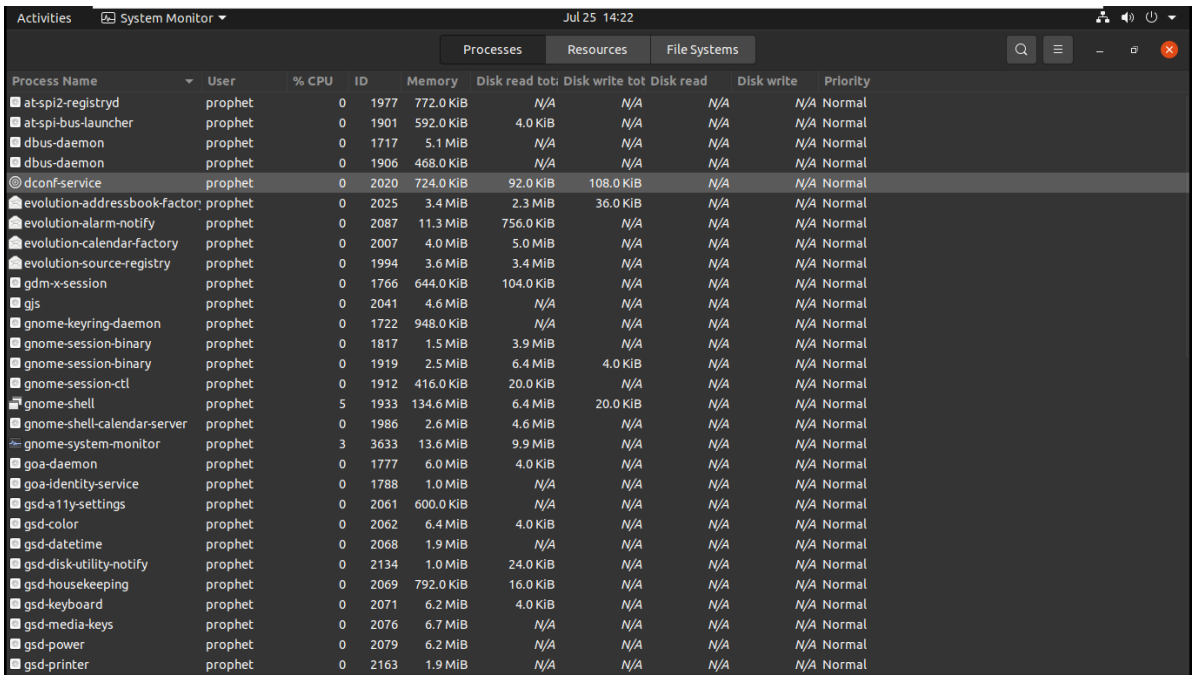


进程管理

Linux 任务管理器

任务管理器，管理电脑正在运行或暂时睡眠的软件（专业属于：正在运行的进程和后台等待随时被唤醒的服务）。

桌面右下角的 show all application 中找到 system monitor 打开



The screenshot shows the 'System Monitor' application window. It has tabs for 'Processes', 'Resources', and 'File Systems'. The 'Processes' tab is active, displaying a table of running processes. The table columns are: Process Name, User, % CPU, ID, Memory, Disk read tot, Disk write tot, Disk read, Disk write, and Priority. The processes listed include system daemons like at-spi2-registryd, dbus-daemon, and dconf-service, as well as user applications like evolution-addressbook-factory, gdm-x-session, and gnome-shell. The 'gnome-shell' process is highlighted.

