一、cd

命令全称

`cd` ——— change directory (更改目录)

基本语法

```
cd [目标目录]
```

- `[目标目录]`：要切换到的目录路径，可以是绝对路径或相对路径。
- 如果不指定路径，默认切换到用户的 **主目录**。

常用场景及选项

1. 切换到指定目录

```
cd /home/user/documents
```

- 直接切换到 `/home/user/documents` 目录。

2. 切换到用户主目录

```
cd
# 或者
cd ~
```

- `~` 表示当前用户的主目录。

3. 切换到上一级目录

```
cd ..
```

- `..` 表示当前目录的上一级目录。

4. 切换到上次访问的目录

```
cd -
```

- 切换到之前的工作目录，并显示切换的路径。

5. 相对路径切换

```
cd ../projects
```

- 从当前目录的上一级目录进入 `projects` 子目录。

6. 切换到根目录

```
cd /
```

- `/` 表示文件系统的根目录。

查看当前工作目录

切换目录后，可以用以下命令查看当前位置：

```
pwd
```

二、ls

命令全称

`ls` —— **list segments**（列出目录内容）。

基本语法

```
ls [选项] [目录或文件]
```

常用参数

参数	作用
`-a`	显示所有文件，包括隐藏文件（以 `.` 开头）。
`-l`	以长格式显示文件信息。
`-h`	配合 `-l`，以人类可读格式显示文件大小。
`-R`	递归显示子目录中的所有文件和目录。
`-d`	只显示目录本身，而不是目录内容。
`-t`	按修改时间排序（最近的优先）。
`-S`	按文件大小排序（大到小）。
`-r`	逆序排列，与其他排序参数组合使用。
`-i`	显示文件的 inode 编号。

基本用法

1. **列出当前目录的内容**

   ```
   ls
   ```

2. **显示所有文件，包括隐藏文件**

```
ls -a
```

3. **长格式显示文件信息**

```
ls -l
```

4. **人类可读的长格式显示**

```
ls -lh
```

5. **递归显示子目录中的内容**

```
ls -R
```

6. **按时间排序显示文件**

```
ls -lt
```

三、mkdir

命令全称

`mkdir` - make directory

基本语法

```
mkdir [选项] 目录名
```

常见参数

- `p`：如果父目录不存在，则一并创建。
- `m`：设置新目录的权限。
- `v`：显示详细的操作过程。
- `-help`：显示帮助信息，列出所有可用选项。

常见用法

- **创建单个目录**：

```
mkdir myfolder
```

- **创建多个目录**：

```
mkdir folder1 folder2 folder3
```

- **创建多层目录**（父目录不存在时自动创建）：

```
mkdir -p /home/user/projects/2024
```

- **设置权限创建目录**：

```
mkdir -m 755 newfolder
```

- **显示创建目录的详细信息**：

```
mkdir -v newfolder
```

- **创建多个目录并设置权限**：

```
mkdir -m 700 dir1 dir2
```

四、touch

命令全称

`touch` - 改变文件的访问和修改时间，或创建空文件。

基本语法

```
touch [选项] 文件名
```

常见参数

- `a`：只改变文件的访问时间。
- `m`：只改变文件的修改时间。
- `c`：如果文件不存在，不创建新文件，仅修改时间。
- `t`：使用指定的时间戳来修改文件的时间，格式为 `[YYYYMMDDhhmm.ss]`。
- `d`：使用指定的日期时间来修改文件的时间，格式为 `YYYY-MM-DD HH:MM:SS`。
- `r`：根据另一个文件的时间戳来修改文件的时间。
- `-help`：显示帮助信息，列出所有可用选项。

常见用法

- **创建空文件**：

```
touch myfile.txt
```

  如果 `myfile.txt` 不存在，`touch` 会创建一个空文件。

- **更新文件的访问和修改时间**：

```
touch existingfile.txt
```

  这个命令会更新 `existingfile.txt` 的访问时间和修改时间为当前时间。

- **只更新访问时间**：

```
touch -a existingfile.txt
```

- **只更新修改时间**：

```
touch -m existingfile.txt
```

- **不创建文件，只修改时间（如果文件不存在）**：

```
touch -c nonexistentfile.txt
```

- **使用指定的时间戳修改文件时间**：

```
touch -t 202412031200.00 myfile.txt
```

这个命令将 `myfile.txt` 的时间戳设置为 2024年12月3日12点00分00秒。

- **根据另一个文件的时间戳来修改文件的时间**：

```
touch -r referencefile.txt targetfile.txt
```

这个命令会将 `targetfile.txt` 的时间戳设置为 `referencefile.txt` 的时间戳。

五、cat

命令全称

`cat` - concatenate and display files 查看文件内容

基本语法

```
cat [选项] [文件...]
```

常见参数

- `n`：为输出的每一行编号，从 1 开始。
- `b`：为非空输出的每一行编号。
- `E`：在每行的末尾显示 `$`，用于标记行尾。
- `T`：将制表符（Tab）显示为 `^I`。
- `A`：显示所有不可见字符，包括换行符、制表符等，类似 `vET` 组合。
- `v`：显示不可打印的字符（除换行符外）。
- `s`：压缩多个连续的空行成一个空行。
- `-help`：显示帮助信息，列出所有可用选项。

