

麒麟信安服务器操作系统

V6 SP1

软件管理员手册

文件编号：KYJS-KS-Server-6-SP1-SAM-V1.1



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	2026.4.27	徐铖	A	ALL	生成全文
V1.1	2026.5.21	余永康	M	封面、页眉、6.1、10.1、11.2	封面、页眉更换新logo; 6.1、10.1、11.2章节更换图片

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	3
4.1 设置语言环境	3
4.1.1 显示当前语言环境	3
4.1.2 显示可用语言环境	3
4.1.3 设置语言环境	3
4.2 设置日期和时间	3
4.2.1 显示当前日期和时间	3
4.2.2 远程服务器时间同步	4
4.2.3 手动修改日期和时间	4

4.3 设置网络	4
4.3.1 nmcli 命令	4
4.3.2 设备管理	6
4.3.3 设备网络连接	6
4.3.4 配置动态 IP 连接	6
4.3.5 配置静态 IP 连接	7
4.3.6 配置主机名	7
5 管理用户和用户组	8
5.1 管理用户	8
5.1.1 增加用户	8
5.1.2 修改用户账号	10
5.1.3 删除用户	12
5.2 管理用户组	12
5.2.1 增加用户组	12
5.2.2 修改用户组	13
5.2.3 删除用户组	13
5.2.4 将用户加入用户组或从用户组中移除	14
5.2.5 切换用户组	14
6 三权分立	15
6.1 三权分立配置	15
6.2 安全开关	16
6.2.1 增加用户	16

6.3 系统特性	17
6.3.1 用户角色	17
6.3.2 默认策略	17
6.3.3 控制策略	17
6.3.4 账号登录	18
6.4 sysadm	18
6.5 secadm	20
6.6 audadm	21
6.7 普通用户	23
7 管理软件包	24
7.1 配置软件源	24
7.2 DNF 管理软件包	25
7.3 DNF 检查并更新	26
8 管理服务	28
8.1 概念介绍	28
8.2 特性说明	29
8.2.1 更快的启动速度	29
8.2.2 提供按需启动能力	29
8.2.3 采用 CGroup 特性跟踪和管理进程的生命周期	30
8.2.4 启动挂载点和自动挂载的管理	30
8.2.5 实现事务性依赖关系管理	31
8.2.6 与 SysV 初始化脚本兼容	31

8.2.7 能够对系统进行快照和恢复	32
8.3 管理系统服务	32
8.3.1 显示所有当前服务	33
8.3.2 显示服务状态	33
8.3.4 关闭服务	35
8.3.5 重启服务	35
8.3.6 启用服务	35
8.3.7 禁用服务	36
8.4 关闭、暂停和休眠系统	36
8.4.1 关闭系统	36
8.4.2 重启系统	36
8.4.3 系统待机	36
8.4.4 系统休眠	36
9 管理进程	37
9.1 查看进程	37
9.1.1 who 命令	37
9.1.2 ps 命令	37
9.1.3 top 命令	38
9.1.4 kill 命令	39
9.2 调度启动进程	40
9.2.1 定时运行一批程序 (at)	40
9.2.2 周期性运行一批程序 (cron)	41

10 管理硬盘	43
10.1 标准分区	43
10.2 LVM 简介	46
10.3 LVM 基本概念	47
10.4 管理物理卷	48
10.4.1 创建物理卷	48
10.4.2 查看物理卷	48
10.4.3 修改物理卷属性	49
10.4.4 删除物理卷	49
10.5 管理卷组	50
10.5.1 创建卷组	50
10.5.2 查看卷组	50
10.5.3 修改卷组属性	51
10.5.4 扩展卷组	51
10.5.5 收缩卷组	52
10.5.6 删除卷组	52
10.6 管理逻辑卷	53
10.6.1 创建逻辑卷	53
10.6.2 查看逻辑卷	53
10.6.3 调整逻辑卷大小	54
10.6.4 扩展逻辑卷	54
10.6.5 收缩逻辑卷	55

10.6.6 删除逻辑卷	55
10.7 创建并挂载文件系统	56
10.7.1 创建文件系统	56
10.7.2 手动挂载文件系统	56
10.7.3 自动挂载文件系统	57
11 虚拟化	59
11.1 环境搭建	59
11.2 配置流程	60
12 FAQ	69
13 帮助	79

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 V6 SP1 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 ~]# ks-showinfo
```

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:用户标识符:组标识符:注释性描述:主目录:登录 shell

3.1.6 查看系统用户组

查看系统中的所有组信息，命令如下：

```
[root@localhost ~]# cat /etc/group
root:x:0:
bin:x:1:
daemon:x:2:
sys:x:3:
```

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  0.1  0.2 75636 38044 ?        Ss   13:49   0:00 /usr/lib/systemd/systemd --switched-root --syste
root         2  0.0  0.0      0     0 ?        S    13:49   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    13:49   0:00 [pool_workqueue_release]
root         4  0.0  0.0      0     0 ?        I<   13:49   0:00 [kworker/R-rcu_g]
root         5  0.0  0.0      0     0 ?        I<   13:49   0:00 [kworker/R-rcu_p]
```

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
C.UTF-8
en_AG.UTF-8
en_AU.UTF-8
en_BW.UTF-8
en_CA.UTF-8
en_DK.UTF-8
```

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-locale LANG=zh_CN.UTF-8
[root@localhost ~]#
```

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：

```
[root@localhost ~]# timedatectl
    Local time: — 2024-07-08 14:38:27 CST
    Universal time: — 2024-07-08 06:38:27 UTC
    RTC time: — 2024-07-08 14:38:27
    Time zone: Asia/Shanghai (CST, +0800)
System clock synchronized: no
    NTP service: active
    RTC in local TZ: yes
```

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  
[root@localhost ~]#
```

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。若 NTP 系统时钟自动同步未被关闭，shell 会提示 NTP 未关闭而无法手动设定时间。

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  
Failed to parse time specification 'YYYY-MM-DD': Invalid argument  
[root@localhost ~]# timedatectl set-time 2024-07-08
```

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：

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-time HH:MM:SS  
Failed to parse time specification 'HH:MM:SS': Invalid argument  
[root@localhost ~]# timedatectl set-time 12:00:00
```

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 命令的基本格式如下：

```
[root@localhost ~]# nmcli help
用法: nmcli [选项] 对象 { 命令 | help }

选项
-a, --ask                询问缺少的参数
-c, --colors auto|yes|no 是否在输出中使用颜色
-e, --escape yes|no      转义值中的列分隔符
-f, --fields <字段,...>|all|common 指定要输出的字段
-g, --get-values <字段,...>|all|common -m tabular -t -f 的快捷方式
-h, --help              打印此帮助
-m, --mode tabular|multiline 输出模式
-o, --overview          概览模式
-p, --pretty            美化输出
-s, --show-secrets      允许显示密码
-t, --terse            简介输出
-v, --version          显示程序版本
-w, --wait <秒数>      设定操作完成的等待超时

对象
g[eneral]              NetworkManager 的常规状态和操作
n[etworking]          整体网络控制
r[adio]               NetworkManager 无线电开关
c[onnection]          NetworkManager 的连接
d[evice]              NetworkManager 管理的设备
a[gent]               NetworkManager 机密 (secret) 或 polkit 代理
m[onitor]             监视 NetworkManager 更改
```

显示 NetworkManager 状态:

```
[root@localhost ~]# nmcli general status
STATE      CONNECTIVITY  WIFI-HW  WIFI    WWAN-HW  WWAN
已连接 (仅本地) 受限      missing  已启用  missing  已启用
```

显示所有连接:

```
[root@localhost ~]# nmcli connection show
NAME      UUID                                  TYPE      DEVICE
```

只显示当前活动连接, 添加-a, --active:

```
[root@localhost ~]# nmcli connection show -a
NAME      UUID                                  TYPE      DEVICE
```

显示由 NetworkManager 识别到设备及其状态:

```
[root@localhost ~]# nmcli device status
DEVICE  TYPE      STATE      CONNECTION
lo      loopback  连接 (外部)  lo
```

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 enp2s0
连接已成功激活 (D-Bus 活动路径: /org/freedesktop/NetworkManager/ActiveConnection/4)
[root@localhost ~]# nmcli device disconnect enp2s0
成功断开设备 "enp2s0"。
```

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
lo         193a2e6f-cdfa-4573-9452-c4cd5944f319 loopback  lo
virbr0     d6b5525e-f527-45bb-be9b-bd6f39e311e3 bridge    virbr0
enp2s0     99d404ba-7ce8-4e28-baab-a16e8b1e56de 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：nmcli device status

DEVICE	TYPE	STATE	CONNECTION
lo	loopback	连接（外部）	lo
virbr0	bridge	连接（外部）	virbr0
enp2s0	ethernet	不可用	--

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
```

type ethernet: 网络类型为以太网。

con-name <Connection-Name>: 连接名称为 <Connection-Name>。

ifname <Interface-Name>: 接口名称为 <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 enp2s0
连接 "net-test" (7e8f28c8-a8e0-4164-87c2-7fd8e684e76c) 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/5)
[root@localhost ~]# nmcli device status
```

DEVICE	TYPE	STATE	CONNECTION
enp2s0	ethernet	已连接	net-test
lo	loopback	连接 (外部)	lo
virbr0	bridge	连接 (外部)	virbr0

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 connection add type ethernet con-name connection-name ifname interface-name ip4 your_ipv4 gw4 ipv4_gw
```

ip4 <Your_IPv4>[/<24>]: 将连接的 IPv4 地址设定为<Your_IPv4>。

gw4 <IPv4_gw>: 设置网关为<IPv4_gw>。

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 connection add type ethernet con-name static-test ifname enp2s0 ip4 10.90.21.111 gw4 10.90.21.1
连接 "static-test" (205a2bea-743c-4ad5-874c-e8cb1a12e01d) 已成功添加。
```

激活连接并检查状态:

```
[root@localhost ~]# nmcli con up static-test ifname enp2s0
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
enp2s0	ethernet	已连接	static-test
lo	loopback	连接 (外部)	lo
virbr0	bridge	连接 (外部)	virbr0

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_you_wanted_to
```

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 [选项] 为相关参数。管理员可以使用 useradd -h 指令查看相关参数的使用方法。

```
[root@localhost ~]# useradd -h
用法: useradd [选项] 登录名
      useradd -D
      useradd -D [选项]

选项:
      --badname          不检查 bad names
  -b, --base-dir BASE_DIR 新账户的主目录的基目录
      --btrfs-subvolume-home use BTRFS subvolume for home directory
  -c, --comment COMMENT  新账户的 GECOS 字段
  -d, --home-dir HOME_DIR 新账户的主目录
  -D, --defaults          显示或更改默认的 useradd 配置
  -e, --expiredate EXPIRE_DATE 新账户的过期日期
  -f, --inactive INACTIVE 新账户的密码不活动期
  -F, --add-subids-for-system add entries to sub[uid]id even when adding a system user
  -g, --gid GROUP         新账户主组的名称或 ID
  -G, --groups GROUPS     新账户的附加组列表
  -h, --help              显示此帮助信息并退出
  -k, --skel SKEL_DIR     使用此目录作为骨架目录
  -K, --key KEY=VALUE     不使用 /etc/login.defs 中的默认值
  -l, --no-log-init       不要将此用户添加到最近登录和登录失败数据库
  -m, --create-home       创建用户的主目录
  -M, --no-create-home    不创建用户的主目录
  -N, --no-user-group     不创建同名的组
  -o, --non-unique        允许使用重复的 UID 创建用户
  -p, --password PASSWORD 加密后的新账户密码
  -r, --system            创建一个系统账户
  -R, --root CHROOT_DIR   chroot 到的目录
  -P, --prefix PREFIX_DIR prefix directory where are located the /etc/* files
  -s, --shell SHELL       新账户的登录 shell
  -u, --uid UID           新账户的用户 ID
  -U, --user-group        创建与用户同名的组
  -Z, --selinux-user SEUSER 为 SELinux 用户映射使用指定 SEUSER
      --selinux-range SERANGE use a specific MLS range for the SELinux user mapping
```

例如新建一个用户名为 KylinTest 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。

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

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。

使用 id 命令查看新建的用户信息，命令如下：

```
[root@localhost ~]# id KylinTest
uid=1001(KylinTest) gid=1001(KylinTest) 组=1001(KylinTest)
```

修改用户 KylinTest 的密码：

```
[root@localhost ~]# passwd KylinTest
更改用户 KylinTest 的密码。
新的密码：
重新输入新的密码：
passwd: 所有的身份验证令牌已经成功更新。
```

