虚拟机的含义、功能

虚拟机（Virtual Machine，简称VM）是一种在现有的硬件系统之上，通过软件模拟出来的完整的、可独立运行的计算机系统。它可以执行程序，如一个物理计算机一样。每个虚拟机都有自己的CPU、内存、硬盘、网络接口等资源，以及操作系统，甚至还可以有自己的BIOS。

虚拟机的主要功能和应用如下：

1. **隔离和安全性**：每个虚拟机都是独立的，它们之间互不影响。如果一个虚拟机发生故障，不会影响其他虚拟机的运行。这对于测试新的软件或操作系统很有用，如果新软件或操作系统发生问题，不会影响主机的正常运行。
2. **资源利用率**：使用虚拟机可以提高物理硬件的使用率。例如，一个物理服务器可能由于资源过剩而浪费，但如果在上面运行多个虚拟机，可以使每个虚拟机利用部分资源，从而提高总体使用率。
3. **灵活性和便捷性**：虚拟机可以轻松地创建、删除和移动，可以根据需要快速调整资源配置。此外，虚拟机也支持快照和回滚功能，这使得管理和恢复变得更加方便。
4. **硬件和操作系统的兼容性**：虚拟机可以运行在各种硬件和操作系统上，这使得一些老的、不兼容的软件可以在新的硬件和操作系统上运行。
5. **负载均衡和灾备**：虚拟机可以实现负载均衡和灾备。如果一个物理服务器发生故障，可以快速将上面的虚拟机迁移到另一个物理服务器上，从而实现高可用。

Linux系统产生、创始人（之父）、特点、版本、功能及作用；重点掌握Linux系统的特点、系统的组成。

Linux系统产生、创始人（之父）

Linux操作系统是由芬兰赫尔辛基大学的学生**林纳斯·托瓦兹（Linus Torvalds）**在1991年创造的。最初，林纳斯只是为了提高他的个人计算机的性能而开发了这个操作系统的内核。他在新闻组上发布了一份帖子，分享了他的项目，这个项目后来演变成了我们今天所知道的Linux。

Linux的特点

1. **开源性**：Linux是一种开源操作系统，这意味着它的源代码可以被任何人自由查看、修改和分发。这使得Linux得到了全世界开发者的广泛贡献和支持。
2. **可靠性和稳定性**：Linux系统非常稳定，很少崩溃。它可以连续运行数年而不需要重启。这使得Linux成为服务器和大型系统的理想选择。
3. **多用户、多任务**：Linux系统支持多用户和多任务。多个用户可以同时登录并运行程序，一个用户也可以同时运行多个程序。
4. **良好的安全性**：Linux系统具有很强的安全性。它有许多安全机制，如权限管理、防火墙和强大的文件系统。
5. **可移植性**：Linux可以在各种硬件平台上运行，包括个人电脑、服务器、超级计算机、嵌入式设备等。

Linux的版本

Linux有许多不同的版本，被称为“发行版”。每个发行版都有自己的特点和目标用户。其中最知名的发行版包括Ubuntu、Debian、Fedora、CentOS、Arch Linux、Red Hat Enterprise Linux等。

Linux的功能及作用

1. **操作系统**：Linux是一种全功能的操作系统。它可以提供所有常见的操作系统服务，如文件管理、进程管理、内存管理、设备管理等。
2. **网络服务器**：由于其稳定性和可靠性，Linux被广泛用于运行网络服务器，如Web服务器、数据库服务器、邮件服务器等。
3. **开发环境**：Linux提供了一种优秀的程序开发环境。它支持各种编程语言，如C、C++、Python、Java等，以及各种开发工具，如gcc、make、gdb等。
4. **嵌入式系统**：由于其可移植性和开源性，Linux也被广泛用于嵌入式系统，如智能家居、无人驾驶、工业自动化等。

Linux系统的组成

1. **Linux内核**：这是Linux系统的核心，负责管理系统的硬件资源，如CPU、内存、硬盘等，以及提供各种系统服务，如进程管理、文件系统、网络协议等。
2. **Shell**：这是用户和Linux内核进行交互的界面。用户可以输入命令来操作系统，也可以编写脚本来自动执行一系列的命令。
3. **文件系统**：这是Linux系统存储和组织文件的方式。Linux支持各种文件系统，如ext4、xfs、btrfs等。
4. **系统程序和工具**：Linux系统包含了大量的系统程序和工具，用于完成各种系统管理和维护任务，如系统安装、包管理、设备驱动、网络配置等。
5. **应用软件**：Linux系统支持各种应用软件，如文档处理、图像处理、音频视频、游戏等。这些软件可以从各种软件仓库中获取。

Ubuntu Linux默认的图形界面、默认的应用及功能。

Ubuntu的默认图形界面

自Ubuntu 17.10版本以来，Ubuntu的默认图形用户界面（GUI）是GNOME。GNOME是一个开源的桌面环境，提供了图形用户界面，用户可以使用鼠标和键盘进行操作。GNOME桌面环境包括一系列基本的桌面应用程序和工具，如文件管理器、Web浏览器、终端模拟器、文本编辑器等。