常见用法

- **显示文件内容**：

```
cat myfile.txt
```

显示 `myfile.txt` 文件的内容。

- **创建一个新文件并输入内容**：

```
cat > newfile.txt
```

使用 `cat` 创建 `newfile.txt` 文件并输入内容，输入结束后按 `Ctrl + D` 退出。

- **连接多个文件并显示其内容**：

  ```
  cat file1.txt file2.txt
  ```

  将 `file1.txt` 和 `file2.txt` 的内容连接起来并显示。

- **将多个文件内容合并到一个文件中**：

  ```
  cat file1.txt file2.txt > mergedfile.txt
  ```

  将 `file1.txt` 和 `file2.txt` 的内容合并并输出到 `mergedfile.txt` 中。

- **为文件内容每行添加行号**：

  ```
  cat -n myfile.txt
  ```

  为 `myfile.txt` 文件中的每一行添加行号。

- **显示不可见字符（如换行符和制表符）**：

  ```
  cat -v myfile.txt
  ```

- **删除空行并合并为一个空行**：

  ```
  cat -s myfile.txt
  ```

- **显示文件内容，并在每行末尾显示 `$`**：

  ```
  cat -E myfile.txt
  ```

- **显示所有特殊字符和制表符**：

  ```
  cat -A myfile.txt
  ```

六、more

命令全称

`more` - 分页查看文件内容

基本语法

```
more [选项] [文件...]
```

常见参数

- `n`：设置每页显示的行数。
- `s`：压缩连续的空行成一行。
- `c`：强制在清屏模式下显示内容，避免屏幕闪烁。
- `d`：启用 "按任意键继续" 提示，类似 `less` 的行为。
- `p`：清屏显示，不保留历史内容。
- `f`：强制将文件视为一系列的行而不是按行拆分（处理长行时有用）。

- `-help`：显示帮助信息，列出所有可用选项。

常见用法

- **查看文件内容**：

  ```
  more myfile.txt
  ```

  分页显示 `myfile.txt` 文件的内容。

- **分页查看长文件内容**：

  ```
  more longfile.txt
  ```

  通过分页显示 `longfile.txt` 文件内容，用户可以通过按空格键继续查看下一页内容，按 `q` 退出查看。

- **查看大文件时，指定每页显示行数**：

  ```
  more -n 20 myfile.txt
  ```

  设置每页显示 20 行内容。

- **查看文件并压缩连续空行**：

  ```
  more -s myfile.txt
  ```

  连续的空行会被压缩为一个空行。

- **清屏后显示内容**：

  ```
  more -c myfile.txt
  ```

  在清屏模式下分页显示文件内容，避免屏幕闪烁。

- **启用 "按任意键继续" 提示**：

  ```
  more -d myfile.txt
  ```

  启用提示 "Press space to continue" 等，帮助用户导航文件。

- **分页显示多个文件**：

  ```
  more file1.txt file2.txt
  ```

  分页显示 `file1.txt` 和 `file2.txt` 文件的内容。

七、cp

命令全称

`cp` - copy files and directories

基本语法

```
cp [选项] 源文件 目标文件/目标目录
```

常见参数

- `r` 或 `R`：递归复制整个目录及其内容（包括子目录和文件）。
- `i`：交互模式，在覆盖文件之前进行确认。
- `f`：强制复制，若目标文件无法覆盖，会先删除目标文件。
- `u`：仅在源文件较新或者目标文件不存在时才进行复制。
- `v`：显示详细的复制过程。
- `a`：归档模式，等同于 `dR --preserve=all`，保留所有文件属性，并递归复制目录。
- `p`：保留源文件的属性，包括时间戳、权限等。
- `l`：创建硬链接，而不是复制文件。
- `-help`：显示帮助信息，列出所有可用选项。

常见用法

- **复制文件**：

  ```
  cp file1.txt file2.txt
  ```

  将 `file1.txt` 复制为 `file2.txt`。

- **复制文件到目录**：

  ```
  cp file1.txt /path/to/directory/
  ```

  将 `file1.txt` 复制到指定的目录。

- **递归复制目录及其内容**：

  ```
  cp -r dir1/ dir2/
  ```

  将目录 `dir1` 及其所有内容（包括子目录和文件）复制到 `dir2`。

- **覆盖时进行确认**：

  ```
  cp -i file1.txt file2.txt
  ```

  在目标文件 `file2.txt` 被覆盖之前，要求用户确认。

- **复制文件并显示详细信息**：

  ```
  cp -v file1.txt file2.txt
  ```

  显示文件复制的详细过程。

- **复制并保留文件的属性（如权限、时间戳等）**：

  ```
  cp -p file1.txt file2.txt
  ```

  复制文件时，保留文件的原始属性。

- **归档模式复制目录**（递归并保留所有属性）：

  ```
  cp -a dir1/ dir2/
  ```

  将 `dir1` 目录及其所有文件和子目录，连同属性一起复制到 `dir2`。

- **只在源文件较新时复制文件**：

  ```
  cp -u file1.txt /path/to/directory/
  ```

  仅当 `file1.txt` 比目标目录中的文件更新，或者目标文件不存在时，才进行复制。

- **复制多个文件到目录**：

  ```
  cp file1.txt file2.txt /path/to/directory/
  ```

  将 `file1.txt` 和 `file2.txt` 复制到指定目录。

- **创建硬链接而非复制文件**：

  ```
  cp -l file1.txt link1.txt
  ```

  创建硬链接 `link1.txt` 指向 `file1.txt`，而不是复制文件内容。

八、mv

命令全称

`mv` - move or rename files or directories 移动/重命名

基本语法

```
mv [选项] 源文件/目录 目标文件/目录
```

常见参数

- `i`：交互模式，在覆盖文件之前进行确认。
- `f`：强制执行操作，若目标文件存在且不可写，`mv` 会先删除目标文件再进行移动。
- `u`：仅在源文件较新或目标文件不存在时才进行移动。
- `v`：显示详细的操作过程。
- `n`：不覆盖目标文件，若目标文件已存在，操作会被取消。
- `b`：在覆盖文件之前，先备份目标文件。
- `-help`：显示帮助信息，列出所有可用选项。