对于 passwd 命令管理员亦可以通过 passwd --help 命令查看其相关配置参数，当 passwd 后不加任何参数时，默认修改目前进行 passwd 操作的用户密码。

```
[root@localhost ~]# passwd --help
用法: passwd [选项...] <帐号名称>
  -k, --keep-tokens      保持身份验证令牌不过期
  -d, --delete           删除命名帐户的密码 (仅限 root 用户) ; 也删除密码锁 (如果有)
  -l, --lock             锁定指名帐户的密码 (仅限 root 用户)
  -u, --unlock           解锁指名帐户的密码 (仅限 root 用户)
  -e, --expire           终止指名帐户的密码 (仅限 root 用户)
  -f, --force            强制执行操作
  -x, --maximum=DAYS    密码的最长有效时限 (只有 root 用户才能进行此操作)
  -n, --minimum=DAYS    密码的最短有效时限 (只有 root 用户才能进行此操作)
  -w, --warning=DAYS    在密码过期前多少天开始提醒用户 (只有 root 用户才能进行此操作)
  -i, --inactive=DAYS   当密码过期后经过多少天该帐号会被禁用 (只有 root 用户才能进行此操作)
  -S, --status           报告已命名帐号的密码状态 (只有 root 用户才能进行此操作)
  --stdin               从标准输入读取令牌 (只有 root 用户才能进行此操作)

帮助选项:
  -?, --help            显示这个帮助信息
  --usage               显示简短的使用说明
```

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 KylinTest 的 shell 改为 csh，命令如下：

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

修改主目录：可以在 root 权限下执行如下命令，其中 <New_Home_Path> 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_Path> <Username>
```

修改用户 ID：在 root 权限下执行如下命令，其中 <New_UID> 代表目标用户的新 ID，<Username>代表用户名，该命令依据用户名进行新 UID 的分派，请根据实际情况修改：

```
[root@localhost ~]# usermod -u <New_UID> <Username>
```

管理员通过 usermod -h 可以查询 usermod 相关参数的使用方法：

```
[root@localhost ~]# usermod -h
用法：usermod [选项] 登录名

选项：
  -a, --append GROUP          将用户追加至上边 -G 中提到的附加组中，
                              并不从其它组中删除此用户
  -q, --badname               允许 bad name
  -c, --comment COMMENT      GECOS 字段的新值
  -d, --home HOME_DIR        用户的新主目录
  -e, --expiredate EXPIRE_DATE 设定帐户过期的日期为 EXPIRE_DATE
  -f, --inactive INACTIVE    过期 INACTIVE 天数后，设定密码为失效状态
  -g, --gid GROUP             强制使用 GROUP 为新主组
  -G, --groups GROUPS        新的附加组列表 GROUPS
  -h, --help                  显示此帮助信息并退出
  -l, --login NEW_LOGIN      新的登录名称
  -L, --lock                  锁定用户帐号
  -m, --move-home             将家目录内容移至新位置（仅于 -d 一起使用）
  -o, --non-unique            允许使用重复的（非唯一的）UID
  -p, --password PASSWORD    将加密过的密码（PASSWORD）设为新密码
  -P, --prefix PREFIX_DIR    prefix directory where are located the /etc/* files
  -r, --remove                remove the user from only the supplemental GROUPS
                              mentioned by the -G option without removing
                              the user from other groups
  -R, --root CHROOT_DIR      chroot 到的目录
  -s, --shell SHELL          该用户帐号的新登录 shell
  -u, --uid UID              用户帐号的新 UID
  -U, --unlock                解锁用户帐号
  -v, --add-subuids FIRST-LAST 添加子 UID 范围
  -V, --del-subuids FIRST-LAST 移除子 UID 范围
  -w, --add-subgids FIRST-LAST 添加子 GID 范围
  -W, --del-subgids FIRST-LAST 移除子 GID 范围
  -Z, --selinux-user SEUSER   用户的新的 SELinux 用户映射
  --selinux-range SERANGE    new SELinux MLS range for the user account
```

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 KylinTest` 命令。

通过 `userdel <Username>` 删除指定的用户，例如，删除用户 `KylinTest` 命令如下：

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

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>
```

根据 `groupadd -h` 可以查询到相关参数的用法：

```
[root@localhost ~]# groupadd -h
用法：groupadd [选项] 组

选项：
  -f, --force           如果组已经存在则成功退出
                        并且如果 GID 已被使用则取消 -g
  -g, --gid GID         为新组使用 GID
  -h, --help            显示此帮助信息并退出
  -K, --key KEY=VALUE   不使用 /etc/login.defs 中的默认值
  -o, --non-unique       允许创建有重复 GID 的组
  -p, --password PASSWORD 为新组使用此加密过的密码
  -r, --system          创建一个系统账户
  -R, --root CHROOT_DIR  chroot 到的目录
  -P, --prefix PREFIX_DIR directory prefix
  -l, --list            该组的用户成员列表
```

据此，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 <New_GID>代表目标用户组的新 ID，<Groupname> 代表用户组名称，这条命令是依据用户组名进行新 GID 的委派，请根据实际情况修改：

```
[root@localhost ~]# groupmod -g <New_GID> <Groupname>
```

修改用户组名：在 root 权限下执行如下命令，其中 <New_Groupname> 代表新用户组名，<Groupname>代表指定的用户组名，这条命令依据用户组名指派新的用户组名称，执行完成后原来叫做<Groupname>的用户组就更名为<New_Groupname>，请根据实际情况修改：

```
[root@localhost ~]# groupmod -n <New_Groupname> <Groupname>
```

根据 groupmod -h 还可以查询到其相关参数和使用方法

```
[root@localhost ~]# groupmod -h
用法：groupmod [选项] 组

选项：
-a, --append          将-U选项提到的用户添加到组中
                        无需删除现有用户成员
-g, --gid GID         将组 ID 改为 GID
-h, --help            显示此帮助信息并退出
-n, --new-name NEW_GROUP 改名为 NEW_GROUP
-o, --non-unique       允许使用重复的 GID
-p, --password PASSWORD 将密码更改为(加密过的) PASSWORD
-R, --root CHROOT_DIR  chroot 到的目录
-P, --prefix PREFIX_DIR prefix directory where are located the /etc/* files
-l, --list            该组的用户成员列表
```

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 grouptest 命令。

例如，删除用户组 groupdel 命令如下：

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

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。

```
[root@localhost ~]# gpasswd -h
用法: gpasswd [选项] 组

选项:
-a, --add USER          向组 GROUP 中添加用户 USER
-d, --delete USER       从组 GROUP 中删除用户
-h, --help               显示此帮助信息并退出
-Q, --root CHROOT_DIR   要 chroot 进的目录
-r, --remove-password    移除组 GROUP 的密码
-R, --restrict           向其成员限制访问组 GROUP
-M, --members USER,...  设置组 GROUP 的成员列表
-A, --administrators ADMIN,... 设置组的管理员列表

除非使用 -A 或 -M 选项，不能结合使用这些选项。
```

据此，例如将用户 KylinTest 加入用户组 groupdel，命令如下：

```
[root@localhost ~]# gpasswd -a KylinTest groupdel
```

例如，将用户 KylinTest 从 groupdel 用户组中移除，命令如下：

```
[root@localhost ~]# gpasswd -d KylinTest groupdel
```

5.2.5 切换用户组

newgrp 指令类似 login 指令，一个用户同时属于多个用户组时，则在用户登录后，使用 newgrp 命令可以再次登入系统切换到其他用户组，以便具有其他用户组的权限。欲使用 newgrp 指令切换群组，您必须是该群组的用户，否则将无法登入指定的群组。单一用户要同时隶属多个群组，需利用交替用户的设置。若不指定群组名称，则 newgrp 指令会登入该用户名称的预设群组。

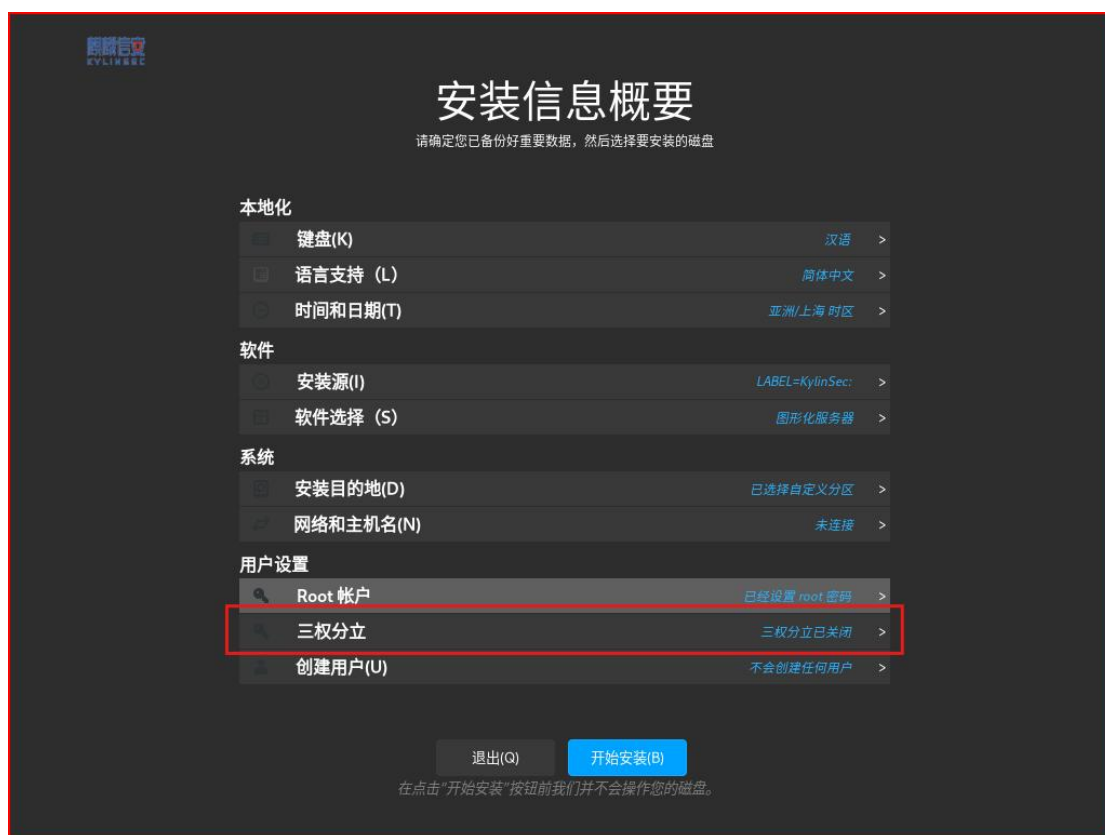
例如，将用户 test 切换到 groupdel 用户组，命令如下：

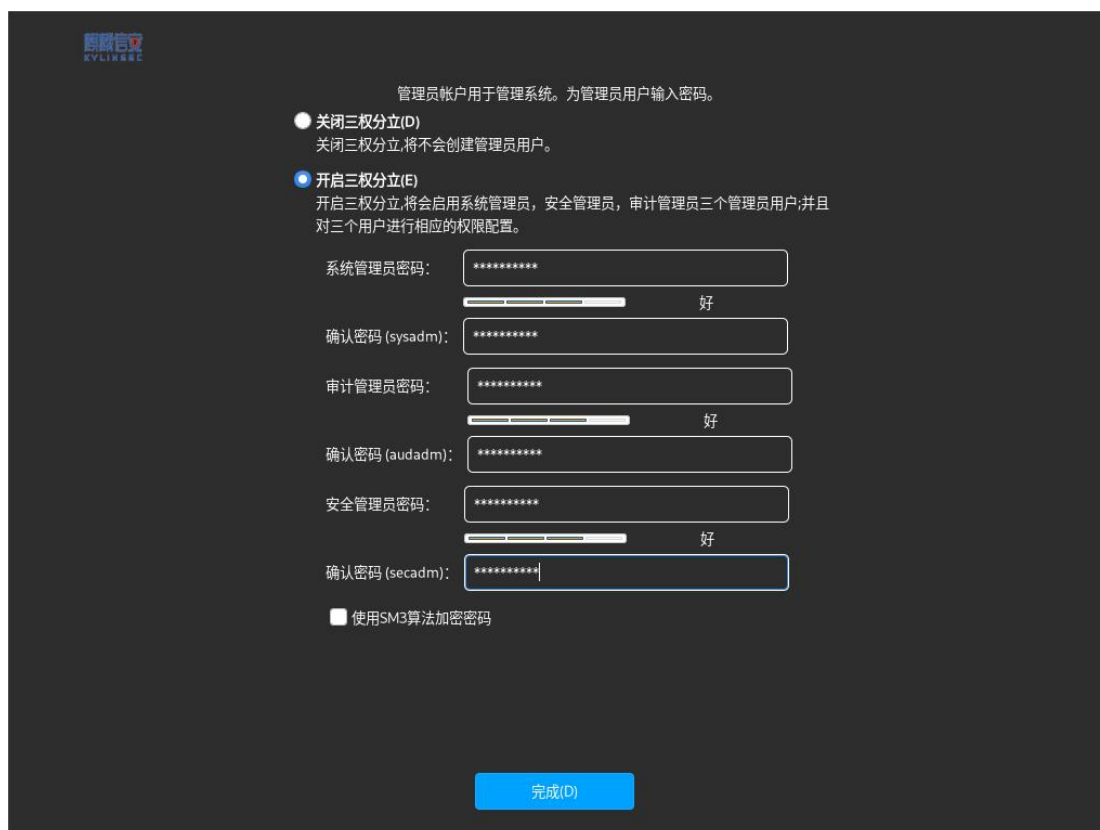
```
[test@localhost ~]$ newgrp groupdel
```

