Linux服务器构建与运维管理

从基础到实战（基于 openEuler）

第3章：系统配置

阮晓龙

13938213680 / ruanxiaolong@hactcm.edu.cn

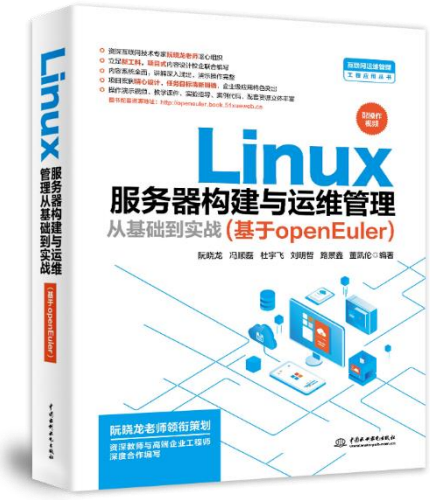
https://internet.hactcm.edu.cn
http://www.51xueweb.cn

河南中医药大学信息技术学院互联网技术教学团队
河南中医药大学医疗健康信息工程技术研究所

2024.9

提纲

- 存储管理
  - tar、zip/unzip、gzip/gunzip
  - df、du、mount/umount
  - fdisk、mkfs、badblocks
  - LVM、RAID
- 网络管理
  - nmcli、nmtui、Bond
- 进程管理
  - ps、nice、renice、jobs、kill
- 任务计划
  - at、crontab



1. 存储管理

- tar 是 Linux 操作系统下经常用到的归档工具。
  - tar 命令用于把多个文件和目录打包成一个文件并归档。
  - 在 Linux 中很多压缩程序都只能针对一个文件进行压缩，因此要压缩一个目录或多个文件时，需要将其先打包为一个文件，然后再进行压缩。
  - tar通常和压缩工具结合使用。

- 打包与压缩的不同
  - 打包：将多个文件或者目录变为一个文件。
  - 压缩：将一个文件通过压缩算法变为一个更小的文件，占用更少的存储空间。

1. 存储管理

tar [选项] [参数]

功能：

- 文件和目录创建档案。

参数/命令：

- 文件或目录：指定要打包的文件或目录列表。

主要选项：

- -A：新增文件到以存在的备份文件
- -B：设置区块大小
- -c或--create：建立新的备份文件
- -C <目录>：
  - 用在解压缩，在特定目录解压缩。
- -d：记录文件的差别
- -x或--extract或--get：从备份文件中还原文件
- -t或--list：列出备份文件的内容
- -z或--gzip或--ungzip：通过gzip指令处理备份文件
- -Z或--compress或--uncompress：
  - 通过compress指令处理备份文件
- -f<备份文件>或--file=<备份文件>：指定备份文件
- -v或--verbose：显示指令执行过程

1. 存储管理

zip [选项] [参数]

功能：

- 创建.zip格式的压缩文件。

说明：

- 常用选项：
  - -r：递归压缩，用于压缩目录及其子目录中的所有文件。
  - -q：安静模式，不显示压缩进度信息。
  - -m：压缩后删除原文件。
  - -v：显示详细的压缩过程信息。
- 备注：
  - zip 和 unzip 通常使用 DEFLATE 压缩算法。
  - DEFLATE 是一种无损数据压缩算法，它结合了 LZ77 算法和哈夫曼编码。

unzip [选项] [参数]

功能：

- 解压缩.zip格式的归档文件。

说明：

- 常用选项：
  - -l：列出压缩包中的文件列表，但不进行解压缩。
  - -d：指定解压缩的目标目录。
  - -n：不覆盖已存在的文件。
  - -o：覆盖已存在的文件且不提示。

1. 存储管理

df [选项] [参数]

功能：

- 显示磁盘分区上的可使用的磁盘空间。

说明：

- 常用选项：
  - -a或--all：包含全部的文件系统
  - -h或--human-readable：以可读性较高的方式来显示信息
  - -H或--si：与-h参数相同，但在计算时是以1000 Bytes为换算单位而非1024 Bytes
  - -k或--kilobytes：指定区块大小为1024字节
  - -m或--megabytes：指定区块大小为1048576字节
  - -l或--local：仅显示本地端的文件系统
  - -i或--inodes：显示inode的信息
  - -t<文件系统类型>或--type=<文件系统类型>：仅显示指定文件系统类型的磁盘信息
  - -T或--print-type：显示文件系统的类型

du [选项] [参数]

功能：

- 查看文件和目录使用的磁盘空间。

说明：

- 常用选项：
  - -a或-all：显示目录中个别文件的大小。
  - -c或--total：除了显示个别目录或文件的大小外，同时也显示所有目录或文件的总和。
  - -s或--summarize：仅显示总计，只列出最后加总的值。
  - -S或--separate-dirs：显示目录大小时不含其子目录。
  - -h或--human-readable：
    - 以K，M，G为单位，提高信息的可读性。
  - -b或-bytes：显示目录或文件大小时，以byte为单位。
  - -k或--kilobytes：以KB(1024bytes)为单位输出。
  - -m或--megabytes：以MB为单位输出。

10.10.2.31 ([openEuler])

Terminal Sessions View X server Tools Games Settings Macros Help

Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit

Quick connect...

/root/

Name

..
.ssh
.bash_history
.bash_logout
.bash_profile
.bashrc
.cshrc
.tcshrc
anaconda-ks.cfg

Remote monitoring

Follow terminal folder

3. 10.10.2.31 ([openEuler])

```
[root@Project-03-Task-01 ~]# df
文件系统                   1K的块    已用     可用 已用% 挂载点
/dev/mapper/openeuler-root 17365776 1817328 14640980   12% /
devtmpfs                      4096       0     4096    0% /dev
tmpfs                       214024       0   214024    0% /dev/shm
tmpfs                         4096       0     4096    0% /sys/fs/cgroup
tmpfs                        85612    2860    82752    4% /run
tmpfs                       214024       0   214024    0% /tmp
/dev/sda2                   996780  266048   661920   29% /boot
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# df -ah
文件系统                   大小  已用  可用 已用% 挂载点
/dev/mapper/openeuler-root   17G  1.8G   14G   12% /
devtmpfs                   4.0M     0  4.0M    0% /dev
tmpfs                      210M     0  210M    0% /dev/shm
devpts                        0     0     0     - /dev/pts
sysfs                         0     0     0     - /sys
securityfs                    0     0     0     - /sys/kernel/security
tmpfs                      4.0M     0  4.0M    0% /sys/fs/cgroup
cgroup                        0     0     0     - /sys/fs/cgroup/systemd
cgroup                        0     0     0     - /sys/fs/cgroup/hugetlb
cgroup                        0     0     0     - /sys/fs/cgroup/net_cls,net_prio
cgroup                        0     0     0     - /sys/fs/cgroup/cpu,cpuacct
cgroup                        0     0     0     - /sys/fs/cgroup/rdma
cgroup                        0     0     0     - /sys/fs/cgroup/memory
cgroup                        0     0     0     - /sys/fs/cgroup/freezer
cgroup                        0     0     0     - /sys/fs/cgroup/blkio
cgroup                        0     0     0     - /sys/fs/cgroup/cpuset
cgroup                        0     0     0     - /sys/fs/cgroup/misc
cgroup                        0     0     0     - /sys/fs/cgroup/perf_event
cgroup                        0     0     0     - /sys/fs/cgroup/devices
cgroup                        0     0     0     - /sys/fs/cgroup/pids
bpf                           0     0     0     - /sys/fs/bpf
configfs                      0     0     0     - /sys/kernel/config
proc                          0     0     0     - /proc
tmpfs                       84M  2.8M   81M    4% /run
selinuxfs                     0     0     0     - /sys/fs/selinux
systemd-1                     -     -     -     - /proc/sys/fs/binfmt_misc
hugetlbfs                     0     0     0     - /dev/hugepages
tracefs                       0     0     0     - /sys/kernel/tracing
mqueue                        0     0     0     - /dev/mqueue
debugfs                       0     0     0     - /sys/kernel/debug
tmpfs                      210M     0  210M    0% /tmp
fusectl                       0     0     0     - /sys/fs/fuse/connections
/dev/sda2                  974M  260M  647M   29% /boot
binfmt_misc                   0     0     0     - /proc/sys/fs/binfmt_misc
[root@Project-03-Task-01 ~]#
```

Project-03-Task-01 | 1% | 0.22 GB / 0.41 GB | 0.01 Mb/s | 0.00 Mb/s | 6 hours | root | /: 12% | /tmp: 0% | /boot: 29%

10.10.2.31 ([openEuler])
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
3. 10.10.2.31 ([openEuler])
/root/
Name
..
.ssh
.bash_history
.bash_logout
.bash_profile
.bashrc
.cshrc
.tcshrc
anaconda-ks.cfg
[root@Project-03-Task-01 ~]# du
4 ./.ssh
40 .
[root@Project-03-Task-01 ~]# du /var/log
4 /var/log/chrony
4 /var/log/private
12 /var/log/tuned
740 /var/log/audit
4 /var/log/samba/old
8 /var/log/samba
4308 /var/log/anaconda
4 /var/log/sssd
6796 /var/log
[root@Project-03-Task-01 ~]# du -h /var/log
4.0K /var/log/chrony
4.0K /var/log/private
12K /var/log/tuned
740K /var/log/audit
4.0K /var/log/samba/old
8.0K /var/log/samba
4.3M /var/log/anaconda
4.0K /var/log/sssd
6.7M /var/log
[root@Project-03-Task-01 ~]#
Remote monitoring
Follow terminal folder
Project-03-Task-01 1% 0.22 GB / 0.41 GB 0.01 Mb/s 0.00 Mb/s 6 hours root /: 12% /tmp: 0% /boot: 29%
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

1. 存储管理

- 挂载磁盘是将存储设备（如硬盘、U 盘、光盘等）连接到文件系统树中的过程，以便能够访问存储设备上的文件和目录。
- 在 Linux 中，挂载磁盘包括临时挂载和开机自动挂载两种方式。
  - 临时挂载：
    - 使用 mount 进行挂载，使用 umount 删除挂载。
    - 临时挂载的流程：确定磁盘信息 -> 创建空目录 -> mount挂载
  - 开机自动挂载：
    - /etc/fstab 文件是 Linux 系统中用于配置文件系统自动挂载的文件。
    - 修改配置文件 /etc/fstab ：
      - # <设备文件> <挂载点> <文件系统类型> <挂载选项> <dump 选项> <fsck 顺序>
      - UUID=12345678-9abc-def0-1234-56789abcdef0 /mnt/data ext4 defaults 0 2
      - /dev/sdb1 /mnt/usb vfat defaults 0 0
      - 192.168.1.100:/export/share /mnt/remote nfs defaults 0 0

1. 存储管理

- /etc/fstab 文件格式：<设备文件> <挂载点> <文件系统类型> <挂载选项> <dump 选项> <fsck 顺序>
  - 设备文件：可以是磁盘设备的路径，也可以是设备的 UUID（通用唯一识别码）或 LABEL（标签）。
  - 挂载点：文件系统在文件系统树中的挂载位置，即一个已存在的空目录。
  - 文件系统类型：
    - 指定要挂载的文件系统类型，常见的有 ext4、vfat、ntfs、iso9660、nfs等。
  - 挂载选项：用于指定文件系统的挂载选项：
    - defaults：使用默认的挂载选项，通常包括读写权限、自动挂载等。
    - ro（只读）：以只读方式挂载文件系统。
    - rw（读写）：以读写方式挂载文件系统。
    - sync（同步）：所有的文件操作都会同步写入磁盘，确保数据的一致性，但可能会影响性能。
    - async（异步）：文件操作会先保存在内存缓冲区中，然后在合适的时候异步写入磁盘，提高性能但可能会有数据丢失的风险（如果系统突然崩溃，缓冲区中的数据可能没有及时写入磁盘）。
  - dump 选项：
    - 用于备份工具 dump 的参数。通常设置为 0 表示不备份，或者 1 表示需要备份。
  - fsck 顺序：
    - 用于文件系统检查工具 fsck 的参数。
    - 根文件系统通常设置为 1，其他文件系统设置为 2 或 0。
    - 设置为 0 表示在系统启动时不进行文件系统检查。

1. 存储管理

mount [选项] [参数]

功能：

说明：

- 选项：
  - -t <文件系统类型>：指定文件系统类型
  - -l：显示已加载的文件系统列表
  - -v：冗长模式，输出指令执行的详细信息
  - -n：加载没有写入文件“/etc/mtab”中的文件系统
  - -r：将文件系统加载为只读模式
  - -a：加载文件“/etc/fstab”中描述的所有文件系统
- 参数：
  - 设备文件：指定要加载的文件系统对应的设备名
  - 加载点：指定加载点目录

umount [选项] [参数]

功能：

- 卸载已加载的文件系统。

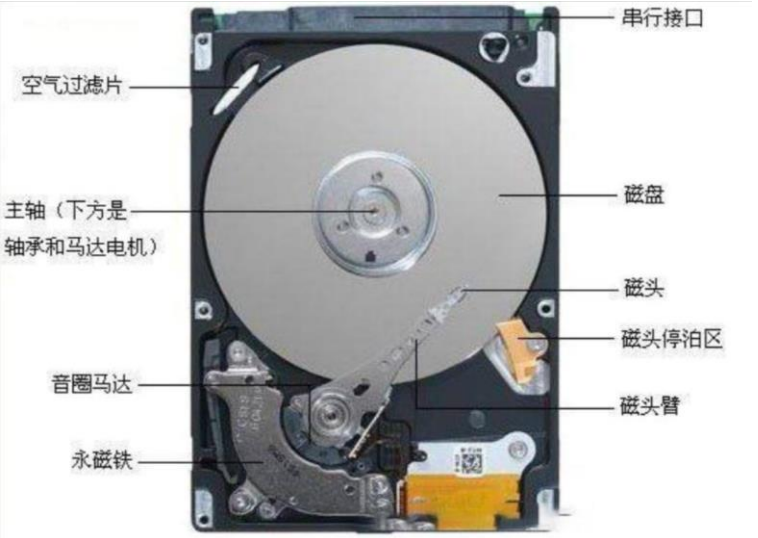
说明：

- 选项：
  - -a：卸除/etc/mtab中记录的所有文件系统
  - -n：卸除时不要将信息存入/etc/mtab文件中
  - -r：
    - 若无法成功卸除，则尝试以只读的方式重新挂入文件系统
  - -t <文件系统类型>：
    - 仅卸除选项中所指定的文件系统
  - -v：执行时显示详细的信息

1. 存储管理

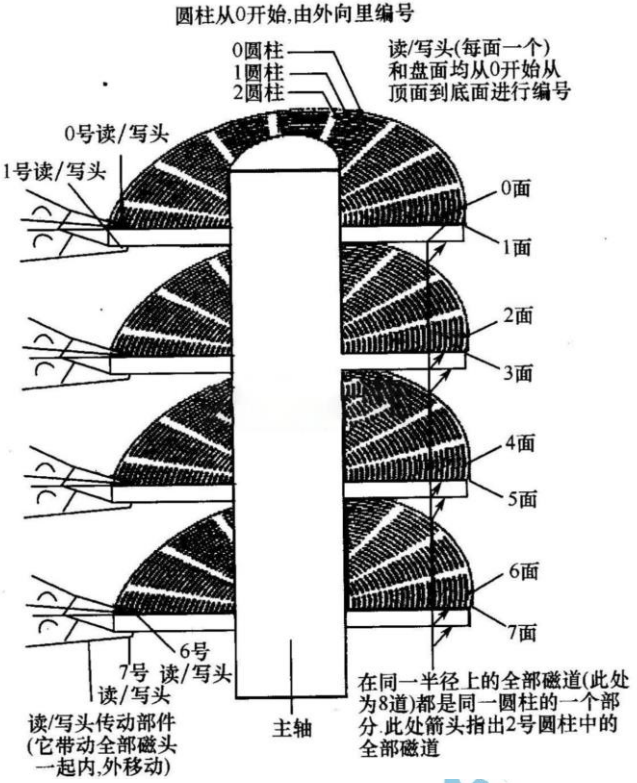
认识磁盘

- 磁盘是一种计算机的外部存储器设备，由一个或多个覆盖有磁性材料的铝制或玻璃制的碟片组成，用来存储用户的信息，这种信息可以反复地被读取和改写。
- 绝大多数磁盘被永久封存在一个密封的盒子里。
- 简单来说就是多个盘片之间靠主轴连接，电机带动主轴做旋转运动，通过多个磁头臂的摇摆和磁盘的旋转，磁头就可以在磁盘旋转的过程中就读取到磁盘中存储的各种数据。



1. 存储管理

- 磁盘的扇区、磁道、柱面：
  - 磁道：磁盘的每个盘面被划分为许多同心圆，这些同心圆的轨道叫做磁道。
  - 扇区：
    - 一个盘面划分为若干个内角相同的扇形，这样盘面上的每个磁道就被分为若干段圆弧，每段圆弧叫做一个扇区。
    - 每个扇区中的数据作为一个单元同时被读入或写入。
    - 每一个扇区是512字节，其中有64个字节存储的是分区表，一条分区信息占16个字节。
  - 柱面：
    - 每一个盘片同一大小的同心圆可以看成连在一起的柱面，磁盘在分区的时候最小单位是柱面，每一个盘片的上下面都可以读取数据，每一个磁头，不可以跨盘面读取数据。

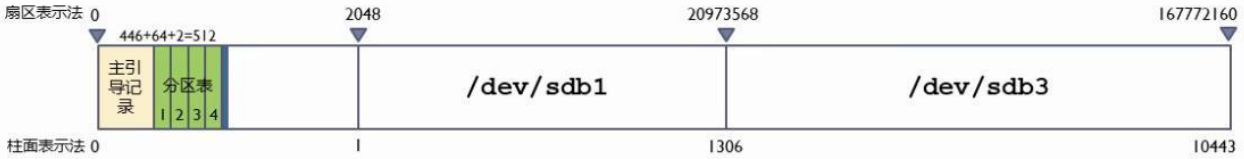


1. 存储管理

- 磁盘的分类：
  - IDE磁盘：
    - 特点价格低廉，兼容性强，性价比高，数据传输慢，不支持热插拔等。
    - Linux的命名为：/dev/hd[a-d]
  - SCSI磁盘：
    - 传输速率高，读写性能好，运行稳定，可连接多个设备。
    - 可支持热插拔，占用CPU低，但是价格相对来说比较贵，一般用于工作站或服务器上。
    - Linux的命名为：/dev/sd[a-p]
  - SATA磁盘：
    - 结构简单、支持热插拔。
    - Linux的命名为：/dev/sd[a-p]
  - 固态硬盘：
    - SSD磁盘，但是严格上来讲不是磁盘，只是习惯性的命名而已。

1. 存储管理

- 为什么要对磁盘进行分区：
  - 易于管理和使用。
  - 有利于数据的安全。
  - 节约寻找文件的时间。
- 磁盘分区的信息存储方式有 MBR 和 GPT 两种。
  - MBR 是传统的磁盘分区表格式，使用 32 位的扇区寻址方式，支持最大 2TB 的磁盘容量。
    - MBR 将磁盘第一个扇区（512 字节）称为主引导记录，包含引导代码、分区表和结束标志等信息。
    - MBR 分区表最多包含四个主分区，或者三个主分区和一个扩展分区，扩展分区可划分逻辑分区。
  - GPT 是较新的磁盘分区表格式，使用全局唯一标识符（GUID）来标识分区，支持更大的磁盘容量和更多的分区数量。
    - GPT 理论上支持无限多个分区，每个分区可以有最大 8ZiB（9.4445 万亿亿字节）的容量。
- 硬盘必须要分区么？
  - 不需要。可根据习惯进行分区。
  - 磁盘可以先分区，再对分区格式化。也可以直接对磁盘格式化进行使用。

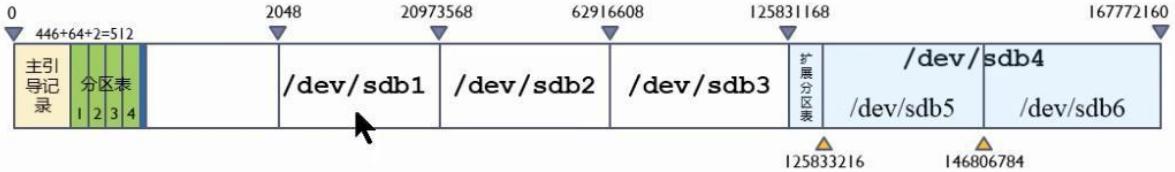



```
# fdisk -l /dev/sdb

Disk /dev/sdb: 85.9 GB, 85899345920 bytes, 167772160 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x0317f738

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048    20973567    10485760   83  Linux
/dev/sdb3        20973568   167772159    73399296   83  Linux
```