常见用法

- **移动文件**：

  ```
  mv file1.txt /path/to/destination/
  ```

  将 `file1.txt` 移动到 `/path/to/destination/` 目录。

- **重命名文件**：

```
mv oldname.txt newname.txt
```

将 `oldname.txt` 文件重命名为 `newname.txt`。

- **移动多个文件到目录**：

```
mv file1.txt file2.txt /path/to/destination/
```

将 `file1.txt` 和 `file2.txt` 移动到指定目录 `/path/to/destination/`。

- **覆盖文件时进行确认**：

```
mv -i file1.txt /path/to/destination/
```

如果目标文件已存在，`mv` 会提示是否覆盖。

- **强制覆盖文件**：

```
mv -f file1.txt /path/to/destination/
```

即使目标文件存在且不可写，`mv` 也会强制覆盖它。

- **移动目录**：

```
mv dir1/ /path/to/destination/
```

将 `dir1` 目录及其中的所有内容移动到 `/path/to/destination/`。

- **移动文件并显示详细信息**：

```
mv -v file1.txt /path/to/destination/
```

在移动过程中显示详细的操作信息。

- **仅在源文件较新时才移动文件**：

```
mv -u file1.txt /path/to/destination/
```

只有当源文件 `file1.txt` 比目标目录中的同名文件更新，或者目标文件不存在时，才进行移动。

- **不覆盖现有文件**：

```
mv -n file1.txt /path/to/destination/
```

如果目标文件已存在，`mv` 不会覆盖它。

- **备份目标文件**：

```
mv -b file1.txt /path/to/destination/
```

如果目标文件存在，`mv` 会先备份目标文件，然后进行移动。

九、rm

命令全称

- **英文全称**: `rm` - remove files or directories
- **中文解释**: `rm` - 删除文件或目录

基本语法

```
rm [选项] 文件/目录
```

常见参数

- `r` 或 `R`：递归删除目录及其内容（包括子目录和文件）。
- `f`：强制删除文件，不提示确认，即使文件不可写或不存在。
- `i`：交互模式，在删除每个文件之前询问用户确认。
- `I`：删除多个文件时，只有在删除之前询问一次确认。
- `v`：显示详细的删除过程。
- `d`：删除空目录。
- `-help`：显示帮助信息，列出所有可用选项。

常见用法

- **删除文件**:

  ```
  rm file1.txt
  ```

  删除 `file1.txt` 文件。

- **删除多个文件**:

  ```
  rm file1.txt file2.txt
  ```

  删除 `file1.txt` 和 `file2.txt` 两个文件。

- **递归删除目录及其内容**:

  ```
  rm -r dir1/
  ```

  删除 `dir1` 目录及其所有子目录和文件。

- **强制删除文件**:

  ```
  rm -f file1.txt
  ```

  强制删除 `file1.txt`，即使它是只读文件。

- **在删除每个文件前询问确认**:

  ```
  rm -i file1.txt
  ```

  删除 `file1.txt` 时，系统会询问用户确认。

- **删除空目录**:

```
rm -d emptydir/
```

删除空目录 `emptydir`。

- **删除多个文件并显示详细信息**：

```
rm -v file1.txt file2.txt
```

删除 `file1.txt` 和 `file2.txt` 文件，并显示详细的删除过程。

- **使用交互模式删除多个文件**：

```
rm -I file1.txt file2.txt file3.txt
```

删除多个文件时，只会询问一次确认。

- **递归删除目录，并强制删除只读文件**：

```
rm -rf dir1/
```

递归删除 `dir1` 目录及其中的文件和子目录，并强制删除只读文件。

十、grep

命令全称

- **英文全称**: `grep` - global regular expression print
- **中文解释**: `grep` - 全局正则表达式打印

基本语法

```
grep [选项] PATTERN [文件...]
```

常见参数

- `i`：忽略大小写进行匹配。
- `r` 或 `R`：递归搜索目录及其子目录中的文件。
- `v`：反向匹配，显示不包含模式的行。
- `l`：仅显示包含匹配模式的文件名，而不显示匹配的行。
- `n`：显示匹配行的行号。
- `c`：显示匹配模式的行数。
- `H`：在输出结果中显示文件名（即使只有一个文件）。
- `o`：只输出匹配的部分，而不是整行。
- `w`：只匹配完整的单词。
- `A NUM`：显示匹配行及其后面 `NUM` 行。
- `B NUM`：显示匹配行及其前面 `NUM` 行。
- `C NUM`：显示匹配行及其前后 `NUM` 行（上下文匹配）。
- `-help`：显示帮助信息，列出所有可用选项。