Process Name	User	% CPU	ID	Memory	Disk read tot	Disk write tot	Disk read	Disk write	Priority
at-spi2-registryd	prophet	0	1977	772.0 KiB	N/A	N/A	N/A	N/A	Normal
at-spi-bus-launcher	prophet	0	1901	592.0 KiB	4.0 KiB	N/A	N/A	N/A	Normal
dbus-daemon	prophet	0	1717	5.1 MiB	N/A	N/A	N/A	N/A	Normal
dbus-daemon	prophet	0	1906	468.0 KiB	N/A	N/A	N/A	N/A	Normal
dconf-service	prophet	0	2020	724.0 KiB	92.0 KiB	108.0 KiB	N/A	N/A	Normal
evolution-addressbook-factory	prophet	0	2025	3.4 MiB	2.3 MiB	36.0 KiB	N/A	N/A	Normal
evolution-alarm-notify	prophet	0	2087	11.3 MiB	756.0 KiB	N/A	N/A	N/A	Normal
evolution-calendar-factory	prophet	0	2007	4.0 MiB	5.0 MiB	N/A	N/A	N/A	Normal
evolution-source-registry	prophet	0	1994	3.6 MiB	3.4 MiB	N/A	N/A	N/A	Normal
gdm-x-session	prophet	0	1766	644.0 KiB	104.0 KiB	N/A	N/A	N/A	Normal
gjs	prophet	0	2041	4.6 MiB	N/A	N/A	N/A	N/A	Normal
gnome-keyring-daemon	prophet	0	1722	948.0 KiB	N/A	N/A	N/A	N/A	Normal
gnome-session-binary	prophet	0	1817	1.5 MiB	3.9 MiB	N/A	N/A	N/A	Normal
gnome-session-binary	prophet	0	1919	2.5 MiB	6.4 MiB	4.0 KiB	N/A	N/A	Normal
gnome-session-ctl	prophet	0	1912	416.0 KiB	20.0 KiB	N/A	N/A	N/A	Normal
gnome-shell	prophet	5	1933	134.6 MiB	6.4 MiB	20.0 KiB	N/A	N/A	Normal
gnome-shell-calendar-server	prophet	0	1986	2.6 MiB	4.6 MiB	N/A	N/A	N/A	Normal
gnome-system-monitor	prophet	3	3633	13.6 MiB	9.9 MiB	N/A	N/A	N/A	Normal
goa-daemon	prophet	0	1777	6.0 MiB	4.0 KiB	N/A	N/A	N/A	Normal
goa-identity-service	prophet	0	1788	1.0 MiB	N/A	N/A	N/A	N/A	Normal
gsd-a11y-settings	prophet	0	2061	600.0 KiB	N/A	N/A	N/A	N/A	Normal
gsd-color	prophet	0	2062	6.4 MiB	4.0 KiB	N/A	N/A	N/A	Normal
gsd-datetime	prophet	0	2068	1.9 MiB	N/A	N/A	N/A	N/A	Normal
gsd-disk-utility-notify	prophet	0	2134	1.0 MiB	24.0 KiB	N/A	N/A	N/A	Normal
gsd-housekeeping	prophet	0	2069	792.0 KiB	16.0 KiB	N/A	N/A	N/A	Normal
gsd-keyboard	prophet	0	2071	6.2 MiB	4.0 KiB	N/A	N/A	N/A	Normal
gsd-media-keys	prophet	0	2076	6.7 MiB	N/A	N/A	N/A	N/A	Normal
gsd-power	prophet	0	2079	6.2 MiB	N/A	N/A	N/A	N/A	Normal
gsd-printer	prophet	0	2163	1.9 MiB	N/A	N/A	N/A	N/A	Normal

Linux 任务管理器命令操作（输入的命令也算进程）

top 显示或管理执行中的程序

top命令 可以实时动态地查看系统的整体运行情况，是一个综合了多方信息监测系统性能和运行信息的实用工具。通过top命令所提供的互动式界面，用热键可以管理。

选项

- b: 以批处理模式操作；
- c: 显示完整的命令；
- d: 屏幕刷新间隔时间；
- I: 忽略失效过程；
- s: 保密模式；
- S: 累积模式；
- i<时间>: 设置间隔时间；
- u<用户名>: 指定用户名；
- p<进程号>: 指定进程；
- n<次数>: 循环显示的次数。

ps (process status:进程状态) 报告当前系统的进程状态

ps 命令 用于报告当前系统的进程状态。可以搭配kill指令随时中断、删除不必要的程序。**ps** 命令是最基本同时也是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、进程有没有僵死、哪些进程占用了过多的资源等等，总之大部分信息都是可以通过执行该命令得到的。Linux 最复杂的命令之一。

