查看磁盘信息	1
3.1.4 查看系统资源实时信息	1
3.1.5 查看系统用户	1
3.1.6 查看系统用户组	2
3.1.7 查看系统运行的进程	2
3.1.8 查看网络信息	2
4 基础配置	2
4.1 设置语言环境	2
4.1.1 显示当前语言环境	2
4.1.2 显示可用语言环境	2
4.1.3 设置语言环境	2
4.2 设置日期和时间	2
4.2.1 显示当前日期和时间	2
4.2.2 远程服务器时间同步	3
4.2.3 手动修改日期和时间	3
4.3 设置网络	3
4.3.1 nmcli 命令	3
4.3.2 设备管理	4
4.3.3 设备网络连接	4
4.3.4 配置动态 IP 连接	5
4.3.5 配置静态 IP 连接	5

4.3.6 配置主机名	5
5 管理用户和用户组	6
5.1 管理用户	6
5.1.1 增加用户	6
5.1.2 修改用户账号	6
5.1.3 删除用户	7
5.2 管理用户组	7
5.2.1 增加用户组	7
5.2.2 修改用户组	7
5.2.3 删除用户组	8
5.2.4 将用户加入用户组或从用户组中移除	8
5.2.5 切换用户组	8
6 管理软件包	8
6.1 配置软件源	8
6.2 DNF 管理软件包	9
6.3 DNF 检查并更新	11
7 管理服务	12
7.1 概念介绍	12
7.2 特性说明	12
7.2.1 更快的启动速度	13
7.2.2 提供按需启动能力	13
7.2.3 采用 cgroup 特性跟踪和管理进程的生命周期	13
7.2.4 启动挂载点和自动挂载的管理	14
7.2.5 实现事务性依赖关系管理	14
7.2.6 与 SysV 初始化脚本兼容	14
7.2.7 能够对系统进行快照和恢复	14
7.3 管理系统服务	15
7.3.1 显示所有当前服务	16
7.3.2 显示服务状态	16

7.3.3 运行服务	17
7.3.4 关闭服务	17
7.3.5 重启服务	18
7.3.6 启用服务	18
7.3.7 禁用服务	18
7.4 关闭、暂停和休眠系统	18
7.4.1 关闭系统	18
7.4.2 重启系统	18
7.4.3 系统待机	18
7.4.4 系统休眠	18
8 管理进程	19
8.1 查看进程	19
8.1.1 who 命令	19
8.1.2 ps 命令	19
8.1.3 top 命令	20
8.1.4 kill 命令	20
8.2 调度启动进程	21
8.2.1 定时运行一批程序（at）	21
8.2.2 周期性运行一批程序（cron）	21
9 管理硬盘	22
9.1 标准分区	22
9.2 LVM 简介	25
9.3 LVM 基本概念	26
9.4 管理物理卷	26
9.4.1 创建物理卷	26
9.4.2 查看物理卷	27
9.4.3 修改物理卷属性	27
9.4.4 删除物理卷	28
9.5 管理卷组	28

9.5.1 创建卷组	28
9.5.2 查看卷组	28
9.5.3 修改卷组属性	29
9.5.4 扩展卷组	29
9.5.5 收缩卷组	30
9.5.6 删除卷组	30
9.6 管理逻辑卷	30
9.6.1 创建逻辑卷	30
9.6.2 查看逻辑卷	31
9.6.3 调整逻辑卷大小	31
9.6.4 扩展逻辑卷	32
9.6.5 收缩逻辑卷	32
9.6.6 删除逻辑卷	33
9.7 创建并挂载文件系统	33
9.7.1 创建文件系统	33
9.7.2 手动挂载文件系统	33
9.7.3 自动挂载文件系统	34
10 虚拟化	35
10.1 环境搭建	35
10.2 配置流程	37
11 FAQ	45
12 帮助	51

1 目的

本文档主要讲解麒麟信安服务器操作系统的管理员操作,方便管理员更好地使用麒麟信安服务器操作系统。

2 适用范围

本文档适用于所有使用麒麟信安服务器操作系统 3.5.3 的管理员。

3 查看系统信息

3.1 查看版本信息

命令如下:

```
[root@localhost ~]# cat /etc/.kyinfo
```

3.1.1 查看 cpu 信息

命令如下:

```
[root@localhost ~]# lscpu
```

3.1.2 查看内存信息

命令如下:

```
[root@localhost ~]# free
```

3.1.3 查看磁盘信息

命令如下:

```
[root@localhost ~]# fdisk -l
```

3.1.4 查看系统资源实时信息

命令如下:

```
[root@localhost ~]# top
```

3.1.5 查看系统用户

查看系统创建了哪些用户,命令如下:

```
[root@localhost ~]# cat /etc/passwd
```

显示的内容为: 用户名:口令:用户标识号:组标识号:注释性描述:主目录:登录 shell

3.1.6 查看系统用户组

查看系统中所有组信息，命令如下：

```
[root@localhost ~]# cat /etc/group
root:x:0:
bin:x:1:
daemon:x:2:
```

显示的内容为：用户组:用户组口令:GID:包含的用户。

3.1.7 查看系统运行的进程

命令如下：

```
[root@localhost ~]# ps -aux
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root             1  0.0  0.5 170912 14344 ?        Ss   7月27   0:10 /usr/lib/systemd/systemd rhg
root             2  0.0  0.0      0     0 ?        S    7月27   0:00 [kthreadd]
root             3  0.0  0.0      0     0 ?        I<   7月27   0:00 [rcu_gp]
```

3.1.8 查看网络信息

命令如下：

```
[root@localhost ~]# ip a
```

4 基础配置

4.1 设置语言环境

4.1.1 显示当前语言环境

命令如下：

```
[root@localhost ~]# localectl status
System Locale: LANG=zh_CN.UTF-8
VC Keymap: cn
X11 Layout: cn
```

4.1.2 显示可用语言环境

命令如下：

```
[root@localhost ~]# localectl list-locales
```

4.1.3 设置语言环境

例如设置为简体中文语言环境，命令如下：

```
[root@localhost ~]# localectl set-locales LANG=zh_CN.UTF-8
```

4.2 设置日期和时间

4.2.1 显示当前日期和时间

命令如下：


```
[系统未激活][root@localhost ~]# timedatectl
Local time: 二 2023-07-18 18:51:26 CST
Universal time: 二 2023-07-18 10:51:26 UTC
RTC time: 二 2023-07-18 11:21:13
Time zone: Asia/Shanghai (CST, +0800)
System clock synchronized: no
NTP service: inactive
RTC in local TZ: no
```

4.2.2 远程服务器时间同步

您可以启用 NTP 远程服务器进行系统时钟的自动同步。若启用了 NTP 远程服务器进行系统时钟自动同步，则不能手动修改日期和时间；若需要手动修改日期或时间，则需确保已经关闭 NTP 系统时钟自动同步。

例如开启自动远程的时间同步，命令如下：

```
[root@localhost ~]# timedatectl set-ntp yes
```

4.2.3 手动修改日期和时间

手动修改日期或时间，请确保已经关闭 NTP 系统时钟自动同步。

修改当前的日期，其中 YYYY 代表年份，MM 代表月份，DD 代表某天，命令如下：

```
[root@localhost ~]# timedatectl set-time YYYY-MM-DD
```

修改当前的时间，其中 HH 代表小时，MM 代表分钟，SS 代表秒，命令如下：

```
[root@localhost ~]# timedatectl set-time HH:MM:SS
```

修改时区，其中 time_zone 是您想要设置的时区，命令如下：

```
[root@localhost ~]# timedatectl set-timezone time_zone
```

4.3 设置网络

4.3.1 nmcli 命令

nmcli 是 NetworkManager 的一个命令行工具，使用 nmcli 命令配置的网络配置可以立即生效且系统重启后配置也不会丢失。

nmcli 命令的基本格式格式如下：

```
[root@localhost ~]# nmcli help
Usage: nmcli [OPTIONS] OBJECT { COMMAND | help }
```

显示 NetworkManager 状态:

```
[root@localhost ~]# nmcli general status
```

显示所有连接:

```
[root@localhost ~]# nmcli connection show
```

只显示当前活动连接, 添加-a, --active:

```
[root@localhost ~]# nmcli connection show --active
```

显示由 NetworkManager 识别到设备及其状态:

```
[root@localhost ~]# nmcli device status
```

使用 nmcli 工具启动和停止网络接口, 在 root 权限下执行如下命令:

```
[root@localhost ~]# nmcli connection up id p8p1
连接已成功激活 (D-Bus 活动路径: /org/freedesktop/NetworkManager/ActiveConnection/4)
[root@localhost ~]# nmcli device disconnect p8p1
成功断开设备 "p8p1"。
```

4.3.2 设备管理

连接到设备: 使用如下命令, NetworkManager 将连接到对应网络设备, 尝试找到合适的连接配置, 并激活配置。如果不存在相应的配置连接, NetworkManager 将创建并激活具有默认设置的新配置文件。

```
[root@localhost ~]# nmcli device connect "$IFNAME"
```

断开设备连接: 使用如下命令, NetworkManager 将断开设备连接, 并防止设备自动激活。

```
[root@localhost ~]# nmcli device disconnect "$IFNAME"
```

4.3.3 设备网络连接

列出目前可用的网络连接:

```
[root@localhost ~]# nmcli con show
NAME      UUID                                  TYPE      DEVICE
p8p1      d3395f17-0052-43c0-8d18-b77f92d7df1f ethernet  p8p1
virbr0    48a6f597-2eb8-4156-b0dc-ac4f22cf9c63 bridge    virbr0
p2p1      2514ee11-be1b-4aef-a950-4555e456d929 ethernet  --
```

添加一个网络连接会生成相应的配置文件, 并与相应的设备关联。检查可用的设备, 方法如下:

```
[root@localhost ~]# nmcli dev status
DEVICE      TYPE      STATE      CONNECTION
p8p1        ethernet  已连接      p8p1
virbr0       bridge    连接（外部） virbr0
p2p1        ethernet  不可用      --
lo          loopback  未托管      --
virbr0-nic  tun       未托管      --
```

4.3.4 配置动态 IP 连接

要使用 DHCP 分配网络时，可以使用动态 IP 配置添加网络配置文件，命令格式如下：

```
[root@localhost ~]# nmcli connection add type ethernet con-name connection-name ifname interface-name
```

例如创建名为 net-test 的动态连接配置文件，在 root 权限下使用以下命令：

```
[root@localhost ~]# nmcli connection add type ethernet con-name net-test ifname p8p1
连接 "net-test" (9666362e-ef06-46ef-b6a3-6787cf2cf905) 已成功添加。
```

激活连接并检查状态：

```
[root@localhost ~]# nmcli con up net-test
连接已成功激活 (D-Bus 活动路径: /org/freedesktop/NetworkManager/ActiveConnection/6)
[root@localhost ~]# nmcli device status
DEVICE      TYPE      STATE      CONNECTION
p8p1        ethernet  已连接      net-test
virbr0       bridge    连接（外部） virbr0
p2p1        ethernet  不可用      --
lo          loopback  未托管      --
virbr0-nic  tun       未托管      --
```

4.3.5 配置静态 IP 连接

添加静态 IPv4 配置的网络连接，可使用以下命令：

```
[root@localhost ~]# nmcli con add type ethernet con-name connection-name ifname interface-name ip4 address gw4 address
```

例如创建名为 net-static 的静态连接配置文件，在 root 权限下使用以下命令：

```
[root@localhost ~]# nmcli con add type ethernet con-name net-static ifname p8p1 ip4 10.10.7.58 gw4 10.60.7.1
连接 "net-static" (e9427b45-058f-477c-87c8-261bf28f2eec) 已成功添加。
```

激活连接并检查状态：

```
[root@localhost ~]# nmcli con up net-static ifname p8p1
连接已成功激活 (D-Bus 活动路径: /org/freedesktop/NetworkManager/ActiveConnection/13)
[root@localhost ~]# nmcli device status
DEVICE      TYPE      STATE      CONNECTION
p8p1        ethernet  已连接      net-static
virbr0       bridge    连接（外部） virbr0
p2p1        ethernet  不可用      --
lo          loopback  未托管      --
virbr0-nic  tun       未托管      --
```

4.3.6 配置主机名

查看当前的主机名，使用如下命令：

```
[root@localhost ~]# hostnamectl status
```

在 root 权限下，设定系统中的所有主机名，使用如下命令：

```
[root@localhost ~]# hostnamectl set-hostname name
```

5 管理用户和用户组

在 Linux 中，每个普通用户都有一个账户，包括用户名、密码和主目录等信息。除此之外，还有一些系统本身创建的特殊用户，它们具有特殊的意义，其中最重要的是管理员账户，默认用户名是 `root`。同时 Linux 也提供了用户组，使每一个用户至少属于一个组，从而便于权限管理。

用户和用户组管理是系统安全管理的重要组成部分，本章主要介绍麒麟信安服务器操作系统提供的用户管理和组管理命令。

5.1 管理用户

5.1.1 增加用户

在 `root` 权限下，通过 `useradd` 命令可以为系统添加新用户信息，其中 `options` 为相关参数，`username` 为用户名称。

```
[root@localhost ~]# useradd [options] username
```

例如新建一个用户名为 `test` 的用户，在 `root` 权限下执行如下命令，没有任何提示，表明用户建立成功。这时并没有设置用户的口令，请使用 `passwd` 命令修改用户的密码，没有设置密码的新账号不能登录系统。

```
[root@localhost ~]# useradd test
```

使用 `id` 命令查看新建的用户信息，命令如下：

```
[root@localhost ~]# id test
用户id=1000(test) 组id=1000(test) 组=1000(test)
```