常见用法

- **基本文本搜索**：

  ```
  grep "pattern" file.txt
  ```

  在 `file.txt` 中查找匹配字符串 `pattern` 的行。

- **忽略大小写的搜索**：

  ```
  grep -i "pattern" file.txt
  ```

  在 `file.txt` 中查找匹配 `pattern` 的行，忽略大小写。

- **递归搜索目录中的所有文件**：

  ```
  grep -r "pattern" /path/to/directory/
  ```

  在指定的目录及其子目录中的所有文件中查找匹配 `pattern` 的行。

- **显示匹配行的行号**：

  ```
  grep -n "pattern" file.txt
  ```

  在 `file.txt` 中查找 `pattern`，并显示匹配行的行号。

- **显示匹配模式的文件名**：

  ```
  grep -l "pattern" *.txt
  ```

  查找当前目录下所有 `.txt` 文件中包含 `pattern` 的文件名。

- **反向匹配**（显示不包含 `pattern` 的行）：

  ```
  grep -v "pattern" file.txt
  ```

  显示 `file.txt` 中不包含 `pattern` 的所有行。

- **仅输出匹配的部分**：

  ```
  grep -o "pattern" file.txt
  ```

  只输出文件中匹配的部分，而不是整行。

- **显示匹配行及其上下文**：

  ```
  grep -C 2 "pattern" file.txt
  ```

  显示 `file.txt` 中匹配 `pattern` 的行及其前后两行（总共显示 5 行）。

- **统计匹配的行数**：

  ```
  grep -c "pattern" file.txt
  ```

  显示 `file.txt` 中匹配 `pattern` 的行数。

- **只匹配完整的单词**:

  ```
  grep -w "pattern" file.txt
  ```

  仅匹配完整的单词 `pattern`，而不是其一部分。

- **在输出中显示文件名**:

  ```
  grep -H "pattern" file1.txt file2.txt
  ```

  在输出中显示 `file1.txt` 和 `file2.txt` 中包含 `pattern` 的行，并显示文件名。

- **查找并显示多个文件中的匹配内容**:

  ```
  grep "pattern" file1.txt file2.txt
  ```

  查找 `file1.txt` 和 `file2.txt` 中包含 `pattern` 的行，并输出这些行。

十一、|

命令全称

- **英文全称**: Pipe (`|`)
- **中文解释**: 管道符，常用于将一个命令的输出传递给另一个命令作为输入。

基本语法

```
command1 | command2
```

常见用法

- **将一个命令的输出传递给另一个命令**:

  ```
  ls | grep "pattern"
  ```

  这个命令首先列出当前目录的文件，然后通过管道将输出传递给 `grep`，查找包含 `pattern` 的文件名。

- **将命令输出传递给 `more`，分页显示内容**:

  ```
  cat largefile.txt | more
  ```

  将 `largefile.txt` 的内容通过管道传递给 `more`，分页显示内容。

- **将多个命令的输出组合**:

  ```
  ps aux | grep "process_name" | wc -l
  ```

  这个命令首先列出所有进程信息，然后筛选出包含 `process_name` 的行，最后计算这些行的数量。

- **管道与 `sort` 配合使用，排序输出**:

```
cat file.txt | sort
```

将 `file.txt` 文件的内容传递给 `sort`，按字母顺序对内容进行排序。

- **查找文件内容并显示行数**:

```
cat file.txt | grep "pattern" | wc -l
```

查找 `file.txt` 中包含 `pattern` 的行，并显示这些行的数量。

管道符 (`|`) 是 Linux 和 Unix 系统中非常强大的工具，它使得用户能够将多个命令组合在一起，实现复杂的操作。

十二、wc

命令全称

- **英文全称**: `wc` - word count
- **中文解释**: `wc` - 计算文件的行数、字数、字节数等统计信息

基本语法

```
wc [选项] [文件...]
```

常见参数

- `l`：仅显示文件的行数。
- `w`：仅显示文件的单词数。
- `c`：仅显示文件的字节数。
- `m`：仅显示文件的字符数（与字节数略有不同，考虑多字节字符的情况）。
- `L`：显示文件中最长一行的字符数。
- `h`：以易读的格式（如 KB、MB）显示大小（适用于 `c` 和 `w` 等统计选项）。
- `-help`：显示帮助信息，列出所有可用选项。

常见用法

- **查看文件的行数、字数、字节数**:

```
wc file.txt
```

默认情况下，`wc` 会显示 `file.txt` 的行数、单词数和字节数。

- **查看文件的行数**:

```
wc -l file.txt
```

只显示 `file.txt` 的行数。

- **查看文件的单词数**:

```
wc -w file.txt
```

只显示 `file.txt` 中的单词数。

- **查看文件的字节数**:

  ```
  wc -c file.txt
  ```

  只显示 `file.txt` 的字节数。

- **查看文件的字符数**:

  ```
  wc -m file.txt
  ```

  显示 `file.txt` 中的字符数（与字节数不同，考虑到多字节字符）。

- **查看文件中最长行的字符数**:

  ```
  wc -L file.txt
  ```

  显示 `file.txt` 中最长一行的字符数。

- **查看多个文件的统计信息**:

  ```
  wc file1.txt file2.txt
  ```

  显示 `file1.txt` 和 `file2.txt` 的行数、单词数、字节数等信息。

- **结合管道统计命令输出**:

  ```
  cat file.txt | wc -l
  ```

  使用管道统计 `file.txt` 中的行数。

- **以易读的格式显示文件大小**（适用于字节数统计）：

  ```
  wc -c -h file.txt
  ```

  显示文件的字节数，并将输出转换为易读的格式（例如 KB、MB）。

十三、which

命令全称

- **英文全称**: `which` - locate a command
- **中文解释**: `which` - 查找命令的位置

基本语法

```
which [选项] 命令
```

常见参数

- `-help`：显示帮助信息，列出所有可用选项。
- `a`：显示所有匹配的命令路径，而不仅仅是第一个匹配项。
- `-show-dot`：在显示路径时，如果命令是从当前目录执行的，显示 `./`。