Ubuntu的默认应用及功能

1. **文件管理器**（Nautilus）：用于浏览和管理文件和目录。
2. **Web浏览器**（Firefox）：用于上网浏览网页。
3. **办公套件**（LibreOffice）：包括文字处理（Writer）、电子表格（Calc）、演示文稿（Impress）等应用程序，与Microsoft Office有类似的功能。
4. **邮件客户端**（Thunderbird）：用于管理电子邮件。
5. **终端模拟器**（GNOME Terminal）：用于运行命令行界面。
6. **音视频播放器**（Rhythmbox and Totem）：用于播放音乐和视频。
7. **图片查看器**（Eye of GNOME）：用于查看和简单编辑图片。
8. **软件中心**：用于安装和卸载软件。

shell字符界面：SSH端口（22）；普通用户和超级用户默认的提示符（$、#），PS1、PS2变量设置方法及作用，命令补齐键（Tab）；多个命令分隔符（；）等

SSH端口（22）

SSH，全称Secure Shell，是一种网络协议，用于加密方式远程登录计算机系统。默认情况下，SSH服务使用的网络端口是22，但可以在配置文件中修改。

普通用户和超级用户默认的提示符（$、#）

在shell界面中，不同的用户类型有不同的提示符。普通用户的提示符通常是一个美元符号（$），而超级用户（通常被称为”root”用户）的提示符是一个井号（#）。这些提示符是为了帮助用户快速识别他们当前的用户类型。

PS1、PS2变量设置方法及作用

PS1和PS2是bash shell的环境变量，用于设置命令行提示符。

- PS1：这是主提示符，它定义了命令行提示符的默认外观。你可以在.bashrc文件中设置PS1变量，以改变你的命令行提示符。例如，你可以将PS1设置为”[\u@\h \W]$”，这将使你的提示符显示为”[用户名@主机名 当前工作目录]$”。
- PS2：这是次要提示符，当一个命令跨越多行时显示。默认情况下，PS2变量设置为”>”。

命令补齐键（Tab）

Tab键在shell中用于自动补全命令、文件名或目录名。例如，如果你键入”cd /usr/lo”，然后按Tab键，shell将自动完成路径，变为”cd /usr/local”（如果这个路径存在的话）。这个功能可以大大提高命令行操作的效率。

多个命令分隔符（；）

在shell中，你可以使用分号（;）来在一行中输入多个命令，这些命令将依次执行。例如，”cd /usr/local; ls”将会先将当前目录改为”/usr/local”，然后列出这个目录下的文件和目录。

常用命令date、who、alias，history的功能、使用场景和使用方法

date

`date`命令用于显示和设置系统的日期和时间。

- 功能：显示或设置系统的日期和时间。
- 使用场景：当你需要查看当前日期和时间，或者需要修改系统的日期和时间时。
- 使用方法：
  - `date`：显示当前日期和时间。
  - `date -s "2023-06-17 10:30:00"`：设置系统的日期和时间。

who

`who`命令用于显示当前登录系统的用户信息。

- 功能：显示当前登录的用户信息。
- 使用场景：当你需要查看当前系统中谁在在线，或者需要查看用户登录的详细信息时。

- 使用方法：
  - `who`：显示当前登录的用户列表。
  - `who -a`：显示所有的登录信息。

alias

`alias`命令用于为命令创建别名。

- 功能：为命令创建别名，使得复杂的命令行可以简化输入。
- 使用场景：当你需要经常输入一些长的或者复杂的命令时，可以使用`alias`命令为这些命令创建简单的别名。
- 使用方法：
  - `alias ll='ls -l'`：为`ls -l`命令创建别名`ll`。以后只需要输入`ll`就可以执行`ls -l`命令。
  - `unalias ll`：删除别名`ll`。

history

`history`命令用于显示命令历史记录。

- 功能：显示命令历史记录，可以查看你之前执行过的命令。
- 使用场景：当你需要查看或者重复之前执行过的命令时。
- 使用方法：
  - `history`：显示所有的命令历史记录。
  - `history 10`：显示最近10条命令历史记录。
  - `!10`：执行第10条命令历史记录。
  - `!!`：执行上一条命令。

软件更新、安装的命令（apt-get、dpkg）的使用场景、区别和用法，了解与yum、rpm的区别。

apt-get

apt（Advanced Package Tool）是一种命令行工具，用于处理软件包。它可以自动处理依赖关系，例如自动下载和安装一个软件包的所有依赖。

- 使用场景：当你需要安装、升级或删除软件包时，特别是当这些软件包有很多依赖关系时。
- 使用方法：
  - `sudo apt-get update`：更新软件包列表，这是安装新软件之前的必要步骤。
  - `sudo apt-get upgrade`：升级所有已安装的软件包。
  - `sudo apt-get install <package-name>`：安装一个软件包。
  - `sudo apt-get remove <package-name>`：删除一个软件包。

dpkg

dpkg（Debian Package）是一个更低级别的包管理工具。它可以安装、卸载和提供有关.deb软件包的信息，但不能自动处理依赖关系。

- 使用场景：当你有一个.deb软件包文件，需要安装或者查询这个软件包的信息时。
- 使用方法：
  - `sudo dpkg -i <package-file.deb>`：安装一个.deb软件包。
  - `sudo dpkg -r <package-name>`：删除一个已安装的软件包。
  - `dpkg -l`：列出所有已安装的软件包。
  - `dpkg -s <package-name>`：查看一个已安装的软件包的信息。