修改用户 `test` 的密码：

```
[root@localhost ~]# passwd test
```

5.1.2 修改用户账号

修改密码：普通用户可以用 `passwd` 修改自己的密码，只有管理员才能用 `passwd username` 为其他用户修改密码。

修改用户 `shell`：用户可以使用 `usermod` 命令修改 `shell` 信息，在 `root` 权限下执行如下命令，其中 `new_shell_path` 为目标 `shell` 路径，`username` 为要修改用户的用户名，请根据实际情况修改：

```
[root@localhost ~]# usermod -s new_shell_path username
```

例如，将用户 test 的 shell 改为 csh，命令如下：

```
[root@localhost ~]# usermod -s /bin/csh test
```

修改主目录：可以在 root 权限下执行如下命令，其中 new_home_directory 为已创建的目标主目录的路径，username 为要修改用户的用户名，请根据实际情况修改：

```
[root@localhost ~]# usermod -d new_home_directory username
```

修改用户 ID：在 root 权限下执行如下命令，其中 UID 代表目标用户 ID，username 代表用户名，请根据实际情况修改：

```
[root@localhost ~]# usermod -u UID username
```

5.1.3 删除用户

删除用户：在 root 权限下，使用 userdel 命令可删除现有用户。如果想同时删除该用户的主目录以及其中所有内容，要使用 -r 参数递归删除。不建议直接删除已经进入系统的用户，如果需要强制删除，请使用 userdel -f test 命令。

例如，删除用户 test 命令如下：

```
[root@localhost ~]# userdel test
```

5.2 管理用户组

5.2.1 增加用户组

在 root 权限下，通过 groupadd 命令可以为系统添加新用户组信息，其中 options 为相关参数，groupname 为用户组名称。

```
[root@localhost ~]# groupadd [options] groupname
```

例如新建一个用户组名为 grouptest 的用户，在 root 权限下执行如下命令：

```
[root@localhost ~]# groupadd grouptest
```

5.2.2 修改用户组

修改 GID：在 root 权限下执行如下命令，其中 GID 代表目标用户组 ID，groupname 代表用户组，请根据实际情况修改：

```
[root@localhost ~]# groupmod -g GID groupname
```

修改用户组名：在 root 权限下执行如下命令，其中 newgroupname 代表新用

户组名， `oldgroupname` 代表已经存在的待修改的用户组名，请根据实际情况修改：

```
[root@localhost ~]# groupmod -n newgroupname oldgroupname
```

5.2.3 删除用户组

删除用户组：在 `root` 权限下，使用 `groupdel` 命令可删除用户组。`groupdel` 不能直接删除用户的主组，如果需要强制删除用户主组，请使用 `groupdel -f groupname` 命令。

例如，删除用户组 `groupname` 命令如下：

```
[root@localhost ~]# groupdel groupname
```

5.2.4 将用户加入用户组或从用户组中移除

在 `root` 权限下，使用 `gpasswd` 命令将用户加入用户组或从用户组中移除。

例如，将用户 `test` 加入用户组 `groupname`，命令如下：

```
[root@localhost ~]# gpasswd -a test groupname
```

例如，将用户 `test` 从 `groupname` 用户组中移除，命令如下：

```
[root@localhost ~]# gpasswd -d test groupname
```

5.2.5 切换用户组

一个用户同时属于多个用户组时，则在用户登录后，使用 `newgrp` 命令可以切换到其他用户组，以便具有其他用户组的权限。

例如，将用户 `test` 切换到 `groupname` 用户组，命令如下：

```
[root@localhost ~]# newgrp groupname
```

6 管理软件包

DNF 是一款 Linux 软件包管理工具，用于管理 RPM 软件包。DNF 可以查询软件包信息，从指定软件库获取软件包，自动处理依赖关系以安装或卸载软件包，以及更新系统到最新可用版本。

注意：DNF 与 YUM 完全兼容，提供了 YUM 兼容的命令行以及扩展和插件提供的 API。使用 DNF 需要管理员权限。

6.1 配置软件源

网络源配置：配置文件位于/etc/yum.repos.d/目录下，在主机网络正常访问外面的情况下，可以直接使用默认网络源，不需要对配置文件做任何修改。

```
[root@localhost ~]# ls /etc/yum.repos.d/
```

本地光盘作为软件源：在无法联网的情况下，可以考虑用本地光盘（或安装映像文件）作为软件源。

放入安装光盘，并挂载光盘到指定位置。命令如下：

```
[root@localhost ~]# mkdir /mnt/cdrom  
[root@localhost ~]# mount /dev/cdrom /mnt/cdrom/
```

修改光盘源配置文件，设置 baseurl=file:///mnt/cdrom

```
[root@localhost ~]# vim /etc/yum.repos.d/KylinSec.repo
```

注意，其中 enabled 的值决定是否启用软件源，1 表示启用，0 表示禁用。

gpgkey 是验证签名用的公钥。

6.2 DNF 管理软件包

您可以使用 rpm 包名称、缩写或者描述搜索需要的 RPM 包，使用命令如下：

```
[root@localhost ~]# dnf search term
```

示例如下：

```
[root@localhost ~]# dnf search httpd
Last metadata expiration check: 11:21:08 ago on 2023年07月21日 星期五 00时00分46秒.
===== Name Exactly Matched: httpd =====
httpd.aarch64 : Apache HTTP Server
===== Name & Summary Matched: httpd =====
httpd-debuginfo.aarch64 : Debug information for package httpd
httpd-debugsource.aarch64 : Debug sources for package httpd
httpd-devel.aarch64 : Development files for httpd
keycloak-httpd-client-install.noarch : Provides tools to configure Apache HTTPD as Keycloak
                                     : client
keycloak-httpd-client-install-help.noarch : Provides help and man docs for
                                     : keycloak-httpd-client-install
kylinsec-logos-httpd.noarch : KylinSec-related icons and pictures used by httpd
libmicrohttpd-debuginfo.aarch64 : Debug information for package libmicrohttpd
libmicrohttpd-debugsource.aarch64 : Debug sources for package libmicrohttpd
libmicrohttpd-devel.aarch64 : Development files for libmicrohttpd
libmicrohttpd-help.noarch : This help package for libmicrohttpd
lighttpd-debuginfo.aarch64 : Debug information for package lighttpd
lighttpd-debugsource.aarch64 : Debug sources for package lighttpd
lighttpd-fastcgi.aarch64 : FastCGI module and spawning helper for lighttpd and PHP
                           : configuration
lighttpd-filesystem.noarch : The basic directory layout for lighttpd
lighttpd-mod_authn_gssapi.aarch64 : Authentication module for lighttpd that uses GSSAPI
lighttpd-mod_authn_mysql.aarch64 : Authentication module for lighttpd that uses a MySQL
                                  : database
lighttpd-mod_authn_pam.aarch64 : Authentication module for lighttpd that uses PAM
lighttpd-mod_mysql_vhost.aarch64 : Virtual host module for lighttpd that uses a MySQL database
python3-keycloak-httpd-client-install.noarch : Provides tools to configure Apache HTTPD as
```

列出软件包清单，可使用如下命令：

```
[root@localhost ~]# dnf list all
```

列出系统中特定的 rpm 包，示例如下：

```
[root@localhost ~]# dnf list httpd
Last metadata expiration check: 11:27:00 ago on 2023年07月21日 星期五 00时00分46秒.
Available Packages
httpd.aarch64                                2.4.51-12.ky3_5.kb3                everything
[root@localhost ~]#
```

显示某个 rpm 包的详细信息，可使用如下命令：

```
[root@localhost ~]# dnf info package_name
```

示例如下：


```
[root@localhost ~]# dnf info httpd
Last metadata expiration check: 11:31:59 ago on 2023年07月21日 星期五 00时00分46秒.
Available Packages
Name      : httpd
Version   : 2.4.51
Release   : 12.ky3_5.kb3
Architecture : aarch64
Size      : 1.3 M
Source    : httpd-2.4.51-12.ky3_5.kb3.src.rpm
Repository : everything
Summary   : Apache HTTP Server
URL       : https://httpd.apache.org/
License   : ASL 2.0
Description : Apache HTTP Server is a powerful and flexible HTTP/1.1 compliant web server.
```

要安装一个软件包及其所有未安装的依赖，请在 root 权限下执行如下命令：

```
[root@localhost ~]# dnf install package_name
```

下载软件包，可在 root 权限下使用如下命令：

```
[root@localhost ~]# dnf download package_name
```

如果需要同时下载未安装的依赖，则加上--resolve，可使用如下命令：

```
[root@localhost ~]# dnf download --resolve package_name
```

删除软件包：要卸载软件包以及相关的依赖软件包，请在 root 权限下执行如下命令：

```
[root@localhost ~]# dnf remove package_name
```

6.3 DNF 检查并更新

检查当前系统可用的更新，使用如下命令：

```
[root@localhost ~]# dnf check-update
Last metadata expiration check: 12:01:13 ago on 2023年07月21日 星期五 00时00分46秒.
Obsoleting Packages
kde-filesystem.aarch64      4-62.kb1.ky3_5      @anaconda
kde-filesystem.aarch64      4-62.kb1.ky3_5      @anaconda
kde-filesystem.aarch64      4-62.kb1.ky3_5      everything
kde-filesystem.aarch64      4-62.kb1.ky3_5      @anaconda
[root@localhost ~]#
```

升级：需要升级单个软件包，在 root 权限下使用如下命令：

```
[root@localhost ~]# dnf update package_name
```

若需要更新所有的包和依赖，可使用如下命令：

```
[root@localhost ~]# dnf update
```

7 管理服务

本章介绍如何使用 `systemd` 进行系统和服务管理。

`systemd` 是在 Linux 下，与 SysV 和 LSB 初始化脚本兼容的系统和服管理器。`systemd` 使用 `socket` 和 `D-Bus` 来开启服务，提供基于守护进程的按需启动策略，支持快照和系统状态恢复，维护挂载和自挂载点，实现了各服务间基于从属关系的一个更为精细的逻辑控制，拥有更高的并行性能。

7.1 概念介绍

`systemd` 开启和监督整个系统是基于 `unit` 的概念。`unit` 是由一个与配置文件对应的名字和类型组成的（例如：`avahi.service unit` 有一个具有相同名字的配置文件，是守护进程 `Avahi` 的一个封装单元）。`unit` 有多重类型，如表 1 所示。

表 1 unit 说明

unit 名称	后缀名	描述
Service unit	.service	系统服务。
Target unit	.target	一组 <code>systemd units</code> 。
Automount unit	.automount	文件系统挂载点。
Device unit	.device	内核识别的设备文件。
Mount unit	.mount	文件系统挂载点。
Path unit	.path	在一个文件系统中的文件或目录。
Scope unit	.scope	外部创建的进程。
Slice unit	.slice	一组用于管理系统进程分层组织的 <code>units</code> 。
Socket unit	.socket	一个进程间通信的 <code>Socket</code> 。
Swap unit	.swap	<code>swap</code> 设备或者 <code>swap</code> 文件。
Timer unit	.timer	<code>systemd</code> 计时器。

所有的可用 `systemd unit` 类型，可在如表 2 所示的路径下查看。

表 2 可用 `systemd unit` 类型

路径	描述
<code>/usr/lib/systemd/system/</code>	随安装的 RPM 产生的 <code>systemd units</code> 。
<code>/run/systemd/system/</code>	在运行时创建 <code>systemd units</code> 。
<code>/etc/systemd/system/</code>	由系统管理员创建和管理的 <code>systemd units</code> 。

7.2 特性说明

7.2.1 更快的启动速度

systemd 提供了比 UpStart 更激进的并行启动能力，采用了 socket/D-Bus activation 等技术启动服务，带来了更快的启动速度。

为了减少系统启动时间，systemd 的目标是：

1. 尽可能启动更少的进程。
2. 尽可能将更多进程并行启动。

7.2.2 提供按需启动能力

当 sysvinit 系统初始化的时候，它会将所有可能用到的后台服务进程全部启动运行。并且系统必须等待所有的服务都启动就绪之后，才允许用户登录。这种做法有两个缺点：首先是启动时间过长；其次是系统资源浪费。

某些服务很可能在很长一段时间内，甚至整个服务器运行期间都没有被使用过。比如 CUPS，打印服务在多数服务器上很少被真正使用到。您可能没有想到，在很多服务器上 SSHD 也是很少被真正访问到的。花费在启动这些服务上的时间是不必要的；同样，花费在这些服务上的系统资源也是一种浪费。

systemd 可以提供按需启动的能力，只有在某个服务被真正请求的时候才启动它。当该服务结束，systemd 可以关闭它，等待下次需要时再次启动它。

7.2.3 采用 cgroup 特性跟踪和管理进程的生命周期

init 系统的一个重要职责就是负责跟踪和管理服务进程的生命周期。它不仅启动一个服务，也能够停止服务。这看上去没有什么特别的，然而在真正用代码实现的时候，您或许会发现停止服务比一开始想的要困难。

服务进程一般都会作为守护进程（daemon）在后台运行，为此服务程序有时候会派生（fork）两次。在 UpStart 中，需要在配置文件中正确地配置 expect 小节。这样 UpStart 通过对 fork 系统调用进行计数，从而获知真正的精灵进程的 PID 号。

cgroup 已经出现了很久，它主要用来实现系统资源配额管理。cgroup 提供了类似文件系统的接口，使用方便。当进程创建子进程时，子进程会继承父进程的 cgroup。因此无论服务如何启动新的子进程，所有的这些相关进程都会属于同一个 cgroup，systemd 只需要简单地遍历指定的 cgroup 即可正确地找到所有的相关进程，将它们逐一停止即可。