常见用法

- **查找命令的路径**:

  ```
  which ls
  ```

  查找并显示 `ls` 命令的路径。例如，可能输出 `/bin/ls`。

- **查找多个命令的路径**:

  ```
  which ls pwd cat
  ```

  查找并显示 `ls`、`pwd` 和 `cat` 命令的路径。

- **显示所有匹配的命令路径**:

  ```
  which -a python
  ```

  查找所有匹配 `python` 命令的路径，并显示所有结果。

- **显示命令的完整路径（如果命令存在于当前目录）**:

  ```
  which --show-dot ./my_script.sh
  ```

  查找当前目录下的 `my_script.sh` 命令，并显示路径。

十四、find

命令全称

- **英文全称**: `find` - search for files in a directory hierarchy
- **中文解释**: `find` - 在目录层次中查找文件

基本语法

```
find [路径] [选项] [表达式]
```

常见参数

- `name`：按文件名匹配，支持通配符。
- `iname`：按文件名匹配，忽略大小写。
- ```
  type
  ```

  ：根据文件类型匹配，常见类型有：

  - `f`：普通文件
  - `d`：目录
  - `l`：符号链接
- `size`：根据文件大小查找。例如，`+10M` 查找大于 10MB 的文件，`100k` 查找小于 100KB 的文件。
- `mtime`：按修改时间查找文件。例如，`+7` 查找 7 天前修改过的文件，`30` 查找 30 天内修改过的文件，`0` 查找当天修改的文件。
  ```

- `atime`：按访问时间查找文件，类似于 `mtime`。
- `ctime`：按状态改变时间查找文件，类似于 `mtime`。
- `exec`：对查找到的文件执行指定命令。例如，`exec rm {} \\;` 删除找到的文件。
- `print`：显示找到的文件路径（默认行为）。
- `maxdepth`：限制查找的目录层次深度。
- `mindepth`：指定查找的最小目录深度。
- `empty`：查找空文件或空目录。
- `user`：查找指定用户的文件。
- `group`：查找属于指定组的文件。
- `perm`：按文件权限查找文件。

常见用法

- **查找某个目录下的所有文件**：

  ```
  find /path/to/directory
  ```

  查找 `/path/to/directory` 目录及其子目录中的所有文件。

- **根据文件名查找文件**：

  ```
  find /path/to/directory -name "pattern"
  ```

  查找 `/path/to/directory` 目录中所有匹配 `pattern` 的文件。例如，`"*.txt"` 查找所有 `.txt` 文件。

- **查找文件时忽略大小写**：

  ```
  find /path/to/directory -iname "pattern"
  ```

  查找 `/path/to/directory` 中所有忽略大小写的匹配 `pattern` 的文件。

- **查找指定类型的文件（例如目录）**：

  ```
  find /path/to/directory -type d
  ```

  查找 `/path/to/directory` 中所有目录（`-type f` 查找普通文件，`-type l` 查找符号链接）。

- **查找指定大小范围的文件**：

  ```
  find /path/to/directory -size +10M
  ```

  查找大于 10MB 的文件。

- **查找最近修改的文件（例如，7 天内修改过的文件）**：

  ```
  find /path/to/directory -mtime -7
  ```

  查找 7 天内修改过的文件。

- **查找并删除特定类型的文件**：

```
find /path/to/directory -type f -name "*.log" -exec rm {} \\;
```

查找并删除 `/path/to/directory` 目录下所有扩展名为 `.log` 的文件。

- **查找空文件或空目录**:

```
find /path/to/directory -empty
```

查找 `/path/to/directory` 目录下的所有空文件或空目录。

- **限制查找的目录深度**:

```
find /path/to/directory -maxdepth 2
```

限制查找最多只遍历 `/path/to/directory` 目录的两级子目录。

- **查找指定用户的文件**:

```
find /path/to/directory -user username
```

查找 `/path/to/directory` 目录中属于用户 `username` 的文件。

- **根据文件权限查找文件**:

```
find /path/to/directory -perm 644
```

查找 `/path/to/directory` 目录中权限为 `644` 的文件。

- **查找文件并列出其路径**:

```
find /path/to/directory -print
```

显示 `/path/to/directory` 目录下所有匹配条件的文件路径（默认行为）。

十五、echo

命令全称

- **英文全称**: `echo` - display a line of text
- **中文解释**: `echo` - 显示一行文本

基本语法

```
echo [选项] [字符串...]
```

常见参数

- `n`：不在输出的末尾添加换行符。
- `e`：启用转义字符的解释（如 `\\n`、`\\t` 等）。
- `E`：禁用转义字符的解释（默认行为）。
- `-help`：显示帮助信息，列出所有可用选项。

常见用法

- **简单输出文本**：