常用命令

由于 **ps** 命令能够支持的系统类型相当的多，所以选项多的离谱！这里只展示最常用的命令。

```
ps axo pid,comm,pcpu # 查看进程的PID、名称以及CPU 占用率
ps -aux | grep named # 查看named进程详细信息
ps -l或l # 采用详细的格式来显示程序状况。
ps -A # 显示所有程序。
-f: #显示UID,PPID,C与STIME栏位
--forest 或 f #用ASCII字符显示树状结构，表达程序间的相互关系。
```

grep 命令：强大的文本搜索工具。全称：global search regular expression(RE) and print out the line：全面搜索正则表达式并把行打印出来。

在工作中，有关进程的多数要使用 **ps** 命令

kill

- 配合 **ps -aux | grep named**
- **kill pid** 杀死进程

pid,相同的进程每次运行的时候 **pid** 是不同的。

```
# 下面是常用的信号。
# 只有第9种信号(SIGKILL)才可以无条件终止进程，其他信号进程都有权利忽略。
kill - 数字
HUP    1    终端挂断
INT     2    中断（同 Ctrl + C）
QUIT    3    退出（同 Ctrl + \）
KILL     9    强制终止
TERM    15   终止
CONT    18   继续（与STOP相反，fg/bg命令）
STOP    19   暂停（同 Ctrl + Z）
```

储存设备操作

站在储存设备的角度操作文件

挂载

介绍

计算机插入一个新的硬盘或者其他设备，就是挂载。`Windows` 是以直接映射一个虚拟的盘符，通过盘这个盘符来访问设备。

`Linux` 没有 `Windows` 盘符感念那是如何完成挂载的呢：

- 在 `Linux` 根目录中：`/dev` 目录（设备分区），存放着连接计算机所有设备，的二级制文件，但那你操作不了，即使能操作你也看懂。。。
- 因此我们需要读取他们，并把它们映射到，**符合我们需要的操作权限**的目录中。（只读，只写，等这些权限）
- `Linux` 提供两个根目录来给我们挂载设备使用，当然也可以挂载到其他符合需求条件的目录。
 - `/media` 存放自动挂载的硬件(载点都是由系统自动建立和删除的)
 - `/mnt` 手动挂载目录（由个人手动建立或删除载点，并非系统建立或删除）

提醒

- 现在的个人版 `Linux` 是可以实现自动挂载的。
 - 在企业中，尤其是企业的服务器在挂载时，可能会使用手动挂载，注重安全性，防止恶意播放脚本 auto 病毒。
 - 手动挂载时，看好选择的目录权限，是否符合需求。
 - 当前位置在挂载目录中，无法卸载该挂载。
-

文件挂载使用的命令

`fdisk -l` 命令查看硬盘及分区信息

查到我们插入的设备分区所在的目录

`mount` 命令用于挂载Linux系统外的文件

语法

```
mount [-fnrsvw] [-t vfstype] [-o options] device(dev目录) dir
```

`[-t vfstype]` 指定文件系统的类型，通常不必指定，`mount`会自动选择正确的类型

- 光盘或光盘镜像iso9660
- DOS fat16文件系统msdos
- Windows 9x fat32文件系统vfat
- Windows NT ntfs文件系统ntfs
- Mount Windows文件网络共享smbfs
- UNIX(LINUX)文件网络共享nfs