7.2.4 启动挂载点和自动挂载的管理

传统的 Linux 系统中，用户可以用 `/etc/fstab` 文件来维护固定的文件系统挂载点。这些挂载点在系统启动过程中被自动挂载，一旦启动过程结束，这些挂载点就会确保存在。这些挂载点都是对系统运行至关重要的文件系统，比如 HOME 目录。和 `sysvinit` 一样，`systemd` 管理这些挂载点，以便能够在系统启动时自动挂载它们。`systemd` 还兼容 `/etc/fstab` 文件，您可以继续使用该文件管理挂载点。

有时候用户还需要动态挂载点，比如打算访问 DVD 内容时，才临时执行挂载以便访问其中的内容，而不访问光盘时该挂载点被取消 (`umount`)，以便节约资源。传统地，人们依赖 `autofs` 服务来实现这种功能。

`systemd` 内建了自动挂载服务，无需另外安装 `autofs` 服务，可以直接使用 `systemd` 提供的自动挂载管理能力来实现 `autofs` 的功能。

7.2.5 实现事务性依赖关系管理

系统启动过程是由很多的独立工作共同组成的，这些工作之间可能存在依赖关系，比如挂载一个 NFS 文件系统必须依赖网络能够正常工作。`systemd` 虽然能够最大限度地并发执行很多有依赖关系的工作，但是类似“挂载 NFS”和“启动网络”这样的工作还是存在天生的先后依赖关系，无法并发执行。对于这些任务，`systemd` 维护一个“事务一致性”的概念，保证所有相关的服务都可以正常启动而不会出现互相依赖，以至于死锁的情况。

7.2.6 与 SysV 初始化脚本兼容

和 `UpStart` 一样，`systemd` 引入了新的配置方式，对应用程序的开发也有一些新的要求。如果 `systemd` 想替代目前正在运行的初始化系统，就必须和现有程序兼容。任何一个 Linux 发行版都很难为了采用 `systemd` 而在短时间内将所有的服务代码都修改一遍。

`systemd` 提供了和 `sysvinit` 以及 `LSB initscripts` 兼容的特性。系统中已经存在的服务和进程无需修改。这降低了系统向 `systemd` 迁移的成本，使得 `systemd` 替换现有初始化系统成为可能。

7.2.7 能够对系统进行快照和恢复

`systemd` 支持按需启动，因此系统的运行状态是动态变化的，人们无法准确地知道系统当前运行了哪些服务。`systemd` 快照提供了一种将当前系统运行状态

保存并恢复的能力。

比如系统当前正运行服务 A 和 B，可以用 `systemd` 命令行对当前系统运行状况创建快照。然后将进程 A 停止，或者做其他的任意的对系统的改变，比如启动新的进程 C。在这些改变之后，运行 `systemd` 的快照恢复命令，就可立即将系统恢复到快照时刻的状态，即只有服务 A 和 B 在运行。一个可能的应用场景是调试：比如服务器出现一些异常，为了调试用户将当前状态保存为快照，然后可以进行任意的操作，比如停止服务等等。等调试结束，恢复快照即可。

7.3 管理系统服务

`systemd` 提供 `systemctl` 命令来运行、关闭、重启、显示、启用/禁用系统服务。

`systemd` 提供 `systemctl` 命令与 `sysvinit` 命令的功能类似。当前版本中依然兼容 `service` 和 `chkconfig` 命令，相关说明如表 3，但建议用 `systemctl` 进行系统服务管理。

表 3 sysvinit 命令和 systemd 命令的对照表

sysvinit 命令	systemd 命令	备注
<code>service network start</code>	<code>systemctl start network.service</code>	用来启动一个服务 (并不会重启现有的)。
<code>service network stop</code>	<code>systemctl stop network.service</code>	用来停止一个服务 (并不会重启现有的)。
<code>service network restart</code>	<code>systemctl restart network.service</code>	用来停止并启动一个服务。
<code>service network reload</code>	<code>systemctl reload network.service</code>	当支持时，重新装载配置文件而不中断等待操作。
<code>service network condrestart</code>	<code>systemctl condrestart network.service</code>	如果服务正在运行那么重启它。
<code>service network status</code>	<code>systemctl status network.service</code>	检查服务的运行状态。
<code>chkconfig network on</code>	<code>systemctl enable network.service</code>	在下次启动时或满足其他触发条件时设置服务为启用。
<code>chkconfig network off</code>	<code>systemctl disable network.service</code>	在下次启动时或满足其他触发条件时设置服务为禁用。
<code>chkconfig network</code>	<code>systemctl is-enabled network.service</code>	用来检查一个服务在当前环境下被配置为启用还是禁用。

chkconfig --list	systemctl list-unit-files --type=service	输出在各个运行级别下服务的启用和禁用情况。
chkconfig network --list	ls /etc/systemd/system/*.wants/network.service	用来列出该服务在哪些运行级别下启用和禁用。
chkconfig network --add	systemctl daemon-reload	当您创建新服务文件或者变更设置时使用。

7.3.1 显示所有当前服务

如果您需要显示当前正在运行的服务，使用命令如下：

```
[root@localhost ~]# systemctl list-units --type service
```

如果您需要显示所有的服务（包括未运行的服务），需要添加-all 参数，使用命令如下：

```
[root@localhost ~]# systemctl list-units --type service --all
```

7.3.2 显示服务状态

如果您需要显示某个服务的状态，可执行如下命令：

```
[root@localhost ~]# systemctl status name.service
```

表 4 状态参数说明

参数	描述
Loaded	说明服务是否被加载，并显示服务对应的绝对路径以及是否启用。
Active	说明服务是否正在运行，并显示时间节点。
Main PID	相应的系统服务的 PID 值。
CGroup	相关控制组（CGroup）的其他信息。

如果您需要鉴别某个服务是否运行，可执行如下命令：

```
[root@localhost ~]# systemctl is-active name.service
```

表 5 is-active 命令的返回结果

状态	含义
active	这个服务正在运行。
inactive	这个服务没有运行。

如果您需要判断某个服务是否被启用，可执行如下命令：

```
[root@localhost ~]# systemctl is-enabled name.service
```

表 6 is-enabled 命令的返回结果

状态	含义
"enabled"	已经通过 /etc/systemd/system/ 目录下的 Alias= 别名、.wants/ 或 .requires/ 软连接被永久启用。
"enabled-runtime"	已经通过 /run/systemd/system/ 目录下的 Alias= 别名、.wants/ 或 .requires/ 软连接被临时启用。
"linked"	虽然单元文件本身不在标准单元目录中,但是指向此单元文件的一个或多个软连接已经存在于 /etc/systemd/system/ 永久目录中。
"linked-runtime"	虽然单元文件本身不在标准单元目录中,但是指向此单元文件的一个或多个软连接已经存在于 /run/systemd/system/ 临时目录中。
"masked"	已经被 /etc/systemd/system/ 目录永久屏蔽(软连接指向 /dev/null 文件), 因此 start 操作会失败。
"masked-runtime"	已经被 /run/systemd/systemd/ 目录临时屏蔽(软连接指向 /dev/null 文件), 因此 start 操作会失败。
"static"	尚未被启用, 并且单元文件的 "[Install]" 小节中没有可用于 enable 命令的选项。
"indirect"	尚未被启用, 但是单元文件的 "[Install]" 小节中 Also= 选项的值列表非空(也就是列表中的某些单元可能已被启用)、或者它拥有一个不在 Also= 列表中的其他名称的别名软连接。对于模版单元来说, 表示已经启用了不同于 DefaultInstance= 的实例。
"disabled"	尚未被启用, 但是单元文件的 "[Install]" 小节中存在可用于 enable 命令的选项
"generated"	单元文件是被单元生成器动态生成的。被生成的单元文件可能并未被直接启用, 而是被单元生成器隐含的启用了。
"transient"	单元文件是被运行时 API 动态临时生成的。该临时单元可能并未被启用。
"bad"	单元文件不正确或者出现其他错误。 is-enabled 不会返回此状态, 而是会显示一条出错信息。 list-unit-files 命令有可能会显示此单元。

7.3.3 运行服务

如果您需要运行某个服务, 请在 root 权限下执行如下命令:

```
[root@localhost ~]# systemctl start name.service
```

7.3.4 关闭服务

如果您需要关闭某个服务, 请在 root 权限下执行如下命令:

```
[root@localhost ~]# systemctl stop name.service
```

7.3.5 重启服务

如果您需要重启某个服务，请在 root 权限下执行如下命令：

```
[root@localhost ~]# systemctl restart name.service
```

7.3.6 启用服务

如果您需要在开机时启用某个服务，请在 root 权限下执行如下命令：

```
[root@localhost ~]# systemctl enable name.service
```

7.3.7 禁用服务

如果您需要在开机时禁用某个服务，请在 root 权限下执行如下命令：

```
[root@localhost ~]# systemctl disable name.service
```

7.4 关闭、暂停和休眠系统

7.4.1 关闭系统

关闭系统并下电，在 root 权限下执行如下命令：

```
[root@localhost ~]# systemctl poweroff
```

关闭系统但不下电机器，在 root 权限下执行如下命令：

```
[root@localhost ~]# systemctl halt
```

7.4.2 重启系统

重启系统，在 root 权限下执行如下命令：

```
[root@localhost ~]# systemctl reboot
```

7.4.3 系统待机

使系统待机，在 root 权限下执行如下命令：

```
[root@localhost ~]# systemctl suspend
```

7.4.4 系统休眠

使系统休眠，在 root 权限下执行如下命令：

```
[root@localhost ~]# systemctl hibernate
```

使系统待机且处于休眠状态，在 root 权限下执行如下命令：

```
[root@localhost ~]# systemctl hybrid-sleep
```


8 管理进程

操作系统管理多个用户的请求和多个任务。大多数系统都只有一个 CPU 和一个主要存储，但一个系统可能有多个二级存储磁盘和多个输入/输出设备。操作系统管理这些资源并在多个用户间共享资源，当用户提出一个请求时，造成好像系统被用户独占的假象。实际上操作系统监控着一个等待执行的任务队列，这些任务包括用户任务、操作系统任务、邮件和打印任务等。本章节将从用户的角度讲述如何控制进程。

8.1 查看进程

Linux 是一个多任务系统，经常需要对这些进程进行一些调配和管理。要进行管理，首先就要知道现在的进程情况：有哪些进程、进程的状态如何等。Linux 提供了多种命令来了解进程的状况。

8.1.1 who 命令

who 命令主要用于查看当前系统中的用户情况。如果用户想和其他用户建立即时通讯，比如使用 talk 命令，那么首先要确定的就是该用户确实在线上，不然 talk 进程就无法建立起来。又如，系统管理员希望监视每个登录的用户此时此刻的所作所为，也要使用 who 命令。who 命令应用起来非常简单，可以比较准确地掌握用户的情况，所以使用非常广泛。

例如查看系统中的用户及其状态。使用如下：

```
[root@localhost ~]# who
root    tty1      2022-12-12 22:38 (:0)
root    pts/1    2022-12-13 04:47 (10.200.6.194)
```

8.1.2 ps 命令

ps 命令是最基本又非常强大的进程查看命令。使用该命令可以确定有哪些进程正在运行和运行的状态、进程是否结束、进程有没有僵尸、哪些进程占用了过多的资源等，大部分进程信息都是可以通过执行该命令得到的。

ps 命令最常用的还是用来监控后台进程的工作情况，因为后台进程是不与屏幕、键盘这些标准输入/输出设备进行通信的，所以如果需要检测其状况，就可使用 ps 命令。ps 命令的常见选项如表 7 所示。

表 7 选项说明

选项	描述
----	----

-e	显示所有进程。
-f	全格式。
-h	不显示标题。
-l	使用长格式。
-w	宽行输出。
-a	显示终端上的所有进程，包括其他用户的进程。
-r	只显示正在运行的进程。
-x	显示没有控制终端的进程。

例如显示系统中终端上的所有进行进程。命令如下：

```
[root@localhost ~]# ps -a
  PID TTY          TIME CMD
 235013 pts/1    00:00:00 ps
```

8.1.3 top 命令

top 命令和 ps 命令的基本作用是相同的，显示系统当前的进程和其他状况，但是 top 是一个动态显示过程，即可以通过用户按键来不断刷新进程的当前状态，如果在前台执行该命令，它将独占前台，直到用户终止该程序为止。其实 top 命令提供了实时的对系统处理器的状态监视。它将显示系统中 CPU 的任务列表。该命令可以按 CPU 使用、内存使用和执行时间对任务进行排序，而且该命令的很多特性都可以通过交互式命令或者在定制文件中进行设定。

top 命令输出的实例如下：

```
top - 03:14:23 up 1 day, 4:36, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 270 total, 1 running, 269 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 99.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 15231.6 total, 10406.4 free, 1323.8 used, 3501.4 buff/cache
MiB Swap: 7940.0 total, 7940.0 free, 0.0 used, 13372.5 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 232946 root        20   0       0        0        0 I   0.7   0.0   0:02.48 kworker/u2+
 1834 root        20   0  332096  27796 12524 S   0.3   0.2   0:04.11 lvmdbusd
 9937 root        20   0  539268  49056 34936 S   0.3   0.3   0:35.95 marco
    1 root        20   0 169444  16080  9312 S   0.0   0.1   0:01.81 systemd
    2 root        20   0        0        0        0 S   0.0   0.0   0:00.01 kthreadd
    3 root         0 -20        0        0        0 I   0.0   0.0   0:00.00 rcu_gp
    4 root         0 -20        0        0        0 I   0.0   0.0   0:00.00 rcu_par_gp
    6 root         0 -20        0        0        0 I   0.0   0.0   0:00.00 kworker/0:
    8 root         0 -20        0        0        0 I   0.0   0.0   0:00.00 mm_percpu_
    9 root        20   0        0        0        0 S   0.0   0.0   0:00.00 rcu_tasks_
   10 root        20   0        0        0        0 S   0.0   0.0   0:00.00 rcu_tasks_
   11 root        20   0        0        0        0 S   0.0   0.0   0:00.01 ksoftirqd/0
   12 root        20   0        0        0        0 I   0.0   0.0   1:15.85 rcu_sched
```