6 三权分立

6.1 三权分立配置

三权分立的配置需要在系统安装前，在系统安装界面勾选开启三权分立，然后分别设置三个管理员用户的密码，如下图所示：





6.2 安全开关

6.2.1 增加用户

使用 secadm 用户登录系统，在终端中输入 `vim /etc/sysconfig/selinux` 命令，可以设置 selinux 状态。如图所示：

```
secadm@localhost:~  
文件(F) 编辑(E) 视图(V) 搜索(S) 终端(T) 帮助(H)  
# This file controls the state of SELinux on the system.  
# SELINUX= can take one of these three values:  
#   enforcing - SELinux security policy is enforced.  
#   permissive - SELinux prints warnings instead of enforcing.  
#   disabled - No SELinux policy is loaded.  
SELINUX=enforcing  
# SELINUXTYPE= can take one of these three values:  
#   targeted - Targeted processes are protected,  
#   minimum - Modification of targeted policy. Only selected processes are protected.  
#   mls - Multi Level Security protection.  
SELINUXTYPE=targeted
```

当设置 `SELINUX=disabled` 状态时，root 用户可以登录系统。

当设置 SELINUX=permissive 状态时，root 用户可以登录系统，还可以查看到 root 用户登录后的审计日志。

当设置 SELINUX=enforcing 状态时，root 用户不可以登录系统。

6.3 系统特性

6.3.1 用户角色

系统内置有系统管理员（sysadm）、审计管理员（audadm）、安全管理员（secadm）。

使用 secadm 登录系统，sysadm、secadm、audadm 三个用户从属于 sysadm_r、secadm_r、audadm_r 角色。

使用 sysadm 登录系统，新建普通用户，普通用户从属于普通用户各自的分组。

6.3.2 默认策略

不同用户的默认策略不同，默认普通用户不能更改（/etc/下任意文件）的权限。

6.3.3 控制策略

1) 目录访问控制策略

系统可以通过 facl 限制用户访问,使用 secadm 用户登录系统,配置 testuser1 普通用户，打开终端，输入命令：

```
sudo setfacl -m u:testuser1:--- /tmp/dir
```

sudo getfacl /tmp/dir，可以对/tmp/dir 这个目录设置权限。

2) 文件访问控制策略

系统可以通过 facl 控制对文件的访问，使用 secadm 用户登录系统，配置

testuser1 普通用户, 对/tmp/test2 文件可读可写的权限, 打开终端: 输入命令:

```
sudo setfacl -m u:testuser1:rw /tmp/test2
```

sudo getfacl /tmp/test2 , 可以对/tmp/test2 这个文件设置可读可写权限。

3) ssh 访问控制策略

系统可以通过使用 sysadm 用户修改/etc/hosts.deny 文件控制 ssh 访问。

如: 在/etc/hosts.deny 文件增加 sshd:10.20.1.23:deny, 使用 10.20.1.23 客户端登录系统, 显示登录失败。

6.3.4 账号登录

使用不存在的内置用户无法登录系统, 任意帐号使用空密码无法登陆系统。

6.4 sysadm

1) 用户标识

使用 sysadm 用户登录系统, 创建系统用户, 系统用户拥有唯一标识 (UID)。

2) 密码规则配置

使用 sysadm 用户登录系统, 打开终端, 输入命令: cat /etc/pam.d/system-auth 可以查看默认密码策略。可以根据需求更改参数。

参数详解:

retry=N: 定义登录/修改密码失败时, 可以重试的次数;

Difok=N: 定义新密码中必须有几个字符要与旧密码不同。但是如果新密码中有 1/2 以上的字符与旧密码不同时, 该新密码将被接受;

minlen=N: 定义用户密码的最小长度;

dcredit=N: 定义用户密码中必须包含多少个数字;

ucredit=N: 定义用户密码中必须包含多少个大写字母;

lcredit=N: 定义用户密码中必须包含多少个小些字母;

ocredit=N: 定义用户密码中必须包含多少个特殊字符(除数字、字母之外);

其中=-1 表示, 至少有一个。

3) 密码登录规则配置

此规则只对新增用户生效, 即在新建用户的时候会去读取此规则。打开终端, 输入命令: `sudo cat /etc/login.defs` 可以查看密码登录规则。可以根据需求更改参数。参数详解:

`PASS_MAX_DAYS 99999` #密码的最大有效期, 99999:永久有期。

`PASS_MIN_DAYS 0` #是否可修改密码,0 可修改,非 0 多少天后可修改。

`PASS_MIN_LEN 5` #密码最小长度,使用 `pam_cracklib` module,该参数不再有效。

`PASS_WARN_AGE 7` #密码失效前多少天在用户登录时通知用户修改密码。

4) 默认配置的锁定用户

默认密码规则配置, 输入 3 次错误密码, 用户锁定。锁定 60s 后, 可以重新登录。

5) 手动解锁

默认密码规则配置, 输入 3 次错误密码, 用户 (testuser2) 锁定, 可以使用管理员手动解锁, 输入命令:

`sudo faillock --user testuser2 --reset`, 再次登录。

6) 进行审计服务管理

系统用户可以查看审计服务, 但是不可以停止审计服务。

7) sysadm 不被允许的操作

当安全开关状态打开，系统 sysadm 用户不可以修改安全开关，查询布尔变量，用户切换，查看审计日志。

6.5 secadm

只有安全管理员能够进行与安全有关的管理：

1) 开关 selinux

\$sudo setenforce 0 关闭

\$sudo setenforce 1 开启

2) bool 值

查看 bool 值 httpd_use_nfs 状态

\$ sudo getsebool httpd_use_nfs

设置 bool 值 httpd_use_nfs 状态

\$ sudo setsebool httpd_use_nfs=on/off

查看登陆用户的标记

\$id -Z

3) 新增角色和关联用户

安全管理员创建安全用户 test_u

```
[secadm@localhost ~]$ sudo semanage user -a -R user_r -r s0 test_u
```

安全管理员将安全用户与系统用户 test2 关联

```
[secadm@localhost ~]$ sudo semanage login -a -s test_u test2
```

```
[secadm@localhost ~]$ sudo semanage login -l |grep test2
```

用户 test2 登陆系统，获取登陆安全上下文

\$ id -Z

获取文件上下文标记:

```
$ ls -Z /tmp/test
```

修改文件安全上下文:

```
$sudo chcon -t type_t /tmp/test
```

4) secadm 不被允许的操作

在系统安全开关打开状态下, secadm 用户不被允许的操作有: 查看审计日志, 管理审计服务, 用户切换, 管理用户。

6.6 audadm

只有审计管理员才能读取审计日志, 其他用户均无权限访问日志文件。

1) 查看审计日志:

切换到审计管理员用户, 查看审计日志文件内容

```
$ sudo cat /var/log/audit/audit.log
```

```
$sudo ausearch -l 查看所有审计记录
```

2) 审计规则

用 audadm 用户登录系统, 不能查看, 停止和开启审计服务。

切换到审计管理员角色添加规则

```
$sudo auditctl -a exit,always -S all -F uid=UID
```

审计管理员角色执行\$sudo ausearch -ui UID,可以查看到以上 uid 用户所有操作记录

审计管理员角色执行\$sudo aureport -m 可以查看对系统用户所有操作, 如用户的删除, 新增, 修改密码等操作;

切换到审计管理员角色, 添加规则监控/tmp/testfile 的所有操作, 并添加关

键字-k test_file;

文件监视审计:

```
$sudo auditctl -w /tmp/testfile -k test_file
```

通过审计管理员执行 `$sudo ausearch -k test_file` 可以查看到 /tmp/testfile 的所有审计日志;

目录监视审计:

切换到审计管理员, 添加规则

```
$sudo auditctl -w /tmp/test.dir -k test_dir
```

切换到审计管理员, 添加对进程的审计规则

```
$sudo auditctl -a exit,always -F pid=PID
```

系统调用审计:

切换到审计管理员, 添加对系统调用如 open 的审计规则

```
$sudo auditctl -a exit,always -S open
```

安全标记审计:

切换到审计管理员, 添加以对属性为 type_t 的审计规则:

```
$auditctl -a exit,always -S all -F obj_type=type_t -k ky_event
```

3) 审计事件存储

使用 sysadm 用户, 移动 audit 日志信息, audit 日志信息无法移动。

4) audadm 不被允许的操作

在系统安全开关打开状态下, audadm 用户不被允许的操作有: 关闭安全开关, 查询布尔变量, 用户切换, 管理用户, 修改安全开关。

6.7 普通用户

在系统安全开关打开状态下，普通用户不被允许的操作有：关闭安全开关，查询布尔变量，用户切换，查看审计日志，关闭审计，新增用户，修改安全开关。

7 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。

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。

7.1 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部网络的情况下，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/
kylinsec.repo
[root@localhost ~]# cat /etc/yum.repos.d/kylinsec.repo
```

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

[CDROM]
name=KylinSec-$releasever - CDROM
gpgcheck=1
enabled=0
baseurl=file:///run/media/$user/KylinSec/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-kylinsec-release
```

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。

7.2 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 <Description>
```

示例如下:

```
[root@localhost ~]# dnf search gcc
Last metadata expiration check: 1:52:36 ago on 2026年07月08日 星期三 00时00分00秒.
===== Name Exactly Matched: gcc =====
gcc.x86_64 : Various compilers (C, C++, Objective-C, ...)
===== Name & Summary Matched: gcc =====
gcc-c++.x86_64 : C++ support for GCC
gcc-gdb-plugin.x86_64 : GCC plugin for GDB
gcc-objc.x86_64 : Objective-C support for GCC
gcc-objc++.x86_64 : Objective-C++ support for GCC
libgcc.x86_64 : GCC version 12 shared support library
===== Name Matched: gcc =====
gcc-gfortran.x86_64 : Fortran support
===== Summary Matched: gcc =====
libgomp.x86_64 : GCC OpenMP v4.5 shared support library
libquadmath.x86_64 : GCC __float128 shared support library
libquadmath-devel.x86_64 : GCC __float128 support
[root@localhost ~]#
```

列出所有已经安装的软件包清单,可使用如下命令:

```
[root@localhost ~]# dnf list all
Last metadata expiration check: 1:53:17 ago on 2026年07月08日 星期三 00时00分00秒.
Installed Packages
ImageMagick.x86_64 1:7.1.2-9-1.ks6 @anaconda
Imath.x86_64 3.1.9-1.ks6 @anaconda
LibRaw.x86_64 0.21.1-4.ks6 @anaconda
ModemManager.x86_64 1.22.0-2.ks6 @anaconda
ModemManager-glib.x86_64 1.22.0-2.ks6 @anaconda
NetworkManager.x86_64 1:1.44.2-4.ks6.kb2 @anaconda
NetworkManager-config-server.noarch 1:1.44.2-4.ks6.kb2 @anaconda
NetworkManager-l2tp.x86_64 1.20.10-2.ks6.kb1 @anaconda
NetworkManager-l2tp-gnome.x86_64 1.20.10-2.ks6.kb1 @anaconda
NetworkManager-libnm.x86_64 1:1.44.2-4.ks6.kb2 @anaconda
NetworkManager-ppp.x86_64 1:1.44.2-4.ks6.kb2 @anaconda
NetworkManager-team.x86_64 1:1.44.2-4.ks6.kb2 @anaconda
NetworkManager-wwan.x86_64 1:1.44.2-4.ks6.kb2 @anaconda
OpenEXR-libs.x86_64 3.1.11-3.ks6 @anaconda
PackageKit.x86_64 1.2.8-1.ks6.kb1 @anaconda
PackageKit-devel.x86_64 1.2.8-1.ks6.kb1 @anaconda
Rocky3.x86_64 1.10.0-3.ks6 @anaconda
SDL.x86_64 1.2.15-40.ks6.kb1 @anaconda
```

列出系统中特定的 rpm 包, 示例如下:

```
[root@localhost ~]# dnf list gcc
Last metadata expiration check: 1:54:10 ago on 2026年07月08日 星期三 00时00分00秒.
Installed Packages
gcc.x86_64 12.3.1-98.ks6.kb4 @anaconda
[root@localhost ~]#
```

显示某个名为<Package_Name>的 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 gcc
Last metadata expiration check: 1:54:40 ago on 2026年07月08日 星期三 00时00分00秒.
Installed Packages
Name       : gcc
Version    : 12.3.1
Release    : 98.ks6.kb4
Architecture : x86_64
Size       : 96 M
Source     : gcc-12.3.1-98.ks6.kb4.src.rpm
Repository : @System
From repo  : anaconda
Summary    : Various compilers (C, C++, Objective-C, ...)
URL        : https://gcc.gnu.org
License    : GPLv3+ and GPLv3+ with exceptions and GPLv2+ with exceptions and LGPLv2+ and BSD
Description : The gcc package contains the GNU Compiler Collection version 12.
              You'll need this package in order to compile C code.
```