  ```
  echo "Hello, World!"
  ```

  输出 `Hello, World!`。

- **输出文本并去除末尾的换行符**：

  ```
  echo -n "Hello, World!"
  ```

  输出 `Hello, World!`，并且不添加换行符。

- **使用转义字符输出文本**：

  ```
  echo -e "Hello\\nWorld!"
  ```

  启用转义字符输出，`-e` 使得 `\\n` 被解析为换行符，输出：

  ```
  Hello
  World!
  ```

- **输出包含特殊字符的文本**：

  ```
  echo -e "Tab\\tSpace"
  ```

  启用转义字符输出，`-e` 使得 `\\t` 被解析为制表符，输出：

  ```
  Tab    Space
  ```

- **输出带有变量的文本**：

  ```
  name="Alice"
  echo "Hello, $name!"
  ```

  输出 `Hello, Alice!`，其中变量 `$name` 被替换为其值。

- **将文本输出到文件**：

  ```
  echo "Hello, World!" > file.txt
  ```

  将文本 `Hello, World!` 写入到 `file.txt` 文件中，覆盖文件内容。

- **将文本追加到文件末尾**：

  ```
  echo "Hello again!" >> file.txt
  ```

  将文本 `Hello again!` 追加到 `file.txt` 文件的末尾。

- **输出多行文本**：

  ```
  echo -e "Line 1\\nLine 2\\nLine 3"
  ```

输出：

```
Line 1
Line 2
Line 3
```

- **输出带有环境变量的文本**：

```
echo "Home directory: $HOME"
```

输出当前用户的 home 目录路径。

- **禁用转义字符**：

```
echo -E "Hello\\nWorld!"
```

不启用转义字符，输出：

```
Hello\\nWorld!
```

十六、重定向符号

1. 输出重定向 (`>` 和 `>>`)

- `>`

将命令的标准输出（stdout）写入到指定文件中，如果文件已存在，会覆盖该文件。

  - **语法**：

```
command > filename
```

  - **示例**：将 `Hello, World!` 输出到 `output.txt` 文件中，若文件已存在则覆盖。

```
echo "Hello, World!" > output.txt
```

- `>>`

将命令的标准输出（stdout）追加到指定文件末尾。如果文件不存在，则创建新文件。

  - **语法**：

```
command >> filename
```

  - **示例**：将 `New line of text` 追加到 `output.txt` 文件末尾。

```
echo "New line of text" >> output.tx
```

2. 输入重定向 (`<`)

- **

  ```
  <
  ```

  **将文件内容作为命令的标准输入（stdin）。

  - **语法**:

    ```
    command < filenam
    ```

  - **示例**：将 `input.txt` 文件的内容作为输入传递给 `sort` 命令，排序文件内容。

    ```
    sort < input.txt
    ```

十七、tail

命令全称

- **英文全称**: `tail` - output the last part of files
- **中文解释**: `tail` - 输出文件的最后部分

基本语法

```
tail [选项] [文件...]
```

常见参数

- `n NUM`：显示文件的最后 `NUM` 行，默认为 10 行。
- `f`：实时监视文件内容的变化，常用于查看日志文件的实时更新。
- `c NUM`：显示文件的最后 `NUM` 字节。
- `q`：禁止输出文件名，在显示多个文件内容时不显示文件头。
- `v`：始终显示文件名（即使只有一个文件）。
- `-help`：显示帮助信息，列出所有可用选项。

常见用法

- **查看文件的最后 10 行**（默认行为）：

  ```
  tail file.txt
  ```

  显示 `file.txt` 文件的最后 10 行内容。

- **查看文件的最后 N 行**:

  ```
  tail -n 20 file.txt
  ```

  显示 `file.txt` 文件的最后 20 行。

- **查看文件的最后 N 行（不包括换行符）**：

```
tail -n +5 file.txt
```

从第 5 行开始显示文件的内容。

- **实时查看文件的新增内容**:

```
tail -f file.log
```

实时显示 `file.log` 文件的新增内容，常用于监视日志文件。

- **实时查看多个文件的新增内容**:

```
tail -f file1.log file2.log
```

实时显示 `file1.log` 和 `file2.log` 文件的新增内容。

- **查看文件的最后 N 字节**:

```
tail -c 100 file.txt
```

显示 `file.txt` 文件的最后 100 字节内容。

- **实时查看文件的新增内容并显示文件名**:

```
tail -f -v file.log
```

在实时查看 `file.log` 文件的新增内容时，始终显示文件名。

- **查看多个文件的最后部分**:

```
tail -n 10 file1.txt file2.txt
```

显示 `file1.txt` 和 `file2.txt` 文件的最后 10 行内容。

- **结合管道查看命令的最后输出部分**:

```
dmesg | tail -n 20
```

查看 `dmesg` 命令的最后 20 行输出，通常用于查看内核日志或系统日志的最新部分。

- **查看一个文件的最后部分并按字节显示**:

```
tail -c 50 file.txt
```

显示 `file.txt` 文件的最后 50 字节内容。

十八、vim & vi

命令全称

- **英文全称**: `vim` - Vi IMproved
- **中文解释**: `vim` - 改进版的 Vi 编辑器，广泛用于文本编辑，尤其是编程和配置文件编辑。

基本语法