总大小

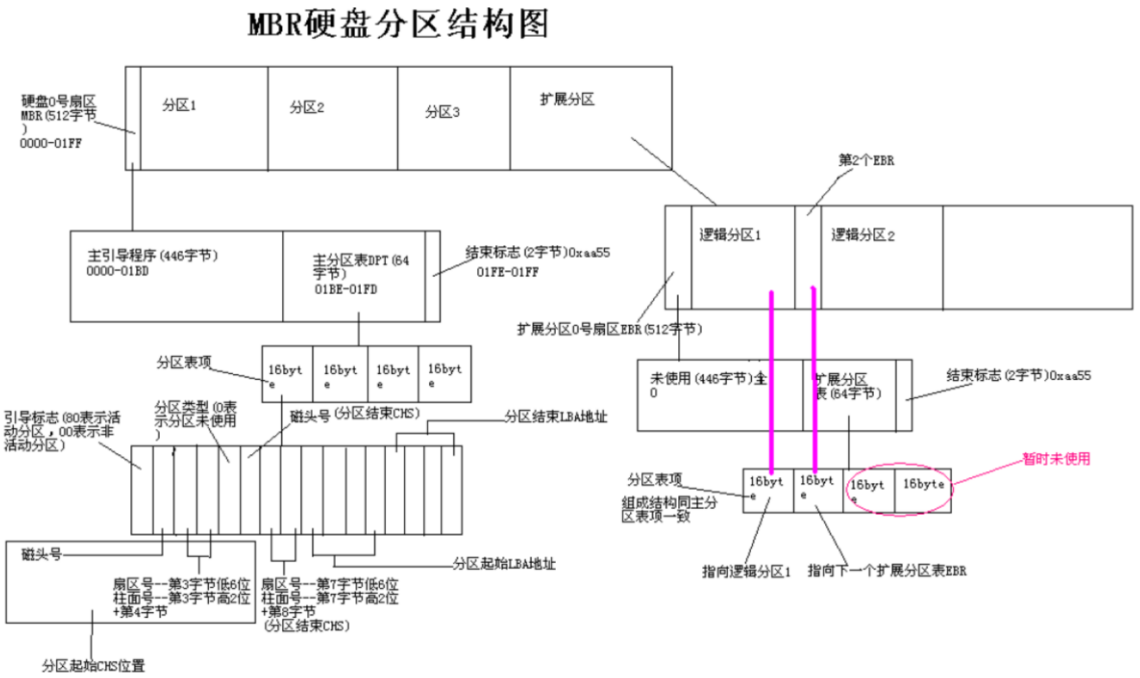
ctyun/刘****林

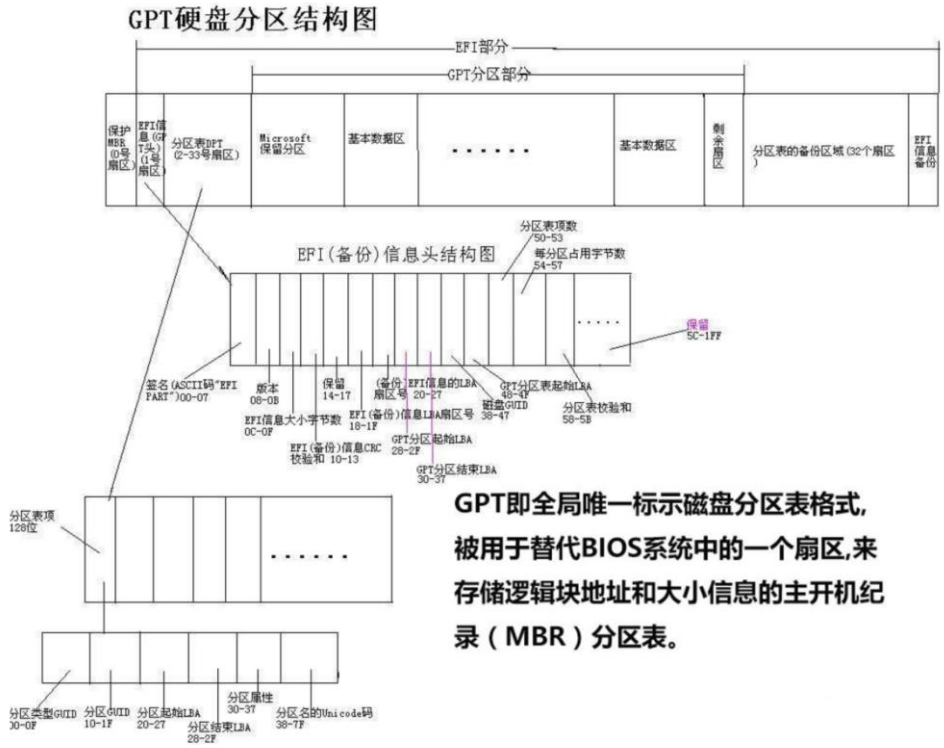
MBR分区的磁盘示例




```
# fdisk -l /dev/sdb

Disk /dev/sdb: 85.9 GB, 85899345920 bytes, 167772160 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x0317f738

 Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048    20973567    10485760   83  Linux
/dev/sdb2        20973568    62916607    20971520   83  Linux
/dev/sdb3        62916608   125831167    31457280   83  Linux
/dev/sdb4       125831168   167772159    20970496    5  Extended
/dev/sdb5       125833216   146804735    10485760   83  Linux
/dev/sdb6       146806784   167772159    10482688   83  Linux
```

ctyun/刘****林





GPT即全局唯一标示磁盘分区表格式，被用于替代BIOS系统中的一个扇区，来存储逻辑块地址和大小信息的主开机纪录（MBR）分区表。

- 如何获知磁盘使用的是 MBR 分区表还是 GPT 分区表，以及两种分区表如何互相转换
- https://forum.openeuler.org/t/topic/4012

1. 存储管理

- 文件系统
  - 是管理文件和目录的一套机制，基本数据单位是文件。
  - 主要是对磁盘上的文件进行组织管理，组织的方式不同，形成的文件系统也会不同。
- 格式化
  - 指将分区格式化成不同的文件系统。
- Linux支持的文件系统
  - Linux下的文件类型有ext2、ext3、ext4、xfs等。
  - 使用 mkfs.，之后按 tab 键盘，可以查看当前操作系统支持的文件系统。
  - 例如：
    - [root@Project-03-Task-01 ~]# mkfs.
    - mkfs.cramfs mkfs.ext2 mkfs.ext3 mkfs.ext4 mkfs.minix mkfs.xfs
  - cat /proc/filesystems
  - ls -l /lib/modules/$(uname -r)/kernel/fs

1. 存储管理

- Linux 安装完成后会创建一些默认的目录，这些默认目录是有特殊功能的。
- 用户在不确定的情况下最好不要更改这些目录下的文件，以免造成系统错误。

表 3-1-2　常用默认目录及其说明

目录	说明
/	Linux 文件系统的入口，也是整个文件系统的最顶层目录
/bin	存放可执行的命令文件，供系统管理员和普通用户使用
/boot	存放内核镜像及引导系统所需要的文件
/dev	存放设备文件
/etc	存放系统配置文件
/home	存放普通用户的个人主目录
/lib	存放库文件
/lost+found	存放因系统意外崩溃或机器意外关机而产生的文件碎片，当系统启动的过程中 fsck 工具会检查这个目录，并修复受损的文件系统
/proc	一个实时的、驻留在内存中的文件系统，用于存放操作系统、运行进程以及内核等信息
/root	用户默认主目录
/tmp	临时文件目录，用户运行程序时所产生的临时文件就存放在这个目录下
/var/log	存放系统日志

1. 存储管理

fdisk [选项] [参数]

功能：

- 创建和维护分区表的程序，兼容 DOS 类型的分区表、BSD 或者 SUN 类型的磁盘列表。

参数/命令：

- 设备文件：指定要进行分区或者显示分区的硬盘设备文件。

主要选项：

- -b<分区大小>：指定每个分区的大小
- -l：列出指定的外围设备的分区表状况
- -s<分区编号>：
  - 将指定的分区大小输出到标准输出上，单位为区块
- -u：
  - 搭配"-l"参数列表，会用分区数目取代柱面数目，来表示每个分区的起始地址
- -v：显示版本信息

1. 存储管理

mkfs [选项] [参数]

功能：

- 在设备上创建Linux文件系统。

说明：

- 选项
  - fs：指定建立文件系统时的参数
  - -t<文件系统类型>：指定要建立何种文件系统
  - -v：显示版本信息与详细的使用方法
  - -V：显示简要的使用方法
  - -c：在制做档案系统前，检查该partition是否有坏轨
- 参数
  - 文件系统：指定要创建的文件系统对应的设备文件名
  - 块数：指定文件系统的磁盘块数

badblocks [选项] [参数]

功能：

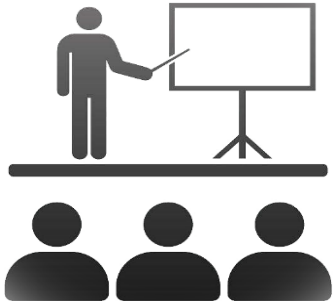
- 用于查找磁盘中损坏的区块。

说明：

- 选项
  - -b <区块大小>：指定磁盘的区块大小，单位为字节
  - -o <输出文件>：将检查结果写入指定的文件
  - -s：检查时显示进度
  - -v：执行时显示详细的信息
  - -w：检查时执行写入
- 参数
  - 磁盘装置：指定要检查的磁盘装置
  - 磁盘区块数：指定磁盘装置的区块总数
  - 启始区块：指定要从哪个区块开始检查

Linux 系统下增加一块新磁盘，通过哪些步骤可以进行应用？

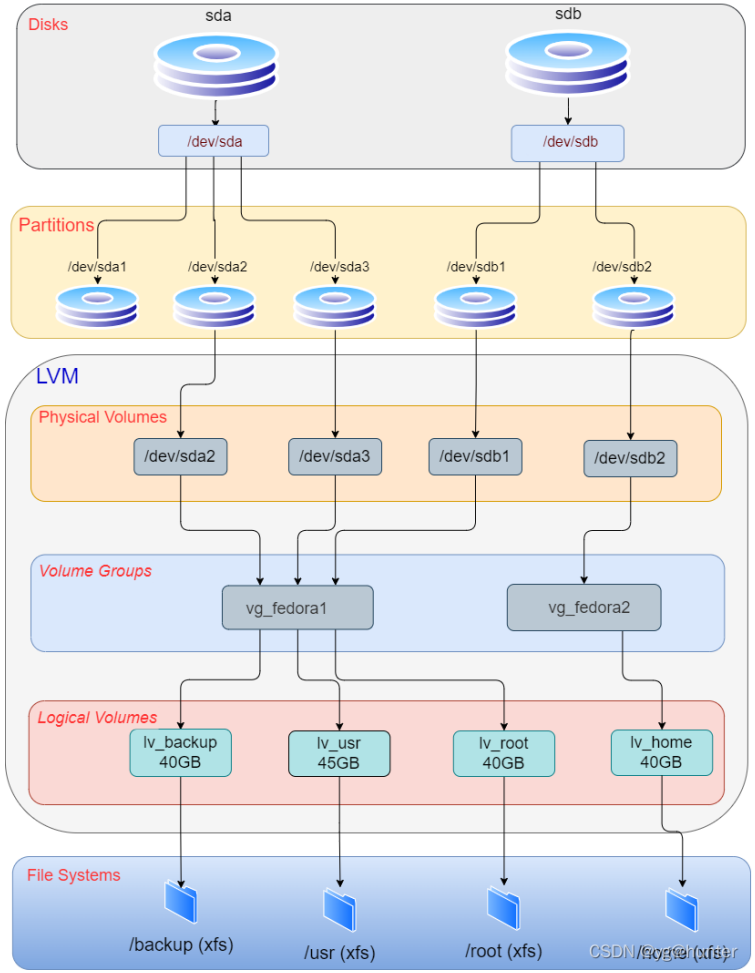
ByteDance 字节跳动 豆包大模型

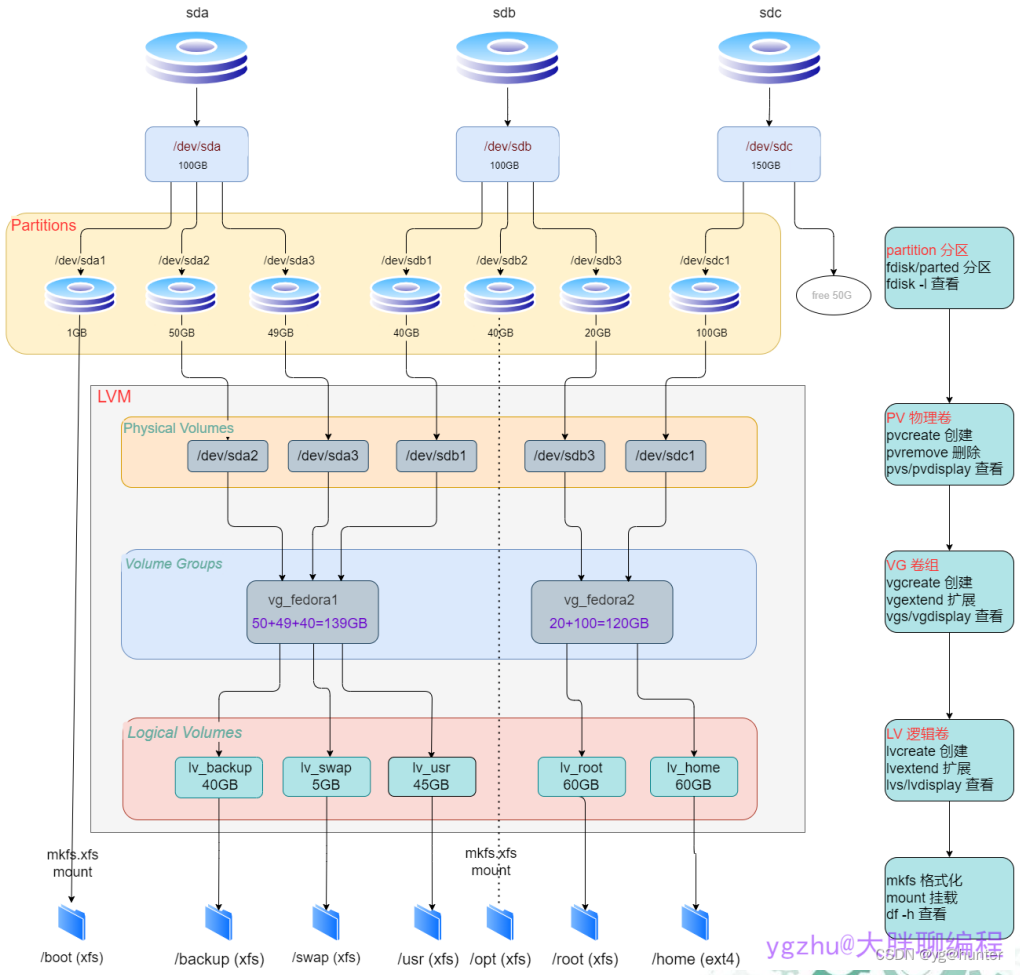


- 为 openEuler 增加一块新磁盘
  - 为 VM 增加新的磁盘
  - 通过 分区->格式化->挂载 操作以使用新磁盘

1. 存储管理

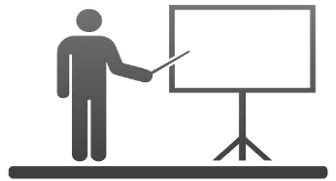
- LVM（Logical Volume Manager），即逻辑卷管理，是Linux环境下对磁盘分区进行管理的一种机制，LVM是建立在硬盘和分区之上的一个逻辑层，用来提高磁盘分区管理的灵活性。
- 物理卷（PV）
  - 物理卷，即物理磁盘分区，是LVM的基本存储逻辑块。
- 卷组（VG）
  - 卷组是将多个物理硬盘整合到一起形成的逻辑卷组。
- 逻辑卷（LV）
  - 逻辑卷是从卷组VG中划分出来的存放数据的磁盘空间。





1. 存储管理

- LVM 的优缺点：
  - 优点：
    - 存储管理灵活，可动态调整容量，组合多个物理设备。
    - 能实现数据冗余和快照，提高可靠性。
    - 简化管理，物理设备抽象为逻辑卷。
    - 高效利用存储资源，避免浪费。
  - 缺点：
    - 配置和管理较复杂，需要技术知识。
    - 对性能有一定影响。
    - 数据迁移和兼容性有风险。
    - 会占用一定系统资源。
- LVM的应用场景：
  - 服务器环境
    - 存储需求变化时可动态调整容量。
    - 关键业务服务器可通过镜像提供冗余。
    - 多块硬盘整合为更大空间方便分配。
  - 虚拟化环境
    - 虚拟机可按需调整容量。
    - 利用快照功能快速部署和克隆虚拟机。
  - 数据中心和云计算环境
    - 存储资源池化以提高资源利用率。
    - 结合自动化管理工具，实现存储资源自动化配置和管理。



- 使用 LVM 管理磁盘
  - 为 VM 增加新的磁盘
  - 使用 LVM 管理磁盘

2. 网络管理

- udev
  - udev 是提供设备事件的 Linux 子系统。
  - 通俗讲就是：当计算机插入了像网卡、外置硬盘（包括 U 盘）、鼠标、键盘、游戏操纵杆和手柄、DVD-ROM 驱动器等设备时，代码能够检测到。
  - 基于这个机制，可写出有用的实用程序去做一些事情，比如当某个硬盘驱动器插入时，执行某个任务（如弹出窗口）。
  - udev 对网卡实施一致的网络接口命名。
    - 进一步了解：
    - https://docs.redhat.com/zh_hans/documentation/red_hat_enterprise_linux/9/html/configuring_and_managing_networking/consistent-network-interface-device-naming_configuring-and-managing-networking
  - openEuler 24.03 默认依然使用 udev 进行设备管理。

10.10.2.31 ([openEuler])

Terminal Sessions View X server Tools Games Settings Macros Help

Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit

Quick connect...

/root/

Name

..
.ssh
.bash_history
.bash_logout
.bash_profile
.bashrc
.cshrc
.tcshrc
anaconda-ks.cfg

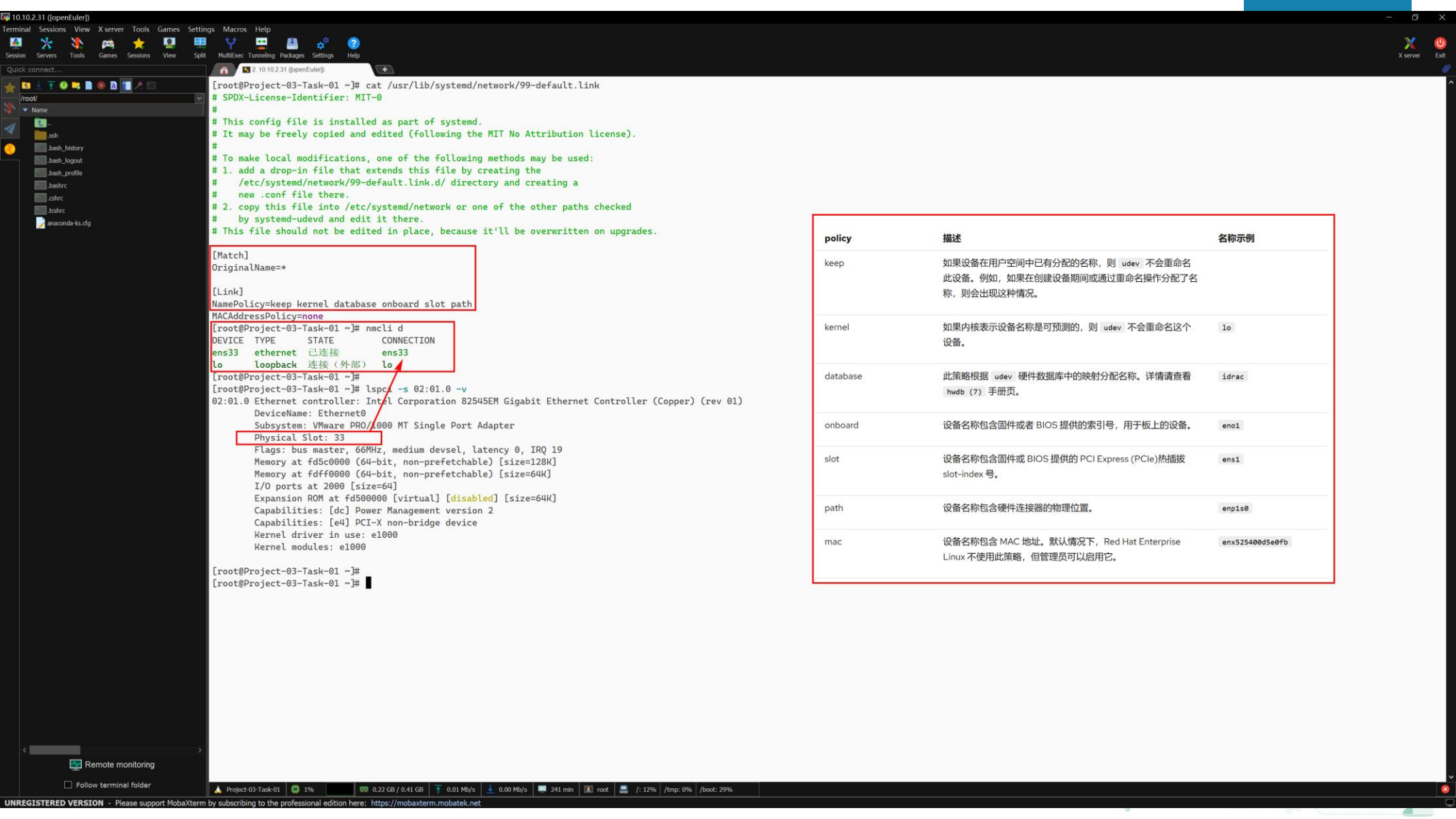
Remote monitoring

Follow terminal folder

2. 10.10.2.31 ([openEuler])

```
[root@Project-03-Task-01 ~]# lspci
00:00.0 Host bridge: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX Host bridge (rev 01)
00:01.0 PCI bridge: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX AGP bridge (rev 01)
00:07.0 ISA bridge: Intel Corporation 82371AB/EB/MB PIIX4 ISA (rev 08)
00:07.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4 IDE (rev 01)
00:07.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 08)
00:07.7 System peripheral: VMware Virtual Machine Communication Interface (rev 10)
00:0f.0 VGA compatible controller: VMware SVGA II Adapter
00:10.0 SCSI storage controller: Broadcom / LSI 53c1030 PCI-X Fusion-MPT Dual Ultra320 SCSI (rev 01)
00:11.0 PCI bridge: VMware PCI bridge (rev 02)
00:15.0 PCI bridge: VMware PCI Express Root Port (rev 01)
00:15.1 PCI bridge: VMware PCI Express Root Port (rev 01)
00:15.2 PCI bridge: VMware PCI Express Root Port (rev 01)
00:15.3 PCI bridge: VMware PCI Express Root Port (rev 01)
00:15.4 PCI bridge: VMware PCI Express Root Port (rev 01)
00:15.5 PCI bridge: VMware PCI Express Root Port (rev 01)
00:15.6 PCI bridge: VMware PCI Express Root Port (rev 01)
00:15.7 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.0 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.1 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.2 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.3 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.4 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.5 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.6 PCI bridge: VMware PCI Express Root Port (rev 01)
00:16.7 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.0 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.1 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.2 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.3 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.4 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.5 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.6 PCI bridge: VMware PCI Express Root Port (rev 01)
00:17.7 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.0 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.1 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.2 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.3 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.4 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.5 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.6 PCI bridge: VMware PCI Express Root Port (rev 01)
00:18.7 PCI bridge: VMware PCI Express Root Port (rev 01)
02:00.0 USB controller: VMware USB1.1 UHCI Controller
02:01.0 Ethernet controller: Intel Corporation 82545EM Gigabit Ethernet Controller (Copper) (rev 01)
02:02.0 USB controller: VMware USB2 EHCI Controller
02:03.0 SATA controller: VMware SATA AHCI controller
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]#
```