8.1.4 kill 命令

当需要中断一个前台进程的时候，通常是使用“Ctrl+C”组合键，而对于后台进程不能用组合键来终止，这时就可以使用 kill 命令。该命令可以终止前台和后

台进程。终止后台进程的原因包括：该进程占用 CPU 的时间过多、该进程已经死锁等。

kill 命令是通过向进程发送指定的信号来结束进程的。如果没有指定发送的信号，那么默认值为 TERM 信号。TERM 信号将终止所有不能捕获该信号的进程。至于那些可以捕获该信号的进程可能就需要使用 KILL 信号（它的编号为 9），而该信号不能被捕捉。

kill 命令的语法格式有以下两种方式：

```
[root@localhost ~]# kill [-s 信号 | -p] [-a] 进程号...
```

其中进程号可以通过 ps 命令的输出得到。-s 选项是给程序发送指定的信号，详细的信号可以用“kill -l”命令查看；-p 选项只显示指定进程的 ID 号。

```
[root@localhost ~]# kill -l [信号]
```

8.2 调度启动进程

有时候需要对系统进行一些比较费时而且占用资源的维护工作，这些工作适合在深夜进行，这时候用户就可以事先进行调度安排，指定任务运行的时间或者场合，到时候系统会自动完成这些任务。要使用自动启动进程的功能，就需要掌握以下几个启动命令。

8.2.1 定时运行一批程序（at）

at 命令：用户使用 at 命令在指定时刻执行指定的命令序列。该命令至少需要指定一个命令和一个执行时间。at 命令可以只指定时间，也可以时间和日期一起指定。

at 命令的语法格式如下：

```
[root@localhost ~]# at [-V] [-q 队列] [-f 文件名] [-mldbv] 时间
```

```
[root@localhost ~]# at -c 作业 [作业...]
```

8.2.2 周期性运行一批程序（cron）

前面介绍 at 命令都会在一定时间内完成一定任务，但是它只能执行一次。也就是说，当指定了运行命令后，系统在指定时间完成任务，以后就不再执行了。但是在很多情况下需要周期性重复执行一些命令，这时候就需要使用 cron 命令来完成任务。

运行机制：首先 `cron` 命令会搜索 `/var/spool/cron` 目录，寻找以 `/etc/passwd` 文件中的用户名命名的 `crontab` 文件，被找到的这种文件将装入内存。比如一个用户名为 `userexample` 的用户，对应的 `crontab` 文件应该是 `/var/spool/cron/userexample`，即以该用户命名的 `crontab` 文件存放在 `/var/spool/cron` 目录下面。

`cron` 命令还将搜索 `/etc/crontab` 文件，这个文件是用不同的格式写成的。`cron` 启动以后，它将首先检查是否有用户设置了 `crontab` 文件，如果没有就转入睡眠状态，释放系统资源。所以该后台进程占用资源极少，它每分钟被唤醒一次，查看当前是否有需要运行的命令。

命令执行结束后，任何输出都将作为邮件发送给 `crontab` 的所有者，或者是 `/etc/crontab` 文件中 `MAILTO` 环境变量中指定的用户。这是 `cron` 的工作原理，但是 `cron` 命令的执行不需要用户干涉，用户只需要修改 `crontab` 中要执行的命令。

crontab 命令：`crontab` 命令用于安装、删除或者显示用于驱动 `cron` 后台进程的表格。用户把需要执行的命令序列放到 `crontab` 文件中以获得执行，而且每个用户都可以有自己的 `crontab` 文件。

`crontab` 命令的常用方法如下：

1.`crontab -u` //设置某个用户的 `cron` 服务，`root` 用户在执行 `crontab` 时需要此参数。

2.`crontab -l` //列出某个用户 `cron` 服务的详细内容。

3.`crontab -r` //删除某个用户的 `cron` 服务。

4.`crontab -e` //编辑某个用户的 `cron` 服务。

例如 `root` 查看自己的 `cron` 设置。命令如下：

```
[root@localhost ~]# crontab -u root -l
```

9 管理硬盘

9.1 标准分区

标准分区可以包含文件系统或交换空间，也能提供一个容器，用于软件 RAID 和 LVM 物理卷。

在安装系统时，您可以手动选择分区的方式为标准分区，也可手动创建分区，如下图所示：



boot 分区：引导分区，包含了系统启动的必要内核文件，即使根分区损坏也能正常引导启动；

/boot/efi 分区：对于 GPT 分区表（UEFI 启动模式），efi 分区是必须的，它用来存放操作系统的引导器（loader）和启动操作系统所必需的引导文件和相关驱动程序；

swap 分区：类似于 Windows 的虚拟内存，在内存不够用时占用硬盘的虚拟内存来进行临时数据的存放，而对于 linux 就是 swap 分区；

/ 分区（根分区）：Linux 系统具有“一切皆文件”的思想和特点，所有的文件都从这里开始；

var 分区（可选）：用于 log 日志的文件的存放，如果不分则默认在/目录下；

home 分区（可选）：存放用户数据，HOME 的结构一般是 HOME/userName/userFile，如果不分则默认在/目录下；

backup 分区：若安装支持备份还原的系统，需要设置该分区。

查看系统分区情况：使用 lsblk 命令查看，如下：


```
[root@localhost ~]# lsblk
NAME                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sr0                  11:0    1   3.7G  0 rom  /run/media/root/KylinSec
vda                  252:0    0  100G  0 disk
├─vda1               252:1    0   600M  0 part /boot/efi
├─vda2               252:2    0    2G   0 part /boot
├─vda3               252:3    0  97.4G  0 part
│   ├─ko-root        253:0    0  37.6G  0 lvm  /
│   ├─ko-swap        253:1    0    4G   0 lvm  [SWAP]
│   └─ko-data        253:2    0  18.3G  0 lvm  /var
│                                   /usr/local
│                                   /root
│                                   /opt
│                                   /home
│                                   /data
└─ko-backup 253:3    0  37.6G  0 lvm
```

查看磁盘设备的具体信息，使用 `fdisk -l` 命令，如下：

```
[root@localhost ~]# fdisk -l
Disk /dev/vda: 100 GiB, 107374182400 字节, 209715200 个扇区
单元：扇区 / 1 * 512 = 512 字节
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节
磁盘标签类型：gpt
磁盘标识符：02544401-781E-4C3C-B5EC-E7B807D17B99

设备            起点      末尾      扇区  大小  类型
/dev/vda1       2048     1230847   1228800  600M  EFI 系统
/dev/vda2     1230848     5425151   4194304    2G  Linux 文件系统
/dev/vda3     5425152    209713151 204288000  97.4G  Linux LVM

Disk /dev/mapper/ko-root: 37.56 GiB, 40332427264 字节, 78774272 个扇区
单元：扇区 / 1 * 512 = 512 字节
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节

Disk /dev/mapper/ko-swap: 3.95 GiB, 4244635648 字节, 8290304 个扇区
单元：扇区 / 1 * 512 = 512 字节
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节

Disk /dev/mapper/ko-data: 18.34 GiB, 19688062976 字节, 38453248 个扇区
单元：扇区 / 1 * 512 = 512 字节
扇区大小(逻辑/物理)：512 字节 / 512 字节
I/O 大小(最小/最佳)：512 字节 / 512 字节

Disk /dev/mapper/ko-backup: 37.56 GiB, 40328232960 字节, 78766080 个扇区
```

对某个磁盘进行分区管理，使用 `fdisk` 设备名，如下：

```
[root@localhost ~]# fdisk /dev/vdb
```

输入 m 可获取帮助信息：

```
root@localhost:~  
文件(F) 编辑(E) 视图(V) 搜索(S) 终端(T) 帮助(H)  
命令(输入 m 获取帮助): m  
帮助:  
  
DOS (MBR)  
a  开关 可启动 标志  
b  编辑嵌套的 BSD 磁盘标签  
c  开关 dos 兼容性标志  
  
常规  
d  删除分区  
F  列出未分区的空闲区  
l  列出已知分区类型  
n  添加新分区  
p  打印分区表  
t  更改分区类型  
v  检查分区表  
i  打印某个分区的相关信息  
  
杂项  
m  打印此菜单  
u  更改 显示/记录 单位  
x  更多功能(仅限专业人员)  
  
脚本  
I  从 sfdisk 脚本文件加载磁盘布局  
O  将磁盘布局转储为 sfdisk 脚本文件  
  
保存并退出  
w  将分区表写入磁盘并退出  
q  退出而不保存更改  
  
新建空磁盘标签  
g  新建一份 GPT 分区表  
G  新建一份空 GPT (IRIX) 分区表  
o  新建一份的空 DOS 分区表
```

9.2 LVM 简介

LVM 是逻辑卷管理（Logical Volume Manager）的简称，它是 Linux 环境下对磁盘分区进行管理的一种机制。LVM 通过在硬盘和文件系统之间添加一个逻辑层，来为文件系统屏蔽下层硬盘分区布局，提高硬盘分区管理的灵活性，

使用 LVM 管理硬盘的基本过程如下：

- 1.将硬盘创建为物理卷
- 2.将多个物理卷组合成卷组
- 3.在卷组中创建逻辑卷
- 4.在逻辑卷之上创建文件系统

通过 LVM 管理硬盘之后，文件系统不再受限于硬盘的大小，可以分布在多个硬盘上，也可以动态扩容。

9.3 LVM 基本概念

物理存储介质（The physical media）：指系统的物理存储设备，如硬盘，系统中为/dev/hda、/dev/sda 等等，是存储系统最低层的存储单元。

物理卷（Physical Volume, PV）：指硬盘分区或从逻辑上与磁盘分区具有同样功能的设备(如 RAID)，是 LVM 的基本存储逻辑块。物理卷包括一个特殊的标签，该标签默认存放在第二个 512 字节扇区，但也可以将标签放在最开始的四个扇区之一。该标签包含物理卷的随机唯一识别符（UUID），记录块设备的大小和 LVM 元数据在设备中的存储位置。

卷组（Volume Group, VG）：由物理卷组成，屏蔽了底层物理卷细节。可在卷组上创建一个或多个逻辑卷且不用考虑具体的物理卷信息。

逻辑卷（Logical Volume, LV）：卷组不能直接用，需要划分成逻辑卷才能使用。逻辑卷可以格式化成不同的文件系统，挂载后直接使用。

物理块（Physical Extent, PE）：物理卷以大小相等的“块”为单位存储，块的大小与卷组中逻辑卷块的大小相同。

逻辑块（Logical Extent, LE）：逻辑卷以“块”为单位存储，在一卷组中的所有逻辑卷的块大小是相同的。

9.4 管理物理卷

通过 `rpm -qa | grep lvm2` 命令查询，若打印信息中包含“lvm2”信息，则表示已安装 LVM，；若无任何打印信息，则表示未安装，可通过 yum 进行安装。

9.4.1 创建物理卷

可在 root 权限下通过 `pvcreate` 命令创建物理卷。

```
[root@localhost ~]# pvcreate [option] devname ...
```

其中：

option：命令参数选项。常用的参数选项有：

-f：强制创建物理卷，不需要用户确认。

-u：指定设备的 UUID。

-y: 所有的问题都回答“yes”。

devname: 指定要创建的物理卷对应的设备名称，如果需要批量创建，可以填写多个设备名称，中间以空格间隔。

示例 1: 将/dev/sdb、/dev/sdc 创建为物理卷。

```
[root@localhost ~]# pvcreate /dev/sdb /dev/sdc
```

9.4.2 查看物理卷

可在 root 权限通过 pvdisplay 命令查看物理卷的信息，包括：物理卷名称、所属的卷组、物理卷大小、PE 大小、总 PE 数、可用 PE 数、已分配的 PE 数和 UUID。

```
[root@localhost ~]# pvdisplay [option] devname
```

其中：

option: 命令参数选项。常用的参数选项有：

-s: 以短格式输出。

-m: 显示 PE 到 LE 的映射。

devname: 指定要查看的物理卷对应的设备名称。如果不指定物理卷名称，则显示所有物理卷的信息。

示例：显示物理卷/dev/sdb 的基本信息。

```
[root@localhost ~]# pvdisplay /dev/sdb
```

9.4.3 修改物理卷属性

可在 root 权限下通过 pvchange 命令修改物理卷的属性。

```
[root@localhost ~]# pvchange [option] pvname ...
```

其中：

option: 命令参数选项。常用的参数选项有：

-u: 生成新的 UUID。

-x: 是否允许分配 PE”。

pvname: 指定要修改属性的物理卷对应的设备名称，如果需要批量修改，可以填写多个设备名称，中间以空格间隔。

示例：禁止分配/dev/sdb 物理卷上的 PE。

```
[root@localhost ~]# pvchange -x n /dev/sdb
```

9.4.4 删除物理卷

可在 root 权限下通过 `pvremove` 命令删除物理卷。

```
[root@localhost ~]# pvremove [option] pvname ...
```

其中：

option: 命令参数选项。常用的参数选项有：

-f: 强制删除物理卷，不需要用户确认。

-y: 所有的问题都回答“yes”。

pvname: 指定要删除的物理卷对应的设备名称，如果需要批量删除，可以填写多个设备名称，中间以空格间隔。

示例：删除物理卷/dev/sdb。

```
[root@localhost ~]# pvremove /dev/sdb
```

9.5 管理卷组

9.5.1 创建卷组

可在 root 权限下通过 `vgcreate` 命令创建卷组。

```
[root@localhost ~]# vgcreate [option] vgname pvname ...
```