apt-get/dpkg与yum/rpm的区别

yum和rpm是基于Red Hat的Linux发行版（如CentOS，Fedora）中的包管理工具，与apt-get和dpkg类似，但处理的是.rpm软件包。

- yum是类似于apt-get的工具，可以自动处理依赖关系。
- rpm是类似于dpkg的工具，是一个更低级别的包管理工具。

在处理软件包的方式上，apt/dpkg和yum/rpm的主要区别在于它们处理的软件包类型（.deb与.rpm）以及它们在处理软件依赖关系时的具体策略。

字符界面下关机和重启的命令shutdown -h/-r/-c，halt，reboot，poweroff和shutdown的区别。

shutdown

`shutdown`命令是最常用也是最安全的用于关闭或重新启动系统的命令，它可以提供给用户一段时间来保存工作。

- `shutdown -h now`：立即关闭系统。
- `shutdown -r now`：立即重新启动系统。
- `shutdown -c`：取消一个已经安排的关机或重启操作。

halt

`halt`命令通知硬件停止所有的CPU功能，但不断电。在较老的系统中，`halt`不会调用`shutdown`，可能导致文件系统损坏。然而，在较新的系统中，`halt`通常等同于`shutdown -h`。

reboot

`reboot`命令用于重新启动系统，等同于`shutdown -r`。

poweroff

`poweroff`命令通知系统关闭所有的CPU和进入停电状态，等同于`shutdown -h`。

总的来说，`shutdown`命令是最通用的命令，它可以用于安全地关闭或重启系统。其他的命令（`halt`，`reboot`，`poweroff`）在功能上是子集，可以在特定的情况下使用，但最好的实践通常是使用`shutdown`命令。

文件系统的概念、重要性（一切皆文件）。Linux文件系统建立和使用的步骤（分区fdisk、格式mkfs、挂载mount），能够对一个新盘创建文件系统。自动加载文件（/etc/fstab）的作用及配置方法。了解umount命令的使用方法。

文件系统的概念与重要性

在操作系统中，文件系统是一种组织和存储数据的方法。它决定了数据如何在磁盘上存储，以及如何在磁盘上检索数据。Linux中的文件系统遵循”一切皆文件”的原则，意味着无论是硬件设备，还是网络连接，都被视为文件。

Linux文件系统的建立和使用的步骤

1. **分区**：使用`fdisk`命令来对硬盘进行分区。例如，`fdisk /dev/sdb`将启动一个交互式菜单，用于创建、删除或修改硬盘的分区。
2. **格式化**：使用`mkfs`命令来格式化分区，这将在分区上创建一个新的文件系统。例如，`mkfs.ext4 /dev/sdb1`将在sdb1分区上创建一个新的ext4文件系统。
3. **挂载**：使用`mount`命令将新的文件系统挂载到Linux文件系统的目录树中。例如，`mount /dev/sdb1 /mnt/data`将将sdb1分区挂载到/mnt/data目录下。

自动加载文件（/etc/fstab）的作用及配置方法

`/etc/fstab`是一个重要的配置文件，用于定义如何挂载文件系统。系统启动时，`mount -a`命令将读取这个文件，然后自动挂载文件系统。

每行定义一个文件系统，字段由制表符或空格分隔：

```
<file system> <mount point>   <type>  <options>       <dump>  <pass>
```

例如：

```
/dev/sdb1       /mnt/data       ext4    defaults        0       2
```

这一行定义了如何挂载/dev/sdb1。它将被挂载到/mnt/data目录下，文件系统类型是ext4，挂载选项是默认的，dump备份被禁用（0），并且在启动时进行fsck文件系统检查的优先级是2。

umount命令的使用方法

`umount`命令用于卸载已经挂载的文件系统。例如，`umount /mnt/data`将卸载/mnt/data目录下的文件系统。在卸载文件系统之前，你需要确保没有进程在使用该文件系统，否则你将收到一个”device is busy”的错误信息。

Linux常见的文件系统格式、特点；当前主流的、默认的文件系统格式。

1. **ext2/ext3/ext4**：这是Linux中最常用的文件系统类型。ext2是没有日志功能的文件系统，ext3添加了日志功能，而ext4则进一步提高了存储限制和性能。
    - ext2：适用于小型系统或嵌入式系统，不需要日志功能。
    - ext3：在ext2的基础上添加了日志功能，提高了数据安全性。
    - ext4：是目前Linux默认的文件系统，支持大容量硬盘，提高了性能和数据安全性。

2. XFS：这是一种高性能的日志文件系统，特别适合大文件和大数据量的处理。
    - 适用于大文件存储和高性能需求，比如大型数据库应用。
3. Btrfs（B-tree file system）：这是一个复制和快照功能丰富的文件系统，目标是提供容错，修复和易管理的功能。
    - 适用于需要高度数据完整性和保护的场景。
4. ReiserFS：这是一种带有日志功能的文件系统，优点是在处理大量小文件时效率高。
    - 适用于邮件服务器和新闻服务器等需要处理大量小文件的场景。
5. FAT32/NTFS：这些是Windows系统中常用的文件系统，Linux也支持这些文件系统。
    - 适用于Linux和Windows系统间的数据共享。