要安装名为<Package_Name>的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>
```

下载名为<Package_Name>的软件包，可在 root 权限下使用如下命令：

```
[root@localhost ~]# dnf download <Package_Name>
```

如果需要同步下载未安装的依赖，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>
```

更多相关用法可以使用 dnf help 命令获得。

7.3 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: 1:55:31 ago on 2026年07月08日 星期三 00时00分00秒.
```

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
```

8 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 务管理 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。

8.1 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	封装守护进程的启动、停止、重启和重载操作,是最常见的一种 Unit 文件
Target unit	.target	用于对 Unit 文件进行逻辑分组,引导其它 Unit 的执行。它替代了 SysV-init 运行级别的作用,并提供更灵活的基于特定设备事件的启动方式
Automount unit	.automount	用于控制自动挂载文件系统,相当于 SysV-init 的 autofs 服务
Device unit	.device	对于 /dev 目录下的设备,主要用于定义设备之间的依赖关系
Mount unit	.mount	定义系统结构层次中的一个挂载点,可以替代过去的 /etc/fstab 配置文件
Path unit	.path	用于监控指定目录或文件的变化,并触发其它 Unit 运行
Scope unit	.scope	这种 Unit 文件不是用户创建的,而是 Systemd 运行时产生的,描述一些系统服务的分组信息
Slice unit	.slice	用于表示一个 CGroup 的树,通常用户不会自己创建这样的 Unit 文件
Socket unit	.socket	监控来自于系统或网络的数据消息,用于实现基于数据自动触发服务启动
Swap unit	.swap	定义一个用户做虚拟内存的交换分区
Timer unit	.timer	用于配置在特定时间触发的任务,替代了 Crontab 的功能
Snapshot	.snapshot	用于表示一个由 systemctl snapshot 命令创建的

unit	Systemd Units 运行状态快照
------	----------------------

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
/etc/systemd/system	系统或用户自定义的配置文件
/run/systemd/system	软件运行时生成的配置文件
/usr/lib/systemd/system	系统或第三方软件安装时添加的配置文件

systemd 默认从目录 /etc/systemd/system/ 读取配置文件。但是，里面存放的大部分文件都是符号链接，指向目录 /usr/lib/systemd/system/，真正的配置文件都存放在该目录。

8.2 特性说明

8.2.1 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。

8.2.2 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。

8.2.3 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。

在 systemd 之前的主流应用管理服务都是使用进程树来跟踪应用的继承关系的，而进程的父子关系很容易通过两次 fork 的方法脱离。

而 systemd 则提供通过 CGroup 跟踪进程关系，弥补了这个缺漏。CGroup 能够实现服务之间访问隔离，用来实现系统资源配额管理，管理服务生命周期。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。

8.2.4 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。

8.2.5 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。

8.2.6 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。

8.2.7 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。

8.3 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
service network start	systemctl start network.service	用来启动一个服务（并不会重启现有的）。
service network stop	systemctl stop network.service	用来停止一个服务（并不会重启现有的）。
service network reload	systemctl reload network.service	当支持时，重新装载配置文件而不中断等待操作。

service network restart	systemctl restart network.service	用来停止并启动一个服务。
service network condrestart	systemctl condrestart network.service	如果服务正在运行那么重启它。
service network status	systemctl status network.service	检查服务的运行状态。
chkconfig network on	systemctl enable network.service	在下次启动时或满足其他触发条件时设置服务为启用。
chkconfig network off	systemctl disable network.service	在下次启动时或满足其他触发条件时设置服务为禁用。
chkconfig network	systemctl is-enabled network.service	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/*.wa nts/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。

8.3.1 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
```

UNIT	LOAD	ACTIVE	SUB	DESCRIPTION
accounts-daemon.service	loaded	active	running	Accounts Service
alsa-state.service	loaded	active	running	Manage Sound Card State (restore and store)
atd.service	loaded	active	running	Deferred execution scheduler
auditd.service	loaded	active	running	Security Auditing Service
avahi-daemon.service	loaded	active	running	Avahi mDNS/DNS-SD Stack
chronyd.service	loaded	active	running	NTP client/server
crond.service	loaded	active	running	Command Scheduler
dbus.service	loaded	active	running	D-Bus System Message Bus

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
```

UNIT	LOAD	ACTIVE	SUB	DESCRIPTION
accounts-daemon.service	loaded	active	running	Accounts Service
alsa-restore.service	loaded	inactive	dead	Save/Restore Sound Card State
alsa-state.service	loaded	active	running	Manage Sound Card State (restore and store)
● apparmor.service	not-found	inactive	dead	apparmor.service
atd.service	loaded	active	running	Deferred execution scheduler
auditd.service	loaded	active	running	Security Auditing Service
auth-rpcgss-module.service	loaded	inactive	dead	Kernel Module supporting RPCSEC_GSS
● auto-cpufreq.service	not-found	inactive	dead	auto-cpufreq.service
● autofs.service	not-found	inactive	dead	autofs.service

8.3.2 显示服务状态

如果您需要显示某个名为<Service_Name>服务的状态，可执行如下命令：

```
[root@localhost ~]# systemctl status <Service_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名为<Service_Name>的服务是否运行，可执行如下命令：

令：

```
[root@localhost ~]# systemctl is-active <Service_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名为<Service_Name>的服务是否被启用，可执行如下命令：

命令：

```
[root@localhost ~]# systemctl is-enabled <Service_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。

8.3.3 运行服务

如果您需要运行某个名为<Service_Name>的服务，请在 root 权限下执行

如下命令：

```
[root@localhost ~]# systemctl start <Service_Name>.service
```

8.3.4 关闭服务

如果您需要关闭某个名为<Service_Name>的服务，请在 root 权限下执行

如下命令：

```
[root@localhost ~]# systemctl stop <Service_Name>.service
```

8.3.5 重启服务

如果您需要重启某个名为<Service_Name>的服务，请在 root 权限下执行

如下命令：

```
[root@localhost ~]# systemctl restart <Service_Name>.service
```

8.3.6 启用服务

如果您需要在开机时启用某个名为<Service_Name>的服务，请在 root 权

限下执行如下命令：

```
[root@localhost ~]# systemctl enable <Service_Name>.service
```

8.3.7 禁用服务

如果您需要在开机时禁用某个名为<Service_Name>的服务，请在 root 权限下执行如下命令：

```
[root@localhost ~]# systemctl disable <Service_Name>.service
```

8.4 关闭、暂停和休眠系统

8.4.1 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
```

8.4.2 重启系统

重启系统，在 root 权限下执行如下命令：

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

8.4.3 系统待机

使系统待机，在 root 权限下执行如下命令：

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

8.4.4 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
```

9 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。

9.1 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。

9.1.1 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      2026-04-08 10:04 (:0)
root    pts/1     2026-04-08 10:16 (10.80.5.41)
root    pts/3     2026-07-08 01:45 (10.80.5.41)
```

9.1.2 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
  4724 pts/0        00:00:00 bash
  4859 pts/0        00:00:00 bash
  8427 pts/0        00:00:00 ps
```

9.1.3 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：

```
[root@localhost ~]# top

top - 15:49:41 up 6:59, 1 user, load average: 0.04, 0.02, 0.00
Tasks: 301 total, 1 running, 300 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.1 us, 0.1 sy, 0.0 ni, 99.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 15170.4 total, 11524.6 free, 2242.5 used, 2148.8 buff/cache
MiB Swap: 7908.0 total, 7908.0 free, 0.0 used, 12927.9 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM     TIME+ COMMAND
 2293 root        20   0  971200 99712 67252 S   1.7   0.6   0:55.78 Xorg
 3758 root        20   0  819100 67060 52208 S   1.0   0.4   0:13.44 mate-terminal
 1283 libstor+    20   0   2536   1740  1556 S   0.3   0.0   0:00.06 lsmd
 5885 root        20   0 2718452 249116 114720 S   0.3   1.6   0:24.01 Isolated Web Co
    1 root        20   0   75808 36704  10072 S   0.0   0.2   0:00.92 systemd
    2 root        20   0      0      0      0 S   0.0   0.0   0:00.00 kthreadd
    3 root        20   0      0      0      0 S   0.0   0.0   0:00.00 pool_workqueue_release
    4 root         0 -20      0      0      0 I   0.0   0.0   0:00.00 kworker/R-rcu_g
    5 root         0 -20      0      0      0 I   0.0   0.0   0:00.00 kworker/R-rcu_p
    6 root         0 -20      0      0      0 I   0.0   0.0   0:00.00 kworker/R-slub_
    7 root         0 -20      0      0      0 I   0.0   0.0   0:00.00 kworker/R-netns
```

9.1.4 kill 命令

当需要中断一个前台进程的时候，通常足使用“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
kill: 用法: kill [-s 信号说明符 | -n 信号编号 | -信号说明符] pid | 任务说明符 ... 或 kill -l [信号说明符]
```

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
1) SIGHUP      2) SIGINT      3) SIGQUIT     4) SIGILL      5) SIGTRAP
6) SIGABRT     7) SIGBUS      8) SIGFPE      9) SIGKILL     10) SIGUSR1
11) SIGSEGV    12) SIGUSR2    13) SIGPIPE    14) SIGALRM    15) SIGTERM
16) SIGSTKFLT  17) SIGCHLD   18) SIGCONT    19) SIGSTOP    20) SIGTSTP
21) SIGTTIN    22) SIGTTOU   23) SIGURG     24) SIGXCPU    25) SIGXFSZ
26) SIGVTALRM  27) SIGPROF   28) SIGWINCH   29) SIGIO       30) SIGPWR
31) SIGSYS     34) SIGRTMIN  35) SIGRTMIN+1 36) SIGRTMIN+2 37) SIGRTMIN+3
38) SIGRTMIN+4 39) SIGRTMIN+5 40) SIGRTMIN+6 41) SIGRTMIN+7 42) SIGRTMIN+8
43) SIGRTMIN+9 44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9 56) SIGRTMAX-8 57) SIGRTMAX-7
58) SIGRTMAX-6 59) SIGRTMAX-5 60) SIGRTMAX-4 61) SIGRTMAX-3 62) SIGRTMAX-2
63) SIGRTMAX-1 64) SIGRTMAX
```

9.2 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。

9.2.1 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 [-q 队列] [-f 文件名] [-u 用户名] [-mMlbv] 时间 ...
```

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

管理员可以通过 at -h 查看更多相关参数和相关用法：

```
[root@localhost ~]# at -h
Usage: at [-V] [-q x] [-f file] [-u username] [-mMlbv] timespec ...
       at [-V] [-q x] [-f file] [-u username] [-mMlbv] -t time
       at -c job ...
       at [-V] -l [-o timeformat] [job ...]
       atq [-V] [-q x] [-o timeformat] [job ...]
       at [ -rd ] job ...
       atrm [-V] job ...
       batch
```

9.2.2 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。

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：

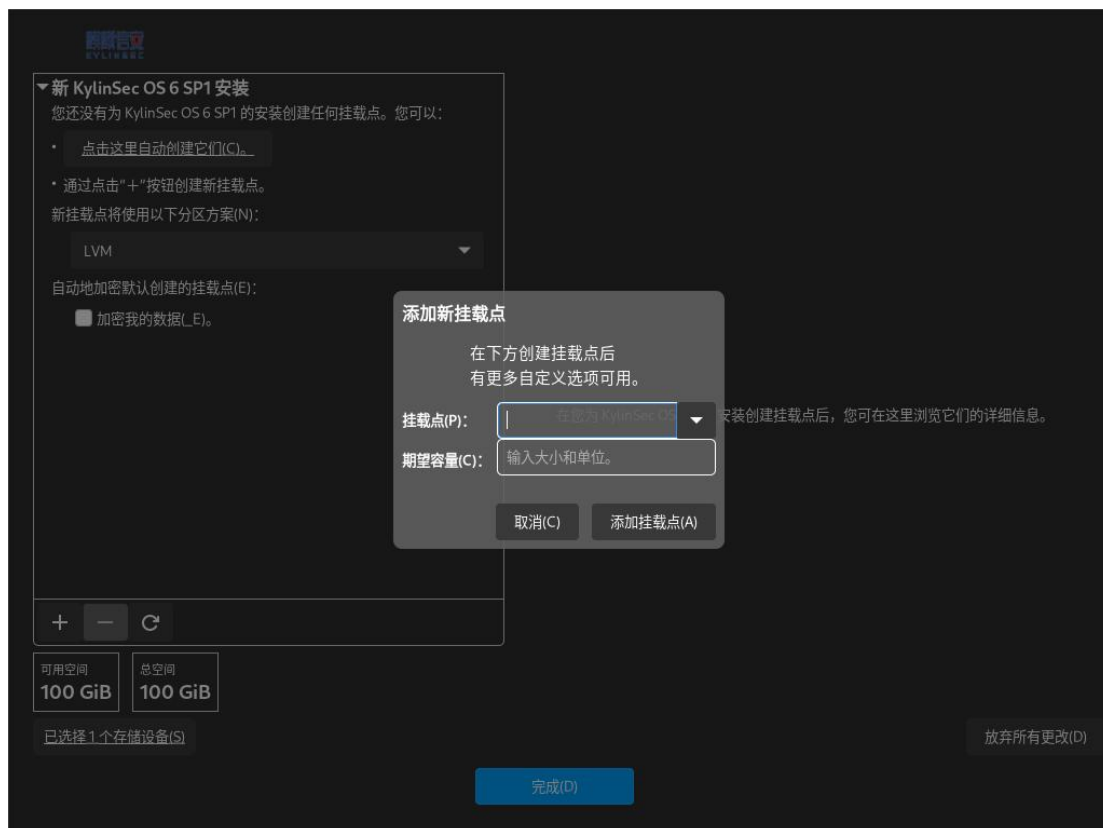
```
[root@localhost ~]# crontab -u root -l  
no crontab for root
```