其中：

option: 命令参数选项。常用的参数选项有：

-l: 卷组上允许创建的最大逻辑卷数。

-p: 卷组中允许添加的最大物理卷数。

-s: 卷组上的物理卷的 PE 大小。

vgname: 要创建的卷组名称。

pvname: 要加入到卷组中的物理卷名称。

示例：创建卷组 `vg1`，并且将物理卷/dev/sdb 和/dev/sdc 添加到卷组中。

```
[root@localhost ~]# vgcreate vg1 /dev/sdb /dev/sdc
```

9.5.2 查看卷组

可在 root 权限下通过 `vgdisplay` 命令查看卷组的信息。

```
[root@localhost ~]# vgdisplay [option] [vgname] []
```

其中：

option: 命令参数选项。常用的参数选项有：

-s: 以短格式输出。

-A: 仅显示活动卷组的属性。

vgname: 指定要查看的卷组名称。如果不指定卷组名称，则显示所有卷组的信息。

示例：显示卷组 **vg1** 的基本信息。

```
[root@localhost ~]# vgdisplay vg1 []
```

9.5.3 修改卷组属性

可在 **root** 权限下通过 **vgchange** 命令修改卷组的属性。

```
[root@localhost ~]# vgchange [option] vgname []
```

其中：

option: 命令参数选项。常用的参数选项有：

-a: 设置卷组的活动状态。

vgname: 指定要修改属性的卷组名称。

示例：将卷组 **vg1** 状态修改为活动。

```
[root@localhost ~]# vgchange -ay vg1 []
```

9.5.4 扩展卷组

可在 **root** 权限下通过 **vgextend** 命令动态扩展卷组。它通过向卷组中添加物理卷来增加卷组的容量。

```
[root@localhost ~]# vgextend [option] vgname pvname ... []
```

其中：

option: 命令参数选项。常用的参数选项有：

-d: 调试模式。

-t: 仅测试。

vgname: 要扩展容量的卷组名称。

pvname: 要加入到卷组中的物理卷名称。

示例：将卷组 vg1 中添加物理卷/dev/sdb。

```
[root@localhost ~]# vgextend vg1 /dev/sdb
```

9.5.5 收缩卷组

可在 root 权限下通过 `vgreduce` 命令删除卷组中的物理卷来减少卷组容量。不能删除卷组中剩余的最后一个物理卷。

```
[root@localhost ~]# vgreduce [option] vgname pvname ...
```

其中：

option: 命令参数选项。常用的参数选项有：

-a: 如果命令行中没有指定要删除的物理卷，则删除所有的空物理卷。

--removemissing: 删除卷组中丢失的物理卷，使卷组恢复正常状态。

vgname: 要收缩容量的卷组名称。

pvname: 要从卷组中删除的物理卷名称。

示例：从卷组 vg1 中移除物理卷/dev/sdb2。

```
[root@localhost ~]# vgreduce vg1 /dev/sdb2
```

9.5.6 删除卷组

可在 root 权限下通过 `vgremove` 命令删除卷组。

```
[root@localhost ~]# vgremove [option] vgname
```

其中：

option: 命令参数选项。常用的参数选项有：

-f: 强制删除卷组，不需要用户确认。

vgname: 指定要删除的卷组名称。

示例：删除卷组 vg1。

```
[root@localhost ~]# vgremove vg1
```

9.6 管理逻辑卷

9.6.1 创建逻辑卷

可在 root 权限下通过 `lvcreate` 命令创建逻辑卷。

```
[root@localhost ~]# lvcreate [option] vgname
```

其中：

option：命令参数选项。常用的参数选项有：

-L：指定逻辑卷的大小，单位为“kKmMgGtT”字节。

-l：指定逻辑卷的大小（LE 数）。

-n：指定要创建的逻辑卷名称。

-s：创建快照。

vgname：要创建逻辑卷的卷组名称。

示例 1：在卷组 vg1 中创建 10G 大小的逻辑卷。

```
[root@localhost ~]# lvcreate -L 10G vg1
```

9.6.2 查看逻辑卷

可在 root 权限下通过 `lvdisplay` 命令查看逻辑卷的信息，包括逻辑卷空间大小、读写状态和快照信息等属性。

```
[root@localhost ~]# lvdisplay [option] [lvname]
```

其中：

option：命令参数选项。常用的参数选项有：

-v：显示 LE 到 PE 的映射

lvname：指定要显示属性的逻辑卷对应的设备文件。如果省略，则显示所有的逻辑卷属性。

示例：显示逻辑卷 lv1 的基本信息。

```
[root@localhost ~]# lvdisplay /dev/vg1/lv1
```

9.6.3 调整逻辑卷大小

可在 root 权限下通过 `lvresize` 命令调整 LVM 逻辑卷的空间大小，可以增大空间和缩小空间。使用 `lvresize` 命令调整逻辑卷空间大小和缩小空间时需要谨慎，因为有可能导致数据丢失。

```
[root@localhost ~]# lvresize [option] vgname
```

其中：

option：命令参数选项。常用的参数选项有：

-L：指定逻辑卷的大小，单位为“kKmMgGtT”字节。

-l: 指定逻辑卷的大小（LE 数）。

-f: 强制调整逻辑卷大小，不需要用户确认。

lvname: 指定要调整的逻辑卷名称。

示例 1: 为逻辑卷/dev/vg1/lv1 增加 200M 空间。

```
[root@localhost ~]# lvresize -L +200 /dev/vg1/lv1
```

9.6.4 扩展逻辑卷

可在 root 权限下通过 `lvextend` 命令动态在线扩展逻辑卷的空间大小，而不中断应用程序对逻辑卷的访问。

```
[root@localhost ~]# lvextend [option] lvname
```

其中：

option: 命令参数选项。常用的参数选项有：

-L: 指定逻辑卷的大小，单位为“kKmMgGtT”字节。

-l: 指定逻辑卷的大小（LE 数）。

-f: 强制调整逻辑卷大小，不需要用户确认。

lvname: 指定要扩展空间的逻辑卷的设备文件。

示例：为逻辑卷/dev/vg1/lv1 增加 100M 空间。

```
[root@localhost ~]# lvextend -L +100M /dev/vg1/lv1
```

9.6.5 收缩逻辑卷

可在 root 权限下通过 `lvreduce` 命令减少逻辑卷占用的空间大小。使用 `lvreduce` 命令收缩逻辑卷的空间大小有可能会删除逻辑卷上已有的数据，所以在操作前必须进行确认。

```
[root@localhost ~]# lvreduce [option] lvname
```

其中：

option: 命令参数选项。常用的参数选项有：

-L: 指定逻辑卷的大小，单位为“kKmMgGtT”字节。

-l: 指定逻辑卷的大小（LE 数）。

-f: 强制调整逻辑卷大小，不需要用户确认。

lvname: 指定要扩展空间的逻辑卷的设备文件。

示例：将逻辑卷/dev/vg1/lv1 的空间减少 100M。

```
[root@localhost ~]# lvreduce -L -100M /dev/vg1/lv1
```

9.6.6 删除逻辑卷

可在 root 权限下通过 `lvremove` 命令删除逻辑卷。如果逻辑卷已经使用 `mount` 命令加载，则不能使用 `lvremove` 命令删除。必须使用 `umount` 命令卸载后，逻辑卷方可被删除。

```
[root@localhost ~]# lvremove [option] vgname
```

其中：

option: 命令参数选项。常用的参数选项有：

-f: 强制删除逻辑卷，不需要用户确认。

vgname: 指定要删除的逻辑卷。

示例：删除逻辑卷/dev/vg1/lv1。

```
[root@localhost ~]# lvremove /dev/vg1/lv1
```

9.7 创建并挂载文件系统

在创建完逻辑卷之后，需要在逻辑卷之上创建文件系统并挂载文件系统到相应目录下。

9.7.1 创建文件系统

可在 root 权限下通过 `mkfs` 命令创建文件系统。

```
[root@localhost ~]# mkfs [option] lvname
```

其中：

option: 命令参数选项。常用的参数选项有：

-t: 指定创建的 linux 系统类型，如 `ext2`，`ext3`，`ext4` 等等，默认类型为 `ext2`。

lvname: 指定要创建的文件系统对应的逻辑卷设备文件名。

示例：在逻辑卷/dev/vg1/lv1 上创建 `ext4` 文件系统。

```
[root@localhost ~]# mkfs -t ext4 /dev/vg1/lv1
```

9.7.2 手动挂载文件系统

手动挂载的文件系统仅在当时有效，一旦操作系统重启则会不存在。

可在 root 权限下通过 mount 命令挂载文件系统。

```
[root@localhost ~]# mount lvname mntpath
```

其中：

lvname：指定要挂载文件系统的逻辑卷设备文件名。

mntpath：挂载路径。

示例：将逻辑卷/dev/vg1/lv1 挂载到/mnt/data 目录。

```
[root@localhost ~]# mount /dev/vg1/lv1 /mnt/data
```

9.7.3 自动挂载文件系统

手动挂载的文件系统在操作系统重启之后会不存在，需要重新手动挂载文件系统。但若在手动挂载文件系统后在 root 权限下进行如下设置，可以实现操作系统重启后文件系统自动挂载文件系统。

1. 执行 blkid 命令查询逻辑卷的 UUID，逻辑卷以/dev/vg1/lv1 为例。

```
[root@localhost ~]# blkid /dev/vg1/lv1
```

查看打印信息，打印信息中包含如下内容，其中 uuidnumber 是一串数字，为 UUID，fstype 为文件系统。

/dev/vg1/lv1: UUID=" uuidnumber " TYPE=" fstype "

2. 执行 vi /etc/fstab 命令编辑 fstab 文件，并在最后加上如下内容。

```
UUID=uuidnumber mntpath fstype defaults 0 0
```

内容说明如下：

第一列：UUID，此处填写 1 查询的 uuidnumber。

第二列：文件系统的挂载目录 mntpath。

第三列：文件的文件格式，此处填写 1 查询的 fstype。

第四列：挂载选项，此处以“defaults”为例；

第五列：备份选项，设置为“1”时，系统自动对该文件系统进行备份；设置为“0”时，不进行备份。此处以“0”为例；

第六列：扫描选项，设置为“1”时，系统在启动时自动对该文件系统进行扫描；设置为“0”时，不进行扫描。此处以“0”为例。

- 3.验证自动挂载功能。

1) 执行 `umount` 命令卸载文件系统，逻辑卷以 `/dev/vg1/lv1` 为例。

```
[root@localhost ~]# umount /dev/vg1/lv1
```

2) 执行如下命令，将 `/etc/fstab` 文件所有内容重新加载。

```
[root@localhost ~]# mount -a
```

3) 执行如下命令，查询文件系统挂载信息，挂载目录以 `/mnt/data` 为例。

```
[root@localhost ~]# mount | grep /mnt/data
```

查看打印信息，若信息中包含如下信息表示自动挂载功能生效。

```
/dev/vg1/lv1 on /mnt/data
```

10 虚拟化

系统虚拟化可通过 `Virt-manager` (Virtual Machine Manager) 实现，`Virt-manager` 是一个轻量级应用程序套件，提供管理虚拟机的命令行或图形用户界面 (GUI) 对虚拟机进行创建、修改及删除、网络连接等管理。除了提供对虚拟机的管理功能外，`virt-manager` 还通过一个嵌入式虚拟网络计算 (VNC) 客户端查看器为 Guest 虚拟机提供一个完整图形控制台。

10.1 环境搭建

首先需要配置好系统源，确认网络正常。在终端输入如下命令：

```
yum makecache
```

```
yum install virt-manager -y
```

```
yum install qemu -y
```

```
yum install libvirt -y
```