Project-03-Task-01 | 0% | 0.22 GB / 0.41 GB | 0.01 Mb/s | 0.00 Mb/s | 239 min | root | /: 12% | /tmp: 0% | /boot: 29%



policy	描述	名称示例
keep	如果设备在用户空间中已有分配的名称，则 `udev` 不会重命名此设备。例如，如果在创建设备期间或通过重命名操作分配了名称，则会出现这种情况。	
kernel	如果内核表示设备名称是可预测的，则 `udev` 不会重命名这个设备。	`lo`
database	此策略根据 `udev` 硬件数据库中的映射分配名称。详情请查看 `hwdb (7)` 手册页。	`idrac`
onboard	设备名称包含固件或者 BIOS 提供的索引号，用于板上的设备。	`eno1`
slot	设备名称包含固件或 BIOS 提供的 PCI Express (PCIe)热插拔 slot-index 号。	`ens1`
path	设备名称包含硬件连接器的物理位置。	`enp1s0`
mac	设备名称包含 MAC 地址。默认情况下，Red Hat Enterprise Linux 不使用此策略，但管理员可以启用它。	`enx525400d5e0fb`

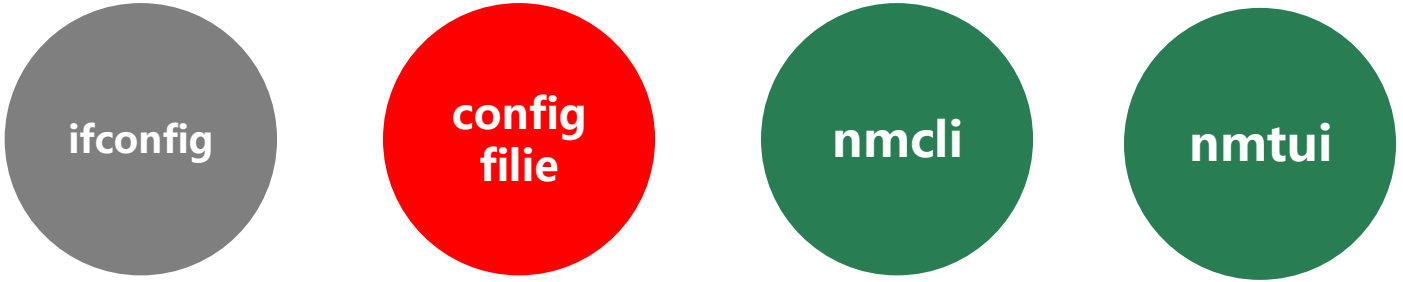
2. 网络管理

- NetworkManager 是一个为系统提供检测和配置功能以便自动连接到网络的程序。
  - NetworkManager 默认情况下被设计为完全自动的，能够管理网络连接和多种网络接口，如以太网、Wi-Fi 和移动宽带设备。
  - NetworkManager 能够有效管理无线网络和有线网络。
  - 对于无线网络：
    - NetworkManager 能够优先选择连接到已知的无线网络中，并有自动切换到最可靠的网络上。
    - NetworkManager 支持应用程序在在线和离线模式之间切换。
  - 对于有线网络：
    - NetworkManager 能够更好的支持有线网络。
    - NetworkManager 支持调制解调器连接和部分类型的 VPN。
  - NetworkManager 最初由 Red Hat 开发，现在由 GNOME 管理。
    - 官方文档：https://networkmanager.dev/
  - openEuler 默认使用 NetworkManager 进行网络配置。

2. 网络管理

操作系统最简网络配置内容

IP / Mask　　Gateway / Route　　DNS



临时修改
net-tools提供的工具
已经废弃，不推荐

推荐使用
一次修改永久生效
使用vi编辑文件

使用方便
NetworkManager
提供的Cli工具

简单直观易用
NetworkManager
提供的TUI工具

2. 网络管理

nmcli [选项] 对象 {命令}

功能：

- NetworkManager 的 工具，用于进行网络配置和管理。

参数/命令：

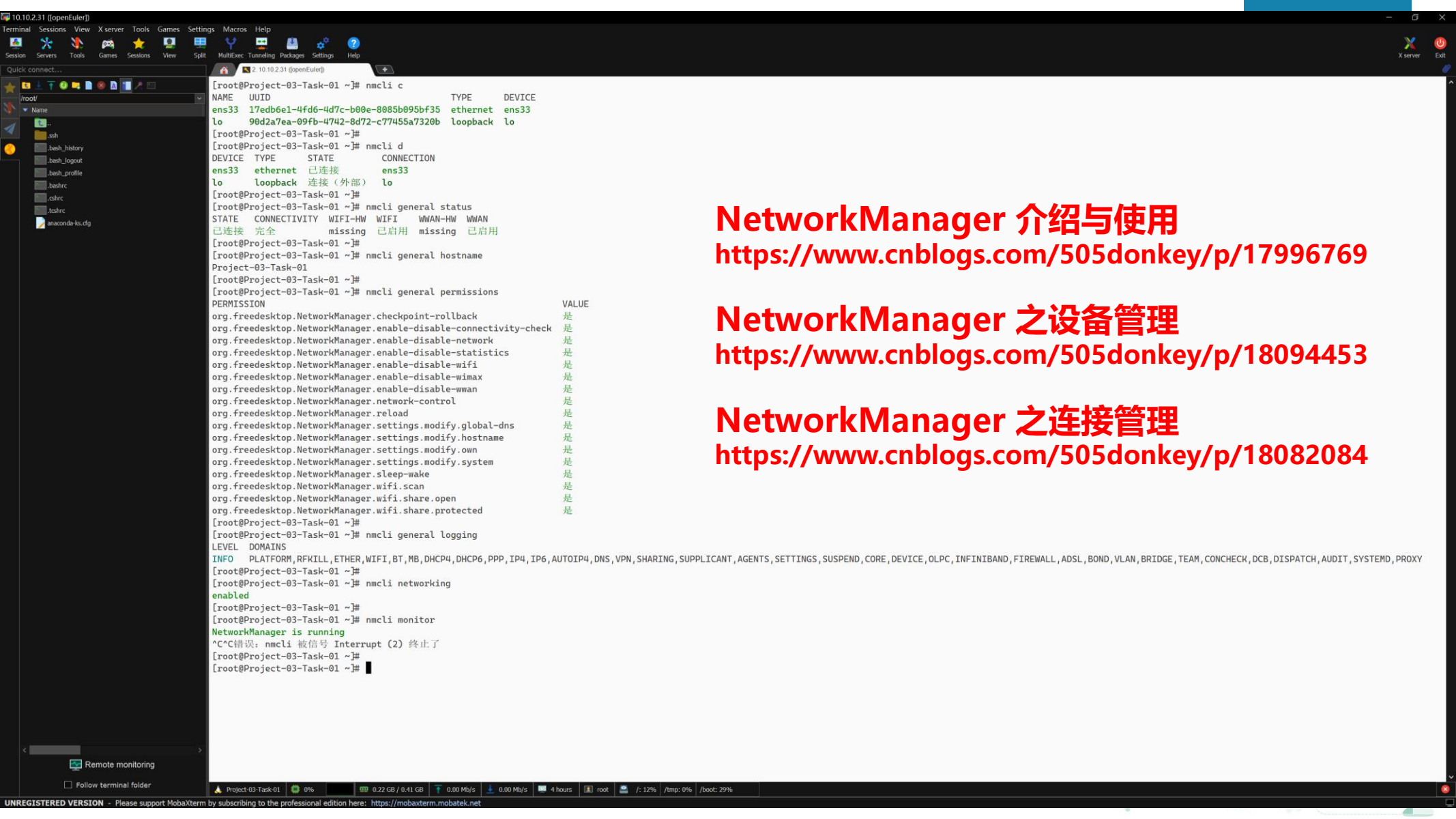
- 选项：
  - -e：是否在值中转义列分隔符
  - -f：指定要输出的字段
  - -m：多行输出模式
  - -o：概述模式
  - -t：简洁输出，仅显示必要信息
  - -w：设置等待完成操作的超时

主要选项：

- g[eneral]：
  - 可查看NetworkManager的整体状态
- n[etworking]：
  - 整体网络控制，可以进行网络连接的开启和关闭等操作
- r[adio]：
  - 可管理无线网络的开启和关闭等操作
- c[onnection]：
  - 可查看和管理当前存在的网络连接
- d[evice]：
  - 可查看和管理连接到NetworkManager的设备
- a[gent]：
  - 可进行与身份验证和授权相关的操作
- m[onitor]：
  - 可实时查看NetworkManager的状态和事件变化

```
[root@Project-03-Task-01 ~]# 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

[root@Project-03-Task-01 ~]# nmcli c --help
用法：nmcli connection { 命令 | help }

命令 := { show | up | down | add | modify | clone | edit | delete | monitor | reload | load | import | export }

  show [--active] [--order <排序规则>]
  show [--active] [id | uuid | path | apath] <ID> ...

  up [[id | uuid | path] <ID>] [ifname <接口名称>] [ap <BSSID>] [passwd-file <密码文件>]

  down [id | uuid | path | apath] <ID> ...

  add 常用选项 类型特定选项 从选项 IP_选项 [-- ([+|-]<设置>.<属性> <值>)+]

  modify [--temporary] [id | uuid | path] <ID> ([+|-]<设置>.<属性> <值>)+

  clone [--temporary] [id | uuid | path ] <ID> <新名称>

  edit [id | uuid | path] <ID>
  edit [type <新连接类型>] [con-name <新连接名称>]

  delete [id | uuid | path] <ID>

  monitor [id | uuid | path] <ID> ...

  reload

  load <文件名> [ <文件名> ... ]

  import [--temporary] type <类型> file <要导入的文件>
```

nmcli 案例：
https://networkmanager.dev/docs/api/latest/nmcli-examples.html

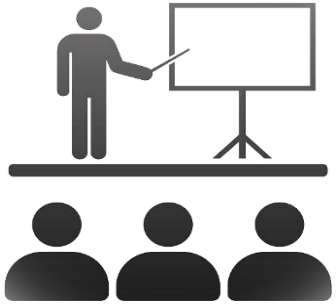

10.10.2.31 ([openEuler])
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
/root/
Name
.ssh
.bash_history
.bash_logout
.bash_profile
.bashrc
.cshrc
.tcshrc
anaconda-ks.cfg
[root@Project-03-Task-01 ~]# nmcli c
NAME UUID TYPE DEVICE
ens33 17edb6e1-4fd6-4d7c-b00e-8085b095bf35 ethernet ens33
lo 90d2a7ea-09fb-4742-8d72-c77455a7320b loopback lo
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# nmcli d
DEVICE TYPE STATE CONNECTION
ens33 ethernet 已连接 ens33
lo loopback 连接（外部） lo
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# nmcli general status
STATE CONNECTIVITY WIFI-HW WIFI WWAN-HW WWAN
已连接 完全 missing 已启用 missing 已启用
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# nmcli general hostname
Project-03-Task-01
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# nmcli general permissions
PERMISSION VALUE
org.freedesktop.NetworkManager.checkpoint-rollback 是
org.freedesktop.NetworkManager.enable-disable-connectivity-check 是
org.freedesktop.NetworkManager.enable-disable-network 是
org.freedesktop.NetworkManager.enable-disable-statistics 是
org.freedesktop.NetworkManager.enable-disable-wifi 是
org.freedesktop.NetworkManager.enable-disable-wimax 是
org.freedesktop.NetworkManager.enable-disable-wwan 是
org.freedesktop.NetworkManager.network-control 是
org.freedesktop.NetworkManager.reload 是
org.freedesktop.NetworkManager.settings.modify.global-dns 是
org.freedesktop.NetworkManager.settings.modify.hostname 是
org.freedesktop.NetworkManager.settings.modify.own 是
org.freedesktop.NetworkManager.settings.modify.system 是
org.freedesktop.NetworkManager.sleep-wake 是
org.freedesktop.NetworkManager.wifi.scan 是
org.freedesktop.NetworkManager.wifi.share.open 是
org.freedesktop.NetworkManager.wifi.share.protected 是
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# nmcli general logging
LEVEL DOMAINS
INFO PLATFORM,RFKILL,ETHER,WIFI,BT,MB,DHCP4,DHCP6,PPP,IP4,IP6,AUTOIP4,DNS,VPN,SHARING,SUPPLICANT,AGENTS,SETTINGS,SUSPEND,CORE,DEVICE,OLPC,INFINIBAND,FIREWALL,ADSL,BOND,VLAN,BRIDGE,TEAM,CONCHECK,DCB,DISPATCH,AUDIT,SYSTEMD,PROXY
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# nmcli networking
enabled
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# nmcli monitor
NetworkManager is running
^C^C错误：nmcli 被信号 Interrupt (2) 终止了
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]#
NetworkManager 介绍与使用
https://www.cnblogs.com/505donkey/p/17996769
NetworkManager 之设备管理
https://www.cnblogs.com/505donkey/p/18094453
NetworkManager 之连接管理
https://www.cnblogs.com/505donkey/p/18082084
Remote monitoring
Follow terminal folder
Project-03-Task-01 0% 0.22 GB / 0.41 GB 0.00 Mb/s 0.00 Mb/s 4 hours root /: 12% /tmp: 0% /boot: 29%
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

2. 网络管理

- Bond 是 Linux 内核提供的一种网络驱动，可以将多个网络接口聚合为一个逻辑接口，从而提高网络带宽、实现负载均衡和故障转移。

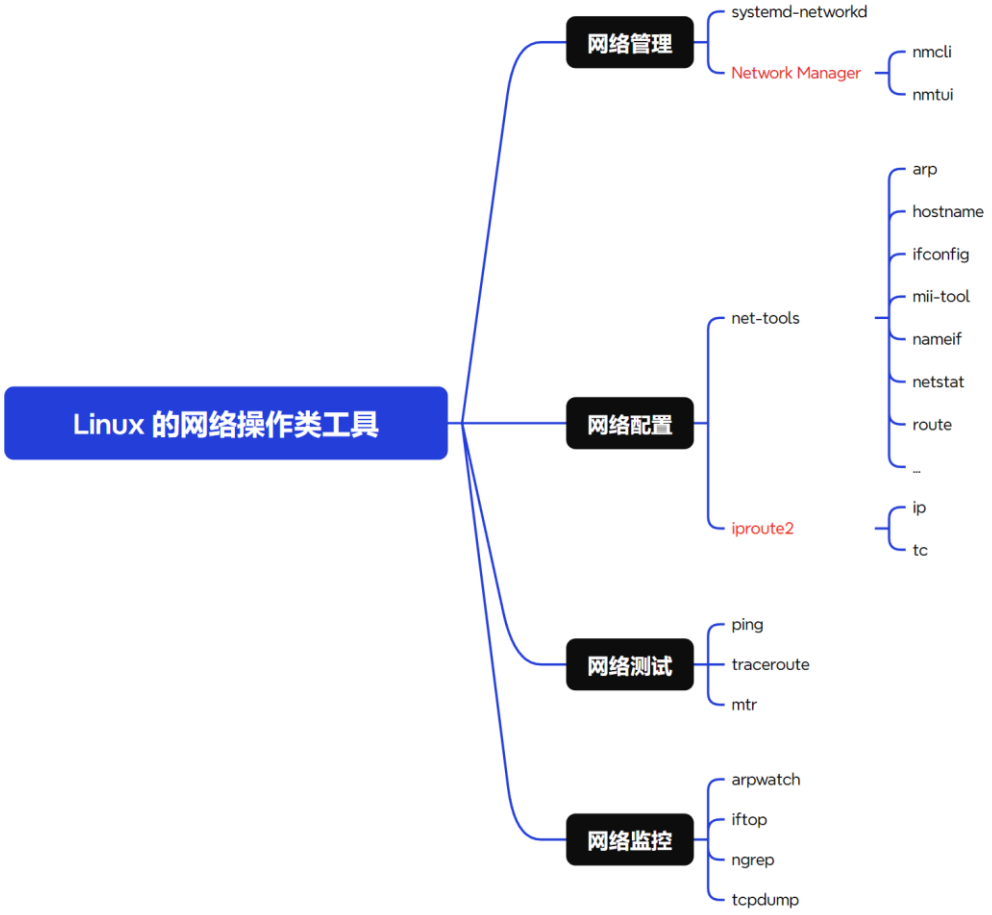
Bond 支持的 7 种模式

模式编号	模式名称	描述	交换机支持要求	常用性
0	Balance-RR （轮询模式）	数据包依次发送到所有网络接口上，实现负载均衡。	需要配置静态链路聚合 mode on	常用
1	Active-Backup （主备模式）	只有一个接口处于活动状态，其他作为备份，活动接口故障时自动切换。	不需要交换机做配置， 仅需要划分对应的vlan	非常常用
2	Balance-XOR （平衡异或模式）	根据源MAC地址和目的MAC地址的异或值来选择发送数据的接口。	需要配置静态链路聚合 mode on， 同时需要设置对应的balance	较少使用
3	Broadcast （广播模式）	所有接口都发送相同的数据包，适用于广播或多播场景。	需要配置静态链路聚合 mode on	很少使用
4	802.3ad （LACP模式）	遵循LACP协议，通过LACP协商实现链路聚合。	需要配置lacp动态链路聚合 mode active	常用
5	Balance-TLB （自适应传输负载均衡模式）	根据每个接口的负载情况动态调整数据包发送。	不需要	较少使用
6	Balance-ALB （自适应负载均衡模式）	Bond5模式的扩展，同时实现发送和接收的负载均衡。	不需要	较少使用

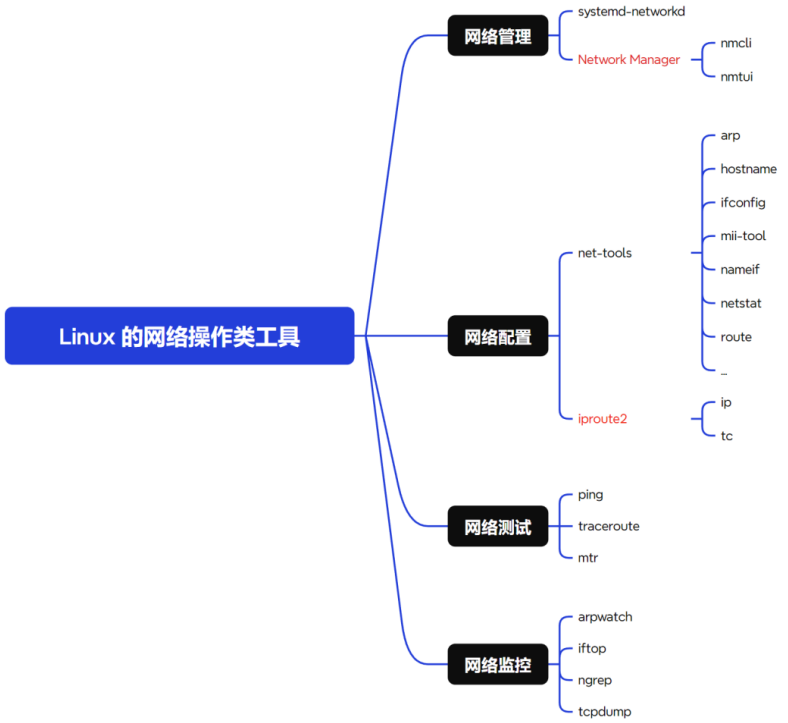


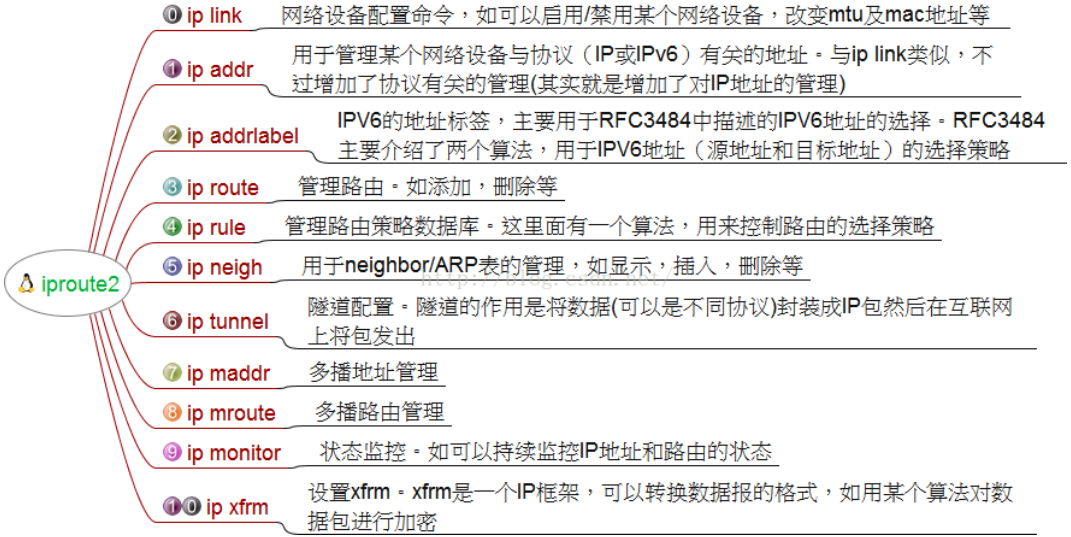
- 通过 nmtui 实现 Bond
  - 在 VM 上增加 NIC
  - 通过 nmtui 创建 Bond
  - 通过 nmtui 配置 Bond

2. 网络管理


Linux 的网络操作类工具
网络管理
systemd-networkd
Network Manager
nmcli
nmtui
网络配置
net-tools
arp
hostname
ifconfig
mii-tool
nameif
netstat
route
...
iproute2
ip
tc
网络测试
ping
traceroute
mtr
网络监控
arpwatch
iftop
ngrep
tcpdump

2. 网络管理





Linux ip Command Examples
https://www.cyberciti.biz/faq/linux-ip-command-examples-usage-syntax/

3. 进程管理

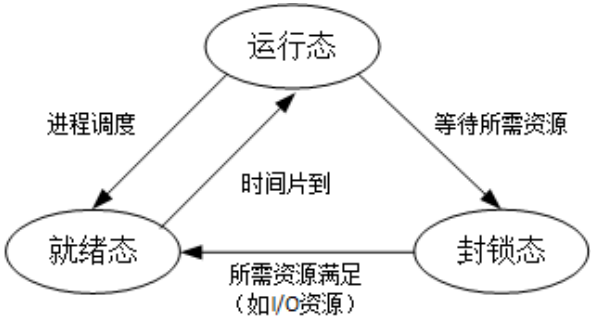
- 程序是存储在磁盘上包含可执行机器指令和数据的静态实体，进程是在操作系统中执行特定任务的动态实体。
  - 一个程序允许有多个进程，而每个运行中的程序至少由一个进程组成。
  - 以 FTP 服务器为例，多个用户使用 FTP 服务，系统会开启多个服务进程以满足用户需求。
- Linux 操作系统作为多用户多任务操作系统，每个进程与其他进程都是彼此独立的，都有独立的权限与职责，用户的应用程序不会干扰到其他用户的程序或操作系统本身。