在2023年，Linux中默认的文件系统通常是ext4，但Btrfs和XFS等新型文件系统在一些特定的场景中也越来越流行。

Linux下文件与目录的结构，理解常见的/etc、/dev、/home、/lib等等目录的功能

Linux文件系统的结构是层级式的，开始于根目录，也就是 ”/”，然后扩展为更多的目录和子目录。这些目录有各自的特定用途，以存储，组织和管理系统和用户数据。以下是一些你提到的目录的功能：

1. `/etc`：这个目录包含了所有的系统管理所需要的配置文件和子目录。这些配置文件用于更改系统的设置，或者允许系统管理者修改应用程序的配置。
2. `/dev`：这个目录包含设备文件。Linux把硬件设备当作特殊的文件，你可以像操作普通文件一样来操作它们。这些设备文件包括硬盘、终端、键盘等等。
3. `/home`：在这个目录下，每个用户都会有一个自己的目录，通常是以用户的用户名命名的。这个目录用于存储用户的数据文件和个人设置。
4. `/lib`：这个目录包含了系统最基本的动态链接共享库，其作用类似于Windows里的DLL文件。几乎所有的应用程序都需要用到这些共享库。

除此之外，还有其他一些重要的目录，例如：

- `/bin` 和 `/usr/bin`：这两个目录存放了大多数用户二进制文件，也就是可执行文件或者程序。
- `/sbin` 和 `/usr/sbin`：这两个目录存放的是供超级用户root使用的系统管理程序。
- `/var`：这个目录包含了在常规操作中经常被修改的文件。这包括各种日志文件、邮件和打印任务队列等等。
- `/tmp`：这个目录用于存放临时文件，这些文件在关闭系统时会被删除。
- `/usr`：这是一个非常重要的目录，用户的很多应用程序和文件都存储在这里，类似于windows下的program files目录。
- `/boot`：这个目录存储了启动Linux时使用的文件，包括Linux核心文件以及引导加载器。

重点掌握Linux文件的类型（–/d/l/b/等）、属性（r、w、x）；10位文件权限的含义和内容；重点掌握ls –l查询出的文件属性的每一个字段的含义。重点掌握权限对操作用户的影响，要能判断用户是否有操作用户的权限。

- `-`: 普通文件
- `d`: 目录
- `l`: 符号链接
- `b`: 块设备
- `c`: 字符设备
- `s`: 套接字
- `p`: 管道文件

对于文件的属性（r、w、x），这些是权限标识，代表着不同的权限：

- `r`: 可读
- `w`: 可写
- `x`: 可执行

当你运行`ls -l`命令，你会看到像这样的输出：

```
-rw-r--r-- 1 root root 0 Jun 18 09:00 example.txt
```

这个输出中的每个字段都有特殊的含义：

- 第一部分`-rw-r--r--`表示文件类型和权限。第一个字符是文件类型（上述已经解释过），接下来的九个字符分为三组，每组三个，分别代表文件所有者的权限，文件所在组的权限和其他用户的权限。
- 第二部分`1`表示文件的链接数。
- 第三部分`root`表示文件所有者的用户名。
- 第四部分`root`表示文件所在的组。
- 第五部分`0`表示文件大小（单位是字节）。
- 第六、七、八部分`Jun 18 09:00`表示文件的最后修改时间。
- 第九部分`example.txt`表示文件名。

对于文件权限的理解和使用，最主要的就是能够判断特定用户是否有权执行特定操作。例如，如果用户是文件的所有者，并且文件的所有者有写权限（即第一组权限中有`w`），那么这个用户就可以修改这个文件。如果用户不是文件的所有者，但是他属于文件所在的组，并且文件的组有读权限（即第二组权限中有`r`），那么这个用户就可以读取这个文件。如果其他用户想要执行文件（即第三组权限中有`x`），那么他们可以执行这个文件。

权限可以通过`chmod`命令修改，例如，`chmod u+x example.txt`会给文件所有者添加执行权限。`chmod`命令也可以使用数字来表示权限，例如`chmod 755 example.txt`，其中7代表所有者的权限（读、写、执行），第一个5代表所在组的权限（读、执行），最后一个5代表其他用户的权限（读、执行）。

重点掌握文件及文件夹相关的操作命令：cat（包括</<</>/>>的区别）、more(less)、head（tail）、echo（双引号、单引号的区别）、ls、pwd、cd（.，..，~）、touch、rm(-r)、mkdir(rmdir)、cp(-rf)、mv、tar(cvzfx)，包括命令使用场景、常用选项等

