1. bash是什么

Bash（全称 Bourne Again SHell）是Linux和Unix-like系统中最常用的Shell（命令行解释器）。简单来说，Shell是用户与操作系统内核之间的“桥梁”——它解释你输入的命令，并执行相应的操作。

Bash 的特点：

它是GNU项目开发的，基于传统的Bourne Shell（sh）改进版。

支持脚本编写（.sh文件），变量、循环、条件判断等编程功能。

默认在大多数Linux发行版（如Ubuntu、CentOS）中作为登录Shell。

命令提示符：

命令提示符（Command Prompt），在Bash Shell中称为PS1（Primary Prompt String）。

以我主机为例：sheldon@sheldon-virtual-machine:~$

| **部分** | **示例值** | **含义与作用** |
| --- | --- | --- |
| **用户名** | sheldon | 当前登录用户的用户名（从/etc/passwd获取）。它告诉你是谁在使用系统。 |
| **@** | @ | 简单分隔符，没有特殊含义，只是视觉上连接用户名和主机名。 |
| **主机名** | sheldon-virtual机 | 当前机器的主机名（hostname，从/etc/hostname或系统配置获取）。这里是虚拟机名，告诉你运行在哪台机器上（在多机环境中特别有用）。 |
| **:** | : | 分隔符，连接主机名和当前目录。 |
| **当前目录** | ~ | 当前工作目录的路径。~ 是家目录的缩写（例如/home/sheldon）。如果是其他目录，会显示如/home/sheldon/docs。它帮你知道命令会影响哪个位置的文件。 |
| **提示符符号** | $ | 命令输入点。$ 表示普通用户（非root）。如果是root用户，会显示#（更醒目，提醒小心操作）。输入命令后，按Enter执行，它会显示输出并返回新提示符。 |

再来一个：

root@iZm5e8dsxce9ufaic7hi3uZ ~# pwd

命令解析：

root：表示用户名；

iZm5e8dsxce9ufaic7hi3uZ：表示主机名；

~：表示目前所在目录为家目录，其中root用户的家目录是 /root普通用户的家目录在 /home 下；

#：指示你所具有的权限（root用户为# ，普通用户为$ ）。

1. 命令与命令格式：

命令格式：command parameters（命令 参数）

长短参数：

单个参数：ls -a（a 是英文 all 的缩写，表示“全部”）

多个参数：ls -al（全部文件 + 列表形式展示）

单个长参数：ls --all

多个长参数：ls --reverse --all

长短混合参数：ls --all -l

参数值：

短参数：command -p 10（例如：ssh root@121.42.11.34 -p 22）

长参数：command --paramters=10（例如：ssh root@121.42.11.34 --port=22）

短参数使用一个短斜杠，长参数使用两个短斜杠。短参数可以连写，长参数不可以。参数的执行效果与顺序无关。除了与位置有关的参数外，顺序可以打乱。

常用快捷指令一览表：

通过上下方向键 ↑ ↓ 来调取过往执行过的Linux命令；

命令或参数仅需输入前几位就可以用 Tab 键补全；

Ctrl + R ：用于查找使用过的命令（输入指令的一部分，会自动查找你之前输入过的指令，之后Enter使用该命令，Ctrl + c退出搜索， Ctrl + G搜索其他匹配的历史指令）

（Research，搜索输入过的命令）  
Ctrl + L：清除屏幕并将当前行移到页面顶部；（与输入clear指令的作用结果相似，不同的是Ctrl + L没有真的清除屏幕，而是相当于把之前的历史向上拉，拉出了一片空地，你可以通过鼠标滚轮向上查看历史，而clear命令则把命令行清空了，并且clear命令会留下指令历史记录，Cltr+L指令没有历史记录）

（Lower，拉至底部）  
Ctrl + C：中止当前正在执行的命令；（常用，发送SIGINT 信号，默认发送给前台进程（用 tcgetpgrp() 获取进程组ID）。进程收到信号：进程的信号处理程序（signal handler）执行，默认行为是终止进程（terminate）。

进程执行到一半退出会有什么后果？取决于进程的“原子性”（atomicity）和状态管理。大多数命令设计为“幂等”或“检查点”，中断后系统会回滚或清理，但不是总是完美。进程退出时，内核会释放资源（如内存、文件句柄），但已完成的“副作用”可能保留。