3. 进程管理

- 进程间有并列关系，也有父进程和子进程的关系，进程间的父子关系实际上是管理和被管理的关系，当父进程终止时，子进程也随之而终止，但子进程终止，父进程并不一定终止。
- Linux 操作系统包括如下 3 种不同类型的进程，每种进程都有其自己的特点和属性。
  - 交互进程：
    - 由Shell启动的进程，可在前台运行，也可在后台运行。
  - 批处理进程：
    - 该进程和终端没有关联，是一个进程序列。
  - 守护进程：
    - 操作系统启动时，随之启动并持续运行的进程。

3. 进程管理

- Linux 操作系统进程具有 3 类状态，分别为：运行态、就绪态和封锁态。
  - 运行态：
    - 当前进程已分配到CPU，正在处理器上执行时的状态。
  - 就绪态：
    - 进程已具备运行条件，但因为其他进程正占用CPU，暂时不能运行而等待分配CPU的状态。
  - 封锁态：
    - 进程因等待某种事件发生而暂时不能运行的状态，也被称为阻塞态。
  - 进程的状态可依据一定的条件和原因而变化。



3. 进程管理

- 进程执行模式划分为用户模式和内核模式。
  - 用户模式。
    - 当前运行的是用户程序、应用程序或者内核之外的系统程序，则对应进程就在用户模式下运行；
  - 内核模式。
    - 在用户程序执行过程中出现系统调用或者发生中断事件，就要运行操作系统（即核心）程序，进程模式就变成内核模式。
- 按照进程的功能和运行程序分类，进程可划分为两大类：
  - 系统进程。
    - 只运行在内核模式，执行操作系统代码，完成一些管理性的工作，例如内存分配、进程切换；
  - 用户进程。
    - 通常在用户模式中执行，并通过系统调用或在出现中断、异常进入内核模式。

3. 进程管理

- 进程管理是操作系统中重要的功能之一，调度和协调进程的执行，控制进程的创建、终止和切换。
- 进行进程管理的基本操作：
  - 进程查看：ps、ps aux、ps lax
  - 后台运行：&、jobs、fg
  - 优先级：nice、renice
  - 进程操作：kill

3. 进程管理

ps [选项]

功能：

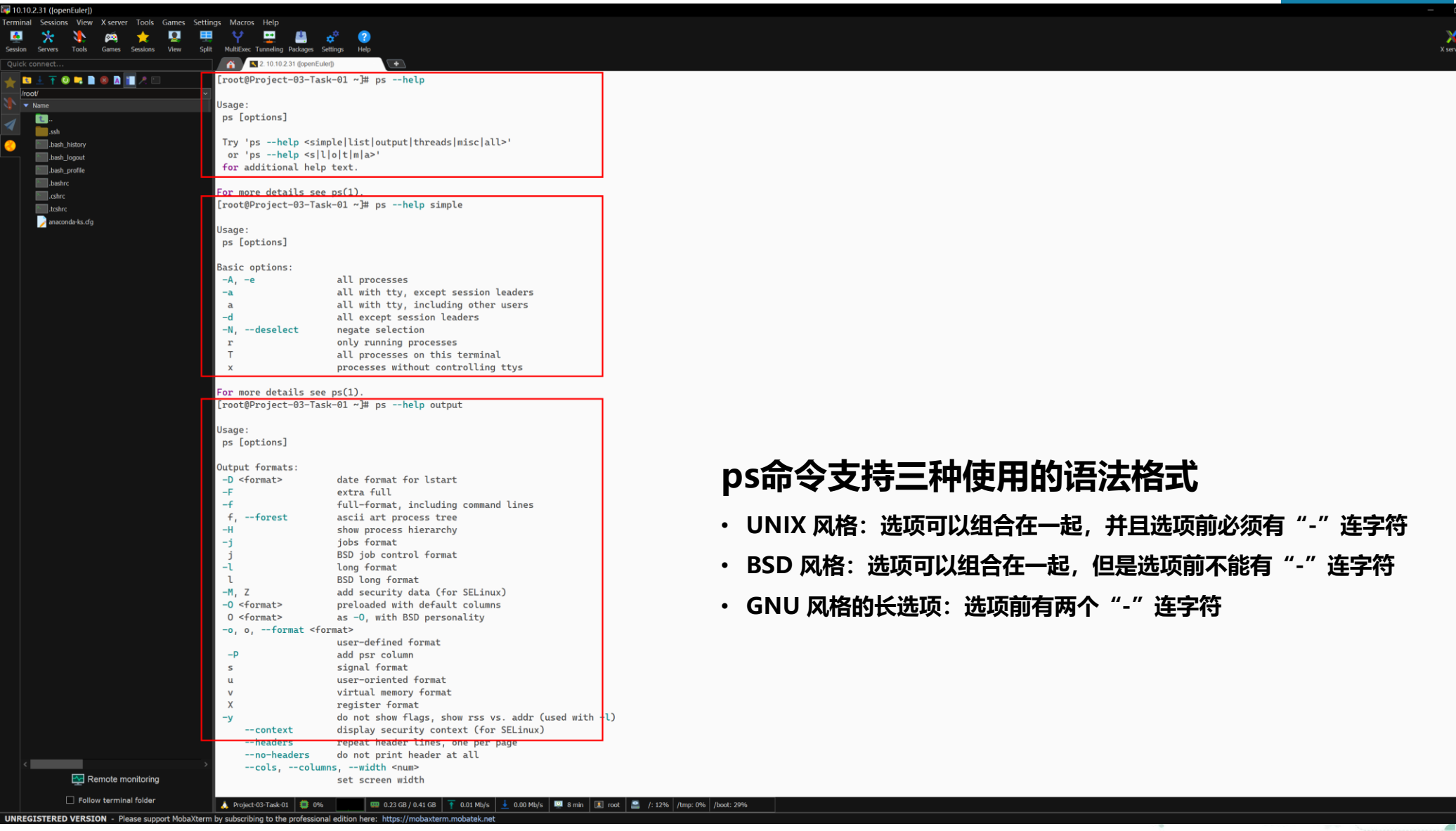
- 查看当前主机的进程信息。

参数/命令：

- a：显示所有终端的进程，包括其他用户
- f：用ASCII字符显示树状结构，表达程序间的相互关系。
- h：不显示标题列。
- l：列出栏位的相关信息。
- s：采用程序信号的格式显示程序状况。
- u：以用户为主的格式来显示程序状况。
- x：显示没有控制终端的进程
- r：显示当前终端中正在执行的进程
- …

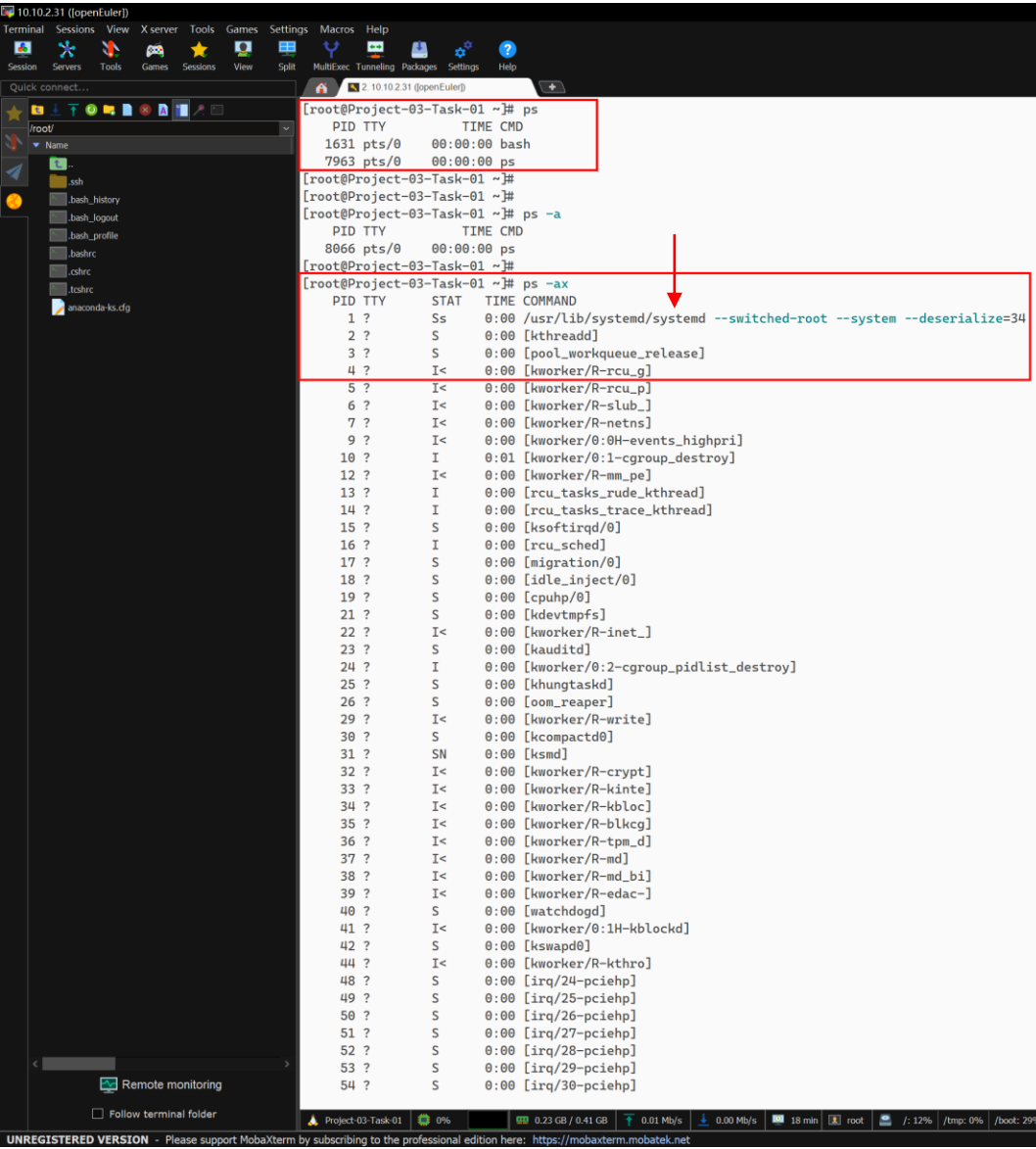
主要选项：

- -e：显示所有进程。
- -f：全格式显示。
- -l：长格式显示。
- -a：显示终端上的所有进程，包括其他用户的进程。
- -u：显示进程的详细状态。
- -x：显示没有控制终端的进程。
- -T：显示当前线程的层次结构。
- -o：
  - 自定义输出格式，其中 <format> 可以是如下的格式选项之一或组合：%cpu、%mem、%tty、%time、%cmd等。
  - 例如，ps -o pid,tty,cmd 将只显示进程ID、终端类型和执行的命令行。
- --sort：按指定的字段对输出进行排序。



ps命令支持三种使用的语法格式

- UNIX 风格：选项可以组合在一起，并且选项前必须有“-”连字符
- BSD 风格：选项可以组合在一起，但是选项前不能有“-”连字符
- GNU 风格的长选项：选项前有两个“-”连字符



不带任何参数的 ps 命令显示运行着的进程信息。

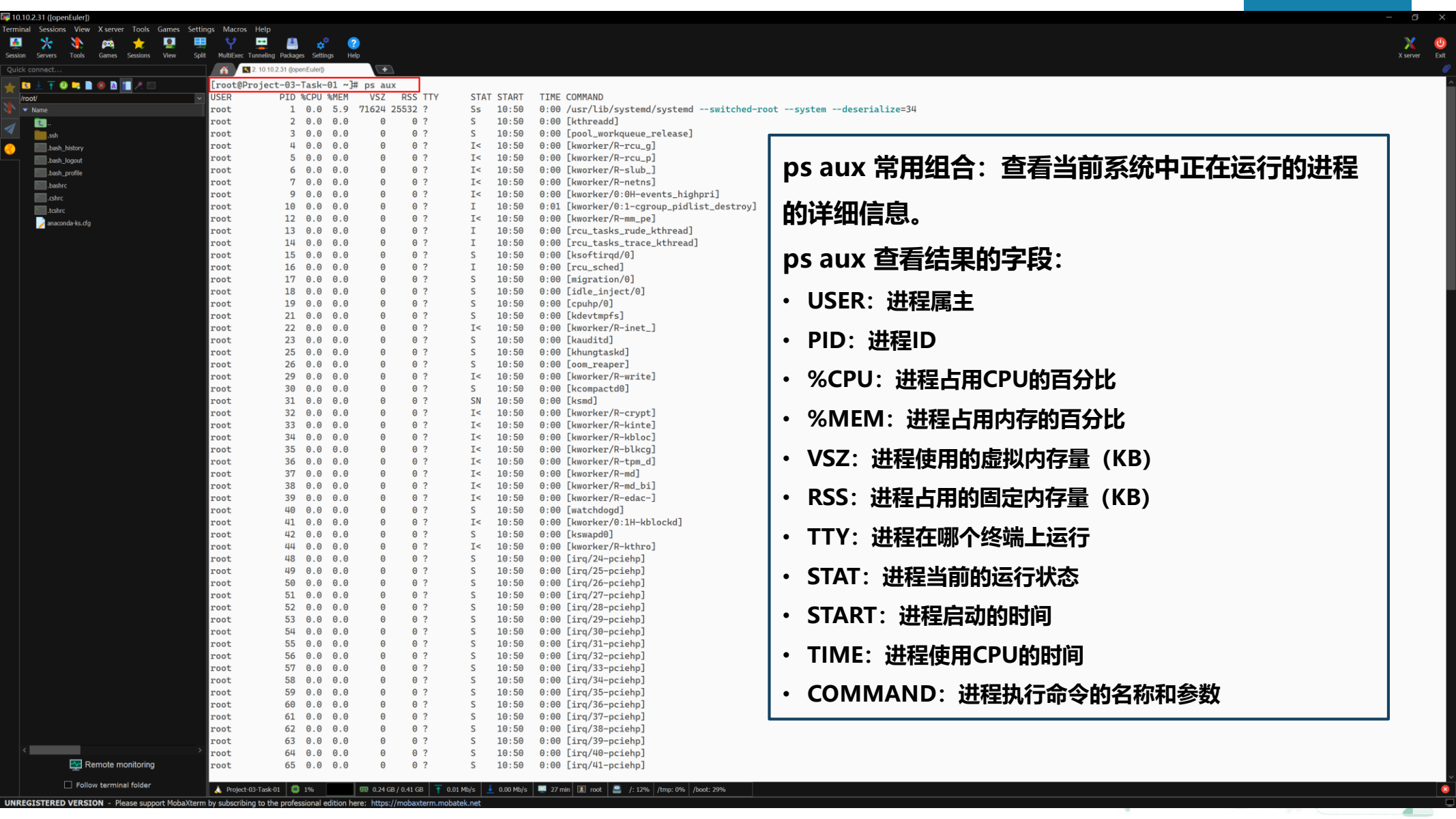
输出结果共有4个字段：

- **PID 运行的命令（CMD）的进程编号**
- **TTY 命令所运行的位置（终端）**
- **TIME 运行该命令所占用的CPU处理时间**
- **CMD 进程运行所执行的命令**

使用 -ax 参数的 ps 命令显示所有的进程信息。

输出结果共有5个字段：

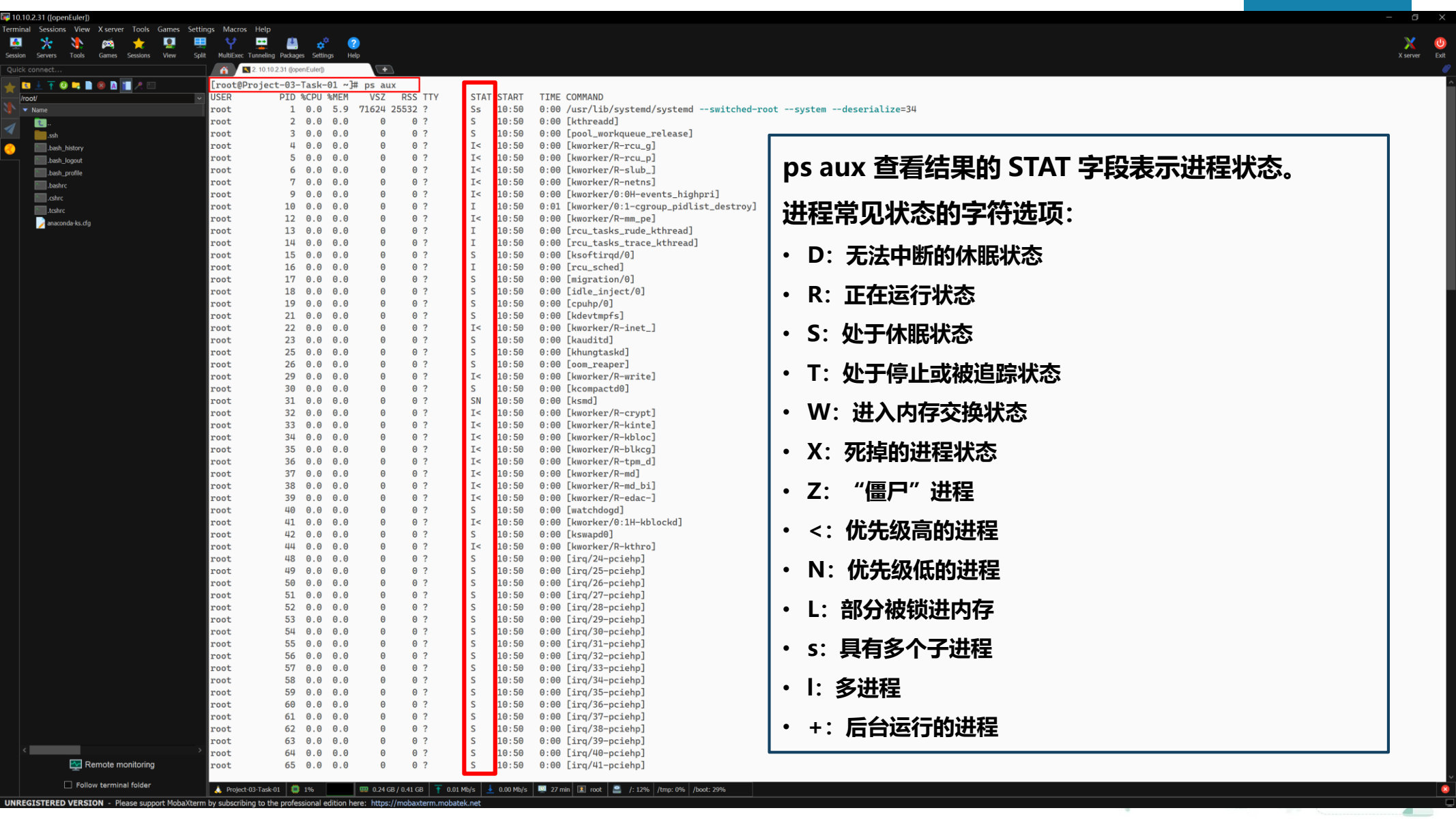
- **PID 运行的命令（CMD）的进程编号**
- **TTY 命令所运行的位置（终端）**
- **STAT 进程的状态，如R（运行）、S（睡眠）、Z（僵尸）等**
- **TIME 运行该命令所占用的CPU处理时间**
- **CMD 进程运行所执行的命令**

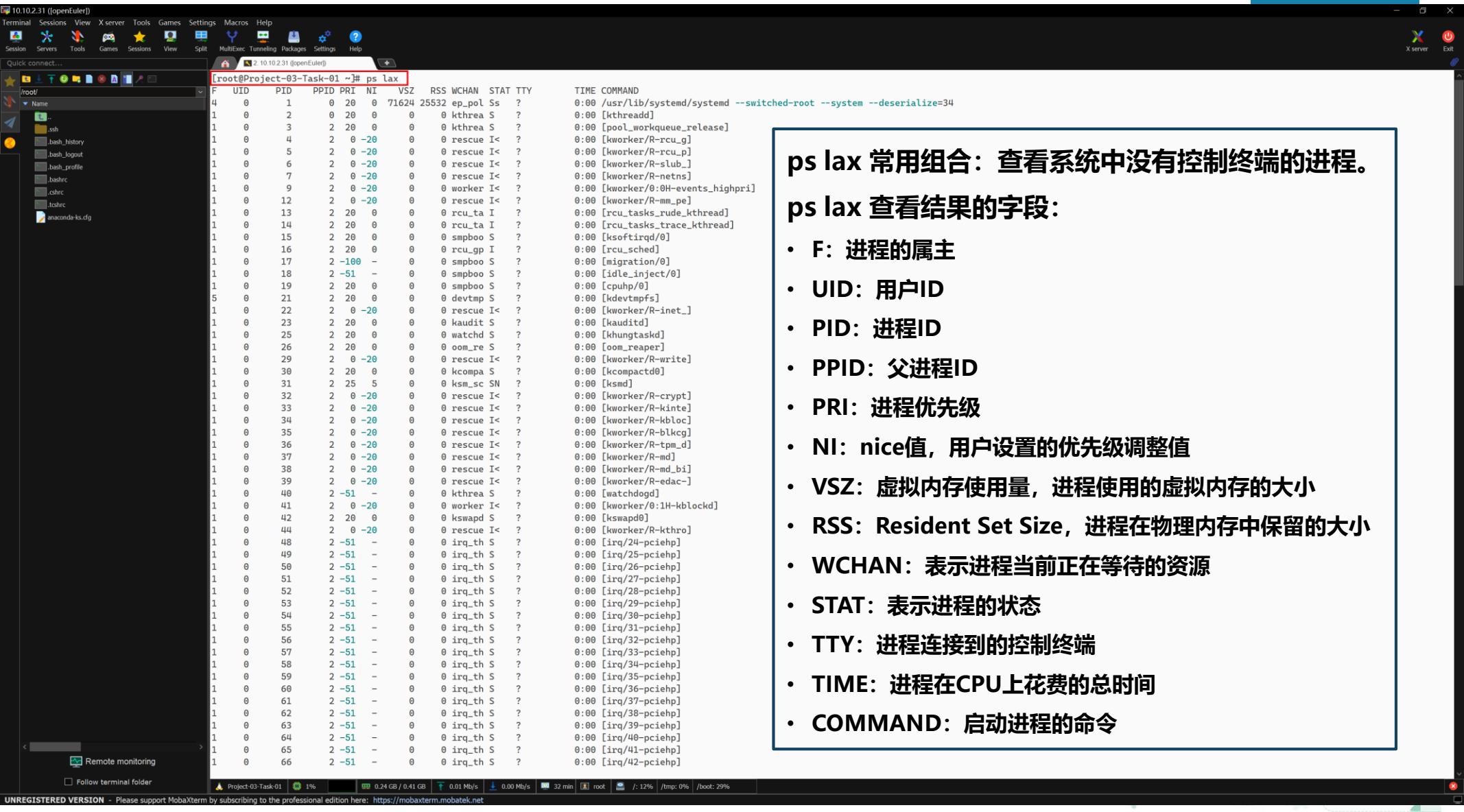


ps aux 常用组合：查看当前系统中正在运行的进程的详细信息。

ps aux 查看结果的字段：

- **USER：进程属主**
- **PID：进程ID**
- **%CPU：进程占用CPU的百分比**
- **%MEM：进程占用内存的百分比**
- **VSZ：进程使用的虚拟内存量（KB）**
- **RSS：进程占用的固定内存量（KB）**
- **TTY：进程在哪个终端上运行**
- **STAT：进程当前的运行状态**
- **START：进程启动的时间**
- **TIME：进程使用CPU的时间**
- **COMMAND：进程执行命令的名称和参数**