`[-o option]` 主要用来描述设备或文件的挂载方式

device要挂载的设备

dir目录，设备在系统上的挂载点

选项

- V: 显示程序版本
- h: 显示辅助讯息
- v: 显示较讯息，通常和 -f 用来除错。
- a: 将 `/etc/fstab` 中定义的所有档案系统挂上。
- F: 这个命令通常和 -a 一起使用，它会为每一个 `mount` 的动作产生一个行程负责执行。在系统需要挂上大量 NFS 档案系统时可以加快挂上的动作。
- f: 通常用在除错的用途。它会使 `mount` 并不执行实际挂上的动作，而是模拟整个挂上的过程。通常会和 -v 一起使用。
- n: 一般而言，`mount` 在挂上后会在 `/etc/mtab` 中写入一笔资料。但在系统中没有可写入档案系统存在的情况下可以用这个选项取消这个动作。
- s-r: 等于 -o ro
- w: 等于 -o rw
- L: 将含有特定标签的硬盘分割挂上。
- U: 将档案分割序号为 的档案系统挂下。-L 和 -U 必须在`/proc/partition` 这种档案存在时才有意义。
- t: 指定档案系统的型态，通常不必指定。`mount` 会自动选择正确的型态。

-o option单独拿出来，因为光这一项就蛮多的

- o async: 打开非同步模式，所有的档案读写动作都会用非同步模式执行。
- o sync: 在同步模式下执行。
- o atime、-o noatime: 当 `atime` 打开时，系统会在每次读取档案时更新档案的『上一次调用时间』。当我们使用 `flash` 档案系统时可能会选项把这个选项关闭以减少写入的次数。
- o auto、-o noauto: 打开/关闭自动挂上模式。
- o defaults: 使用预设的选项 `rw, suid, dev, exec, auto, nouser, and async`。
- o dev、-o nodev-o exec、-o noexec允许执行档被执行。
- o suid、-o nosuid: 允许执行档在 `root` 权限下执行。
- o user、-o nouser: 使用者可以执行 `mount/umount` 的动作。
- o remount: 将一个已经挂下的档案系统重新用不同的方式挂上。例如原先是唯读的系统，现在用可读写的模式重新挂上。
- o ro: 用唯读模式挂上。
- o rw: 用可读写模式挂上。
- o loop=: 使用 `loop` 模式用来将一个档案当成硬盘分割挂上系统

挂载点有内容的文件夹，在挂载后内容消失，卸载后内容重现，也就是说挂载后会将原文件内容掩盖，但并不对其进行其他操作。

umount 用于卸载已经加载的文件系统

利用设备名或挂载点都能umount文件系统，不过最好还是通过挂载点卸载，以免使用绑定挂载（一个设备，多个挂载点）时产生混乱。

选项

- a: 卸除/etc/mtab中记录的所有文件系统；
- h: 显示帮助；
- 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；
- V: 显示版本信息。

使用挂载命令时遇到问题

1.以上这三种命令绝大多数下都需要管理员身份：

- 可以在命令前加 `sudo` 命令执行后输入管理员密码
- 使用 `sudo -s` 直接进入管理员终端。

2.操作移动硬盘挂载服务器时，执行完mount指令，直接报错：

分析：

“Mount is denied because the NTFS volume is already exclusively opened.The volume may be already mounted, or another software may use it which could be identified for example by the help of the 'fuser' command.”

翻译下：“拒绝挂载，因为NTFS卷已经独占地打开了。卷可能已经挂载，或其他软件可能使用它可以通过 `fuser` 命令的帮助来识别。”

解决方案：

```
fuser -a [device] # 根据目录查UID
```

#fuser 使用文件或文件结构识别进程

- a: 显示命令行中指定的所有文件的UID；
- k: 杀死访问指定文件的所有进程；
- i: 杀死进程前需要用户进行确认；
- l: 列出所有已知信号名；
- m: 指定一个被加载的文件系统或一个被加载的块设备；
- n: 选择不同的名称空间；
- u: 在每个进程后显示所属的用户名。

查询挂载设备全部信息

```
kill PID
```

关掉进程

再次查询设备信息，确定是否真的关掉进程。

重新执行挂载命令。

拓展 Android 设备挂载

Android 设备的链接协议为 MTP 全称：Media Transfer Protocol（媒体传输协议）

产生该协议的需求：

- 传统硬盘的挂载方式（UMS），PC对设备有百分之百的控制权。
- Android设备本身也是类 Linux 系统，需要保证自己文件的安全性。
- 以及Android设备会安装SD卡（Android，安装SDcard使用UMS协议），也为保障SD的控制权，和减少SD卡重新挂载回Android。