成功中断：进程收到SIGINT，执行清理（atexit handlers），然后退出。Shell提示符返回。

潜在风险：数据丢失、半成品状态、资源泄漏（罕见）。

| **场景类型** | **示例命令** | **中断后果（按 Ctrl + C）** | **为什么这样？** |
| --- | --- | --- | --- |
| **纯计算/输出** | ping -c 100 google.com（ping 100次） | 立即停止，输出最后一行“---”总结。只丢失未完成的ping，无文件变化。 | 命令无持久状态，简单终止。 |
| **文件下载/复制** | wget http://bigfile.iso 或 cp bigfile1 bigfile2 | 下载/复制中断，目标文件保留已下载/复制的部分（不完整）。需从头重跑。 | 文件I/O是“增量”的，系统不自动回滚。 |
| **文件修改到一半** | echo "新内容" > file.txt（覆盖）或 vim file.txt（编辑中） | - 简单echo：文件可能为空或部分写（取决于缓冲）。 - vim：退出时提示“保存？”（:q! 不保存，回到原状态）。 总体：文件可能保留“半改”状态，或回滚（取决于编辑器）。 | 文件系统用缓冲区（buffered I/O），中断时fsync未调用，变化可能未flush到磁盘。vim有内部状态检查。 |
| **数据库/事务** | mysql 导入数据 | 事务可能回滚（如果支持ACID），或数据库半更新（需手动修复）。 | 现代工具有事务日志，回滚易；简单脚本无。 |
| **后台/守护进程** | 不适用（见下文） | Ctrl + C 只影响前台；后台用 kill。 | 信号只发前台组。 |

文件修改例子详解（你特别问的）：

简单重定向：如 cat newdata.txt > oldfile.txt（复制中）。中断后，oldfile.txt 可能只有部分数据（e.g., 前半文件），因为I/O是流式的（stream）。系统不会“魔法”回滚——文件已打开写模式，变化持久化直到flush。

后果：文件“半改”，需备份恢复（用 mv oldfile.txt oldfile.txt.bak 前跑）。

编辑器如vim：vim有“交换文件”（.swp），中断后重开会提示恢复。选择不保存：回到最初状态（:q!）。但如果已保存一次（:w），半改部分保留。

. 有时 Ctrl + C 没有成功退出，可能原因？

常见原因（按概率排序，用列表）：

进程忽略SIGINT：程序自定义处理信号（如 trap '' INT 在脚本中，或 sigaction(SIGINT, SIG\_IGN) 在C程序）。e.g., dd if=/dev/zero of=file bs=1M 默认忽略（用 Ctrl + \ 发SIGQUIT）。

解决：用 Ctrl + \ (SIGQUIT) 或 kill -TERM <PID>（温和终止）。

进程在系统调用中阻塞：如等待I/O（read/write）、sleep，或NFS挂起。信号排队，但进程忙时延迟响应。

解决：等几秒，或用 kill -KILL <PID>（强制杀，但危险——不清理）。

后台/多进程：Ctrl + C 只中断前台；如果命令fork子进程（如 find / | xargs rm），子进程可能继续。

解决：用 pkill 或 jobs 查后台，fg 带前台。

终端问题：远程SSH冻结（网络延迟），或终端模拟器bug（e.g., VS Code终端）。或Shell配置改了绑定（stty intr ^Z）。

解决：检查 stty -a（看intr键），或新开终端。

权限/容器：在Docker/chroot中，信号可能被容器隔离。

解决：进容器用 docker exec 或 kill 从宿主机。

Ctrl + U：从光标位置剪切到行首；（Up，剪切到行首）

Ctrl + K：从光标位置剪切到行尾；（Kill，剪切到行尾）

Ctrl + W：剪切光标左侧的一个单词；（Word，剪切一个单词）

Ctrl + Y：粘贴Ctrl + U | K | Y剪切的命令；（Yank，粘贴剪切的内容）

Ctrl + A：光标跳到命令行的开头；（Ahead，跳回行开头）

Ctrl + E：光标跳到命令行的结尾；（End，跳至行尾）

Ctrl + D：关闭Shell会话（要清空当前输入的命令，否则执行不了）；（Delete，退出命令行）

常见命令：  
pwd（Print Working Directory）打印当前工作目录

which：查看命令的可执行文件所在路径，Linux下，每一条命令其实都对应一个可执行程序，在终端中输入命令，按回车的时候，就是执行了对应的那个程序，which命令本身对应的程序也存在于Linux中。