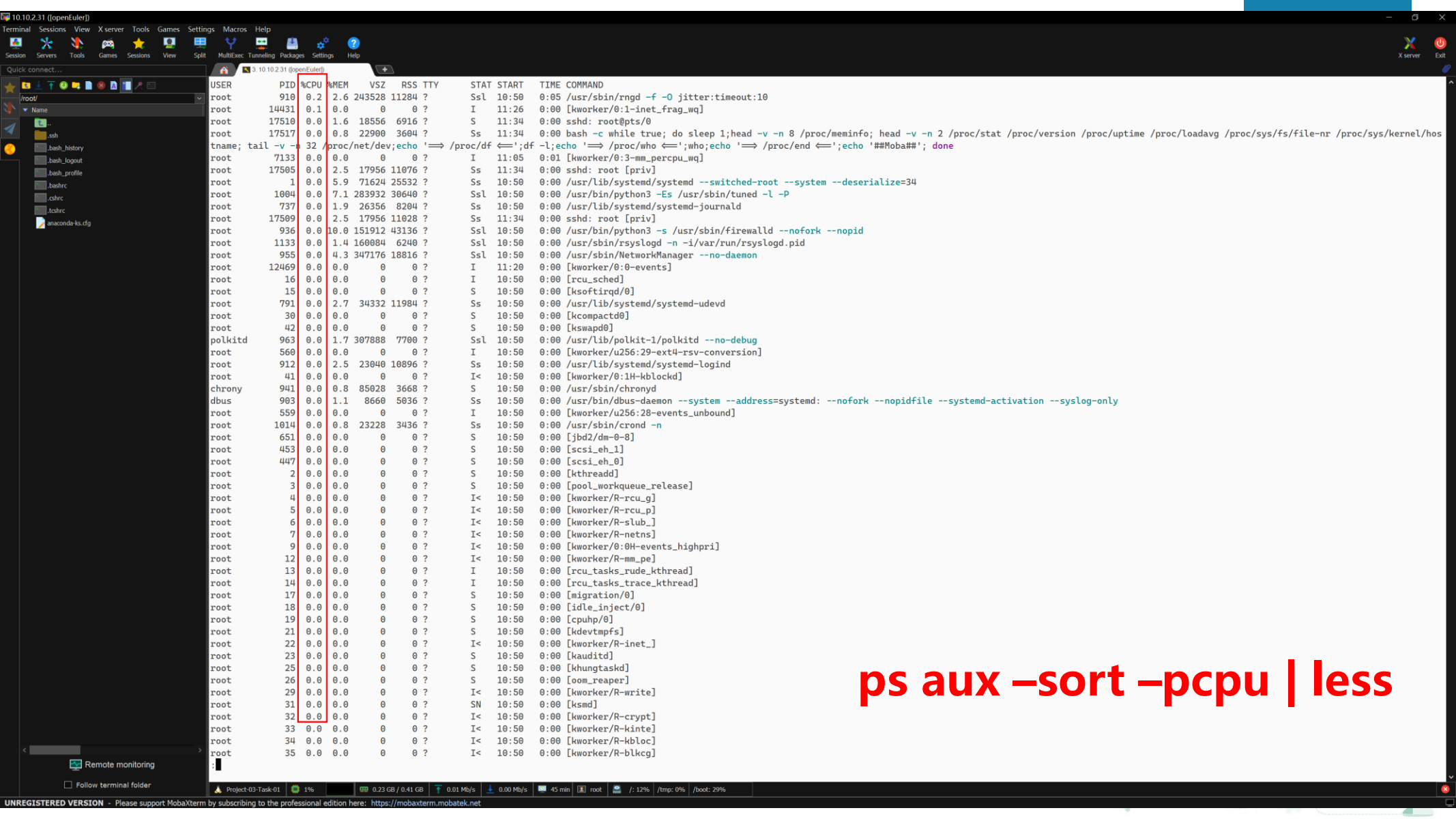

10.10.2.31 ([openEuler])
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
/root/
Name
.ssh
.bash_history
.bash_logout
.bash_profile
.bashrc
.cshrc
.tcshrc
anaconda-ks.cfg
Remote monitoring
Follow terminal folder
[root@Project-03-Task-01 ~]# ps aux
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
root 1 0.0 5.9 71624 25532 ? Ss 10:50 0:00 /usr/lib/systemd/systemd --switched-root --system --deserialize=34
root 2 0.0 0.0 0 0 ? S 10:50 0:00 [kthreadd]
root 3 0.0 0.0 0 0 ? S 10:50 0:00 [pool_workqueue_release]
root 4 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-rcu_g]
root 5 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-rcu_p]
root 6 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-slub_]
root 7 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-netns]
root 9 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/0:0H-events_highpri]
root 10 0.0 0.0 0 0 ? I 10:50 0:01 [kworker/0:1-cgroup_pidlist_destroy]
root 12 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-mm_pe]
root 13 0.0 0.0 0 0 ? I 10:50 0:00 [rcu_tasks_rude_kthread]
root 14 0.0 0.0 0 0 ? I 10:50 0:00 [rcu_tasks_trace_kthread]
root 15 0.0 0.0 0 0 ? S 10:50 0:00 [ksoftirqd/0]
root 16 0.0 0.0 0 0 ? I 10:50 0:00 [rcu_sched]
root 17 0.0 0.0 0 0 ? S 10:50 0:00 [migration/0]
root 18 0.0 0.0 0 0 ? S 10:50 0:00 [idle_inject/0]
root 19 0.0 0.0 0 0 ? S 10:50 0:00 [cpuhp/0]
root 21 0.0 0.0 0 0 ? S 10:50 0:00 [kdevtmpfs]
root 22 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-inet_]
root 23 0.0 0.0 0 0 ? S 10:50 0:00 [kauditd]
root 25 0.0 0.0 0 0 ? S 10:50 0:00 [khungtaskd]
root 26 0.0 0.0 0 0 ? S 10:50 0:00 [oom_reaper]
root 29 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-write]
root 30 0.0 0.0 0 0 ? S 10:50 0:00 [kcompactd0]
root 31 0.0 0.0 0 0 ? SN 10:50 0:00 [ksmd]
root 32 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-crypt]
root 33 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-kinte]
root 34 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-kbloc]
root 35 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-blkcg]
root 36 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-tpm_d]
root 37 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-md]
root 38 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-md_bi]
root 39 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-edac-]
root 40 0.0 0.0 0 0 ? S 10:50 0:00 [watchdogd]
root 41 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/0:1H-kblockd]
root 42 0.0 0.0 0 0 ? S 10:50 0:00 [kswapd0]
root 44 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-kthro]
root 48 0.0 0.0 0 0 ? S 10:50 0:00 [irq/24-pciehp]
root 49 0.0 0.0 0 0 ? S 10:50 0:00 [irq/25-pciehp]
root 50 0.0 0.0 0 0 ? S 10:50 0:00 [irq/26-pciehp]
root 51 0.0 0.0 0 0 ? S 10:50 0:00 [irq/27-pciehp]
root 52 0.0 0.0 0 0 ? S 10:50 0:00 [irq/28-pciehp]
root 53 0.0 0.0 0 0 ? S 10:50 0:00 [irq/29-pciehp]
root 54 0.0 0.0 0 0 ? S 10:50 0:00 [irq/30-pciehp]
root 55 0.0 0.0 0 0 ? S 10:50 0:00 [irq/31-pciehp]
root 56 0.0 0.0 0 0 ? S 10:50 0:00 [irq/32-pciehp]
root 57 0.0 0.0 0 0 ? S 10:50 0:00 [irq/33-pciehp]
root 58 0.0 0.0 0 0 ? S 10:50 0:00 [irq/34-pciehp]
root 59 0.0 0.0 0 0 ? S 10:50 0:00 [irq/35-pciehp]
root 60 0.0 0.0 0 0 ? S 10:50 0:00 [irq/36-pciehp]
root 61 0.0 0.0 0 0 ? S 10:50 0:00 [irq/37-pciehp]
root 62 0.0 0.0 0 0 ? S 10:50 0:00 [irq/38-pciehp]
root 63 0.0 0.0 0 0 ? S 10:50 0:00 [irq/39-pciehp]
root 64 0.0 0.0 0 0 ? S 10:50 0:00 [irq/40-pciehp]
root 65 0.0 0.0 0 0 ? S 10:50 0:00 [irq/41-pciehp]
ps aux 查看结果的 STAT 字段表示进程状态。
进程常见状态的字符选项：
• D：无法中断的休眠状态
• R：正在运行状态
• S：处于休眠状态
• T：处于停止或被追踪状态
• W：进入内存交换状态
• X：死掉的进程状态
• Z：“僵尸”进程
• <：优先级高的进程
• N：优先级低的进程
• L：部分被锁进内存
• s：具有多个子进程
• l：多进程
• +：后台运行的进程
Project-03-Task-01 1% 0.24 GB / 0.41 GB 0.01 Mb/s 0.00 Mb/s 27 min root /: 12% /tmp: 0% /boot: 29%
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net



ps lax 常用组合：查看系统中没有控制终端的进程。

ps lax 查看结果的字段：

- **F：进程的属主**
- **UID：用户ID**
- **PID：进程ID**
- **PPID：父进程ID**
- **PRI：进程优先级**
- **NI：nice值，用户设置的优先级调整值**
- **VSZ：虚拟内存使用量，进程使用的虚拟内存的大小**
- **RSS：Resident Set Size，进程在物理内存中保留的大小**
- **WCHAN：表示进程当前正在等待的资源**
- **STAT：表示进程的状态**
- **TTY：进程连接到的控制终端**
- **TIME：进程在CPU上花费的总时间**
- **COMMAND：启动进程的命令**


10.10.2.31 ([openEuler])
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
/root/
Name
..
.ssh
.bash_history
.bash_logout
.bash_profile
.bashrc
.cshrc
.tcshrc
anaconda-ks.cfg
Remote monitoring
Follow terminal folder
3. 10.10.2.31 ([openEuler])
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
root 910 0.2 2.6 243528 11284 ? Ssl 10:50 0:05 /usr/sbin/rngd -f -O jitter:timeout:10
root 14431 0.1 0.0 0 0 ? I 11:26 0:00 [kworker/0:1-inet_frag_wq]
root 17510 0.0 1.6 18556 6916 ? S 11:34 0:00 sshd: root@pts/0
root 17517 0.0 0.8 22900 3604 ? Ss 11:34 0:00 bash -c while true; do sleep 1;head -v -n 8 /proc/meminfo; head -v -n 2 /proc/stat /proc/version /proc/uptime /proc/loadavg /proc/sys/fs/file-nr /proc/sys/kernel/hos
tname; tail -v -n 32 /proc/net/dev;echo '==> /proc/df <==';df -l;echo '==> /proc/who <==';who;echo '==> /proc/end <==';echo '##Moba##'; done
root 7133 0.0 0.0 0 0 ? I 11:05 0:01 [kworker/0:3-mm_percpu_wq]
root 17505 0.0 2.5 17956 11076 ? Ss 11:34 0:00 sshd: root [priv]
root 1 0.0 5.9 71624 25532 ? Ss 10:50 0:00 /usr/lib/systemd/systemd --switched-root --system --deserialize=34
root 1004 0.0 7.1 283932 30640 ? Ssl 10:50 0:00 /usr/bin/python3 -Es /usr/sbin/tuned -l -P
root 737 0.0 1.9 26356 8204 ? Ss 10:50 0:00 /usr/lib/systemd/systemd-journald
root 17509 0.0 2.5 17956 11028 ? Ss 11:34 0:00 sshd: root [priv]
root 936 0.0 10.0 151912 43136 ? Ssl 10:50 0:00 /usr/bin/python3 -s /usr/sbin/firewalld --nofork --nopid
root 1133 0.0 1.4 160084 6240 ? Ssl 10:50 0:00 /usr/sbin/rsyslogd -n -i/var/run/rsyslogd.pid
root 955 0.0 4.3 347176 18816 ? Ssl 10:50 0:00 /usr/sbin/NetworkManager --no-daemon
root 12469 0.0 0.0 0 0 ? I 11:20 0:00 [kworker/0:0-events]
root 16 0.0 0.0 0 0 ? I 10:50 0:00 [rcu_sched]
root 15 0.0 0.0 0 0 ? S 10:50 0:00 [ksoftirqd/0]
root 791 0.0 2.7 34332 11984 ? Ss 10:50 0:00 /usr/lib/systemd/systemd-udevd
root 30 0.0 0.0 0 0 ? S 10:50 0:00 [kcompactd0]
root 42 0.0 0.0 0 0 ? S 10:50 0:00 [kswapd0]
polkitd 963 0.0 1.7 307888 7700 ? Ssl 10:50 0:00 /usr/lib/polkit-1/polkitd --no-debug
root 560 0.0 0.0 0 0 ? I 10:50 0:00 [kworker/u256:29-ext4-rsv-conversion]
root 912 0.0 2.5 23040 10896 ? Ss 10:50 0:00 /usr/lib/systemd/systemd-logind
root 41 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/0:1H-kblockd]
chrony 941 0.0 0.8 85028 3668 ? S 10:50 0:00 /usr/sbin/chronyd
dbus 903 0.0 1.1 8660 5036 ? Ss 10:50 0:00 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
root 559 0.0 0.0 0 0 ? I 10:50 0:00 [kworker/u256:28-events_unbound]
root 1014 0.0 0.8 23228 3436 ? Ss 10:50 0:00 /usr/sbin/crond -n
root 651 0.0 0.0 0 0 ? S 10:50 0:00 [jbd2/dm-0-8]
root 453 0.0 0.0 0 0 ? S 10:50 0:00 [scsi_eh_1]
root 447 0.0 0.0 0 0 ? S 10:50 0:00 [scsi_eh_0]
root 2 0.0 0.0 0 0 ? S 10:50 0:00 [kthreadd]
root 3 0.0 0.0 0 0 ? S 10:50 0:00 [pool_workqueue_release]
root 4 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-rcu_g]
root 5 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-rcu_p]
root 6 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-slub_]
root 7 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-netns]
root 9 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/0:0H-events_highpri]
root 12 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-mm_pe]
root 13 0.0 0.0 0 0 ? I 10:50 0:00 [rcu_tasks_rude_kthread]
root 14 0.0 0.0 0 0 ? I 10:50 0:00 [rcu_tasks_trace_kthread]
root 17 0.0 0.0 0 0 ? S 10:50 0:00 [migration/0]
root 18 0.0 0.0 0 0 ? S 10:50 0:00 [idle_inject/0]
root 19 0.0 0.0 0 0 ? S 10:50 0:00 [cpuhp/0]
root 21 0.0 0.0 0 0 ? S 10:50 0:00 [kdevtmpfs]
root 22 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-inet_]
root 23 0.0 0.0 0 0 ? S 10:50 0:00 [kauditd]
root 25 0.0 0.0 0 0 ? S 10:50 0:00 [khungtaskd]
root 26 0.0 0.0 0 0 ? S 10:50 0:00 [oom_reaper]
root 29 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-write]
root 31 0.0 0.0 0 0 ? SN 10:50 0:00 [ksmd]
root 32 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-crypt]
root 33 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-kinte]
root 34 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-kbloc]
root 35 0.0 0.0 0 0 ? I< 10:50 0:00 [kworker/R-blkcg]
:
Project-03-Task-01 1% 0.23 GB / 0.41 GB 0.01 Mb/s 0.00 Mb/s 45 min root /: 12% /tmp: 0% /boot: 29%
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
ps aux –sort –pcpu | less

10.10.2.31 ([openEuler])

Terminal Sessions View X server Tools Games Settings Macros Help

Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit

Quick connect...

/root/

Name

..
.ssh
.bash_history
.bash_logout
.bash_profile
.bashrc
.cshrc
.tcshrc
anaconda-ks.cfg