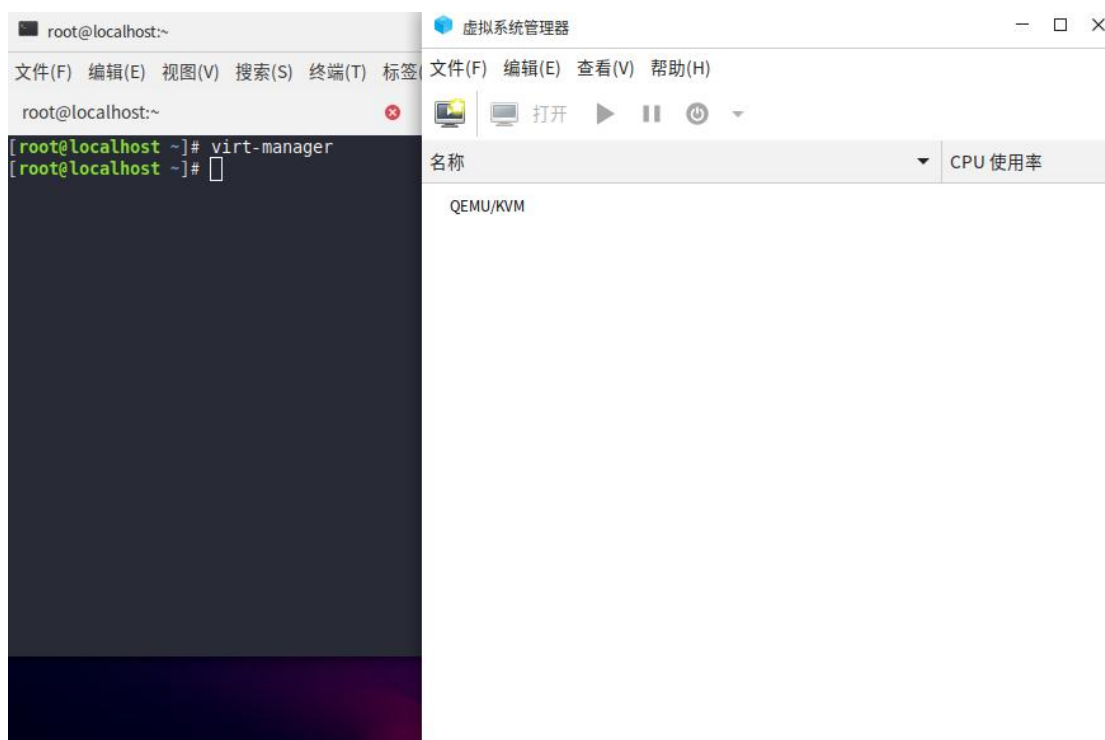
```
yum install edk2-aarch64 (aarch 机器)
```

```
systemctl start libvirtd
```

```
[root@localhost grub2]# systemctl start libvirtd
[root@localhost grub2]# systemctl status libvirtd
● libvirtd.service - Virtualization daemon
   Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2023-08-02 15:55:44 CST; 7s ago
     TriggeredBy: ● libvirtd-ro.socket
                  ● libvirtd.socket
                  ● libvirtd-admin.socket
   Docs: man:libvirtd(8)
         https://libvirt.org
   Main PID: 540371 (libvirtd)
     Tasks: 19 (limit: 32768)
    Memory: 17.4M
   CGroup: /system.slice/libvirtd.service
           └─ 540371 /usr/sbin/libvirtd --timeout 120
              └─ 540484 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf --leasefile-ro --dhcp-script=/usr>
                 └─ 540485 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf --leasefile-ro --dhcp-script=/usr>

8月 02 15:55:43 localhost.localdomain systemd[1]: Starting Virtualization daemon...
8月 02 15:55:44 localhost.localdomain systemd[1]: Started Virtualization daemon.
8月 02 15:55:45 localhost.localdomain dnsmasq[540484]: started, version 2.86 cachesize 150
8月 02 15:55:45 localhost.localdomain dnsmasq[540484]: compile time options: IPv6 GNU-getopt DBus no-UBus no-i18n IDN2 DHCP>
8月 02 15:55:45 localhost.localdomain dnsmasq-dhcp[540484]: DHCP, IP range 192.168.122.2 -- 192.168.122.254, lease time 1h
8月 02 15:55:45 localhost.localdomain dnsmasq-dhcp[540484]: DHCP, sockets bound exclusively to interface virbr0
8月 02 15:55:45 localhost.localdomain dnsmasq[540484]: no servers found in /etc/resolv.conf, will retry
8月 02 15:55:45 localhost.localdomain dnsmasq[540484]: read /etc/hosts - 2 addresses
8月 02 15:55:45 localhost.localdomain dnsmasq[540484]: read /var/lib/libvirt/dnsmasq/default.addnhosts - 0 addresses
8月 02 15:55:45 localhost.localdomain dnsmasq-dhcp[540484]: read /var/lib/libvirt/dnsmasq/default.hostsfile
[root@localhost grub2]#
```

完成后终端输入 `virt-manager`，回车，桌面弹出虚拟系统管理器界面，如图所示：

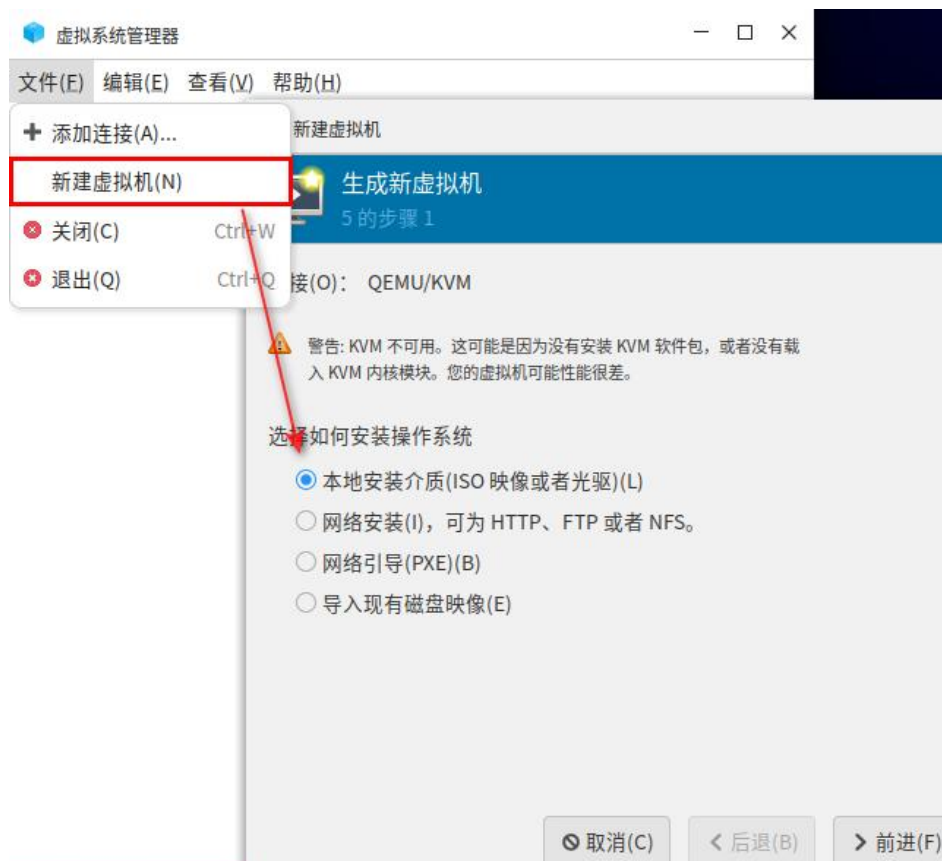


注意：申微架构在 `/boot/grub2/grub.cfg` 的启动项中必须添加保留内存的内核参数，否则 `qemu` 启动时将无法分配内存，并报错 “Cannot allocate memory”。所以在启动时需要在 `grub` 界面按 “e” 编辑，在 `linux` 行添加 `kvm_mem=nn[MG]@start[MG]`，其中 `nn[MG]` 是虚拟机预留的内存大小，`start[MG]`

是预留内存的起始地址，如：`kvm_mem=32G@16G`，添加完成后再进入系统进行 `virt-manager` 虚拟化配置操作。

10.2 配置流程

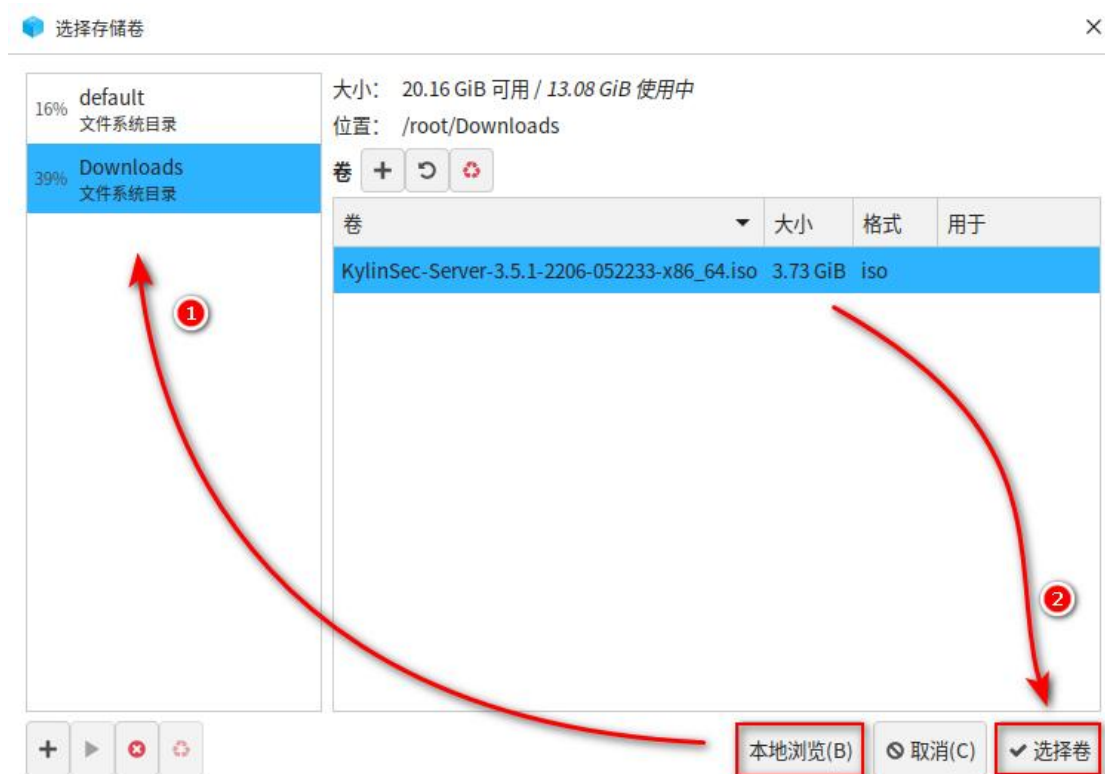
点击文件——新建虚拟机——本地安装介质，如图所示：



点击“前进”后进入步骤 2，此时需要“选择 ISO 或 CDROM 安装介质”，如下图所示：

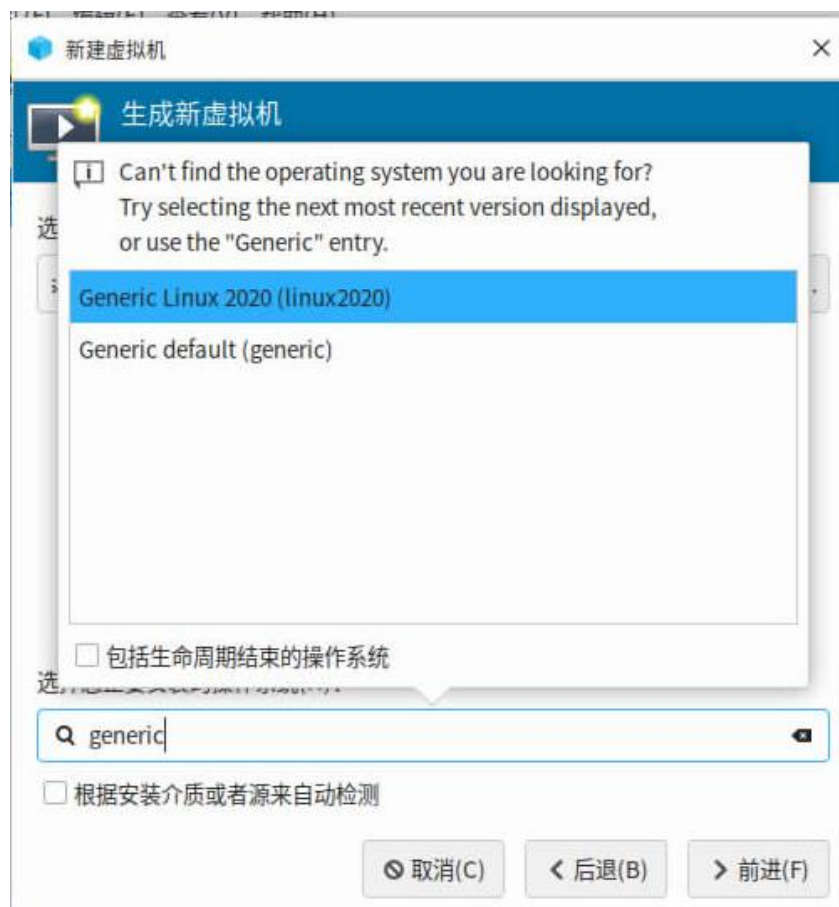


点击“浏览”进入到介质选择界面，如下图所示：



在该界面中，选择下方的<本地浏览>，打开文件浏览器，根据使用者自身镜像放置位置选择路径并确认回到该界面。

需要选择安装的操作系统，取消下方的自动检测勾选项，在搜索输入框输入关键字“Generic”-》在弹出的选项中选择 Generic Linux 2020，如下图所示：