总的来说一个命令就是一个可执行程序。

输入命令等同于执行一个可执行程序，命令对应的可执行程序都放在/usr/bin中，可以使用

**ls查看所有命令（list）**

ls:列出文件和目录，它是Linux最常用的命令之一。（list）

-a显示所有文件和目录包括隐藏的（隐藏文件用.开头）（all)  
为何隐藏文件：  
许多文件是用户/系统配置（如 .git/、.env），含敏感信息（密码、API密钥）。隐藏避免意外泄露或编辑。

普通用户无需碰系统/临时文件（如 .cache/、.DS\_Store）。隐藏减少“乱动”风险，保持目录整洁。

. 像“点缀”，标记“辅助”文件（config、temp、backup）。可见文件是“主角”，隐藏是“配角”。

大目录（如 /tmp）有海量临时 .pid 文件，隐藏后 ls 更快、更易读。

-l：显示详细列表（long format）

详细列出文件/目录的属性，包括：

文件类型和权限。

硬链接数、所有者和组。

文件大小、修改时间、文件名。

默认 ls 只给名字（如“file.txt”），-l 像“文件身份证”，帮你快速诊断（e.g., 文件可读？谁拥有？多大？）。

| **列位置** | **示例输出** | **含义与解释** |
| --- | --- | --- |
| **1** | - rw-r--r-- | **权限字符串**：10字符。 - 第一：类型（-普通文件，d目录，l符号链接）。 - 后9：权限（r=读，w=写，x=执行；- =无）。分三组：用户/组/其他（e.g., rw- = 用户读写无执行）。 |
| **2** | 1 | **硬链接数**：文件被链接的次数（目录常见2+，文件通常1）。 |
| **3** | sheldon | **所有者（owner）**：文件所属用户。 |
| **4** | sheldon | **组（group）**：文件所属用户组。 |
| **5** | 1024 | **大小**：字节数（数字）。用 -h 选项变人性化（如1.0K）。 |
| **6-8** | Oct 10 10:00 | **修改时间**：最后修改日期/时间（年份若旧）。 |
| **9** | example.txt | **文件名**：核心信息。 |

sheldon@sheldon-virtual-machine:/usr/bin$ ls -l

总计 232632

-rwxr-xr-x 1 root root 51632 2月 8 2022 '['

-rwxr-xr-x 1 root root 35344 3月 7 2024 aa-enabled

-rwxr-xr-x 1 root root 35344 3月 7 2024 aa-exec

-rwxr-xr-x 1 root root 31248 3月 7 2024 aa-features-abi

总计 232632

含义：目录总占用大小（total），单位是“块”（通常4KB/块）。这里 232632 块 ≈ 906 MB（232632 × 4KB）。它计算所有文件/子目录的元数据和内容占用。

| **列位置** | **示例输出** | **含义与解释** |
| --- | --- | --- |
| **1** | -rwxr-xr-x | **权限字符串**：10字符，定义文件类型和访问权。 - 第一位：- = 普通文件（非目录 d、非链接 l）。 - 后9位：r=读（read）、w=写（write）、x=执行（execute）；- = 无。 分三组：**用户（owner）** rw x（读写执行）；**组（group）** r-x（读执行）；**其他** r-x（读执行）。 这里：所有人都可读/执行，但只有 root 可写。为什么？系统命令需执行，但防篡改。 |
| **2** | 1 | **硬链接数**（hard links）：文件被链接的次数。通常1（单一文件）；目录常见2（自身+上级）。这里1=标准文件。 |
| **3** | root | **所有者（owner）**：文件所属用户（root=超级用户）。系统文件多 root 拥有，防普通用户改。 |
| **4** | root | **所属组（group）**：文件所属用户组（root 组）。类似所有者，控制组权限。 |
| **5** | 51632 | **文件大小**：字节数（51632 ≈ 50 KB）。这是二进制内容大小，不含元数据。用 -h 选项变 K/M/G。 |
| **6** | 2月 8 2022 | **最后修改时间**：文件内容/属性上次改的日期（月 日 年）。这里 2022 年 2 月 8 日（旧文件）。新文件如 aa-enabled 是 2024 年 3 月 7 日（系统更新）。 |
| **7** | （隐含时间） | **时间**：如果同年，显示时:分（e.g., 10:00）；否则省略。这里无，因为跨年。 |
| **8** | '[' | **文件名**：[ —— 这是 test 命令的别名（用于 [ condition ] 测试）。其他如 aa-exec 是 AppArmor 执行工具。 |