Remote monitoring

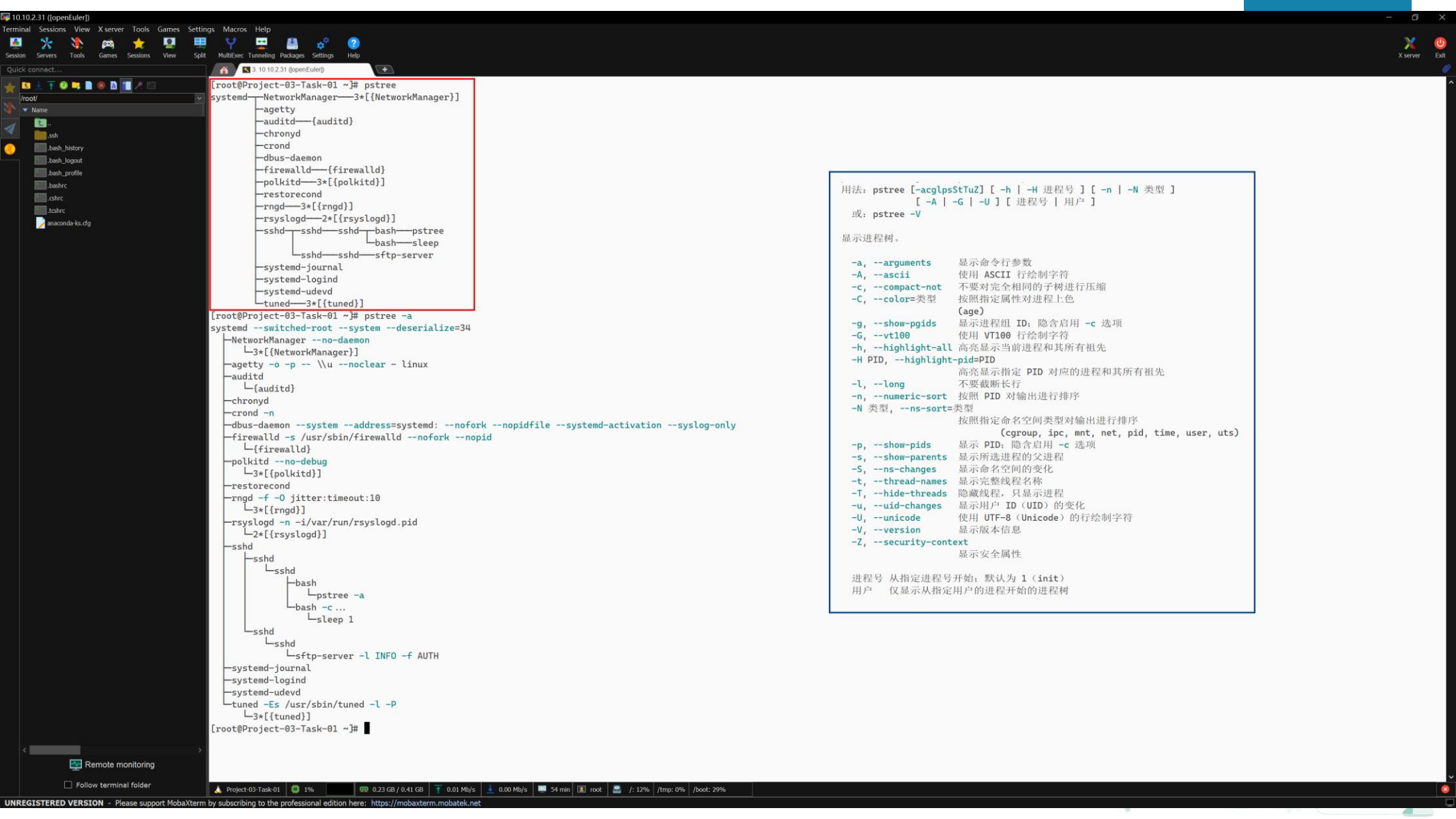
Follow terminal folder

3. 10.10.2.31 ([openEuler])

```
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         936  0.0 10.0 151912 43136 ?        Ssl  10:50   0:00 /usr/bin/python3 -s /usr/sbin/firewalld --nofork --nopid
root        1004  0.0  7.1 283932 30640 ?        Ssl  10:50   0:00 /usr/bin/python3 -Es /usr/sbin/tuned -l -P
root           1  0.0  5.9  71624 25532 ?        Ss   10:50   0:00 /usr/lib/systemd/systemd --switched-root --system --deserialize=34
root         955  0.0  4.3 347176 18816 ?        Ssl  10:50   0:00 /usr/sbin/NetworkManager --no-daemon
root         925  0.0  3.2  62244 13816 ?        Ss   10:50   0:00 /usr/sbin/restorecond
root         791  0.0  2.7  34332 11984 ?        Ss   10:50   0:00 /usr/lib/systemd/systemd-udevd
root         910  0.2  2.6 243528 11284 ?        Ssl  10:50   0:05 /usr/sbin/rngd -f -O jitter:timeout:10
root       17505  0.0  2.5  17956 11076 ?        Ss   11:34   0:00 sshd: root [priv]
root       17509  0.0  2.5  17956 11028 ?        Ss   11:34   0:00 sshd: root [priv]
root         912  0.0  2.5  23040 10896 ?        Ss   10:50   0:00 /usr/lib/systemd/systemd-logind
root         737  0.0  1.9  26356  8204 ?        Ss   10:50   0:00 /usr/lib/systemd/systemd-journald
root        1000  0.0  1.8  15440  8032 ?        Ss   10:50   0:00 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
polkitd      963  0.0  1.7 307888  7700 ?        Ssl  10:50   0:00 /usr/lib/polkit-1/polkitd --no-debug
root       17510  0.0  1.6  18556  6916 ?        S    11:34   0:00 sshd: root@pts/0
root       17650  0.0  1.5  18216  6616 ?        S    11:34   0:00 sshd: root@notty
root        1133  0.0  1.4 160084  6240 ?        Ssl  10:50   0:00 /usr/sbin/rsyslogd -n -i/var/run/rsyslogd.pid
dbus         903  0.0  1.1   8660  5036 ?        Ss   10:50   0:00 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
root       18727  0.0  1.0  25652  4448 pts/0    R+   11:37   0:00 ps aux --sort -pmem
root       17512  0.0  1.0  23084  4440 pts/0    Ss   11:34   0:00 -bash
chrony       941  0.0  0.8  85028  3668 ?        S    10:50   0:00 /usr/sbin/chronyd
root       17517  0.0  0.8  22900  3604 ?        Ss   11:34   0:00 bash -c while true; do sleep 1;head -v -n 8 /proc/meminfo; head -v -n 2 /proc/stat /proc/version /proc/uptime /proc/loadavg /proc/sys/fs/file-nr /proc/sys/kernel/hos
tname; tail -v -n 32 /proc/net/dev;echo '==> /proc/df <==';df -l;echo '==> /proc/who <==';who;echo '==> /proc/end <==';echo '##Moba##'; done
root        1014  0.0  0.8  23228  3436 ?        Ss   10:50   0:00 /usr/sbin/crond -n
root       18728  0.0  0.6  21520  2580 pts/0    S+   11:37   0:00 less
root         882  0.0  0.5  18928  2488 ?        S<sl 10:50   0:00 /sbin/auditd
root       18726  0.0  0.5  21508  2400 ?        S    11:37   0:00 sleep 1
root       17657  0.0  0.5   4992  2232 ?        Ss   11:34   0:00 /usr/libexec/openssh/sftp-server -l INFO -f AUTH
root        1017  0.0  0.4  21548  1932 tty1     Ss+  10:50   0:00 /sbin/agetty -o -p -- \u --noclear - linux
root           2  0.0  0.0      0     0 ?        S    10:50   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        S    10:50   0:00 [pool_workqueue_release]
root           4  0.0  0.0      0     0 ?        I<   10:50   0:00 [kworker/R-rcu_g]
root           5  0.0  0.0      0     0 ?        I<   10:50   0:00 [kworker/R-rcu_p]
root           6  0.0  0.0      0     0 ?        I<   10:50   0:00 [kworker/R-slub_]
root           7  0.0  0.0      0     0 ?        I<   10:50   0:00 [kworker/R-netns]
root           9  0.0  0.0      0     0 ?        I<   10:50   0:00 [kworker/0:0H-events_highpri]
root          12  0.0  0.0      0     0 ?        I<   10:50   0:00 [kworker/R-mm_pe]
root          13  0.0  0.0      0     0 ?        I    10:50   0:00 [rcu_tasks_rude_kthread]
root          14  0.0  0.0      0     0 ?        I    10:50   0:00 [rcu_tasks_trace_kthread]
root          15  0.0  0.0      0     0 ?        S    10:50   0:00 [ksoftirqd/0]
root          16  0.0  0.0      0     0 ?        R    10:50   0:00 [rcu_sched]
root          17  0.0  0.0      0     0 ?        S    10:50   0:00 [migration/0]
root          18  0.0  0.0      0     0 ?        S    10:50   0:00 [idle_inject/0]
root          19  0.0  0.0      0     0 ?        S    10:50   0:00 [cpuhp/0]
root          21  0.0  0.0      0     0 ?        S    10:50   0:00 [kdevtmpfs]
root          22  0.0  0.0      0     0 ?        I<   10:50   0:00 [kworker/R-inet_]
root          23  0.0  0.0      0     0 ?        S    10:50   0:00 [kauditd]
root          25  0.0  0.0      0     0 ?        S    10:50   0:00 [khungtaskd]
root          26  0.0  0.0      0     0 ?        S    10:50   0:00 [oom_reaper]
root          29  0.0  0.0      0     0 ?        I<   10:50   0:00 [kworker/R-write]
root          30  0.0  0.0      0     0 ?        S    10:50   0:00 [kcompactd0]
root          31  0.0  0.0      0     0 ?        SN   10:50   0:00 [ksmd]
root          32  0.0  0.0      0     0 ?        I<   10:50   0:00 [kworker/R-crypt]
root          33  0.0  0.0      0     0 ?        I<   10:50   0:00 [kworker/R-kinte]
root          34  0.0  0.0      0     0 ?        I<   10:50   0:00 [kworker/R-kbloc]
root          35  0.0  0.0      0     0 ?        I<   10:50   0:00 [kworker/R-blkcg]
:
```

ps aux –sort –pmem | less

Project-03-Task-01 1% 0.24 GB / 0.41 GB 0.01 Mb/s 0.00 Mb/s 47 min root /: 12% /tmp: 0% /boot: 29%


10.10.2.31 ([openEuler])
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
/root/
Name
..
.ssh
.bash_history
.bash_logout
.bash_profile
.bashrc
.cshrc
.tcshrc
anaconda-ks.cfg
Remote monitoring
Follow terminal folder
3. 10.10.2.31 ([openEuler])
[root@Project-03-Task-01 ~]# pstree
systemd─┬─NetworkManager───3*[{NetworkManager}]
├─agetty
├─auditd───{auditd}
├─chronyd
├─crond
├─dbus-daemon
├─firewalld───{firewalld}
├─polkitd───3*[{polkitd}]
├─restorecond
├─rngd───3*[{rngd}]
├─rsyslogd───2*[{rsyslogd}]
├─sshd─┬─sshd───sshd─┬─bash───pstree
│ └─bash───sleep
│ └─sshd───sshd───sftp-server
├─systemd-journal
├─systemd-logind
├─systemd-udevd
└─tuned───3*[{tuned}]
[root@Project-03-Task-01 ~]# pstree -a
systemd --switched-root --system --deserialize=34
├─NetworkManager --no-daemon
│ └─3*[{NetworkManager}]
├─agetty -o -p -- \\u --noclear - linux
├─auditd
│ └─{auditd}
├─chronyd
├─crond -n
├─dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
├─firewalld -s /usr/sbin/firewalld --nofork --nopid
│ └─{firewalld}
├─polkitd --no-debug
│ └─3*[{polkitd}]
├─restorecond
├─rngd -f -O jitter:timeout:10
│ └─3*[{rngd}]
├─rsyslogd -n -i/var/run/rsyslogd.pid
│ └─2*[{rsyslogd}]
├─sshd
│ ├─sshd
│ │ └─sshd
│ │ ├─bash
│ │ │ └─pstree -a
│ │ └─bash -c ...
│ │ └─sleep 1
│ └─sshd
│ └─sshd
│ └─sftp-server -l INFO -f AUTH
├─systemd-journal
├─systemd-logind
├─systemd-udevd
└─tuned -Es /usr/sbin/tuned -l -P
└─3*[{tuned}]
[root@Project-03-Task-01 ~]#
用法：pstree [-acglpsStTuZ] [-h | -H 进程号] [-n | -N 类型]
[-A | -G | -U] [进程号 | 用户]
或：pstree -V
显示进程树。
-a, --arguments 显示命令行参数
-A, --ascii 使用 ASCII 行绘制字符
-c, --compact-not 不要对完全相同的子树进行压缩
-C, --color=类型 按照指定属性对进程上色
(age)
-g, --show-pgids 显示进程组 ID；隐含启用 -c 选项
-G, --vt100 使用 VT100 行绘制字符
-h, --highlight-all 高亮显示当前进程和其所有祖先
-H PID, --highlight-pid=PID
高亮显示指定 PID 对应的进程和其所有祖先
-l, --long 不要截断长行
-n, --numeric-sort 按照 PID 对输出进行排序
-N 类型, --ns-sort=类型
按照指定命名空间类型对输出进行排序
(cgroup, ipc, mnt, net, pid, time, user, uts)
-p, --show-pids 显示 PID；隐含启用 -c 选项
-s, --show-parents 显示所选进程的父进程
-S, --ns-changes 显示命名空间的变化
-t, --thread-names 显示完整线程名称
-T, --hide-threads 隐藏线程，只显示进程
-u, --uid-changes 显示用户 ID（UID）的变化
-U, --unicode 使用 UTF-8（Unicode）的行绘制字符
-V, --version 显示版本信息
-Z, --security-context
显示安全属性
进程号 从指定进程号开始；默认为 1（init）
用户 仅显示从指定用户的进程开始的进程树
Project-03-Task-01 1% 0.23 GB / 0.41 GB 0.01 Mb/s 0.00 Mb/s 54 min root /: 12% /tmp: 0% /boot: 29%
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

3. 进程管理

- 在 Linux 终端中执行命名，分前台运行和后台运行两种。
- 前台运行：
  - 在终端中运行命令必须等命令完成后才能运行另一个，称为在前台或前台进程运行命令。
  - 当进程在前台运行时，将占用当前 Shell 会话，并可以使用输入设备与之进行交互。
- 后台运行：
  - 后台运行也成为后台进程。
  - 即在终端启动后，转入到后台运行，不占用当前的 Shell 会话，也无需用户交互的进程。
  - 让程序在后台运行的方法有四种途径：
    - 使用 & 运算符在后台运行命令，通常所说的在命令后面加“&”符号。
    - 使用后台运行工具运行命令，常用工具有 nohup、screen、tmux。
    - 作为系统服务来执行，使用 systemd 进行管理。
    - 作为任务计划来执行，使用 at、crontab。

```
[root@Project-03-Task-01 ~]# ping www.51xueweb.cn
PING www.51xueweb.cn (123.57.38.22) 56(84) 字节的数据。
64 字节，来自 123.57.38.22 (123.57.38.22): icmp_seq=1 ttl=128 时间=27.4 毫秒
64 字节，来自 123.57.38.22 (123.57.38.22): icmp_seq=2 ttl=128 时间=24.7 毫秒
64 字节，来自 123.57.38.22 (123.57.38.22): icmp_seq=3 ttl=128 时间=29.2 毫秒
64 字节，来自 123.57.38.22 (123.57.38.22): icmp_seq=4 ttl=128 时间=25.0 毫秒
^C
--- www.51xueweb.cn ping 统计 ---
已发送 4 个包， 已接收 4 个包, 0% packet loss, time 5189ms
rtt min/avg/max/mdev = 24.687/26.568/29.188/1.832 ms
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# ping www.51xueweb.cn &
[1] 29243
[root@Project-03-Task-01 ~]# PING www.51xueweb.cn (123.57.38.22) 56(84) 字节的数据。
64 字节，来自 123.57.38.22 (123.57.38.22): icmp_seq=1 ttl=128 时间=24.5 毫秒
64 字节，来自 123.57.38.22 (123.57.38.22): icmp_seq=2 ttl=128 时间=24.7 毫秒
64 字节，来自 123.57.38.22 (123.57.38.22): icmp_seq=3 ttl=128 时间=24.9 毫秒
64 字节，来自 123.57.38.22 (123.57.38.22): icmp_seq=4 ttl=128 时间=24.7 毫秒
^C
[root@Project-03-Task-01 ~]# 64 字节，来自 123.57.38.22 (123.57.38.22): icmp_seq=5 ttl=128 时间=24.2 毫秒
64 字节，来自 123.57.38.22 (123.57.38.22): icmp_seq=6 ttl=128 时间=37.3 毫秒
64 字节，来自 123.57.38.22 (123.57.38.22): icmp_seq=7 ttl=128 时间=48.9 毫秒
64 字节，来自 123.57.38.22 (123.57.38.22): icmp_seq=8 ttl=128 时间=30.1 毫秒
^C
[root@Project-03-Task-01 ~]# kill64 字节，来自 123.57.38.22 (123.57.38.22): icmp_seq=9 ttl=128 时间=34.0 毫秒
64 字节，来自 123.57.38.22 (123.57.38.22): icmp_seq=10 ttl=128 时间=23.9 毫秒
 64 字节，来自 123.57.38.22 (123.57.38.22): icmp_seq=11 ttl=128 时间=31.7 毫秒
64 字节，来自 123.57.38.22 (123.57.38.22): icmp_seq=12 ttl=128 时间=30.0 毫秒
29243
[root@Project-03-Task-01 ~]#
[1]+  Terminated              ping www.51xueweb.cn
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# ping www.51xueweb.cn > /dev/null &
[1] 29508
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# jobs
[1]+  运行中               ping www.51xueweb.cn > /dev/null &
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# kill -9 29508
[root@Project-03-Task-01 ~]#
[1]+  Killed                  ping www.51xueweb.cn > /dev/null
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# jobs
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]#
```

3. 进程管理

jobs [选项] [参数]

功能：

- 查看任务列表及任务状态，包括后台运行的任务。
- 可以显示任务号及其对应的进程号。

参数/命令：

- 任务标识号：指定要显示的任务识别号。

主要选项：

- -l：显示进程号
- -p：仅任务对应的显示进程号
- -n：显示任务状态的变化
- -r：仅输出运行状态（running）的任务
- -s：仅输出停止状态（stoped）的任务

3. 进程管理

jobs [选项] [参数]

功能：

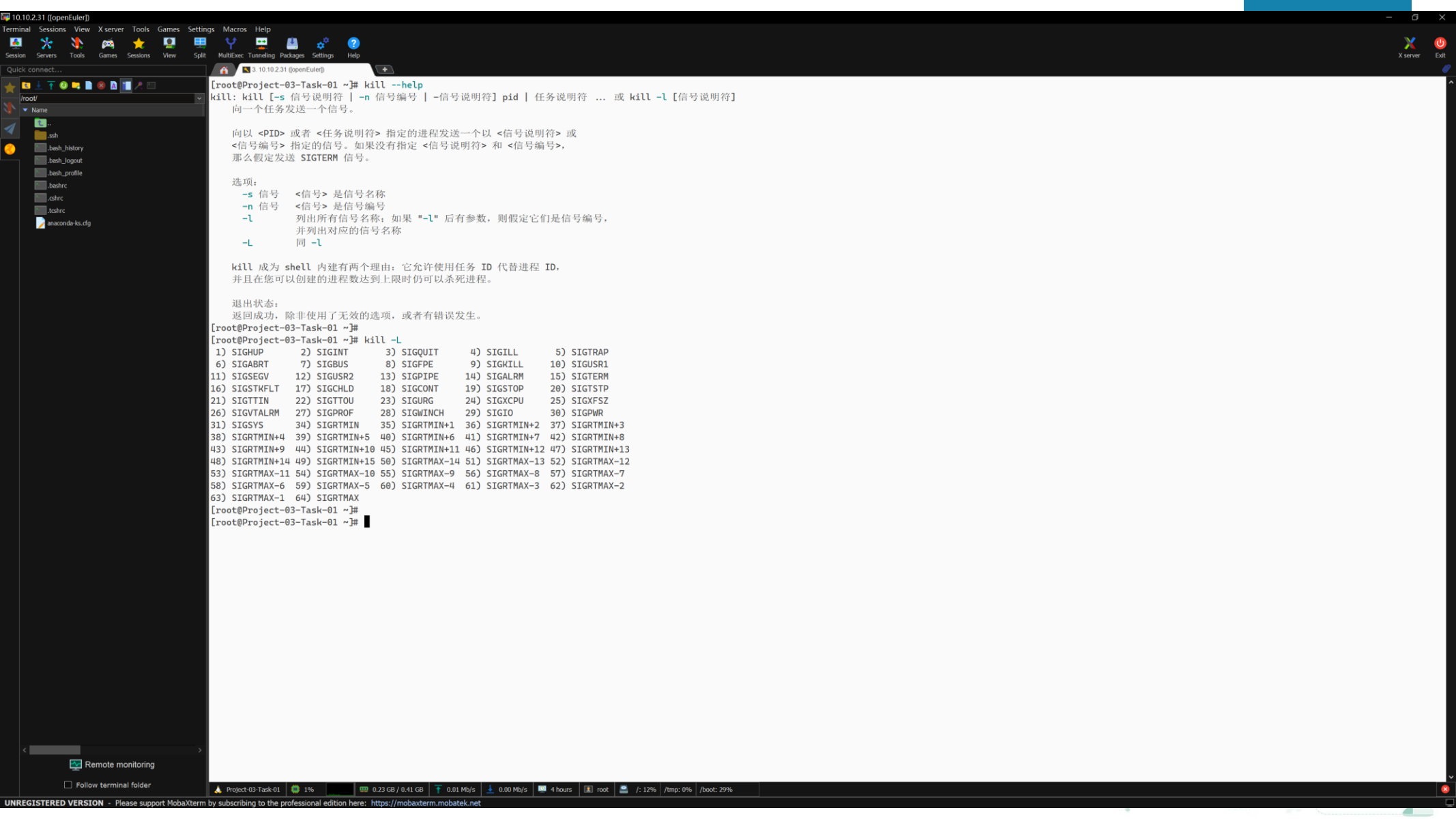
- 向指定进程发送指令信息。
- 不指定，则发送SIGTERM(15)终止进程指令。

参数/命令：

- 进程或作业识别号：
  - 指定要删除的进程或作业。
- 常用的指令信号：
  - HUP 1 终端断线
  - INT 2 中断（同 Ctrl + C）
  - QUIT 3 退出（同 Ctrl + \）
  - KILL 9 强制终止
  - CONT 18 继续（与STOP相反）
  - STOP 19 暂停（同 Ctrl + Z）

主要选项：

- -a：处理当前进程时，不限制命令名和进程号的对应关系；
- -l <信息编号>：
  - 若不加<信息编号>选项，则-l参数会列出全部的信息名称；
- -p：只打印相关进程的进程号，而不发送任何信号
- -s <信息名称或编号>：指定要送出的信息
- -u：指定用户


10.10.2.31 ([openEuler])
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
3. 10.10.2.31 ([openEuler])
/root/
Name
..
.ssh
.bash_history
.bash_logout
.bash_profile
.bashrc
.cshrc
.tcshrc
anaconda-ks.cfg
[root@Project-03-Task-01 ~]# kill --help
kill: 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]
向一个任务发送一个信号。
向以 <PID> 或者 <任务说明符> 指定的进程发送一个以 <信号说明符> 或
<信号编号> 指定的信号。如果没有指定 <信号说明符> 和 <信号编号>，
那么假定发送 SIGTERM 信号。
选项：
-s 信号 <信号> 是信号名称
-n 信号 <信号> 是信号编号
-l 列出所有信号名称；如果 "-l" 后有参数，则假定它们是信号编号，
并列出对应的信号名称
-L 同 -l
kill 成为 shell 内建有两个理由：它允许使用任务 ID 代替进程 ID，
并且在您可以创建的进程数达到上限时仍可以杀死进程。
退出状态：
返回成功，除非使用了无效的选项，或者有错误发生。
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# 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
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]#
Remote monitoring
Follow terminal folder
Project-03-Task-01
1%
0.23 GB / 0.41 GB
0.01 Mb/s
0.00 Mb/s
4 hours
root
/: 12%
/tmp: 0%
/boot: 29%
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

3. 进程管理

- 在 Linux 操作系统中，进程在执行时都会赋予一个优先等级，等级越高，进程获得 CPU 时间就会越多，所以级别越高的进程，运行的时间就会越短，反之则需要较长的运行时间。
- 进程的优先等级范围为-20~19。
  - -20 表示最高等级，而 19 则是最低。
  - 等级 -1~-20 只有 root 用户可以设置，进程运行的默认优先等级为 0。

3. 进程管理

nice [选项] [参数]

功能：

- 指定的进程调度优先级启动其他的程序。

说明：

- 选项
  - -n：指定进程的优先级
- 参数：
  - 指令及选项：需要运行的指令及其他选项。

renice [选项] [参数]

功能：

- 修改正在运行的进程的调度优先级。

说明：

- 选项
  - -g：指定进程组id；
  - -p<程序识别码>：
    - 改变该程序的优先权等级，此参数为预设值。
  - -u：指定开启进程的用户名。
- 参数
  - 进程号：指定要修改优先级的进程。

4. 任务计划

- openEuler 中的任务计划包括一次性定时任务和周期性任务，允许用户在特定的时间自动执行命令或脚本，实现各种自动化的操作，从而提高了工作效率和系统的稳定性。
- 一次性任务计划：at
  - at 命令是一次性定时计划任务，允许用户在特定时间运行命令或脚本。
  - 系统默认没有安装，需提前安装 at 软件包，并开启 atd 服务。
  - yum install at
- 周期性任务计划：crontab

4. 任务计划

at [参数] [时间]

功能：

参数/命令：

- -m：指定的任务完成后，给用户发送邮件
- -v：显示任务将被执行的时间
- -c：打印任务的内容到标准输出
- -q<列队>：使用指定的列队
- -f<文件>：从指定文件读入任务而不是从标准输入读入
- -t<时间参数>：以时间参数的形式提交要运行的任务

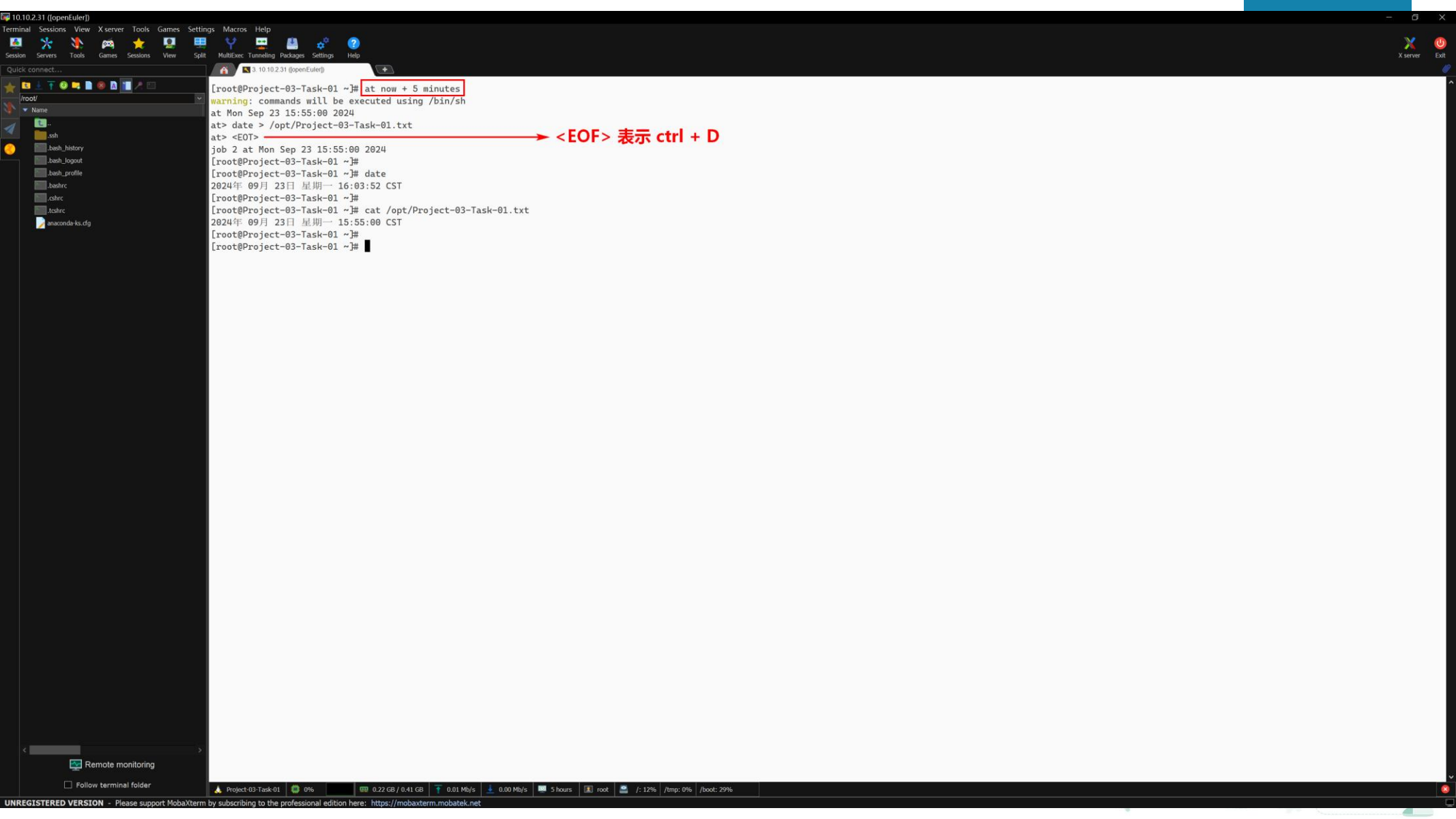
主要选项：

- 时间可以按照以下5类时间格式指定。
  - hh:mm：小时:分钟(当天，如时间已过，则在第二天执行)
  - 描述时间
    - midnight（00:00）
    - noon（12:00 PM）
    - teatime（下午4点）
    - today
    - tomorrow等
  - 12小时计时制：时间后加am(上午)或pm(下午)
  - 指定具体执行日期：mm/dd/yy（月/日/年）或 dd.mm.yy（日.月.年）
  - 相对计时法：
    - now + n units
    - now是现在时刻，n为数字
    - units是单位(minutes、hours、days、weeks)

```
[root@Project-03-Task-01 ~]# at
-bash: at: 未找到命令
[root@Project-03-Task-01 ~]# yum -y install at
Last metadata expiration check: 0:13:46 ago on 2024年09月23日 星期一 15时31分46秒.
Dependencies resolved.
```