点击“前进”到下一步，设置内存和 CPU，建议内存大于等于 4G，CPU 大于等于 2，如下图所示：



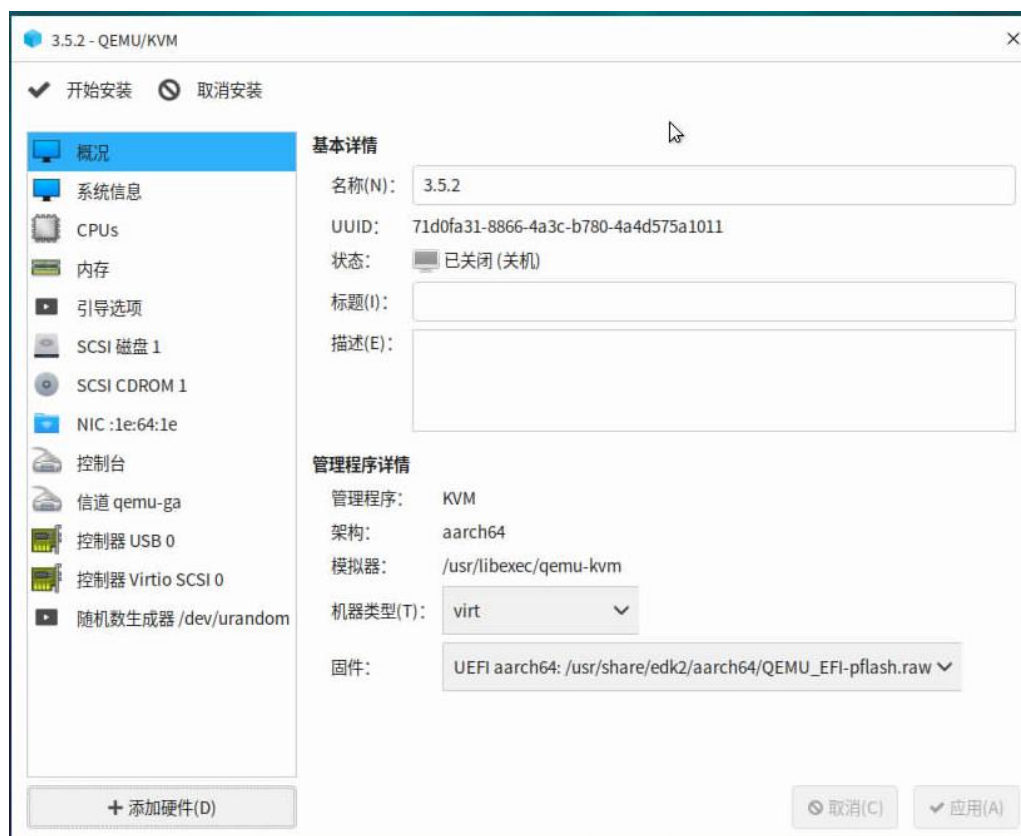
设置完成后点击“前进”到下一步，如下图所示：



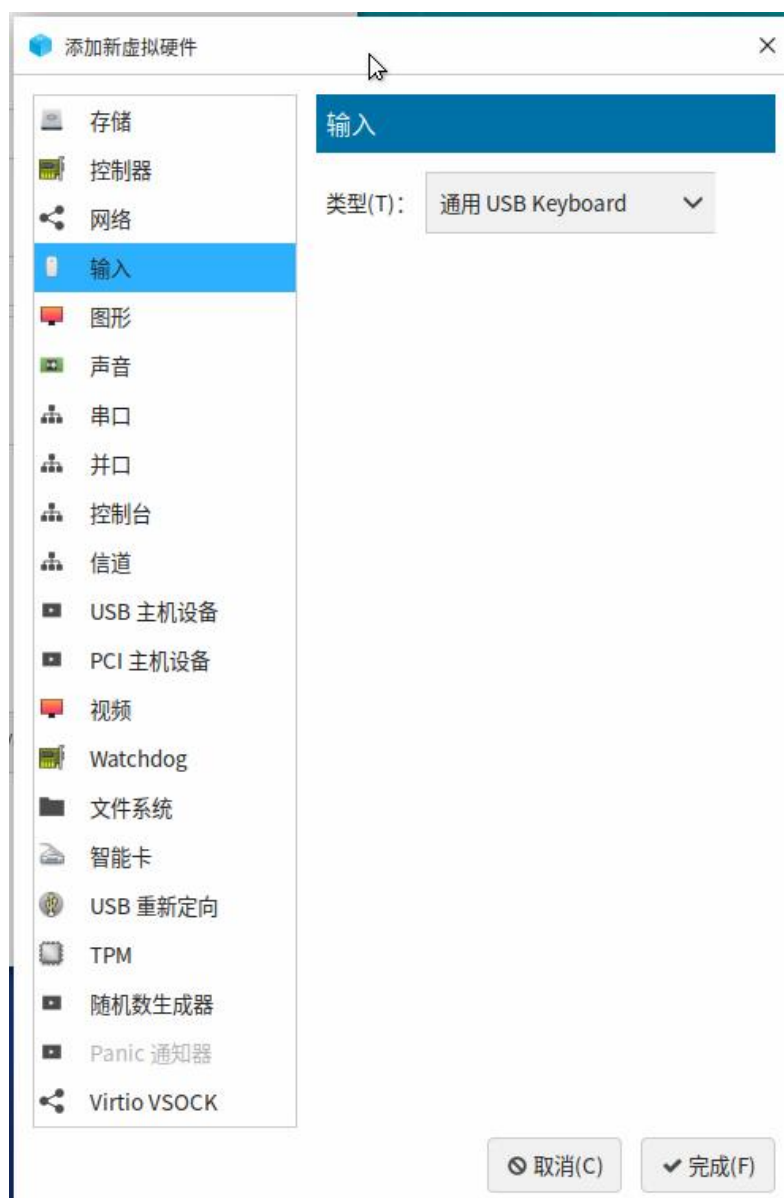
磁盘大小默认是 25G，可以保持默认，也可以自定义大小，设置完成后点击“前进”到下一步，如下图所示：



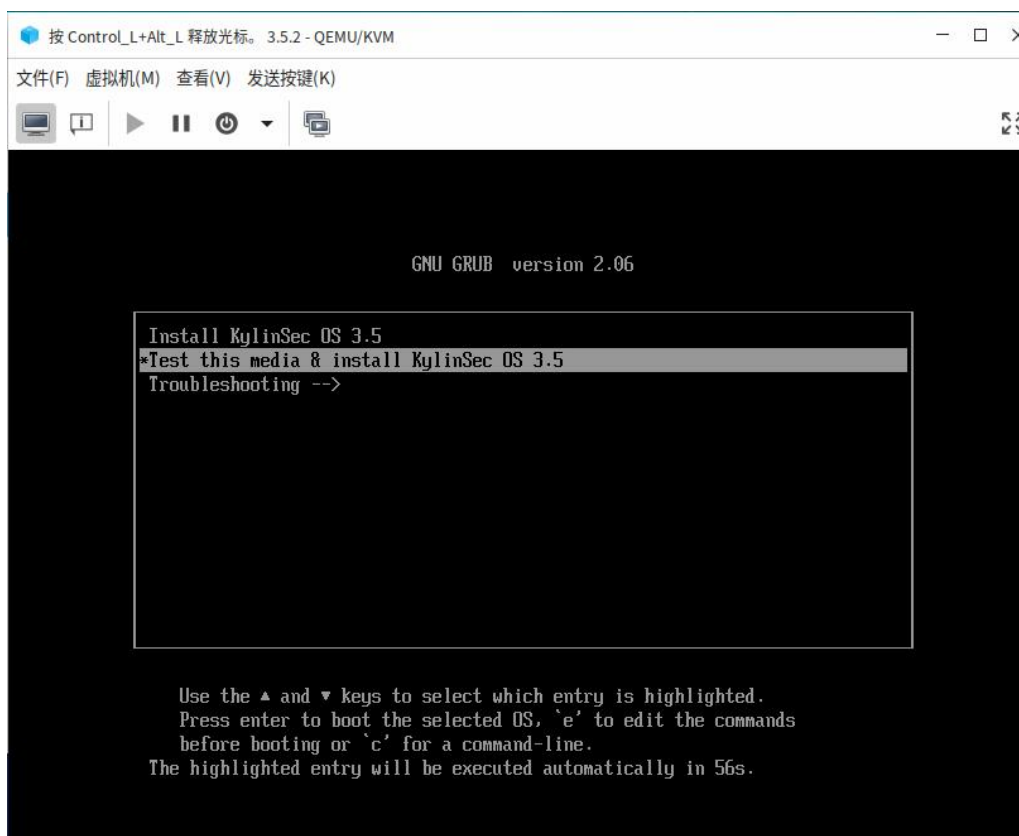
在名称输入框输入需要修改的名称，勾选“在安装前自定义配置”，点击选择网络，然后点击“完成”按钮，弹出配置界面如下图：



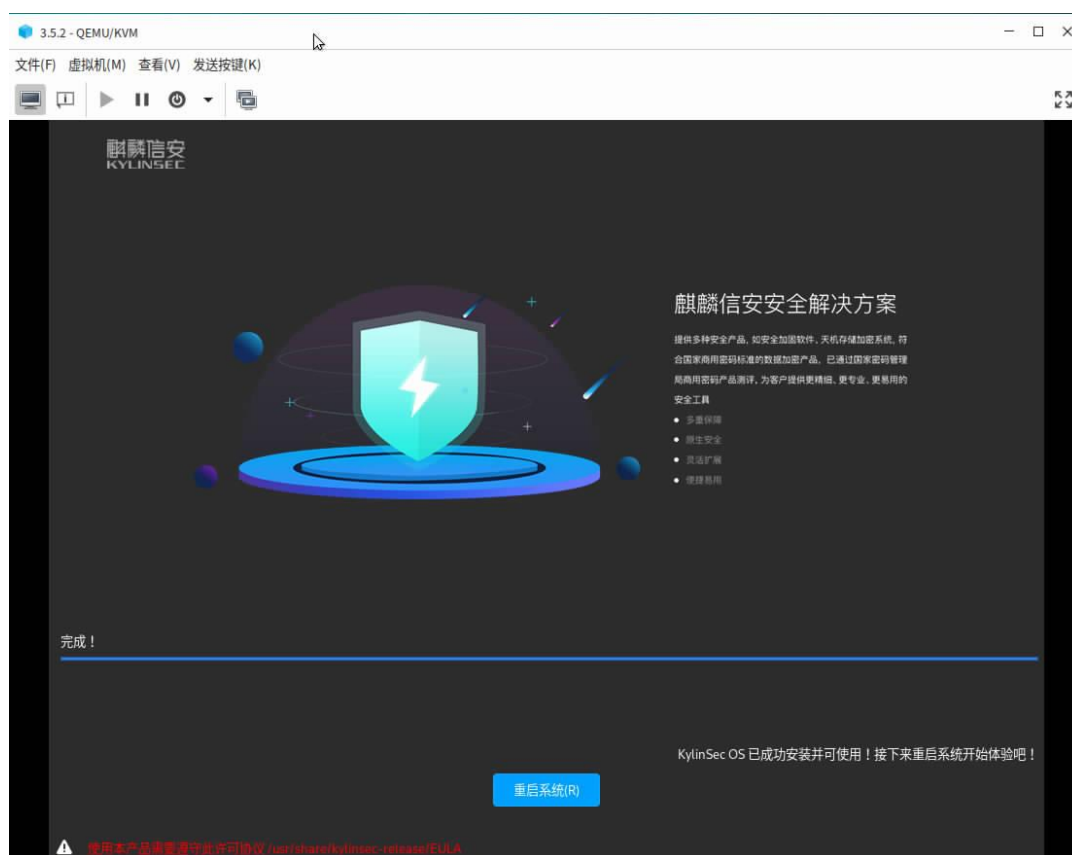
点击“添加硬件”，分别添加输入（USB Keyboard）、输入（USB Mouse）、图形（Spice 服务器）、视频（Virtio），添加完成后点击“开始安装”：



进入系统安装界面，如下图所示，选择“Install KylinSec OS 3.5”并回车后，开始进行常规安装；



进行常规安装前的配置后，开始进行安装，安装完成如下图所示：



点击“重启系统”即可重启进入。

11 FAQ

1. 问题描述：Nvidia 显卡如何适配？

解决方法：

1) 从 Nvidia 官网下载和显卡型号对应的显卡驱动，比如 NVIDIA-Linux-x86_64-xxx.**.run;

2) 禁止 nouveau 公版驱动的加载，修改/boot/grub2/grub.cfg 文件，添加 rd.driver.blacklist=nouveau nouveau.modeset=0 启动项参数；修改 /usr/lib/modprobe.d/dist-blacklist.conf 文件，如下所示：

```
#blacklist nvidiafb
```

```
blacklist nouveau
```

3) 安装显卡驱动依赖，yum install kernel-devel kernel-headers gcc gcc-c++（注意安装软件时 kb 号等小版本号的对应）；

4) 重启计算机，使用集成显卡进入系统，执行 init 3 切换到文字界面；

5) chmod +x NVIDIA-Linux-x86_64-xxx.**.run;

6) 执行 nvidia 驱动安装脚本./NVIDIA-Linux-x86_64-XXX.**.run;

7) 执行 X -configure 生成新的 xorg.conf 配置文件，默认会位于 /root/xorg.conf.new;

8) cp /etc/X11/xorg.conf /etc/X11/xorg.conf.bak;

9) cp /root/xorg.conf.new /etc/X11/xorg.conf;

10) 修改/etc/X11/xorg.conf 文件，将 Section "Device"部分的 Driver 值修改为 nvidia，将 BusID 的值修改为独立显卡的 BusID 值（通过 lspci 获取）；

11) 重新启动计算机。

重新启动计算机后，系统能够进入图形界面，使用 lspci -nvk 可以查看独立显卡使用的驱动为 nvidia。

2. 问题描述：AMD 显卡驱动如何适配？

解决方法：

1) 首先安装 gcc、kernel-devel、kernel-headers

2) 解压获取的驱动压缩包文件，修改 amdgpu-instll 脚本：148 行 rhel|centos

后加 UniKylin（注意不是空格后加，是|后加），保存退出

手动添加源：vim /etc/yum.repos.d/amd.repo

```
[wine]
```

```
name=amd
```

```
gpgcheck=0
```

```
baseurl=file:///root/amdgpu-pro-20.10-1048554-rhel-8.1
```

注：file 后路径为压缩包解压后的路径

3) 执行命令 yum clean all 和 yum makechcke

cd 进入解压目录，运行脚本：./amdgpcc-install -y

4) 终端运行：dracut -f

5) 重启电脑（注意把显示器接口插到显卡上）。

6) lspci -k 查看是否安装使用成功：VGA 行 use 的是 AMDGPU 则安装成功

3. 问题描述：HBA 卡如何适配？

解决方法：HBA 卡适配需要先执行 lspci -nvk 查看需要适配的机器硬件信息，比如，可以看到 Marvell QLE2742 HBA 卡的 VID/PID 是“1077: 2261”，并从 HBA 卡硬件提供商下载驱动。如果不知道 HBA 卡的型号，可以使用搜索引擎搜索 HBA 卡的 VID/PID（比如 1077: 2261）来获取 HBA 卡的型号。在麒麟信安系统上编译驱动,优先编译源代码包，其次选择源代码，以编译 Marvell QLE2742 HBA 卡为例，可用的驱动版本为 qla2xxx-v8.08-1。