10 管理硬盘

10.1 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
sda                  8:0    0 931.5G  0 disk
├─sda1                8:1    0   128M  0 part /boot/efi
├─sda2                8:2    0     2G  0 part
├─sda3                8:3    0    50G  0 part
├─sda4                8:4    0    7.7G  0 part
├─sda5                8:5    0   100G  0 part
├─sda6                8:6    0     2G  0 part /boot
├─sda7                8:7    0 769.6G  0 part
├─ko-root            253:0    0    70G  0 lvm /
├─ko-swap            253:1    0    7.7G  0 lvm [SWAP]
├─ko-backup          253:2    0    50G  0 lvm
└─ko-KSData          253:3    0 641.9G  0 lvm /var
                               /usr/local
                               /root
                               /opt
                               /home
                               /KSData
```

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/sda: 931.51 GiB, 1000204886016 字节, 1953525168 个扇区
磁盘型号: ST1000DM010-2EP1
单元: 扇区 / 1 * 512 = 512 字节
扇区大小(逻辑/物理): 512 字节 / 4096 字节
I/O 大小(最小/最佳): 4096 字节 / 4096 字节
磁盘标签类型: gpt
磁盘标识符: 3C20F1F8-1FA5-4155-A85D-2536F734AFB6
```

设备	起点	末尾	扇区	大小	类型
/dev/sda1	2048	264191	262144	128M	EFI 系统
/dev/sda2	264192	4458495	4194304	2G	Linux 文件系统
/dev/sda3	4458496	109316095	104857600	50G	Linux 文件系统
/dev/sda4	109316096	125552639	16236544	7.7G	Linux swap
/dev/sda5	125552640	335267839	209715200	100G	Linux 文件系统
/dev/sda6	335267840	339462143	4194304	2G	Linux 文件系统
/dev/sda7	339462144	1953523711	1614061568	769.6G	Linux LVM

对某个磁盘进行分区管理，使用 fdisk 设备名，如下：

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

欢迎使用 fdisk (util-linux 2.39.1)。
更改将停留在内存中，直到您决定将更改写入磁盘。
使用写入命令前请三思。
```

输入 m 可获取帮助信息：

```
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  create a new empty MBR (DOS) partition table
s  新建一份空 Sun 分区表
```

10.2 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。

10.3 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。

10.4 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。

10.4.1 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/sda` 创建为物理卷。

```
[root@localhost ~]# pvcreate -y /dev/sdb /dev/sda
```

10.4.2 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
```

10.4.3 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] devname ...
```

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”。

devname: 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
```

10.4.4 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] devname ...
```

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” 。

devname: 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
```

10.5 管理卷组

10.5.1 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
```

10.5.2 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
```

10.5.3 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
```

10.5.4 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
```

10.5.5 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
```

10.5.6 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
```

10.6 管理逻辑卷

10.6.1 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
```

10.6.2 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
```

10.6.3 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。

vgname：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
```

10.6.4 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
```

10.6.5 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
```

10.6.6 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
```

10.7 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。

10.7.1 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
```

10.7.2 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
```

10.7.3 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。

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

执行如下命令，将/etc/fstab 文件所有内容重新加载。

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

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

11 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。

11.1 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
```

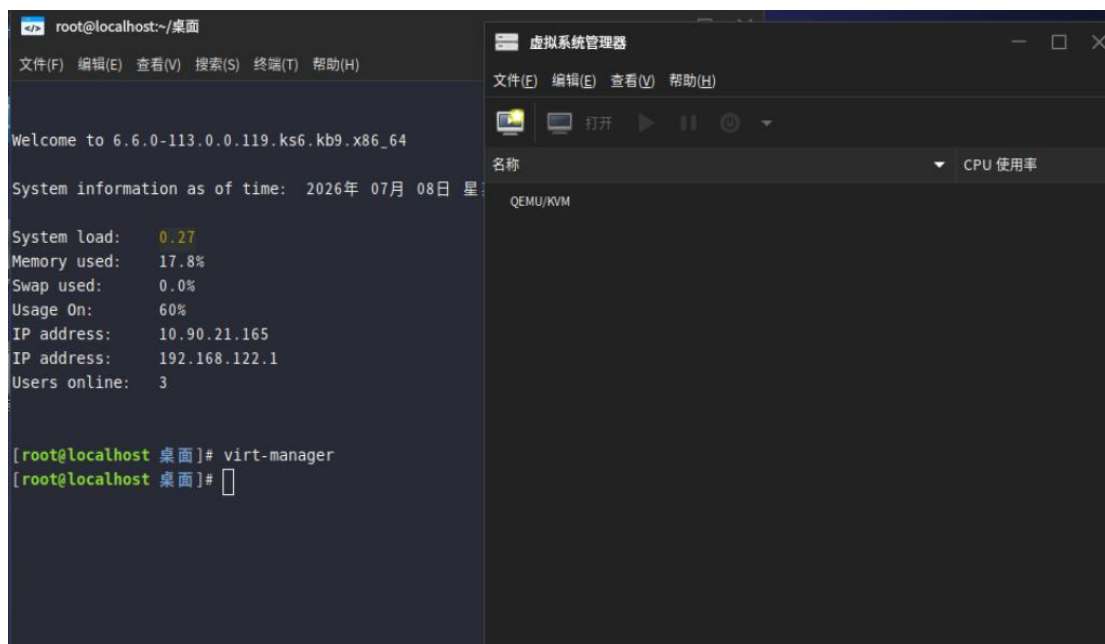
```
yum install edk2-aarch64 (aarch 机器)
```

```
systemctl start libvirtd
```

```
[root@localhost ~]# systemctl start libvirtd
[root@localhost ~]# systemctl status libvirtd
● libvirtd.service - libvirt legacy monolithic daemon
   Loaded: loaded (/usr/lib/systemd/system/libvirtd.service; enabled; preset: enabled)
   Active: active (running) since Wed 2026-07-08 02:16:47 CST; 10s ago
     TriggeredBy: ● libvirtd.socket
                  ● libvirtd-ro.socket
                  ● libvirtd-admin.socket
     Docs: man:libvirtd(8)
           https://libvirt.org/
    Main PID: 150985 (libvirtd)
      Tasks: 22 (limit: 32768)
     Memory: 14.7M ()
    CGroup: /system.slice/libvirtd.service
            └─150985 /usr/sbin/libvirtd --timeout 120
              └─151080 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf --leasefile-ro --dhcp-script=/usr/lib/
                └─151081 /usr/sbin/dnsmasq --conf-file=/var/lib/libvirt/dnsmasq/default.conf --leasefile-ro --dhcp-script=/usr/lib/