Package	Architecture	Version	Repository	Size
Installing:				
at	x86_64	3.2.5-1.oe2403	OS	53 k
Transaction Summary				

```
Install  1 Package

Total download size: 53 k
Installed size: 126 k
Downloading Packages:
at-3.2.5-1.oe2403.x86_64.rpm                                    16 kB/s |  53 kB     00:03
Total                                                           9.5 kB/s |  53 kB     00:05
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                   1/1
  Running scriptlet: at-3.2.5-1.oe2403.x86_64                                          1/1
  Installing       : at-3.2.5-1.oe2403.x86_64                                          1/1
  Running scriptlet: at-3.2.5-1.oe2403.x86_64                                          1/1
Created symlink /etc/systemd/system/multi-user.target.wants/atd.service → /usr/lib/systemd/system/atd.service.

  Verifying        : at-3.2.5-1.oe2403.x86_64                                          1/1

Installed:
  at-3.2.5-1.oe2403.x86_64

Complete!
[root@Project-03-Task-01 ~]# systemctl enable atd
[root@Project-03-Task-01 ~]# systemctl start atd
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# at --help
at: invalid option -- '-'
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
[root@Project-03-Task-01 ~]# at
Garbled time
[root@Project-03-Task-01 ~]#
```


10.10.2.31 ([openEuler])
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
/root/
Name
..
.ssh
.bash_history
.bash_logout
.bash_profile
.bashrc
.cshrc
.tcshrc
anaconda-ks.cfg
Remote monitoring
Follow terminal folder
3. 10.10.2.31 ([openEuler])
[root@Project-03-Task-01 ~]# at now + 5 minutes
warning: commands will be executed using /bin/sh
at Mon Sep 23 15:55:00 2024
at> date > /opt/Project-03-Task-01.txt
at> <EOT>
<EOF> 表示 ctrl + D
job 2 at Mon Sep 23 15:55:00 2024
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# date
2024年 09月 23日 星期一 16:03:52 CST
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# cat /opt/Project-03-Task-01.txt
2024年 09月 23日 星期一 15:55:00 CST
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]#
Project-03-Task-01 0% 0.22 GB / 0.41 GB 0.01 Mb/s 0.00 Mb/s 5 hours root /: 12% /tmp: 0% /boot: 29%
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

4. 任务计划

atq [选项]

功能：

- 列出当前用户的 at 任务列表。

说明：

- 选项
  - -V：显示版本号；
  - -q：查询指定队列的任务。

atrm [选项] [参数]

功能：

- 删除待执行任务队列中的指定任务。

说明：

- 选项
  - -V：显示版本号。
- 参数
  - 任务号：指定待执行队列中要删除的任务。

```
[root@Project-03-Task-01 ~]# at now + 2 minutes
warning: commands will be executed using /bin/sh
at Mon Sep 23 16:20:00 2024
at> date >> /opt/Project-03-Task-01.txt
at> <EOT>
job 6 at Mon Sep 23 16:20:00 2024
[root@Project-03-Task-01 ~]# at now + 3 minutes
warning: commands will be executed using /bin/sh
at Mon Sep 23 16:22:00 2024
at> date >> /opt/Project-03-Task-01.txt
at> <EOT>
job 7 at Mon Sep 23 16:22:00 2024
[root@Project-03-Task-01 ~]# at now + 4 minutes
warning: commands will be executed using /bin/sh
at Mon Sep 23 16:23:00 2024
at> date >> /opt/Project-03-Task-01.txt
at> <EOT>
job 8 at Mon Sep 23 16:23:00 2024
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# atq
8       Mon Sep 23 16:23:00 2024 a root
6       Mon Sep 23 16:20:00 2024 a root
7       Mon Sep 23 16:22:00 2024 a root
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# atrm 7
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# atq
8       Mon Sep 23 16:23:00 2024 a root
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# date
2024年 09月 23日 星期一 16:26:24 CST
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# cat /opt/Project-03-Task-01.txt
2024年 09月 23日 星期一 15:55:00 CST
2024年 09月 23日 星期一 16:10:00 CST
2024年 09月 23日 星期一 16:20:00 CST
2024年 09月 23日 星期一 16:23:00 CST
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]# atq
[root@Project-03-Task-01 ~]#
[root@Project-03-Task-01 ~]#
```

4. 任务计划

- Linux下的任务调度分为两类：系统任务调度和用户任务调度。
  - 系统任务调度：
    - 系统周期性所要执行的工作，比如写缓存数据到硬盘、日志清理等。
    - 在/etc目录下有一个crontab文件，这个就是系统任务调度的配置文件。
    - /etc/crontab文件包括下面几行：

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=""HOME=/
# run-parts
51 * * * * root run-parts /etc/cron.hourly
24 7 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

    - 前四行是用来配置crond任务运行的环境变量：
      - 第一行SHELL变量指定了系统要使用哪个shell，这里是bash。
      - 第二行PATH变量指定了系统执行命令的路径。
      - 第三行MAILTO变量指定了crond的任务执行信息将通过电子邮件发送给root用户，如果MAILTO变量的值为空，则表示不发送任务执行信息给用户。
      - 第四行的HOME变量指定了在执行命令或者脚本时使用的主目录。

4. 任务计划

- Linux下的任务调度分为两类：系统任务调度和用户任务调度。
  - 用户任务调度：
    - 用户定期要执行的任务可以使用 crontab 工具来定制计划任务。
    - 用户定义的都保存在/var/spool/cron目录中，文件名与用户名一致。
    - 使用者权限文件如下：
      - /etc/cron.deny　该文件中所列用户不允许使用crontab命令。
      - /etc/cron.allow　该文件中所列用户允许使用crontab命令。
      - /var/spool/cron/　所有用户crontab文件存放的目录。

4. 任务计划

- Linux下的任务调度分为两类：系统任务调度和用户任务调度。
  - 用户任务调度：
    - 以用户名命名 crontab 文件的含义：
      - 用户所建立的crontab文件中，每一行都代表一项任务，每个字段代表一项设置。
      - 共分为六个字段。前五段是时间设定段，第六段是要执行的命令段。格式如下：
        - minute hour day month week command 顺序：分 时 日 月 周
        - minute： 表示分钟，可以是从0到59之间的任何整数。
        - hour：表示小时，可以是从0到23之间的任何整数。
        - day：表示日期，可以是从1到31之间的任何整数。
        - month：表示月份，可以是从1到12之间的任何整数。
        - week：表示星期几，可以是从0到7之间的任何整数，这里的0或7代表星期日。
        - command：要执行的命令，可以是系统命令，也可以是自己编写的脚本文件。
      - 在以上各个字段中，还可以使用以下特殊字符：
        - 星号（*）：代表所有可能的值。。
        - 逗号（,）：可以用逗号隔开的值指定一个列表范围，例如，“1,2,5,7,8,9”
        - 中杠（-）：可以用整数之间的中杠表示一个整数范围，例如“2-6”表示“2,3,4,5,6”
        - 正斜线（/）：可以用正斜线指定时间的间隔频率，例如“0-23/2”表示每两小时执行一次。
        - 正斜线可以和星号一起使用，例如*/10，如果用在minute字段，表示每十分钟执行一次。

4. 任务计划

crontab [选项] [参数]

功能：

- 用来提交和管理用户的需要周期性执行的任务。系统默认安装此服务，crond进程每分钟会定期检查是否有要执行的任务。

参数/命令：

- crontab文件：
  - 指定包含待执行任务的crontab文件。

主要选项：

- -e：编辑该用户的计时器设置
- -l：列出该用户的计时器设置
- -r：删除该用户的计时器设置
- -u<用户名称>：指定要设定计时器的用户名称

4. 任务计划

crontab [选项] [参数]

操作命令与脚本程序：

- 每1分钟执行一次command
  - * * * * * command
- 每小时的第3和第15分钟执行
  - 3,15 * * * * command
- 在上午8点到11点的第3和第15分钟执行
  - 3,15 8-11 * * * command
- 每隔两天的上午8点到11点的第3和第15分钟执行
  - 3,15 8-11 */2 * * command
- 每星期一的上午8点到11点的第3和第15分钟执行
  - 3,15 8-11 * * 1 command

计划任务条目格式

第一列	第二列	第三列	第四列	第五列	第六列
分钟（0-59）	小时（0-23）	日（1-31）	月（1-12）	周（0-7）	命令（允许空格）

第一列至第五列为时间段，`*` 表示所有时间，`-` 表示一段连续的时间，`,` 表示不连续的时间，`/` 表示间隔的时间。

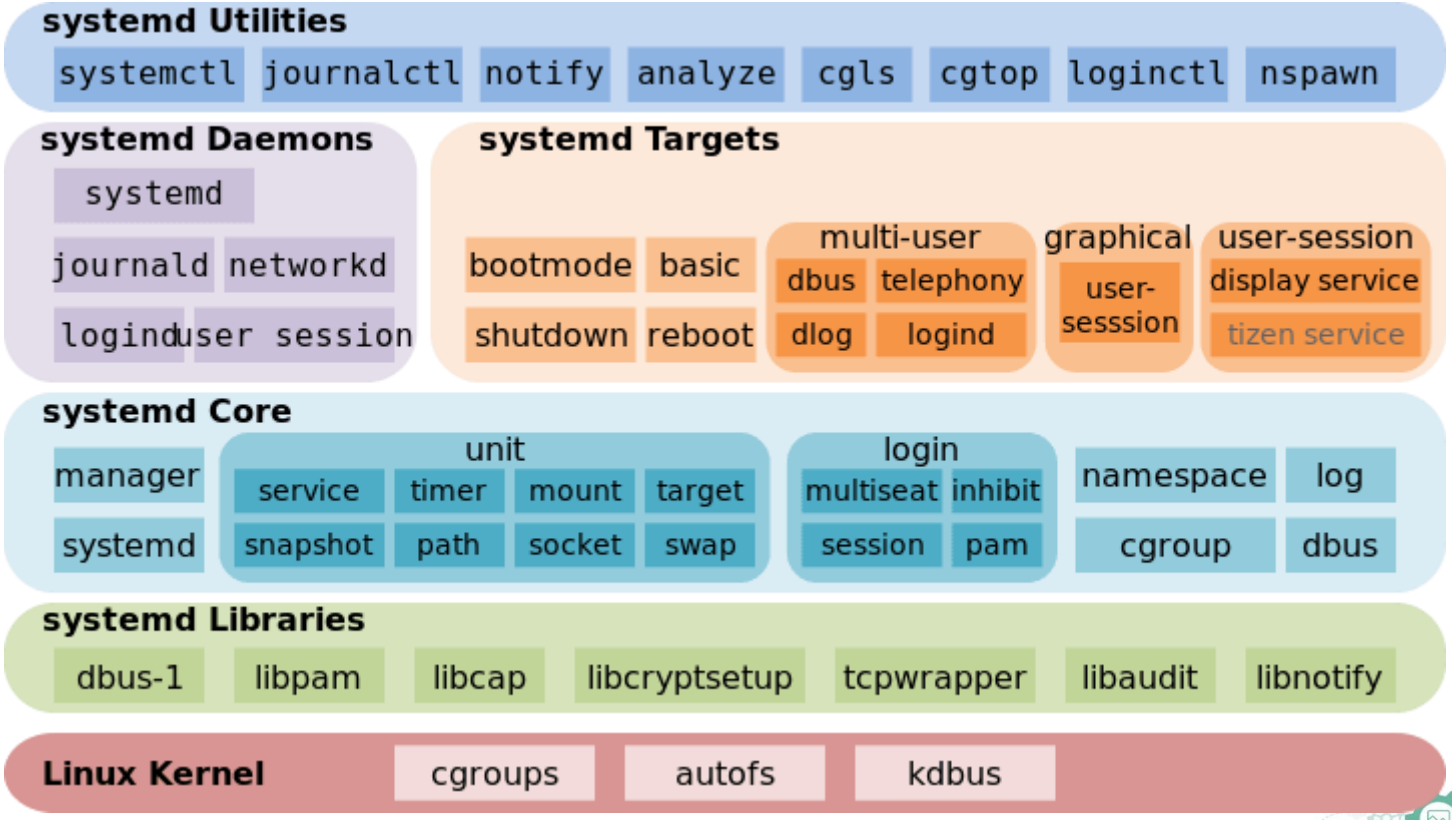
时间格式举例

第一列	第二列	第三列	第四列	第五列	意义
*	*	*	*	*	每分钟执行一次
0	8-22	*	*	*	8-22 点整点执行一次
30	*/2	*	*	*	从 0:30 分开始每隔 2 小时执行一次
0	0	10	2-12/2	*	每偶数月 10 日 0 点执行一次

5. 服务管理

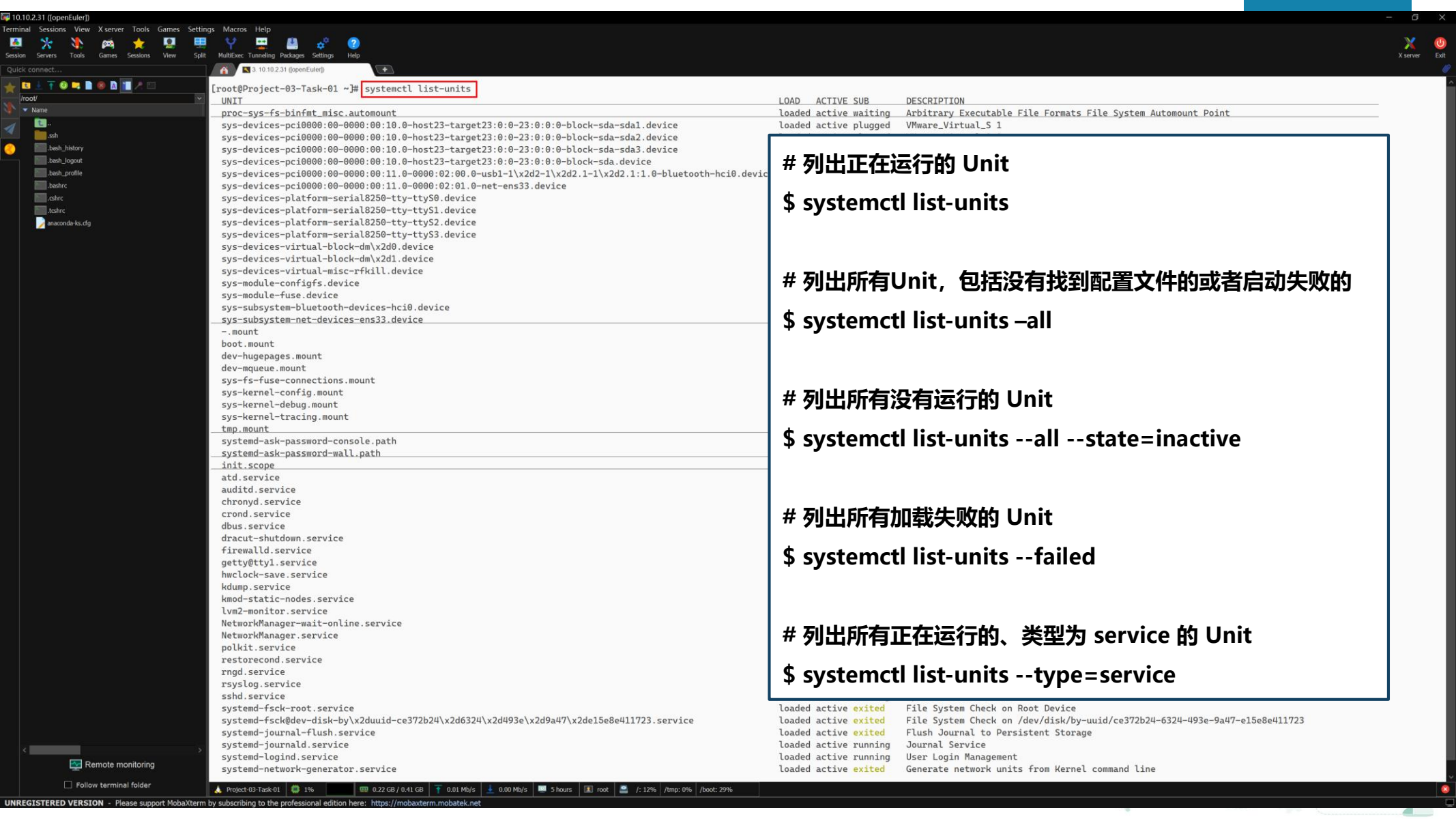
- 服务是一种特殊的进程，可以在后台运行并提供一系列的功能和服务。
- openEuler 默认使用 systemd 进行服务管理。
  - 历史上，Linux 的启动一直采用 init 进程，但 init 有两个缺点。
    - #/etc/init.d/apache2 start
    - # service apache2 start
    - 一是启动时间长。init 进程是串行启动，只有前一个进程启动完，才会启动下一个进程。
    - 二是启动脚本复杂。init 进程只是执行启动脚本，脚本需要处理各种情况，使脚本变得复杂。
  - Systemd 就是为了解决这些问题而诞生的，其设计目标是：
    - 为系统的启动和管理提供一套完整的解决方案。
    - 字母d是守护进程（daemon）的缩写。
    - Systemd 名字的含义是它要守护整个系统。
  - 使用了 Systemd，就不需要再用init了。
    - Systemd 取代了initd，成为系统的第一个进程（PID 等于 1），其他进程都是它的子进程。
  - Systemd 的优点是功能强大，使用方便，缺点是体系庞大，非常复杂。
    - 现在还有很多人反对使用 Systemd，理由就是它过于复杂，
    - 与操作系统的其他部分强耦合，违反“keep simple, keep stupid”的 Unix 哲学。

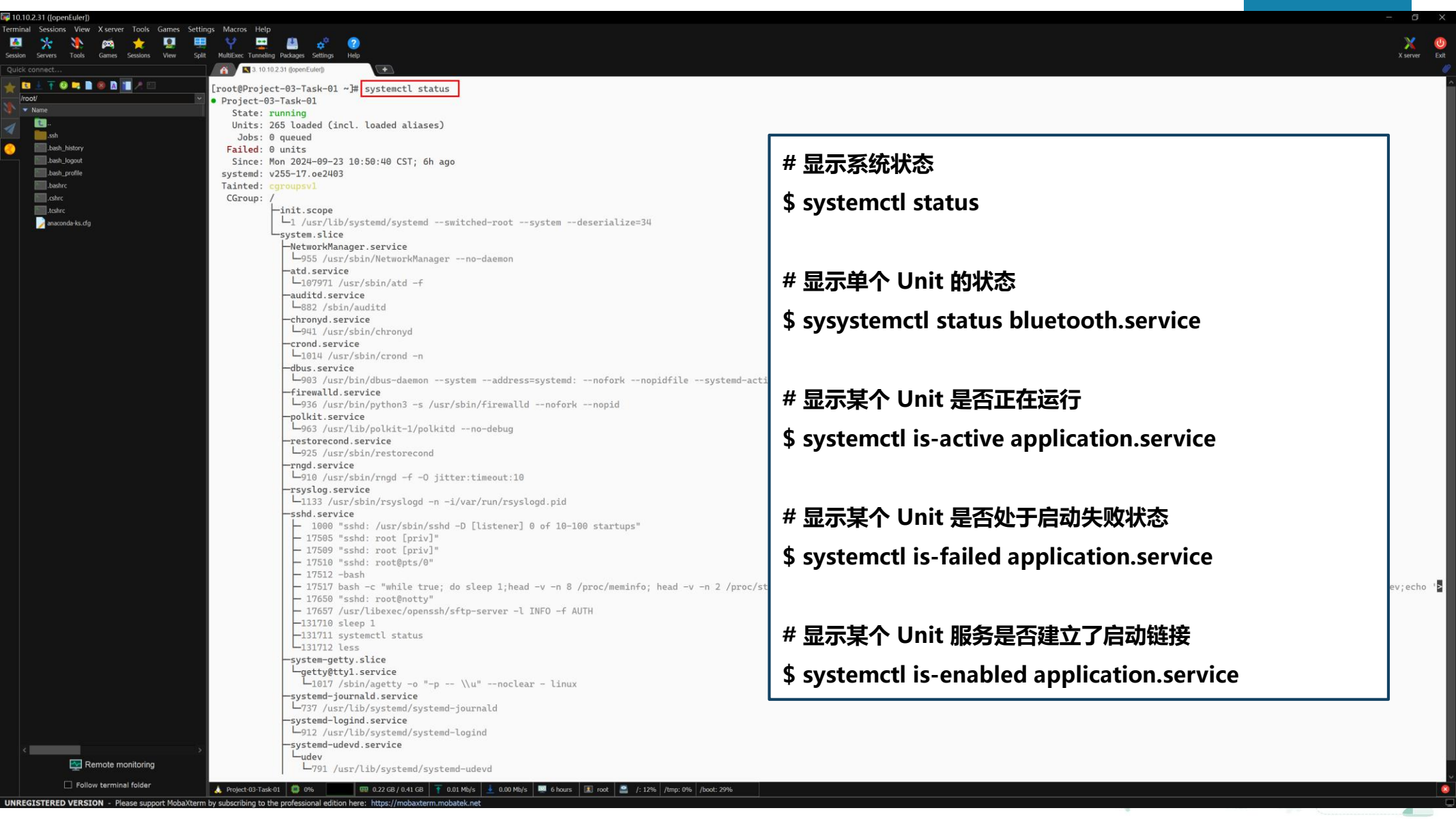
Systemd 架构图


systemd Utilities
systemctl
journalctl
notify
analyze
cgls
cgtop
loginctl
nspawn
systemd Daemons
systemd
journald
networkd
logind
user session
systemd Targets
bootmode
basic
shutdown
reboot
multi-user
dbus
telephony
dlog
logind
graphical
user-sesssion
user-session
display service
tizen service
systemd Core
manager
systemd
unit
service
timer
mount
target
snapshot
path
socket
swap
login
multiseat
inhibit
session
pam
namespace
log
cgroup
dbus
systemd Libraries
dbus-1
libpam
libcap
libcryptsetup
tcpwrapper
libaudit
libnotify
Linux Kernel
cgroups
autofs
kdbus