-h：适合人类阅读的（human readable）

主要就是把文件大小由字节数转换成人类熟悉的单位（KB，MB等）

简单。

-t按文件最近一次修改时间排序（time）

从上至下为最晚修改的文件到最早修改的文件

-i显示文件的inode（inode是文件内容的标识）

2. Inode 是什么？

全称：Index Node（索引节点）。

Index：索引（像文件系统的“目录索引”）。

Node：节点（数据结构节点）。

本质：inode 是文件系统（e.g., ext4、xfs）中的一个元数据结构（metadata structure），用于存储文件的属性信息，但不存储文件实际内容（数据块）。每个文件/目录在文件系统上都有一个唯一的 inode。

**cd：cd是英语 change directory的缩写，表示切换目录。**

cd / --> 跳转到根目录

cd ~ --> 跳转到家目录

cd .. --> 跳转到上级目录

cd ./home --> 跳转到当前目录的home目录下

cd /home/lion --> 跳转到根目录下的home目录下的lion目录

cd --> 不添加任何参数，也是回到家目录

**du:列举目录大小信息。（全称：Disk Usage（磁盘使用量）。）**  
在一个目录下使用du是列出该目录下的目录的大小，看不到该目录下的文件大小

文件大小会被打包到总计大小中，但不会列出。

如果想看到文件大小需要加-a参数（all）会列出所有目录以及文件的大小。

【常用参数】

-h 适合人类阅读的；

-a同时列举出目录下文件的大小信息；

-s只显示总计大小，不显示具体信息。

**cat：一次性显示文件所有内容，更适合查看小的文件。（**

全称：Concatenate（连接）。

Concatenate：连接/串联（英文词，源自拉丁文“con-”（一起）+ “tenere”（持有），意思是“把多个东西连成一串”）。

为什么叫这个？ cat 的设计初衷是连接多个文件的内容，并输出到标准输出（stdout）。早期Unix中，你可以用 cat file1 file2 > output 把两个文件“拼接”成一个。单文件时，它就“连接”文件内容到屏幕（输出显示）。

助记技巧：C-A-T = "Concatenate And Type"（连接并打印）——简单记成“连接文件输出”。

核心作用：读取文件内容，并直接输出到终端（或重定向到其他地方）。它不解释内容，只是“盲读盲输”——适合文本文件（如日志、脚本）。

常见用法：

cat file.txt：显示文件内容。

cat file1 file2：连接多个文件输出。

cat > newfile：从输入创建新文件（Ctrl+D结束）。

选项：-n（加行号）、-E（显示行尾 $）。

为什么“列出”？ 单文件时像“列印”（print），但本质是“dump”内容。

3. 如果用 cat 显示非文本文件，会看到什么？

关键点：cat 不区分文件类型——它无脑输出二进制数据（raw bytes），不管是文本还是可执行程序。结果：屏幕会显示乱码（garbled text），因为非文本文件含二进制码（0-255字节），包括不可打印字符（control codes，如换行、颜色码）、十六进制数据等。

为什么乱码？ 终端假设输入是ASCII/UTF-8文本（可打印字符），二进制会触发奇怪效果：光标乱跳、颜色闪、甚至终端卡顿。不是“错误”，只是人类眼不适。

后果：无害（文件不变），但不实用——用 hexdump 或 xxd 看二进制更好。

具体看到什么？ 取决于文件：

可执行程序（e.g., /bin/ls）：ELF二进制头 + 机器码，看起来像 ^@ ^A ^B（控制符）+ 随机字母/数字字符串。

图片（e.g., PNG/JPG）：二进制头（‰PNG 或 JFIF）+ 像素数据，显示为乱码 + 可能的部分可读字符串（如元数据）。

压缩文件（e.g., ZIP）：头信息 + 压缩块，纯乱码。

**less：分页显示文件内容，更适合查看大的文件。**

本质：less 是一个分页查看器（pager），用于交互式浏览文件内容。它一次只加载一屏内容，支持向上/向下滚动、搜索、跳转等操作，让你像翻书一样“少量阅读”（hence the name）。