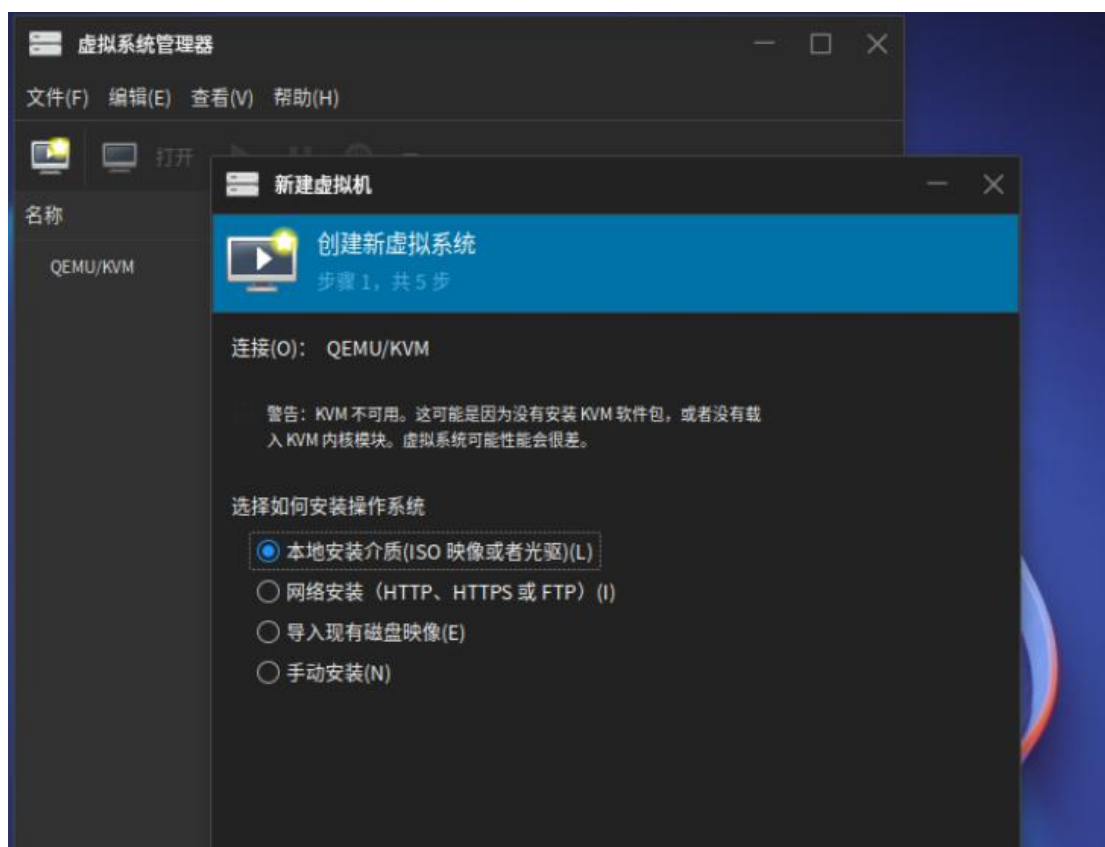
7月 08 02:16:48 localhost.localdomain dnsmasq-dhcp[151080]: DHCP, IP range 192.168.122.2 -- 192.168.122.254, lease time 1h
7月 08 02:16:48 localhost.localdomain dnsmasq-dhcp[151080]: DHCP, sockets bound exclusively to interface virbr0
```

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

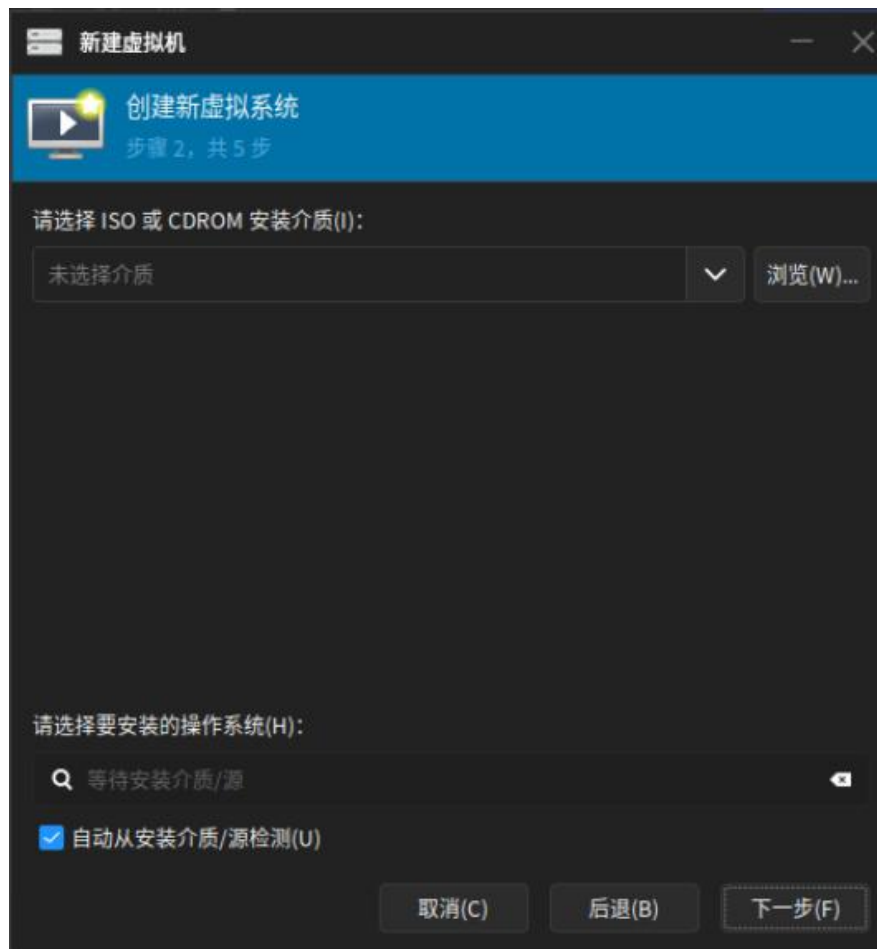


11.2 配置流程

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：

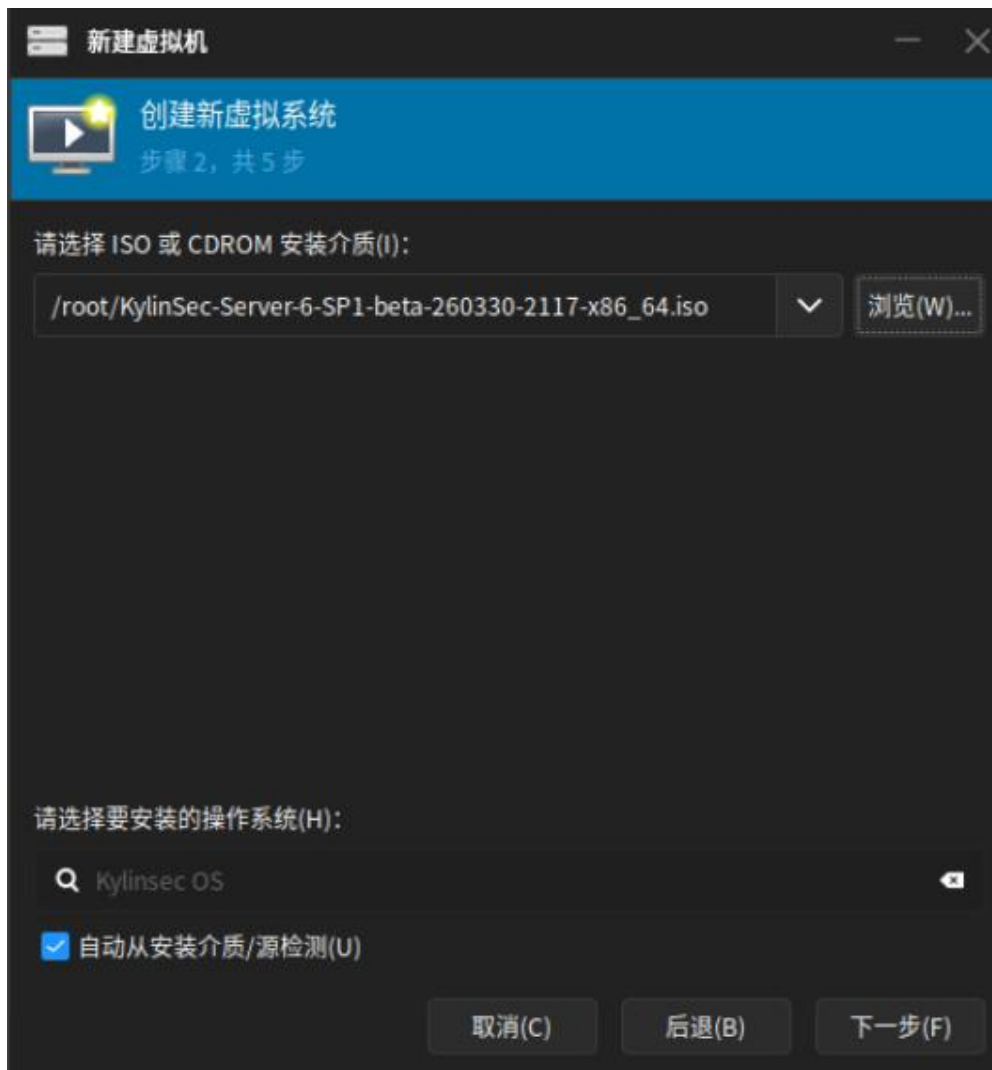


点击“浏览”进入到介质选择界面，如下图所示：

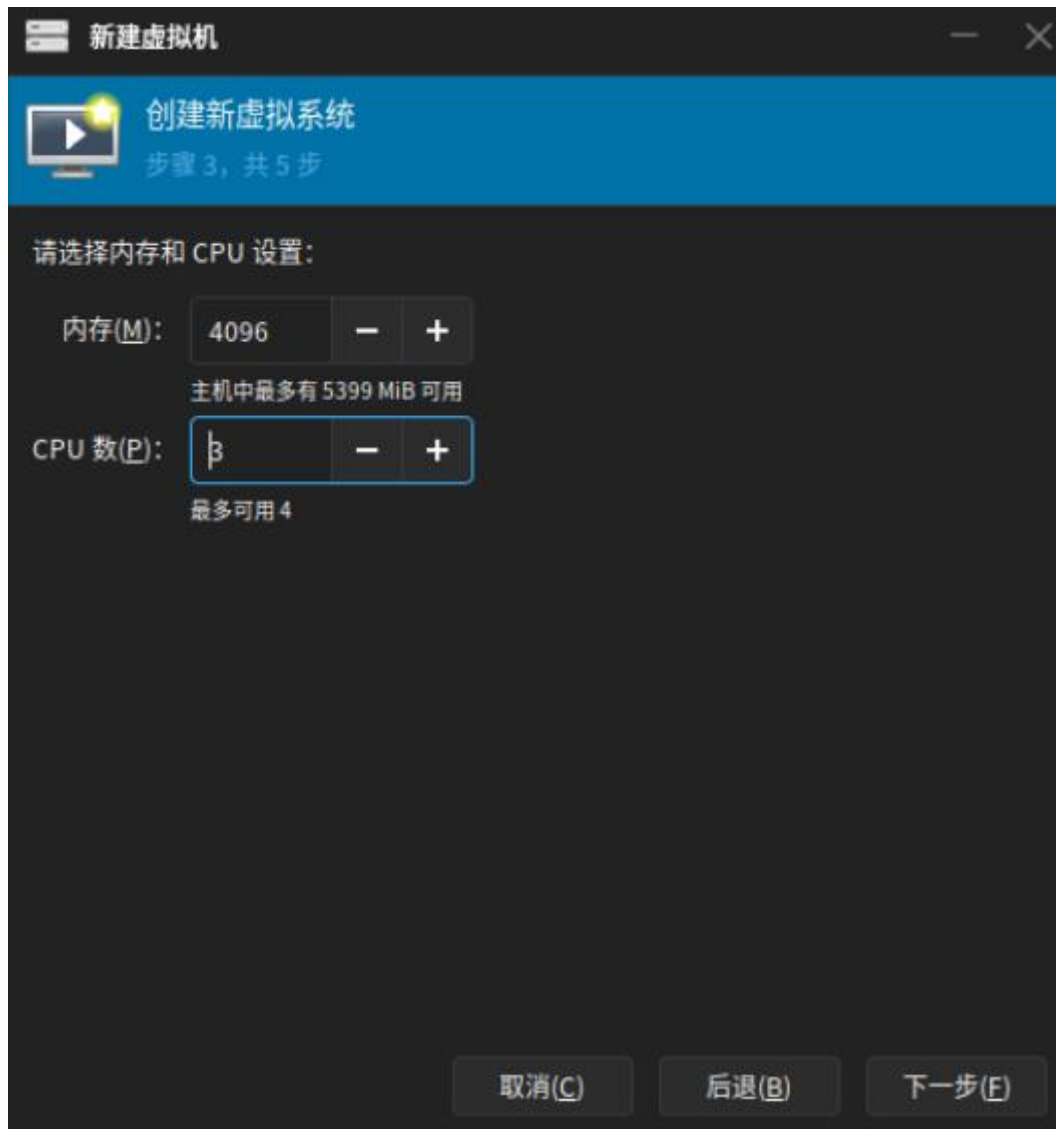


在该界面中，选择下方的<本地浏览>，打开文件浏览器，根据使用者自身镜像放置位置选择路径并确认回到该界面。

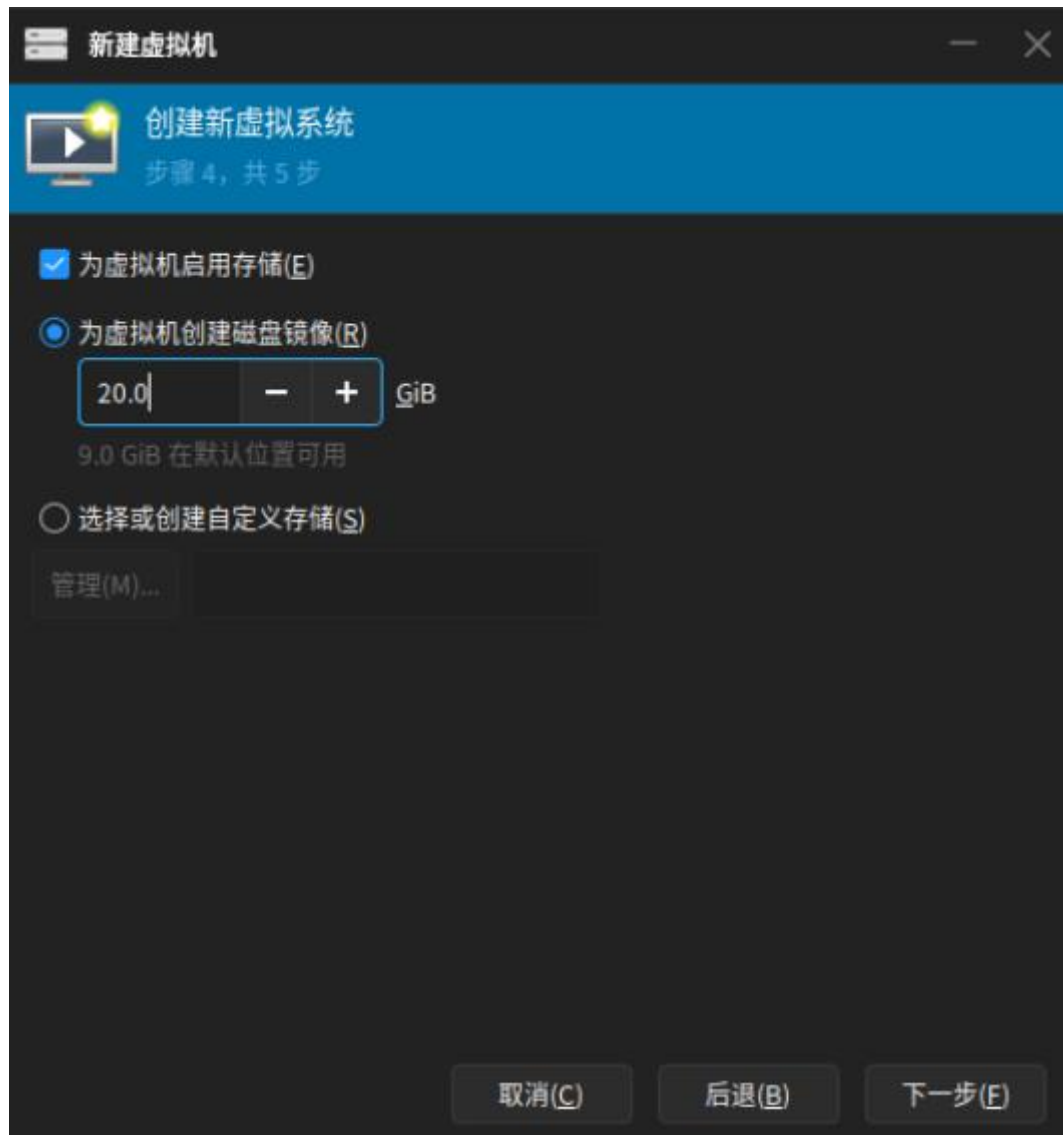
需要选择安装的操作系统，勾选下方的自动检测勾选项，如下图所示：



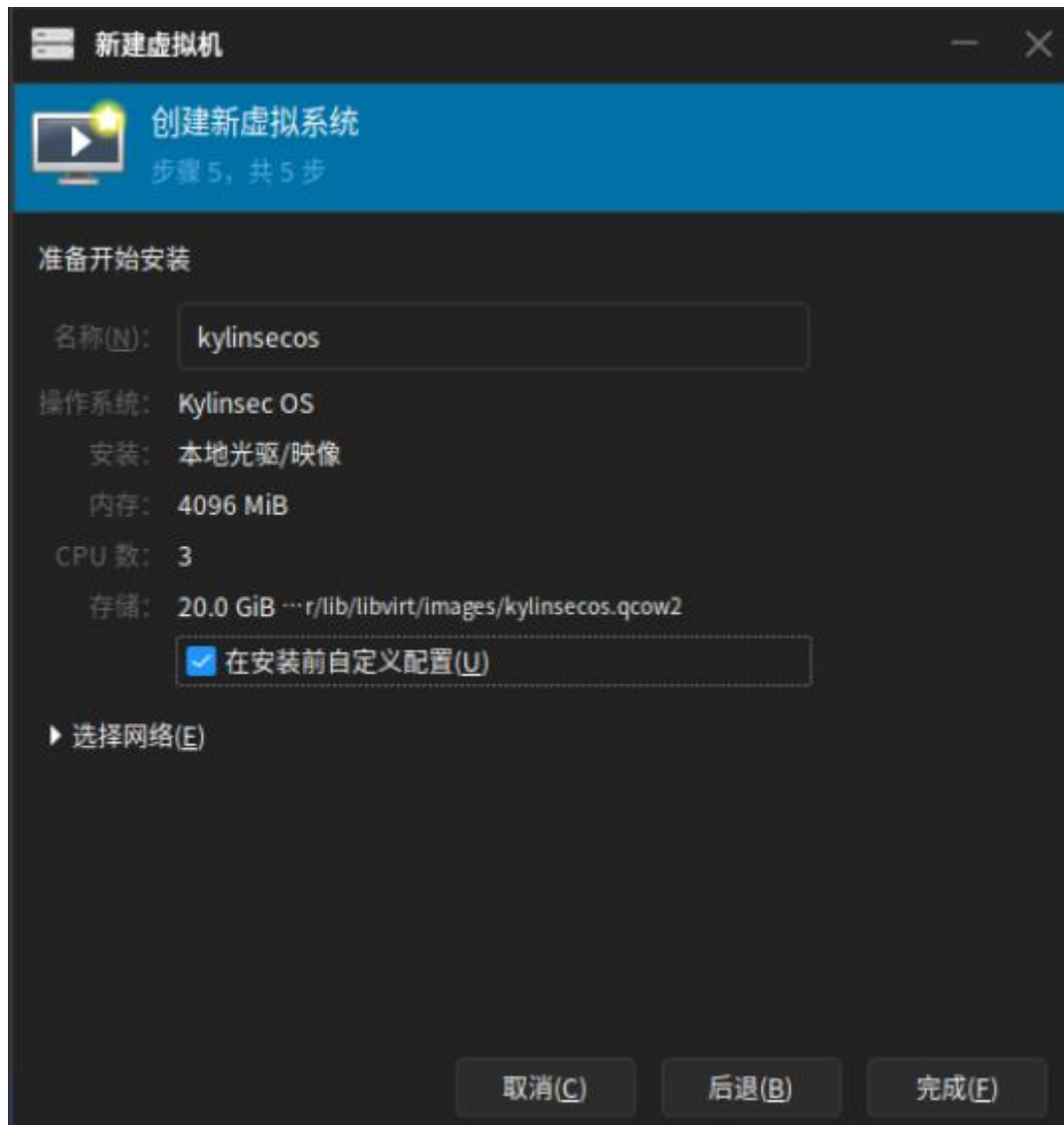