1) 编译得到 qla2xxx-v8.08-1.ky3.x86_64.rpm。

2) 解压缩 qla2xxx-v8.08-1.ky3.x86_64.rpm，得到 qla2xxx.ko 文件

3) 执行 modinfo qla2xxx.ko 可以查看驱动信息；

4) 执行 rpm -ivh qla2xxx-v8.08-1.ky3.x86_64.rpm 安装驱动包

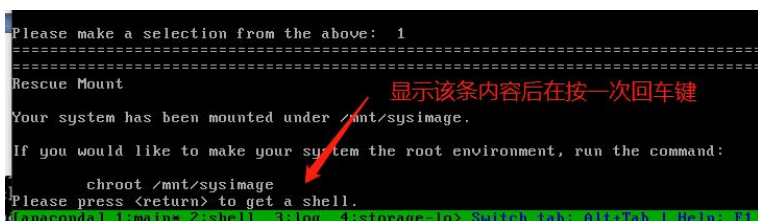
4. 问题描述：如果要强制使用 gpt 分区，该怎么操作？

解决方法：安装时按下 e 键进入编辑模式，在启动参数中加上” inst.gpt”

5. 问题描述：grub 损坏后如何修复？

解决方法:

- 1) 插入刻录好镜像的 U 盘或光盘, 选择从 U 盘或光盘启动, 进入到安装选择界面后选择 “Troubleshooting” -》 “Resure a KylinSec system” ;
- 2) 进入修复界面后按 “1” 继续, 出现 Please press to get a shell, 再按下回车:

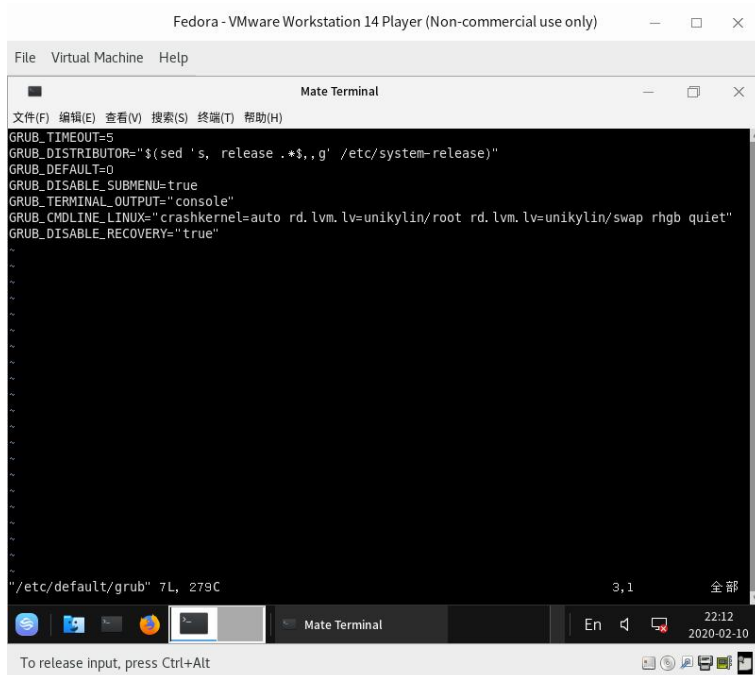


- 4) 输入 `chroot /mnt/sysimage` 命令
- 5) 输入 `grub2-install /dev/sda` 重装 grub
- 6) 安装成功后重启机器。

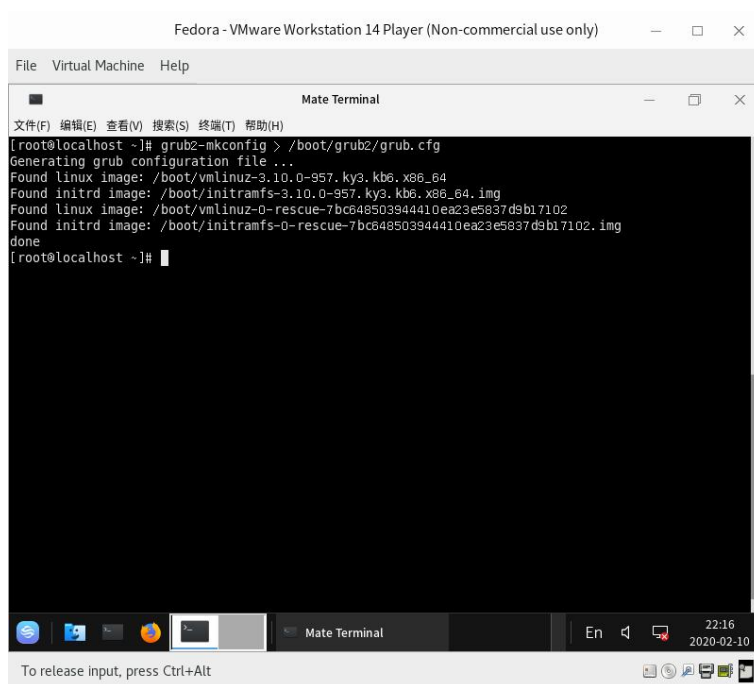
6. 问题描述: 如何通过 grub 指定启动系统和内核

解决方法:

修改/etc/default/grub 文件, 通过修改配置文件里的 GRUB_DEFAULT=0 可以指定要启动的系统, GRUB_DEFAULT=0 为启动第一个系统



再用 `grub2-mkconfig > /boot/grub2/grub.cfg` 指令将更改写入 grub:



7. 问题描述：如何设置系统密码复杂度？

解决方法：

1) 首先查看系统默认配置方案：`cat /etc/pam.d/system-auth`，其中 `password requisite` 字段为默认配置信息，如下图所示：

```
root@localhost:~  
文件(F) 编辑(E) 视图(V) 搜索(S) 终端(T) 帮助(H)  
%PAM-1.0  
# User changes will be destroyed the next time authconfig is run.  
auth      required      pam_env.so  
auth      requisite     pam_faillock.so preauth audit deny=3 even_deny_root unlock_time=60  
-auth     sufficient    pam_fprintd.so  
auth      sufficient    pam_unix.so nullok try_first_pass  
-auth     sufficient    pam_sss.so use_first_pass  
auth      [default=die] pam_faillock.so authfail audit deny=3 even_deny_root unlock_time=60  
auth      sufficient    pam_faillock.so authsucc audit deny=3 even_deny_root unlock_time=60  
auth      requisite     pam_succeed_if.so uid >= 1000 quiet_success  
auth      required      pam_deny.so  
  
account    required      pam_unix.so  
account    required      pam_faillock.so  
account    sufficient    pam_localuser.so  
account    sufficient    pam_succeed_if.so uid < 1000 quiet  
-account   [default=bad success=ok user_unknown=ignore] pam_sss.so  
account    required      pam_permit.so  
  
password   requisite     pam_pwquality.so try_first_pass local_users_only  
password   sufficient    pam_unix.so sha512 shadow nullok try_first_pass use_authtok  
-password  sufficient    pam_sss.so use_authtok  
password   required      pam_deny.so  
  
session    optional     pam_keyinit.so revoke  
session    required    pam_limits.so  
-session   optional     pam_systemd.so  
session    [success=1 default=ignore] pam_succeed_if.so service in crond quiet use_uid  
session    required    pam_unix.so  
-session   optional     pam_sss.so
```

2) 在修改配置文件前先备份，防止错误编辑：

```
cp /etc/pam.d/system-auth /etc/pam.d/system-auth.bak
```

3) 编辑配置文件：vim /etc/pam.d/system-auth

如 将 password requisite pam_pwquality.so try_first_pass
local_users_only 更改为：

```
password requisite pam_pwquality.so try_first_pass minlen=8 difok=5  
dcredit=-1 lcredit=-1 ocredit=-1 retry=1 type=
```

wq 保存配置文件

备注：

try_first_pass 而当 pam_unix 验证模块与 password 验证类型一起使用时，该选项主要用来防止用户新设定的密码与以前的旧密码相同。

minlen=8：最小长度 8 位

difok=5：新、旧密码最少 5 个字符不同

dcredit=-1：最少 1 个数字

lcredit=-1：最少 1 个小写字符，（ucredit=-1：最少 1 个大写字符）

ocredit=-1: 最少 1 个特殊字符

retry=1:1 次错误后返回错误信息

type=xxx: 此选项用来修改缺省的密码提示文本

8. 问题描述：现场机器码每次重启都发生变化问题该如何解决？

目前现场客户遇见最多的问题为机器码每次开机都发生变化，导致系统激活信息失效，机器码发生变化通常是多网卡机器引起的，因为操作系统计算机器码时绑定了机器的主板 UUID、首个网卡 mac 地址，由于有些机器有多个网卡，会导致每次启动时的 mac 地址顺序不一致，而使机器码发生变化。

解决方案：

- 1) 如系统能正常启动进入，则直接进入系统，更新系统内的 kylin-register\kyverify 的 rpm 包到最新版本。
- 2) 更新完包之后，执行 dracut -f.重启系统即可解决问题。
- 3) 如系统启动时卡在机器码验证不一致的界面，而不能进入系统，则需要使用系统光盘，进入修复模式。更新系统内的 kylin-register\kyverify 的 rpm 包到最新版本。更新完包之后，执行 dracut -f.重启系统即可解决问题。

9. 问题描述：查看日志文件中有 fail 报错信息，如下图所示：

```
[root@localhost Desktop]# cat /var/log/messages | grep fail
Apr 23 15:00:09 localhost alsactl[1024]: /usr/sbin/alsactl: do_nice:165sched_setparam failed: No such file or directory
Apr 23 15:00:13 localhost avahi-daemon[1053]: chroot.c: open() failed: No such file or directory
Apr 23 15:00:26 localhost journal[1572]: gtk_container_add: assertion 'GTK_IS_WIDGET (widget)' failed
Apr 23 15:00:26 localhost journal[1572]: gtk_widget_show: assertion 'GTK_IS_WIDGET (widget)' failed
Apr 23 15:00:26 localhost journal[1572]: gtk_container_add: assertion 'GTK_IS_WIDGET (widget)' failed
Apr 23 15:00:26 localhost journal[1572]: gtk_widget_show: assertion 'GTK_IS_WIDGET (widget)' failed
Apr 23 15:00:26 localhost journal[1572]: gtk_container_add: assertion 'GTK_IS_WIDGET (widget)' failed
Apr 23 15:00:26 localhost journal[1572]: gtk_widget_show: assertion 'GTK_IS_WIDGET (widget)' failed
Apr 23 15:00:27 localhost journal[1572]: g_dbus_proxy_new: assertion 'G_IS_DBUS_CONNECTION (connection)' failed
Apr 23 15:00:27 localhost journal[1572]: g_dbus_proxy_new: assertion 'G_IS_DBUS_CONNECTION (connection)' failed
Apr 23 15:00:27 localhost journal[1572]: g_dbus_proxy_new: assertion 'G_IS_DBUS_CONNECTION (connection)' failed
Apr 23 15:00:27 localhost journal[1572]: g_dbus_proxy_new: assertion 'G_IS_DBUS_CONNECTION (connection)' failed
Apr 23 15:00:27 localhost journal[1572]: g_dbus_proxy_new: assertion 'G_IS_DBUS_CONNECTION (connection)' failed
Apr 23 15:00:27 localhost journal[1572]: g_dbus_proxy_new: assertion 'G_IS_DBUS_CONNECTION (connection)' failed
Apr 23 15:03:05 localhost lightdm-kiran-greeter[1805]: UserAvatar: file path[ "" ] load failed.
Apr 23 15:03:12 localhost journal[1773]: g_dbus_connection_call_sync_internal: assertion 'G_IS_DBUS_CONNECTION (connection)' failed
Apr 23 15:03:22 localhost com.redhat.imsettings[1924]: [ 1682233402.098801]: FCITX[2522]: I/O warning : failed to load external entity ""
Apr 23 15:03:22 localhost com.redhat.imsettings[1924]: [ 1682233402.098841]: FCITX[2522]: I/O warning : failed to load external entity ""
Apr 23 15:03:22 localhost com.redhat.imsettings[1924]: [ 1682233402.528780]: FCITX[2522]: I/O warning : failed to load external entity ""
Apr 23 15:03:22 localhost com.redhat.imsettings[1924]: [ 1682233402.528830]: FCITX[2522]: I/O warning : failed to load external entity ""
[root@localhost Desktop]# cat /var/log/messages | grep error
Apr 23 14:59:29 localhost kernel: [ 1.596533] ksl-daemon[217]: segfault at 270 ip 0000564166a2a3f4 sp 00007fff78ae22c0 error 4 in ksl-daemon[564166a17000+74000]
Apr 23 15:00:11 localhost run-initial-setup[1135]: #011(WARNING) warning, (EE) error, (NI) not implemented, (??) unknown.
Apr 23 15:02:52 localhost lightdm-kiran-greeter[1805]: klog_qt5_init error: -1
Apr 23 15:03:19 localhost journal[2182]: Theme parsing error: gtk.css:1:0: unknown @ rule
Apr 23 15:03:20 localhost journal[2183]: Theme parsing error: gtk.css:1:0: unknown @ rule
Apr 23 15:03:20 localhost journal[2178]: Theme parsing error: gtk.css:1:0: unknown @ rule
Apr 23 15:03:20 localhost journal[2531]: Theme parsing error: gtk.css:1:0: unknown @ rule
Apr 23 15:05:36 localhost journal[2973]: Theme parsing error: gtk.css:1:0: unknown @ rule
[root@localhost Desktop]# dmesg | grep fail
[root@localhost Desktop]# dmesg | grep error
[ 1.596533] ksl-daemon[217]: segfault at 270 ip 0000564166a2a3f4 sp 00007fff78ae22c0 error 4 in ksl-daemon[564166a17000+74000]
```

解决方法：这些报错信息不影响系统正常使用，不同报错的详细说明见如下文档：



日志报错信息说明
文档.md

12 帮助

如果您有任何疑问，都可以通过以下方式获得帮助：

- 1) 求助在线帮助：<http://www.kylinsec.com.cn>；
- 2) 热线：400-012-6606；0731-88777708；
- 3) 可扫描下述二维码进行咨询、建议和提工单：