作用：

显示文件内容，但不一次性全倒到终端（避免刷屏）。

适合大文件：e.g., less /var/log/syslog（系统日志，可能几MB）。

交互命令：按 q 退出；Space 下翻页；b 上翻页；/ + 关键词 搜索；g 跳首；G 跳尾。

3. less 和 cat 的区别是什么？

核心区别：cat 是“全速输出”（dump everything），适合小文件或连接；less 是“分页交互”（browse interactively），适合大文件或需要导航的场景。cat 像“倒书到桌上”，less 像“手持书本翻阅”。

| **方面** | **cat** | **less** | **何时选哪个？** |
| --- | --- | --- | --- |
| **输出方式** | 一次性全输出到终端（stdout）。 | 分页显示，一屏一屏加载，支持滚动。 | cat 小文件；less 大文件。 |
| **交互性** | 无（输出后结束）。 | 高：搜索（/）、翻页（Space/b）、退出（q）。 | cat 简单读；less 复杂浏览。 |
| **文件大小适应** | 适合小文件（<1MB），大文件刷屏/卡。 | 专为大文件设计，懒加载（不全读入内存）。 | cat 脚本管道；less 日志调试。 |
| **多文件支持** | 是：cat file1 file2 连接输出。 | 是，但逐个分页：less file1 file2（n 下一个）。 | cat 拼接；less 逐览。 |
| **示例** | cat /etc/passwd（全输出用户列表）。 | less /etc/passwd（分页翻，搜 "root"）。 | cat 复制内容；less 探索。 |
| **性能/资源** | 快，但大文件耗内存/CPU。 | 高效，边读边显。 | cat 自动化；less 交互。 |

【快捷操作】

空格键：前进一页（一个屏幕）；

b 键：后退一页；（back）

回车键：前进一行；

y 键：后退一行；（Yank）

上下键：回退或前进一行；

d 键：前进半页；（down）

u 键：后退半页；（up）

q 键：停止读取文件，中止less命令；（quit）

= 键：显示当前页面的内容是文件中的第几行到第几行以及一些其它关于本页内容的详细信息；（Equals）

h 键：显示帮助文档；（help）

/ 键：进入搜索模式后，按 n 键跳到一个符合项目，按 N 键跳到上一个符合项目，同时也可以输入正则表达式匹配。

**head：显示文件的开头几行（默认是 10 行）：**

【参数】

-n 指定行数head cloud-init.log -n 2

**tail：显示文件的结尾几行（默认是 10 行）**

【参数】

-n指定行数tail cloud-init.log -n 2

-f会每过 1 秒检查下文件是否有更新内容，也可以用 -s 参数指定间隔时间tail -f -s 4 xxx.log

**touch：创建一个文件**

touch new\_file

**mkdir：创建一个目录**

mkdir new\_folder

【常用参数】

-p递归的创建目录结构mkdir -p one/two/three

**cp：拷贝文件和目录**

cp file file\_copy --> file 是目标文件，file\_copy 是拷贝出来的文件

cp file one --> 把 file 文件拷贝到 one 目录下，并且文件名依然为 file

cp file one/file\_copy --> 把 file 文件拷贝到 one 目录下，文件名为file\_copy

cp \*.txt folder --> 把当前目录下所有 txt 文件拷贝到 folder 目录下

【常用参数】

-r递归的拷贝，常用来拷贝一整个目录

**mv：移动（重命名）文件或目录，与 cp 命令用法相似。**

mv file one --> 将 file 文件移动到 one 目录下

mv new\_folder one --> 将 new\_folder 文件夹移动到one目录下

mv \*.txt folder --> 把当前目录下所有 txt 文件移动到 folder 目录下

mv file new\_file --> file 文件重命名为 new\_file

文件的删除和链接

rm：删除文件和目录，由于 Linux 下没有回收站，一旦删除非常难恢复，因此需要谨慎操作

rm new\_file --> 删除 new\_file 文件

rm f1 f2 f3 --> 同时删除 f1 f2 f3 3个文件

【常用参数】

-i向用户确认是否删除；

-f文件强制删除；

-r递归删除文件夹，著名的删除操作 rm -rf。