5. 服务管理

- Systemd 并不是一个命令，而是一组命令，涉及到系统管理的方方面面。
  - Systemd 可以管理所有系统资源。不同的资源统称为 Unit（单位）。
  - Unit 一共分成12种。
    - Service unit：系统服务
    - Target unit：多个 Unit 构成的一个组
    - Device Unit：硬件设备
    - Mount Unit：文件系统的挂载点
    - Automount Unit：自动挂载点
    - Path Unit：文件或路径
    - Scope Unit：不是由 Systemd 启动的外部进程
    - Slice Unit：进程组
    - Snapshot Unit：Systemd 快照，可以切回某个快照
    - Socket Unit：进程间通信的 socket
    - Swap Unit：swap 文件
    - Timer Unit：定时器


10.10.2.31 ([openEuler])
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
/root/
Name
.ssh
.bash_history
.bash_logout
.bash_profile
.bashrc
.cshrc
.tcshrc
anaconda-ks.cfg
Remote monitoring
Follow terminal folder
[root@Project-03-Task-01 ~]# systemctl list-units
UNIT LOAD ACTIVE SUB DESCRIPTION
proc-sys-fs-binfmt_misc.automount loaded active waiting Arbitrary Executable File Formats File System Automount Point
sys-devices-pci0000:00-0000:00:10.0-host23-target23:0:0-23:0:0:0-block-sda-sda1.device loaded active plugged VMware_Virtual_S 1
sys-devices-pci0000:00-0000:00:10.0-host23-target23:0:0-23:0:0:0-block-sda-sda2.device
sys-devices-pci0000:00-0000:00:10.0-host23-target23:0:0-23:0:0:0-block-sda-sda3.device
sys-devices-pci0000:00-0000:00:10.0-host23-target23:0:0-23:0:0:0-block-sda.device
sys-devices-pci0000:00-0000:00:11.0-0000:02:00.0-usb1-1\x2d2-1\x2d2.1-1\x2d2.1:1.0-bluetooth-hci0.devic
sys-devices-pci0000:00-0000:00:11.0-0000:02:01.0-net-ens33.device
sys-devices-platform-serial8250-tty-ttyS0.device
sys-devices-platform-serial8250-tty-ttyS1.device
sys-devices-platform-serial8250-tty-ttyS2.device
sys-devices-platform-serial8250-tty-ttyS3.device
sys-devices-virtual-block-dm\x2d0.device
sys-devices-virtual-block-dm\x2d1.device
sys-devices-virtual-misc-rfkill.device
sys-module-configfs.device
sys-module-fuse.device
sys-subsystem-bluetooth-devices-hci0.device
sys-subsystem-net-devices-ens33.device
-.mount
boot.mount
dev-hugepages.mount
dev-mqueue.mount
sys-fs-fuse-connections.mount
sys-kernel-config.mount
sys-kernel-debug.mount
sys-kernel-tracing.mount
tmp.mount
systemd-ask-password-console.path
systemd-ask-password-wall.path
init.scope
atd.service
auditd.service
chronyd.service
crond.service
dbus.service
dracut-shutdown.service
firewalld.service
getty@tty1.service
hwclock-save.service
kdump.service
kmod-static-nodes.service
lvm2-monitor.service
NetworkManager-wait-online.service
NetworkManager.service
polkit.service
restorecond.service
rngd.service
rsyslog.service
sshd.service
systemd-fsck-root.service loaded active exited File System Check on Root Device
systemd-fsck@dev-disk-by\x2duuid-ce372b24\x2d6324\x2d493e\x2d9a47\x2de15e8e411723.service loaded active exited File System Check on /dev/disk/by-uuid/ce372b24-6324-493e-9a47-e15e8e411723
systemd-journal-flush.service loaded active exited Flush Journal to Persistent Storage
systemd-journald.service loaded active running Journal Service
systemd-logind.service loaded active running User Login Management
systemd-network-generator.service loaded active exited Generate network units from Kernel command line
# 列出正在运行的 Unit
$ systemctl list-units
# 列出所有Unit，包括没有找到配置文件的或者启动失败的
$ systemctl list-units –all
# 列出所有没有运行的 Unit
$ systemctl list-units --all --state=inactive
# 列出所有加载失败的 Unit
$ systemctl list-units --failed
# 列出所有正在运行的、类型为 service 的 Unit
$ systemctl list-units --type=service
Project-03-Task-01 1% 0.22 GB / 0.41 GB 0.01 Mb/s 0.00 Mb/s 5 hours root /: 12% /tmp: 0% /boot: 29%
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net



 # 显示系统状态

$ systemctl status

 # 显示单个 Unit 的状态

$ sysystemctl status bluetooth.service

 # 显示某个 Unit 是否正在运行

$ systemctl is-active application.service

 # 显示某个 Unit 是否处于启动失败状态

$ systemctl is-failed application.service

 # 显示某个 Unit 服务是否建立了启动链接

$ systemctl is-enabled application.service

5. 服务管理

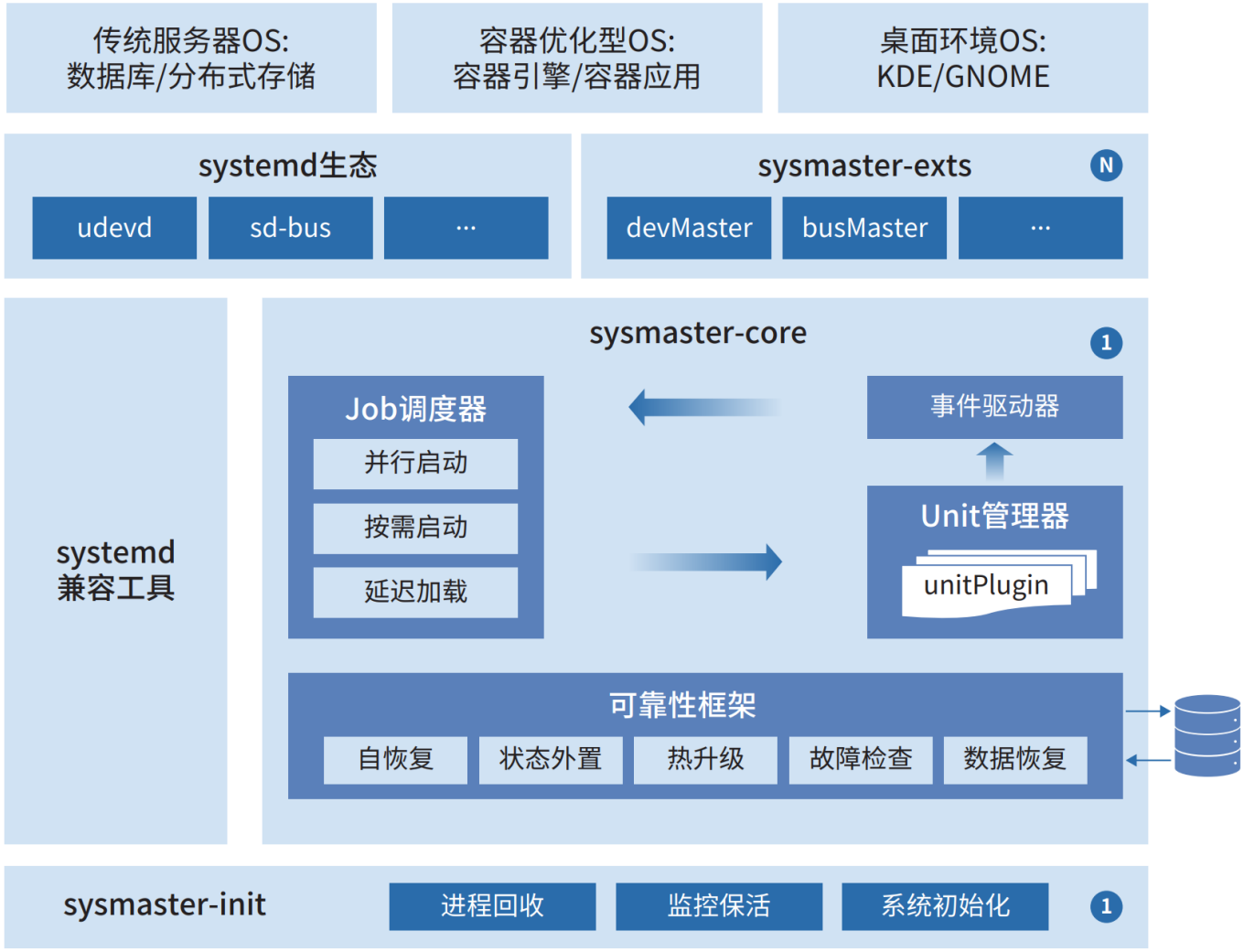
systemctl 用于启动和停止 Unit（主要是 service）。

操作命令与脚本程序：

- # 立即启动一个服务
- systemctl start apache.service
- # 立即停止一个服务
- systemctl stop apache.service
- # 重启一个服务
- systemctl restart apache.service
- # 杀死一个服务的所有子进程
- systemctl kill apache.service
- # 重新加载一个服务的配置文件
- systemctl reload apache.service
- # 重载所有修改过的配置文件
- systemctl daemon-reload
- # 显示某个 Unit 的所有底层参数
- systemctl show httpd.service
- # 显示某个 Unit 的指定属性的值
- systemctl show -p CPUShares httpd.service
- # 设置某个 Unit 的指定属性
- systemctl set-property httpd.service CPUShares=500

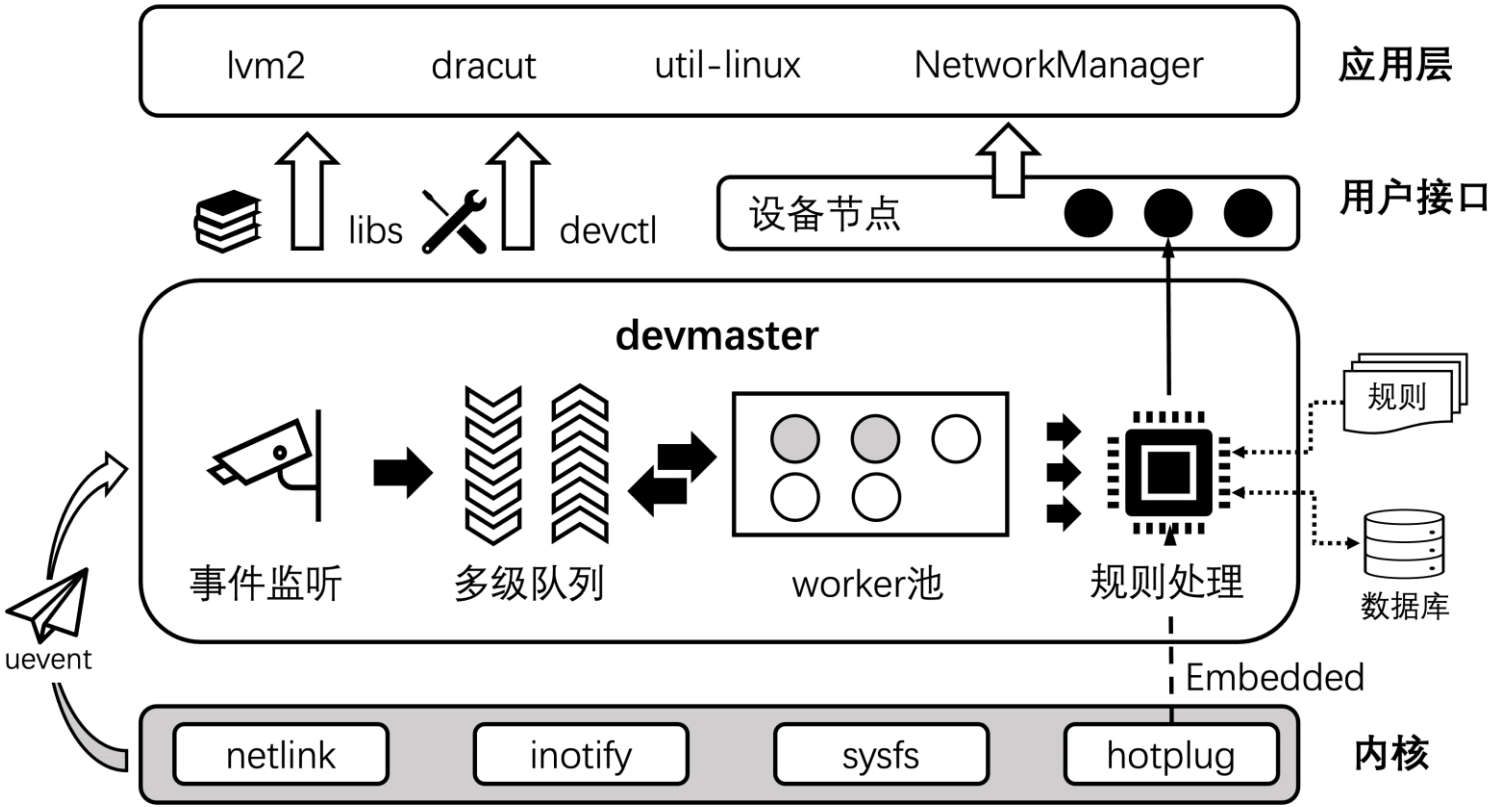
5. 服务管理

- sysMaster
  - sysMaster 是一套超轻量、高可靠的服务管理程序集合，是对1号进程的全新实现，旨在改进传统的 init 守护进程。
  - sysMaster 使用 Rust 编写，具有故障监测、秒级自愈和快速启动等能力，从而提升操作系统可靠性和业务可用度。
  - sysMaster 是 openEuler 对当前 Linux 系统初始化和服务管理在嵌入式，服务器，云化等不同场景下面临的问题和特点进行总结和思考后，开展的一种改进和探索，提供统一的，能够支持嵌入式、服务器、云场景下的系统初始化和服务（进程，容器，虚拟机）管理系统。
    - sysMaster 支持进程、容器和虚拟机的统一管理，其适用于服务器、云计算和嵌入式等多个场景。
    - sysMaster实现思路是将传统 1号进程的功能解耦分层，结合使用场景，拆分出 1+1+N的架构。
  - sysMaster 目前主要由 sysmaster 和 devmaster 两部分功能组成。
    - sysmaster 负责服务的管理。
    - devmaster 负责设备的管理。
      - devmaster 目前可应用于以 sysmaster 为1号进程的虚拟机环境。


传统服务器OS:
数据库/分布式存储
容器优化型OS:
容器引擎/容器应用
桌面环境OS:
KDE/GNOME
systemd生态
udevd
sd-bus
···
sysmaster-exts
N
devMaster
busMaster
···
systemd
兼容工具
sysmaster-core
1
Job调度器
并行启动
按需启动
延迟加载
事件驱动器
Unit管理器
unitPlugin
可靠性框架
自恢复
状态外置
热升级
故障检查
数据恢复
sysmaster-init
进程回收
监控保活
系统初始化
1

5. 服务管理

- devMaster
  - 设备管理器作为连接用户态软件与底层物理设备的桥梁，支撑着 lvm2、NetworkManager 等关键基础软件的运作。
  - devmaster 作为 sysMaster的设备管理组件，一方面支撑 sysMaster 的快速启动以及用户态软件的生态兼容，另一方面通过对 Linux 生态下主流设备管理方案的现状和优劣进行了总结和思考，从而提供一种分层解耦、可扩展性强、面向通用 OS的设备管理能力。
  - devmaster 由常驻进程、客户端工具和动态库组成。
    - 常驻进程 devmaster：
      - 基于内核提供的 netlink、inotify、sysfs 等机制，监听设备事件并触发规则处理任务。
    - 客户端工具 devctl 和动态库 libs：
      - 提供一组命令行指令以及公开接口，用于调试规则、控制常驻进程、查询设备状态等。


lvm2
dracut
util-linux
NetworkManager
应用层
libs
devctl
设备节点
用户接口
devmaster
规则
事件监听
多级队列
worker池
规则处理
数据库
uevent
Embedded
netlink
inotify
sysfs
hotplug
内核

5. 服务管理

- devmaster 对网卡的命名
  - devmaster 网卡重命名功能由内置命令 net_id、net_setup_link 和网卡配置文件配合完成。
  - 在规则文件中，通过 net_id 获取网卡的硬件属性，再使用 net_setup_link 选择某个网卡属性作为新的网卡名。
    - net_setup_link命令基于网卡配置，针对特定网卡设备，控制网卡命名的风格。
  - devmaster 提供的默认网卡配置：
    - [Match]
    - OriginalName = "* “
    - [Link]
    - NamePolicy = ["onboard", "slot", "path"]
    - 网卡配置文件中包含 [Match] 匹配节和 [Link] 控制节，每节中包含若干配置项。
    - 匹配节的配置项用于匹配网卡设备，当网卡满足所有匹配条件时，将控制节中的所有配置项作用在网卡上，比如设置网卡名选取策略、调整网卡参数等等。
    - 默认网卡配置表示将该配置作用在所有网卡设备上，并依次检查 onboard、slot和 path风格的网卡命名风格，如果找到一个可用的风格，就以该风格对网卡进行命名。

Systemd

Systemd 入门教程：命令篇

https://ruanyifeng.com/blog/2016/03/systemd-tutorial-commands.html

Systemd 入门教程：实战篇

https://www.ruanyifeng.com/blog/2016/03/systemd-tutorial-part-two.html

sysMaster

sysMaster用户指南（24.04 LTS）

https://docs.openeuler.org/zh/docs/24.03_LTS/docs/sysMaster/overview.html

sysMaster： 全新1号进程实现方案，秒级自愈，保障系统全天在线

https://www.openeuler.org/zh/blog/20230512-sysMaster/20230512-sysMaster.html

sysmaster：重写一号进程项目

https://forum.openeuler.org/t/topic/567

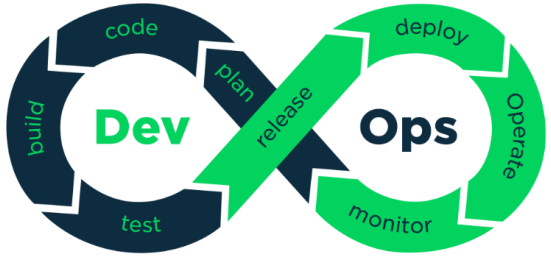
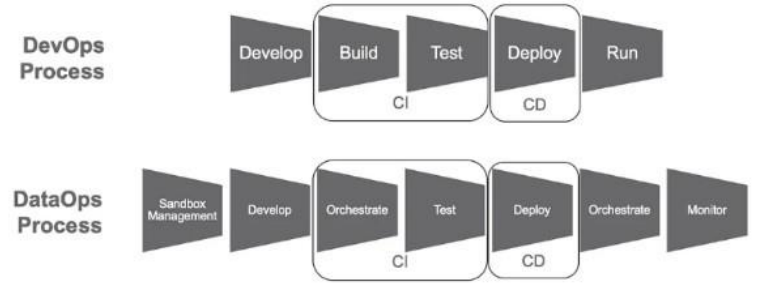
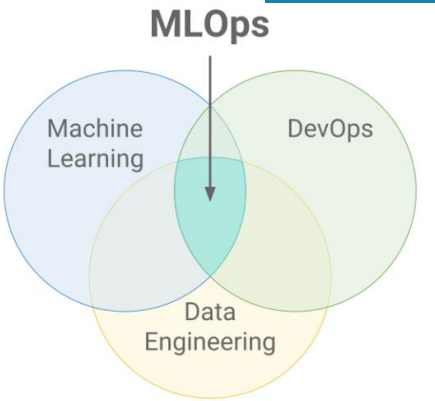


Image source: Devopedia

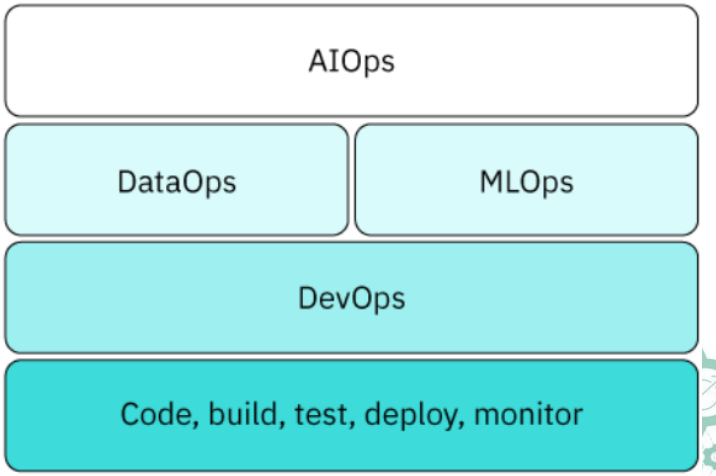


Comparing DevOps processes with DataOps processes. Image: Medium



DevOps vs. DataOps vs. MLOps Vs. AIOps: Comparison of All "Ops "

https://dzone.com/articles/comparison-of-all-ops



网络与信息系统智能运维课程体系学习平台

***

本课程体系由
河南中医药大学信息技术学院建设

课程体系学习平台由河南中医药大学医疗健康信息工程技术研究所开发与技术保障

网络与信息系统智能运维课程体系学习平台
https://internet.hactcm.edu.cn

互联网运维管理工程应用丛书
http://www.51xueweb.cn

扫码学习
并获取课程资源