点击“前进”到下一步，设置内存和 CPU，建议内存大于等于 4G，CPU 大于等于 2，如下图所示：



设置完成后点击“前进”到下一步，如下图所示：



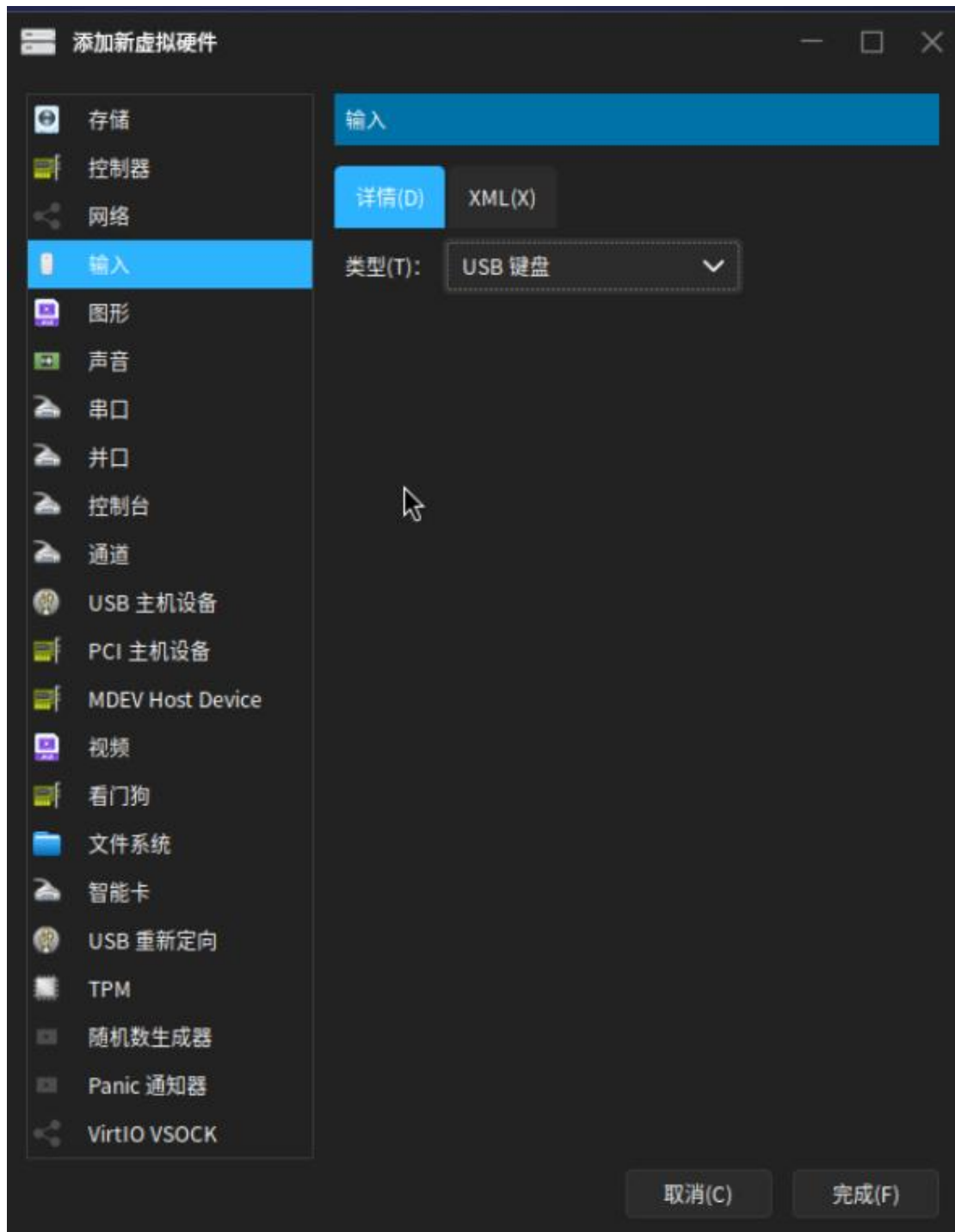
磁盘大小默认是 20G，但是对于 V6 SP1 版本的安装，最小磁盘大小不能低于 40G，建议设置为 40G，设置完成后点击“前进”到下一步，如下图所示：



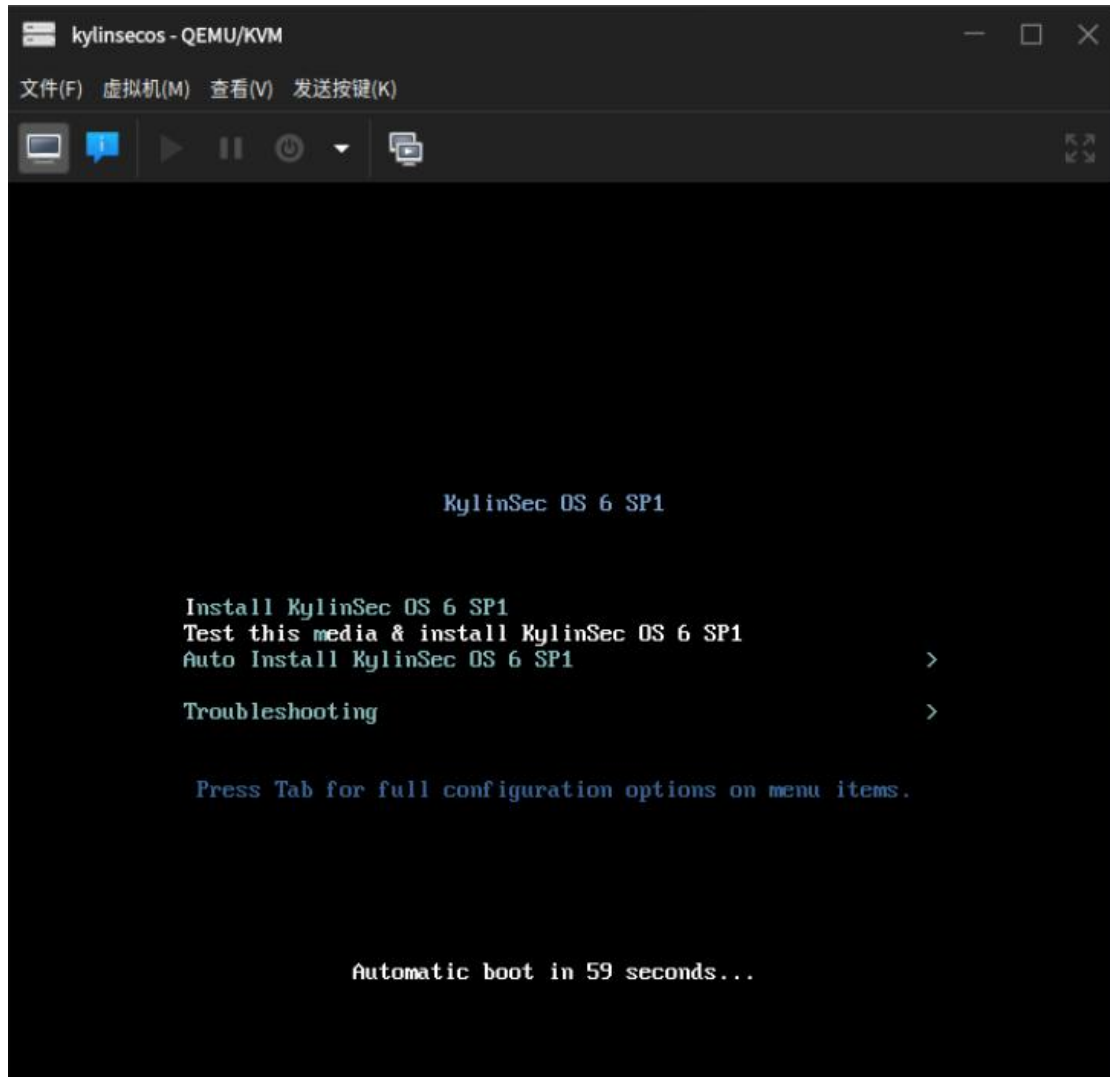
在名称输入框输入需要修改的名称，勾选“在安装前自定义配置”，点击选择网络，然后点击“完成”按钮，弹出配置界面如下图：



点击“添加硬件”，分别添加输入（USB Keyboard）、输入（USB Mouse）、图形（Spice 服务器，新版本的 Virt-Manager 已经自带了 VNC 图形适配，建议根据需求进行添加配置），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 6 SP1”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。



12 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/efi/EFI/KylinSecOS/grub.cfg, 在对应 menuentry 内核条目的 linux 条目中添加 rd.driver.blacklist=nouveau nouveau.modeset=0 参数，修改 /usr/lib/modprobe.d/dist-blacklist.conf 文件，添加下面的条目在末尾