1. `cat`：用于显示文件内容。`>`表示重定向，比如`cat file1 > file2`会把file1的内容写入file2，如果file2已经存在，那么它的内容将被覆盖。`>>`表示追加，比如`cat file1 >> file2`会把file1的内容追加到file2的后面。`<`表示从文件读入，`<<`表示here document。
2. `more` 和 `less`：用于分页显示文件内容，`more`从上到下显示，`less`可以上下翻页。
3. `head` 和 `tail`：`head`显示文件的前几行，`tail`显示文件的后几行。比如`head -n 5 file`会显示file的前5行，`tail -n 5 file`会显示file的后5行。
4. `echo`：用于输出文本或变量的值。如果使用双引号，那么其中的变量会被替换为它的值，如果使用单引号，那么其中的内容会被原样输出。
5. `ls`：列出目录的内容，`-l`选项可以显示详细信息，`-a`选项可以显示隐藏文件。
6. `pwd`：显示当前工作目录的路径。
7. `cd`：改变当前工作目录，`.`表示当前目录，`..`表示父目录，`~`表示用户的家目录。
8. `touch`：用于创建新的空文件，或者更新已有文件的访问和修改时间。
9. `rm`：用于删除文件或目录，`-r`选项表示递归删除，用于删除目录及其内容。
10. `mkdir` 和 `rmdir`：`mkdir`用于创建新的目录，`rmdir`用于删除空目录。
11. `cp`：用于复制文件或目录，`-r`选项表示递归复制，用于复制目录及其内容，`-f`选项表示强制复制，即如果目标文件已经存在，那么它会被覆盖。
12. `mv`：用于移动或重命名文件或目录。
13. `tar`：用于打包和解压缩文件，`c`表示创建新的归档文件，`v`表示详细模式，`z`表示gzip压缩，`f`表示指定归档文件，`x`表示解压缩。比如`tar cvzf file.tar.gz dir`会将dir目录打包并压缩为file.tar.gz，`tar xvfz file.tar.gz`会将file.tar.gz解压缩。

掌握权限管理命令：chgrp、chown、chmod（符号模式、绝对模式），包括命令使用场景、常用选项（-R）等。

1. `chgrp`：这个命令用于改变文件或者目录的组，比如`chgrp groupname filename`会将文件filename的组改为groupname。`-R`选项表示递归操作，即改变目录及其内容的组。
2. `chown`：这个命令用于改变文件或者目录的所有者，比如`chown username filename`会将文件filename的所有者改为username。你也可以同时改变所有者和组，比如`chown username:groupname filename`。`-R`选项同样表示递归操作。
3. `chmod`：这个命令用于改变文件或者目录的权限。有两种模式来指定权限：
    - 符号模式：使用u（用户）、g（组）、o（其他用户）、a（所有人）来代表用户，使用+（添加权限）、-（删除权限）、=（设置权限）来修改权限，使用r（读权限）、w（写权限）、x（执行权限）来指定权限。比如`chmod u+x filename`会给文件filename的所有者添加执行权限，`chmod g-w filename`会删除文件filename所在组的写权限，`chmod o=r filename`会设置其他用户的权限为只读。

- 绝对模式：使用三位数字来设置权限，每位数字代表一个用户组的权限，数字是0-7，分别代表不同的权限组合，数字是r、w、x权限值的和，r=4，w=2，x=1。比如`chmod 755 filename`会设置文件filename的所有者权限为rwx（读、写、执行，对应7），所在组的权限为r-x（读、执行，对应5），其他用户的权限也为r-x。
- `-R`选项表示递归操作，即改变目录及其内容的权限。

掌握find、grep命令的作用、使用场景、常用参数、会使用简单的正则表达式和通配符。

1. `find`：这个命令用于在文件系统中查找文件。它允许你按照各种条件来查找，如按照名称、大小、修改时间等。以下是一些常见的使用例子：
   - `find /home -name myfile.txt`：在/home目录及其子目录中查找名为myfile.txt的文件。
   - `find /home -type d -name mydir`：在/home目录及其子目录中查找名为mydir的目录。
   - `find /home -size +100M`：在/home目录及其子目录中查找大于100M的文件。
2. `grep`：这个命令用于在文本中查找匹配的字符串。你可以使用正则表达式来匹配复杂的模式。以下是一些常见的使用例子：
   - `grep 'my pattern' myfile.txt`：在myfile.txt文件中查找包含'my pattern'的行。
   - `grep -r 'my pattern' /home`：在/home目录及其子目录中的文件中查找包含'my pattern'的行。
   - `grep -i 'my pattern' myfile.txt`：在myfile.txt文件中查找包含'my pattern'的行，忽略大小写。
   - `grep -v 'my pattern' myfile.txt`：在myfile.txt文件中查找不包含'my pattern'的行。

对于正则表达式和通配符，以下是一些常见的例子：

- `*`：匹配任意多个任意字符，比如`*.txt`会匹配所有以.txt结尾的文件。
- `?`：匹配任意一个字符，比如`?.txt`会匹配a.txt但不会匹配ab.txt。
- `[abc]`：匹配方括号中的任意一个字符，比如`[abc].txt`会匹配a.txt，b.txt和c.txt。
- `{a,b,c}`：匹配大括号中的任意一个字符串，比如`file{1,2,3}.txt`会匹配file1.txt，file2.txt和file3.txt。
- `.`：在正则表达式中，匹配任意一个字符，比如`a.c`会匹配abc，acc等。
- `*`：在正则表达式中，表示前一个字符可以出现任意多次，比如`a*`会匹配a，aa，aaa等。
- `?`：在正则表达式中，表示前一个字符可以出现0次或1次，比如`a?`会匹配a和""。

掌握标准输入、输出概念，输入重定向（<,<<）、输出重定向（>,>>）的作用和区别。

在Linux中，每一个运行的进程通常都会有三个标准的I/O通道：标准输入（stdin）、标准输出（stdout）、标准错误（stderr）。

1. 标准输入（stdin）：默认是键盘作为输入设备。
2. 标准输出（stdout）：默认是屏幕作为输出设备，用于显示进程的输出。
3. 标准错误（stderr）：默认也是屏幕，用于显示进程的错误输出。