MTP不能直接修改文件本身。只能先拷贝到本地修改，完毕后再拷贝回去。

- 正因如此直接往SD卡传输数据手机本身要有一定4G剩余空间。安卓11以后该问题得到解决。
- 这也就是为什么安卓手机单个文件大小不能超过4G。安卓11以后该问题得到解决。
- 这也是为什么PC操作Android设备传输文件，传输删除慢，但是传输取消时快。
- 然而PC操作移动储存设备时传输删除虽快，但是中途想取消时慢。
- 但如今 Type-C 接口的普及，加大拷贝的速度，传输速度还是提升的。

结合 MTP 协议的性质也就是为什么Android的挂载目录在 `/run/user/1000/gvfs` 中。

最直接的回答就是PC获得Android的设备信息必须通过 MTP 协议询问Android设备：

- 因此Android设备的挂载信息是动态的。
- `/run` 目录管理正在运行的进程，是存在内存中的，正好符合条件。
- `./user/.` 文件夹，当前登录用户空间。
- `./1000/.` 当前正在运行的进程之一。
- `./gvfs/.` 虚拟文件系统文件夹，用来放置 Android 设备的挂载。

综上安卓设备不能手动挂载

df和du命令

df

用于显示磁盘分区上的可使用的磁盘空间。默认显示单位为KB。可以利用该命令来获取硬盘被占用了多少空间，目前还剩下多少空间等信息。

常用命令

```
df -h #以可读性较高的方式来显示信息
```

du

显示每个文件和目录的磁盘使用空间

常用命令

```
du -h #以可读性较高的方式来显示信息
du -s, --summarize #仅显示总计，只列出最后加总的值。
```

sort 文本文件中所有行进行排序

- 将所有输入文件的内容排序后并输出(不改变源文件)。
- 默认对每一行第一个字符排序，并展示出来，并不会改变原来的文件。

sort 命令有排序选项，和其他功能选项，详细看文档。

```
-n, --numeric-sort #根据数字排序
-M, --month-sort #按照非月份、一月、十二月的顺序排序(常用于日志 log 文件)
-r, --reverse #将结果倒序排列
```

tar 打包 归档、gzip压缩、gunzip解压缩

- Linux 系统自带的压缩软件 gzip 后缀为 .gz。
- 首先要弄清两个概念：打包和压缩。打包是指将一大堆文件或目录变成一个总的文件；压缩则是将一个大的文件通过一些压缩算法变成一个小文件。
- Linux中很多压缩程序只能针对一个文件进行压缩，这样当你想要压缩一大堆文件时，你得先将这一大堆文件先打成一个包（tar命令），然后再用压缩程序进行压缩（gzip bzip2命令）。
- 当然 tar 的选项也可以自己完成打包压缩，解压缩拆包。

tar

语法

```
tar [选项...] [FILE]...
```

常用选项

```
-A, --catenate, --concatenate  #追加 tar 文件至归档
-c, --create                    #创建一个新归档
-d, --diff, --compare          #找出归档和文件系统的差异
--delete                        #从归档(非磁带!)中删除
-r, --append                    #追加文件至归档结尾
-t, --list                      #列出归档内容
--test-label                    #测试归档卷标并退出
-u, --update                    #仅追加比归档中副本更新的文件
-x, --extract, --get            #从归档中解出文件
-f, --file=ARCHIVE             #使用归档文件或 ARCHIVE 设备(不可少, 少了会报错)
-v, --verbose                   #详细地列出处理的文件
```

#压缩格式选项:

```
-a, --auto-compress            #使用归档后缀名来决定压缩程序
-z, --gzip, --gunzip, --ungzip  #通过 gzip 过滤归档, 后缀 .gz
-j, --bzip2                    #通过 bzip2 过滤归档, 后缀 .bz2
```

常用命令

```
tar -cvf 1.tar 1.txt #仅打包, 不压缩
tar -zcvf 1.tar.gz 1.txt #打包后, 以 gzip 格式压缩
tar -zxvf /opt/soft/test/1.tar.gz #将tar 解压缩并解包
```