```
#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) 安装完成前选择装配 X Windows 配置并自动备份相关配置 conf 文件
- 8) 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：（一般情况下直接使用本地驱动就行）

- 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-install` 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：`./amdgpc-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-xxx.x86_64.rpm`。
- 2) 解压缩 `qla2xxx-v8.08-xxx.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-xxx.x86_64.rpm` 安装驱动包

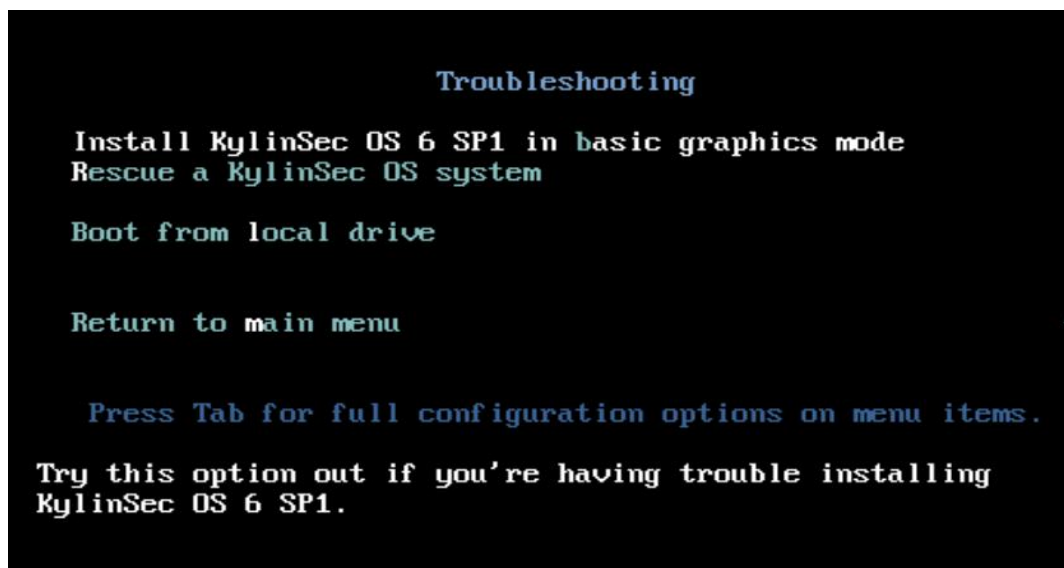
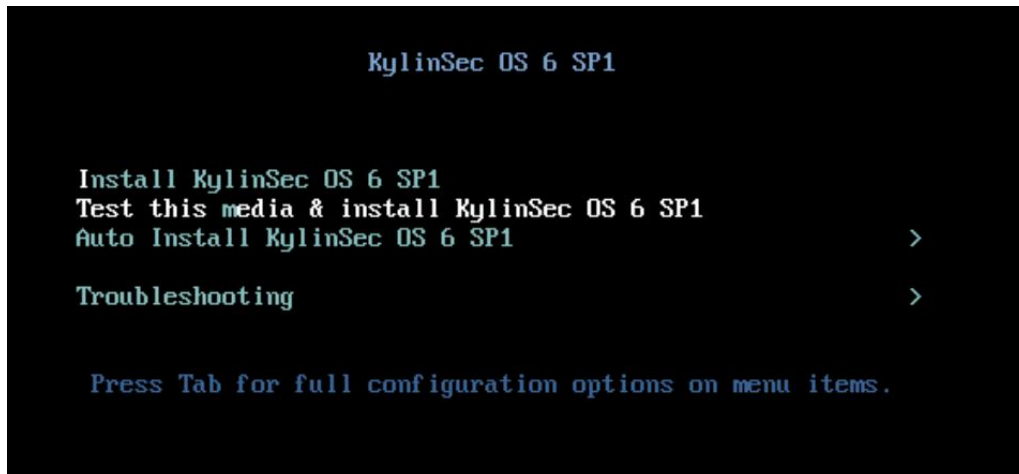
4、问题描述：如果要强制使用 gpt 分区，该怎么操作？

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？

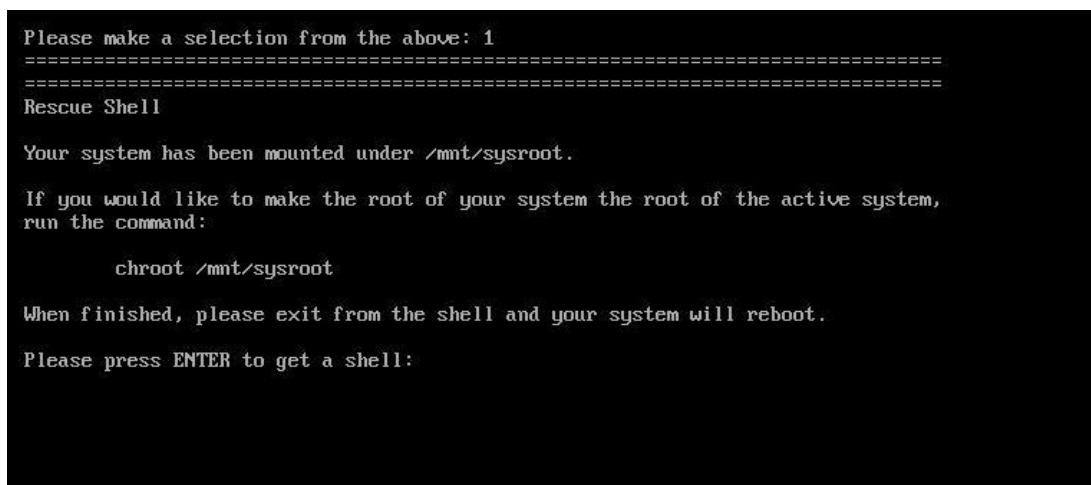
解决方法：

- 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” -> “Rescue a KylinSec OS 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：



3) 输入 chroot /mnt/sysimage 命令

4) 输入 `grub2-install /dev/sda` 重装 grub

5) 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：

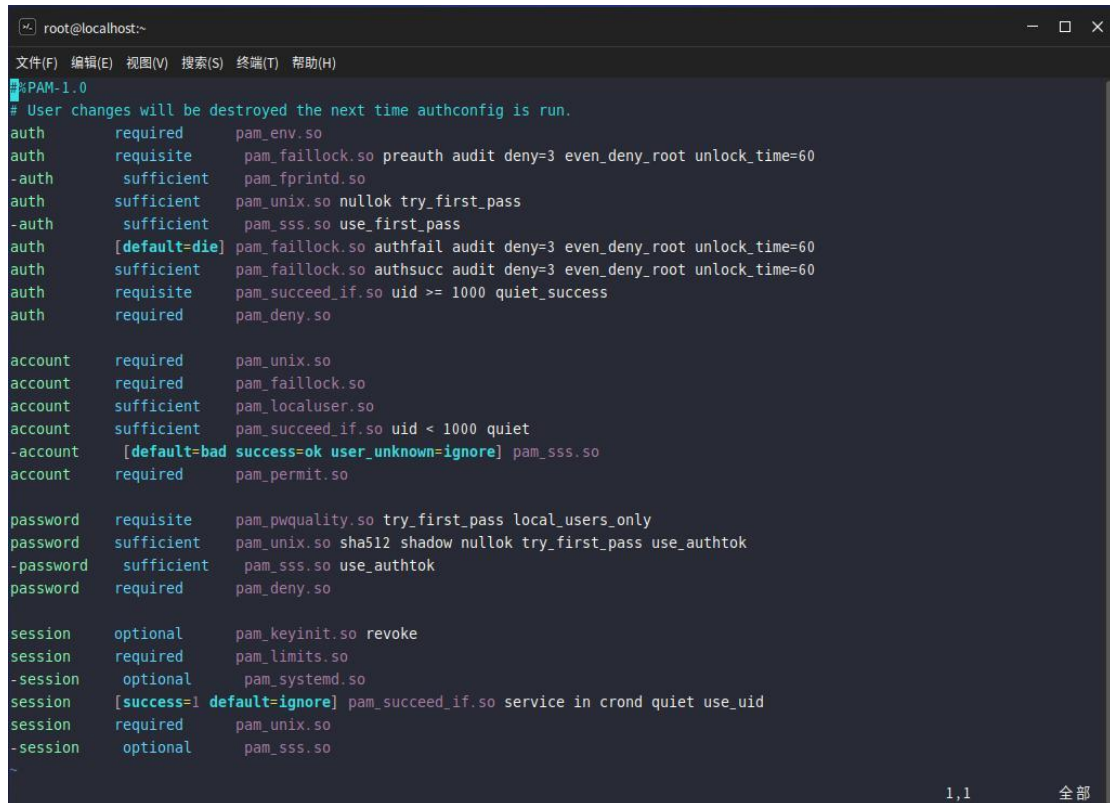
修改 `/etc/default/grub` 文件，通过修改配置文件里的 `GRUB_DEFAULT=0` 可以指定要启动的系统，`GRUB_DEFAULT=0` 为启动第一个系统

再用 `grub2-mkconfig > /boot/efi/EFI/KylinSecOS/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_denied_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_denied_root unlock_time=60
auth      sufficient     pam_faillock.so authsucc audit deny=3 even_denied_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

1,1 全部
```

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
```

```
dcrcdit=-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

dcrcdit=-1: 最少 1 个数字

lccredit=-1: 最少 1 个小写字符, (uccredit=-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.insettings[1924]: [ 1682233402.098801]: FCITX[2522]: I/O warning : failed to load external entity ""
Apr 23 15:03:22 localhost com.redhat.insettings[1924]: [ 1682233402.098841]: FCITX[2522]: I/O warning : failed to load external entity ""
Apr 23 15:03:22 localhost com.redhat.insettings[1924]: [ 1682233402.528780]: FCITX[2522]: I/O warning : failed to load external entity ""
Apr 23 15:03:22 localhost com.redhat.insettings[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(WN) warning, (EE) error, (NI) not implemented, (??) unknown.
Apr 23 15:02:52 localhost lightdm-kiran-greeter[1805]: klog.qts.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 15:03:36 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：

/var/log

1./var/log/messages：包括整体系统信息，其中也包含系统启动期间的日志。此外，mail，cron，daemon，kern 和 auth 等内容也记录在 var/log/messages 日志中。

2./var/log/dmesg：包含内核缓冲信息（kernel ring buffer）。在系统启动时，会在屏幕上显示许多与硬件有关的信息。可以用 dmesg 查看它们。

3./var/log/audit/：包含被 Linux audit daemon 储存的信息。例如：selinux 开启的时候，这里就会有有关 selinux 审计的日志。

4./var/log/boot.log：包含系统启动时的日志，包括自启动的服务。

5. /var/log/daemon.log：包含各种系统后台守护进程日志信息。

6./var/log/dpkg.log：包括安装或 dpkg 命令清除软件包的日志。

7./var/log/kern.log：包含内核产生的日志，有助于在定制内核时解决问题。

8./var/log/lastlog: 记录所有用户的最近信息。这不是一个 ASCII 文件, 因此需要用 lastlog 命令查看内容。

9./var/log/maillog 或 /var/log/mail.log: 包含来自系统运行电子邮件服务器的日志信息。例如, sendmail 日志信息就全部送到这个文件中。

10./var/log/user.log: 记录所有等级用户信息的日志。

11./var/log/Xorg.x.log: 来自 X 的日志信息。

12./var/log/alternatives.log: 更新替代信息都记录在这个文件中。

13./var/log/btmp: 记录所有失败登录信息。使用 last 命令可以查看 btmp 文件。例如, " last -f /var/log/btmp | more "。

14./var/log/cups: 涉及所有打印信息的日志, 即 cups 打印服务运行的日志。

15./var/log/anaconda.log 或 /var/log/anaconda/: 在安装 Linux 时, 所有安装信息都储存在这个文件中。

16./var/log/yum.log: 包含使用 yum 安装的软件包信息。

17./var/log/cron: 每当 cron 进程开始一个工作时, 就会将相关信息记录在这个文件中。

18./var/log/secure: 包含验证和授权方面信息。例如, sshd 会将所有信息记录 (其中包括失败登录) 在这里。

19./var/log/wtmp 或 /var/log/utmp: 包含登录信息。使用 wtmp 可以找出谁正在登陆进入系统, 谁使用命令显示这个文件或信息等。

20./var/log/faillog - 包含用户登录失败信息。此外, 错误登录命令也会记录在本文件中。

除了上述 Log 文件以外，/var/log 还基于系统的具体应用包含以下一些子目录：

- 1./var/log/httpd/ 或 /var/log/apache2：包含服务器 access_log 和 error_log 信息。
- 2./var/log/lighttpd/：包含 light HTTPD 的 access_log 和 error_log。
- 3./var/log/mail/：这个子目录包含邮件服务器的额外日志。
- 4./var/log/prelink/：包含.so 文件被 prelink 修改的信息。
- 5./var/log/audit/：包含被 Linux audit daemon 储存的信息。例如：selinux 开启的时候，这里就会有有关 selinux 审计的日志。
- 6./var/log/samba/：包含由 samba 存储的信息。
- 7./var/log/sa/：包含每日由 sysstat 软件包收集的 sar 文件。
- 8./var/log/sss/：用于守护进程安全服务。

13 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：<https://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：



📍 总部: 湖南省长沙市岳麓区麒麟科技园
📈 股票代码: 688152
🌐 www.kylinsec.com.cn
☎ 咨询热线: 400-012-6606
湖南麒麟信安科技股份有限公司