重定向就是改变这些默认设备，把输出发送到其他地方，或者从其他地方获取输入。

1. **输入重定向：使用`<`和`<<`来重定向输入。**
    - `<`：将文件的内容作为命令的输入。例如，`sort < file.txt`，`sort`命令会从`file.txt`文件中读取输入，而不是从键盘。
    - `<<`：这是一个特殊的重定向，称为Here Document。它允许你指定一个文本块作为输入。例如：

      ```
      cat << EOF
      Hello
      World
      EOF
      ```

      上述命令会把两行文本”Hello”和”World”作为`cat`命令的输入。
2. **输出重定向：使用`>`和`>>`来重定向输出。**
    - `>`：将命令的输出写入到文件中，如果文件已经存在，它会被覆盖。例如，`ls > file.txt`，`ls`命令的输出会被写入到`file.txt`文件中。
    - `>>`：将命令的输出追加到文件中，如果文件已经存在，新的输出会被添加到文件的末尾。例如，`ls >> file.txt`，`ls`命令的输出会被追加到`file.txt`文件中。

掌握管道（|）的概念、作用以及使用方法，以及与定向符的异同点。

管道（pipe）是一种非常强大的Linux命令行特性，允许你将多个命令串联起来，使一个命令的输出成为另一个命令的输入。管道的符号是竖线`|`。

例如，你可能已经看到过类似`ls | grep txt`的命令，这就是使用了管道。在这个例子中，`ls`命令的输出被作为`grep txt`命令的输入。所以，这个命令序列的作用就是列出所有包含”txt”的文件名。

管道和重定向的主要区别在于它们操作的对象不同：

- 重定向主要用于将命令的输出发送到文件（使用`>`或`>>`），或者从文件获取输入（使用`<`或`<<`）。比如，`ls > files.txt`将ls命令的输出写入到files.txt文件中。
- 管道则是将一个命令的输出直接作为另一个命令的输入。比如，`ls | grep txt`将ls命令的输出直接作为grep命令的输入。

在实际使用中，管道和重定向通常会结合起来使用，以实现更复杂的任务。比如，`ls | grep txt > files.txt`将列出所有包含”txt”的文件名，并将结果写入到files.txt文件中。

链接的两种方式及区别。

1. **硬链接（Hard Link）**：硬链接是一个文件的别名，和原始文件共享相同的i节点和存储块，也就是说，硬链接与其源文件共享相同的数据和属性。硬链接在文件系统中具有以下特点：
    - 删除源文件不会影响硬链接。
    - 修改源文件或硬链接，会影响彼此的内容，因为他们共享相同的数据。
    - 硬链接不能跨文件系统（即不能在不同的磁盘分区之间创建硬链接）。
    - 硬链接不能链接目录，只能链接文件。
2. **软链接（Symbolic Link）**：软链接相当于Windows系统中的快捷方式，它是一个单独的文件，只是这个文件包含了对另一个文件或目录的引用。软链接在文件系统中具有以下特点：
    - 删除源文件会导致软链接失效，因为它只是对源文件的引用。

- 修改源文件会影响软链接，因为软链接指向源文件，但修改软链接不会影响源文件。
- 软链接可以跨文件系统。
- 软链接可以链接文件或目录。

用户的概念。重点掌握分类及特点。了解用户组的。

在Linux系统中，用户是对使用系统资源的个体或应用程序的抽象。每个用户都会有一个唯一的用户名和用户ID，以及一组其他属性，如用户的全名、用户的主目录、用户使用的shell等。

Linux用户可以大致分为以下三类：

1. **超级用户（Root）**：在Linux系统中，超级用户（通常被称为root）具有所有的权限，可以访问系统的任何资源。Root用户可以执行任何命令，修改任何文件，创建和删除任何用户。由于root用户的权限非常大，所以通常情况下应该避免使用root用户，除非绝对必要。
2. **系统用户**：系统用户主要用于运行系统服务或守护进程。这些用户通常不允许登录，也没有主目录。比如，邮件服务可能会有一个名为mail的用户，Web服务器可能会有一个名为www或httpd的用户。这些用户的权限通常被严格限制，以提高系统的安全性。
3. **普通用户**：普通用户是进行日常工作的用户，如登录、运行应用程序、创建和修改个人文件等。普通用户的权限受到限制，通常不能访问其他用户的文件或系统文件，除非有明确的权限设置。每个普通用户都会有一个主目录，用于存储个人文件。

用户组：用户组是一种将用户组织起来的方式，主要用于共享资源和权限管理。每个用户都属于至少一个用户组。用户组可以使管理员更容易地管理一组有相同权限需求的用户。比如，你可以创建一个名为developers的用户组，然后将所有开发人员添加到这个组，然后给这个组分配对某些文件或目录的访问权限。用户组的信息存储在`/etc/group`文件中。

用户和用户组相关文件(4个文件），管理命令（重点掌握useradd（-s/-d/-m/-g）、passwd、groupadd等。usermod、userdel(-r)、groupmod、groupdel、gpasswd等）的使用场景、格式、常用选项等

在Linux系统中，与用户和用户组相关的主要有四个文件：

1. `/etc/passwd`：存储用户信息，每一行代表一个用户账户，包括用户名、用户ID、组ID、家目录、默认shell等信息。
2. `/etc/shadow`：存储用户的加密密码及其相关信息，如密码的修改日期、密码的有效期等。
3. `/etc/group`：存储用户组信息，每一行代表一个用户组，包括组名、组ID，以及组成员。
4. `/etc/gshadow`：存储用户组的密码和组管理员信息。

重点掌握

1. `useradd`：添加新用户。常用选项包括：
   - `-m`：创建用户的家目录。
   - `-s`：设置用户的默认shell。
   - `-d`：指定用户的家目录。

- `-g`：指定用户的初始用户组。
  例如，`useradd -m -s /bin/bash -d /home/newuser -g users newuser`将创建一个新用户newuser，设置其默认shell为/bin/bash，家目录为/home/newuser，并将其添加到users组。

2. `passwd`：修改用户密码。例如，`passwd newuser`将提示你输入新用户的新密码。
3. `groupadd`：添加新用户组。例如，`groupadd newgroup`将创建一个新的用户组newgroup。
4. `usermod`：修改用户属性。例如，`usermod -s /bin/tcsh -d /home/olduser newuser`将修改newuser的默认shell为/bin/tcsh，并将其家目录更改为/home/olduser。
5. `userdel`：删除用户。使用`-r`选项将同时删除用户的家目录和邮件池。例如，`userdel -r olduser`。
6. `groupmod`：修改用户组的属性，如更改组名。例如，`groupmod -n oldgroup newgroup`将更改oldgroup的名称为newgroup。
7. `groupdel`：删除用户组。例如，`groupdel oldgroup`。
8. `gpasswd`：管理用户组的成员和密码。例如，`gpasswd -a user group`将添加user到group组，`gpasswd -d user group`将从group组删除user。

su、sudo命令的作用和使用方法

`su`和`sudo`是两个用于权限切换的命令：

1. `su`：切换用户。例如，`su newuser`将切换到newuser，你将需要输入newuser的密码。
2. `sudo`：以其他用户（默认为root）的身份执行命令。例如，`sudo ls /root`将以root的身份列出/root目录的内容。使用sudo执行的命令需要在`/etc/sudoers`文件中进行配置，并且通常需要输入当前用户的密码。

ubuntu下硬盘的分区和命名方式，主分区、扩展分区、逻辑分区

在Ubuntu（及其他Linux发行版）下，硬盘分区和命名方式遵循特定的规则。

对于传统的机械硬盘（HDD）和SATA接口的固态硬盘（SSD），其设备名称通常以`/dev/sd`开头，接着跟一个由a开始的字母，比如`/dev/sda`，`/dev/sdb`等。其中，`/dev/sda`通常是第一块硬盘，`/dev/sdb`是第二块硬盘，以此类推。

硬盘上的每个分区都会有一个唯一的编号，从1开始。例如，`/dev/sda1`是`/dev/sda`硬盘上的第一个分区，`/dev/sda2`是第二个分区，以此类推。

对于NVMe接口的固态硬盘，设备名称则是`/dev/nvme0n1`，`/dev/nvme0n2`等，其中，分区编号的命名方式类似，例如`/dev/nvme0n1p1`，`/dev/nvme0n1p2`等。

在硬盘分区方面，一个硬盘可以有多个分区，分区类型主要有以下三种：

1. **主分区（Primary Partition）**：一个硬盘最多可以有四个主分区。主分区可以被系统直接识别，可以安装操作系统。
2. **扩展分区（Extended Partition）**：当你需要在一个硬盘上创建超过四个分区时，就需要创建扩展分区。扩展分区本身不能被直接使用，它是包含逻辑分区的容器。
3. **逻辑分区（Logical Partition）**：逻辑分区是在扩展分区内创建的分区，数量上没有限制。逻辑分区从逻辑上扩展了主分区的数量限制。

注意：在使用GPT分区表的硬盘上，没有主分区和扩展分区的概念，可以创建多达128个分区。

你可以使用诸如`fdisk`、`parted`、`gparted`等工具来查看和管理硬盘分区。

进程管理的常用命令：ps（静态）、top（动态）、kill（杀）

1. `ps`：`ps`命令用于显示当前进程的状态。默认情况下，它只显示当前shell中的进程。一些常用的选项包括：
   - `-e`或`--every`：列出所有进程。
   - `-f`或`--full`：提供全格式输出。
   - `-u`或`--user`：列出某个用户的所有进程。
   - `-x`：列出没有控制终端的进程。
     例如，`ps -ef`将列出所有进程的详细信息。
2. `top`：`top`命令提供一个实时的动态视图，用于显示系统中的进程和其状态。它可以按CPU使用率、内存使用率等进行排序，还可以发送信号来控制进程。按`q`可以退出`top`命令。
3. `kill`：`kill`命令用于向进程发送信号。默认情况下，如果没有指定信号，`kill`命令会发送TERM信号，请求进程自行退出。如果这不起作用，可以使用`kill -9`来强制杀死进程（发送KILL信号）。例如，`kill 12345`将发送TERM信号给进程ID为12345的进程，`kill -9 12345`将强制杀死这个进程。

重点掌握vi编辑器三种工作模式（命令行模式、输入模式、末行模式）以及模式之间切换的方法；掌握三种模式下常用的操作。能够描述修改文件的操作过程。

1. **命令行模式（Normal Mode）**：这是`vi`启动后的默认模式。在这种模式下，你可以使用键盘输入各种命令来导航、复制、粘贴、搜索和替换等。例如，`h`、`j`、`k`和`l`用于左、下、上、右移动光标；`dd`删除当前行，`yy`复制当前行，`p`粘贴等。
2. **输入模式（Insert Mode）**：在这种模式下，你可以插入文本。从命令行模式进入输入模式，可以使用`i`（在当前光标位置插入）、`a`（在当前光标后插入）、`I`（在当前行首插入）、`A`（在当前行尾插入）、`o`（在当前行下插入新行）、`O`（在当前行上插入新行）等命令。要从输入模式返回命令行模式，可以按`Esc`键。
3. **末行模式（Command-line Mode 或 Ex Mode）**：在这种模式下，你可以输入保存文件、退出`vi`、设置选项等命令。从命令行模式进入末行模式，可以按`:`、`/`或`?`键。例如，`:w`保存文件，`:q`退出`vi`，`:wq`保存并退出，`:set number`显示行号等。输入命令后，按`Enter`键执行。

修改文件的操作过程可能如下：

1. 启动`vi`并打开文件，如`vi filename`。
2. 进入输入模式，进行文本插入或修改，如按`i`进入输入模式。
3. 按`Esc`键返回命令行模式。
4. 在命令行模式下进行导航、复制、粘贴、删除等操作。
5. 如果需要，可以再次进入输入模式进行修改。
6. 按`Esc`键返回命令行模式，然后按`:`进入末行模式。
7. 在末行模式下输入`wq`保存并退出，或者`q!`强制退出不保存。

重点掌握shell脚本的基本格式，执行方式（三种）、脚本变量（系统（位置参数）变量、环境变量以及自定义变量）。

一个基本的shell脚本通常以`#!/bin/bash`开头，这一行被称为shebang。它告诉系统此脚本需要使用哪个解释器来执行，这里是`/bin/bash`。接下来就是一些shell命令，就像你在命令行中输入的命令一样。

例如，以下是一个基本的shell脚本：

```
#!/bin/bash
echo "Hello, world!"
```

这个脚本非常简单，只做一件事：打印”Hello, world!”。

脚本的执行方式有三种：

1. 直接调用解释器执行脚本，如：`bash script.sh`
2. 使脚本具有执行权限，然后直接执行脚本，如：`chmod +x script.sh`，然后`./script.sh`
3. 在当前shell环境下执行脚本，如：`source script.sh`或`. script.sh`

在shell脚本中，有三种类型的变量：

1. **位置参数变量**：这些变量代表命令行参数。例如，`$0`是脚本名，`$1`是第一个参数，`$2`是第二个参数，依此类推。`$#`是参数的个数，`$*`和`$@`是所有参数的列表，但在引用时行为略有不同。
2. **环境变量**：这些变量对整个系统环境有效。例如，`$PATH`代表可执行文件的搜索路径，`$HOME`代表用户的家目录，`$USER`代表当前用户的用户名。你可以使用`export`命令在shell和它的子进程中创建或修改环境变量。
3. **自定义变量**：你可以在脚本中创建自己的变量。例如，`my_variable="Hello, world!"`创建了一个名为`my_variable`的变量。在引用这个变量时，需要在前面加上`$`，如：`echo $my_variable`。注意，等号两边不能有空格。

重点掌握shell脚本的编程步骤，会写简单的shell脚本：参数的输入及判断、输出echo，运算符（数值、文件、字符串比较），算数运算符，特殊字符，控制语句（test、if、while、for）。能使用shell脚本实现简单的逻辑：求和、阶乘和、偶数和等

基本步骤：

1. 设定解释器：通常，我们使用`#!/bin/bash`指定bash作为解释器。
2. 写脚本逻辑：这包括定义变量、编写条件判断和循环等。
3. 测试和调试：运行脚本，查看是否存在错误或者是否符合预期。
4. 调整权限：使用`chmod`命令调整脚本的权限，使得它可以执行。

```
#!/bin/bash

# check if argument is provided
if [ $# -eq 0 ]; then
    echo "Please provide an argument"
    exit 1
fi
```

```
n=$1
sum=0

# calculate sum
for (( i=1; i<=n; i++ ))
do
    sum=$((sum + i))
done

echo "The sum from 1 to $n is $sum"
```

上面的脚本首先检查是否提供了一个参数。如果没有，脚本将输出错误消息并退出。然后，脚本将输入的参数存储在变量`n`中，并初始化变量`sum`为0。接着，脚本使用for循环从1迭代到`n`，并在每次迭代时将迭代变量`i`加到`sum`上。最后，脚本输出计算结果。

保存上述脚本为`sum.sh`，然后可以这样运行：`bash sum.sh 10`。脚本将输出从1到10的整数和。

这只是一个基本的例子，shell脚本的可能性非常多。你可以根据需要添加更复杂的逻辑，比如使用if/else条件判断，使用while或until循环，定义和使用函数，处理数组等。