```
vim [选项] [文件名]
```

常见参数

- `y`：只读模式打开文件，类似 `view` 命令。
- `R`：以只读模式打开文件。
- `c`：在打开文件后执行命令。例如，`vim -c 'set number' filename` 打开文件并设置行号。
- `o`：水平分屏打开多个文件。
- `O`：垂直分屏打开多个文件。
- `-version`：显示 `vim` 的版本信息。
- `h`：显示帮助信息。

常见用法

1. 打开文件

- **打开文件**：

  ```
  vim filename
  ```

  打开 `filename` 文件，进入 `vim` 编辑器。

- **以只读模式打开文件**：

  ```
  vim -R filename
  ```

  打开 `filename` 文件并以只读模式打开，不能修改文件内容。

2. `vim` 编辑器模式

`vim` 编辑器有三种基本模式：

- **普通模式**（Normal Mode）：用于浏览和操作文本。
- **插入模式**（Insert Mode）：用于输入文本。
- **命令模式**（Command Mode）：用于执行命令。

3. 在普通模式和插入模式之间切换

- 进入插入模式

  ：

  - 按 `i` 键：在当前光标位置插入文本。
  - 按 `I` 键：在行首插入文本。
  - 按 `a` 键：在当前光标后插入文本。
  - 按 `A` 键：在行尾插入文本。
  - 按 `o` 键：在当前行下方插入新行。
  - 按 `O` 键：在当前行上方插入新行。

- 从插入模式返回普通模式

  ：

  - 按 `Esc` 键：退出插入模式并回到普通模式。

4. 常用命令

- **保存文件**:
  - 在普通模式下，输入 `:w` 并按回车键保存文件。
  - 如果文件是只读文件，输入 `:w!` 强制保存。
- **退出 `vim`**:
  - 输入 `:q` 并按回车键退出。
  - 如果文件有未保存的更改，`vim` 会提示，输入 `:q!` 强制退出不保存更改。
  - 输入 `:wq` 保存文件并退出 `vim`。
- **保存并退出**:

  ```
  :wq
  ```

  保存文件并退出 `vim`。
- **退出不保存**:

  ```
  :q!
  ```

  强制退出 `vim`，不保存任何更改。

5. 查找和替换

- 查找文本

  :
  - 在普通模式下，输入 `/` 后跟要查找的文本，然后按回车键:

    ```
    /search_term
    ```
  - 向下查找文本，可以使用 `n` 跳到下一个匹配项，`N` 跳到上一个匹配项。
- 查找并替换文本

  :
  - 在普通模式下，输入 `:s/old/new/g`，替换当前行中的所有匹配项:

    ```
    :s/old/new/g
    ```
  - 替换整篇文档中的所有匹配项:

    ```
    :%s/old/new/g
    ```

6. 移动光标

- **向上移动一行**：按 `k` 键。
- **向下移动一行**：按 `j` 键。
- **向左移动一个字符**：按 `h` 键。
- **向右移动一个字符**：按 `l` 键。
- **移动到行首**：按 `0` 键。

- **移动到行尾**：按 `$` 键。
- **跳转到指定行**：输入 `:n`，`n` 为行号：

  ```
  :5
  ```

  跳转到第 5 行。
- **跳转到文件开头**：输入 `gg`。
- **跳转到文件结尾**：输入 `G`。

7. 剪切、复制和粘贴

- **复制当前行**：在普通模式下，按 `yy` 键（即 "yank"）。
- **剪切当前行**：按 `dd` 键（即 "delete"）。
- **粘贴内容**：按 `p` 键粘贴到当前行后，或按 `P` 键粘贴到当前行前。

8. 分屏和窗口管理

- **水平分屏**：输入 `:split`，可以同时查看多个文件。
- **垂直分屏**：输入 `:vsplit`，在窗口中创建垂直分屏。
- **切换窗口**：使用 `Ctrl+w` 后跟方向键（如 `h`、`j`、`k`、`l`）切换不同的分屏窗口。

9. 撤销和重做

- **撤销**：按 `u` 键撤销操作。
- **重做**：按 `Ctrl+r` 键重做操作。

10. 配置文件

- **设置行号**：在命令模式下输入：

  ```
  :set number
  ```

- **禁用行号**：输入：

  ```
  :set nonumber
  ```

常见快捷键

- `i`：进入插入模式。
- `Esc`：返回普通模式。
- `:w`：保存文件。
- `:q`：退出编辑器。
- `:wq`：保存并退出编辑器。
- `/pattern`：查找文本。
- `dd`：删除一行。
- `yy`：复制一行。
- `p`：粘贴。

十九、uname

命令全称

- **英文全称**: `uname` - print system information
- **中文解释**: `uname` - 打印系统信息

基本语法

```
uname [选项]
```

常见参数

- `a`：显示所有可用的系统信息（操作系统、内核版本、主机名等）。
- `s`：显示操作系统名称（如 Linux）。
- `n`：显示主机名。
- `r`：显示内核版本。
- `v`：显示内核版本的详细信息。
- `m`：显示机器硬件名称（如 x86_64）。
- `p`：显示处理器架构类型（如果可用）。
- `i`：显示硬件平台信息（如果可用）。
- `o`：显示操作系统名称。

常见用法

- **显示所有系统信息**:

  ```
  uname -a
  ```

  显示操作系统、内核、主机名、硬件平台等信息。例如：

  ```
  Linux myhost 5.4.0-80-generic #90-Ubuntu SMP Fri May 28 14:44:00 UTC 2021
  x86_64 x86_64 x86_64 GNU/Linux
  ```

- **查看操作系统名称**:

  ```
  uname -s
  ```

  输出操作系统名称，如 `Linux`。

- **查看内核版本**:

  ```
  uname -r
  ```

  输出内核的版本号，如 `5.4.0-80-generic`。

- **查看主机名**:

  ```
  uname -n
  ```

  输出主机名，如 `myhost`。

- **查看硬件架构**:

```
uname -m
```

输出机器的硬件架构，如 `x86_64`。

- **查看处理器架构**:

```
uname -p
```

输出处理器类型（如果可用），如 `x86_64`。

- **查看操作系统类型**:

```
uname -o
```

输出操作系统类型，如 `GNU/Linux`。

`uname` 是一个非常有用的命令，可以帮助用户快速了解系统的基本信息，尤其在调试和系统管理中非常常见。

二十、hostname

命令全称

- **英文全称**: `hostname` - show or set the system's hostname
- **中文解释**: `hostname` - 显示或设置系统的主机名

基本语法

```
hostname [选项] [新主机名]
```

常见参数

- `a`：显示主机的别名（如果有的话）。
- `d`：显示主机的域名部分（DNS 后缀）。
- `f`：显示完全合格的域名（FQDN），包括主机名和域名。
- `i`：显示主机的 IP 地址。
- `I`：显示所有与主机相关的 IP 地址（包括多网卡的 IP 地址）。
- `s`：显示主机名的短版本（不包含域名）。
- `v`：显示主机名的版本信息。
- `-help`：显示帮助信息，列出所有可用选项。

常见用法

- **查看当前主机名**:

```
hostname
```

输出当前系统的主机名（短名称），例如:

```
myhos
```

- **查看完整的主机名（包括域名）**:

```
hostname -f
```

输出主机的完全合格域名（FQDN），如：

```
myhost.example.com
```

- **查看主机的 IP 地址**：

```
hostname -i
```

显示当前主机的 IP 地址，例如：

```
192.168.1.100
```

- **查看主机的所有 IP 地址**：

```
hostname -I
```

显示与主机相关的所有 IP 地址，适用于多网卡的情况，输出示例：

```
192.168.1.100 192.168.2.10
```

- **查看主机的域名部分**：

```
hostname -d
```

显示主机的 DNS 域名部分，例如：

```
example.com
```

- **修改主机名**（临时生效，重启后失效）：

```
sudo hostname newhostname
```

临时将主机名更改为 `newhostname`。这种修改不会在系统重启后保留。

- **永久修改主机名**： 永久修改主机名通常需要修改系统的配置文件：
  - **在 `/etc/hostname` 文件中修改主机名**：

```
sudo nano /etc/hostname
```

编辑文件内容，将其中的主机名修改为新的名称。

  - **在 `/etc/hosts` 文件中修改主机名的映射**：

```
sudo nano /etc/hosts
```

修改文件中的主机名映射行，通常是将当前主机名与 `127.0.0.1` 和 `::1` 映射：

```
127.0.0.1   localhost newhostname
::1         localhost newhostname
```

修改这些文件后，重新启动系统或使用 `sudo systemctl restart systemd-hostnamed` 使更改生效。

`hostname` 命令通常用于查询和设置系统的主机名，对于管理和配置网络、服务器以及域名解析非常重要。

二十一、cat /proc/cpuinfo

查看cpu信息

二十二、lsmod

命令全称

- **英文全称**: `lsmod` - list loaded modules
- **中文解释**: `lsmod` - 列出当前加载的内核模块

基本语法

```
lsmod
```

常见参数

- `h`：显示帮助信息，列出所有可用选项。
- `-help`：同 `h`，显示帮助信息。

常见用法

- **列出当前加载的内核模块**:

  ```
  lsmod
  ```

  输出当前系统中加载的所有内核模块，列出的内容通常包括：

  - **模块名**（Module）：模块的名称。
  - **大小**（Size）：模块的大小。
  - **使用计数**（Used by）：该模块被多少其他模块或进程使用。
  - **依赖关系**（Dependencies）：显示该模块的依赖模块。

  示例输出：

  ```
  Module                  Size  Used by
  nvidia              12345678  32
  i2c_core              123456  1 nvidia
  ```

- **查看特定模块的状态**：`lsmod` 默认会列出所有加载的模块。如果你只关心某个特定模块，可以结合 `grep` 命令使用：

  ```
  lsmod | grep nvidia
  ```

这将显示与 `nvidia` 模块相关的信息，输出可能如下：

```
nvidia              12345678  32
i2c_core            123456  1 nvidia
```

- **检查模块是否已加载**： 你可以通过检查 `lsmod` 的输出或直接使用 `modinfo` 命令来验证某个模块是否已加载。例如，检查 `nvidia` 模块是否已加载：

  ```
  lsmod | grep nvidia
  ```

  或者使用 `modinfo` 命令来获取模块的详细信息：

  ```
  modinfo nvidia
  ```

二十三、top

命令全称

- **英文全称**: `top` - task manager for Linux
- **中文解释**: `top` - 实时显示系统中各个进程的状态和系统资源使用情况

基本语法

```
top [选项]
```

常见参数

- `d SECONDS`：设置更新频率，单位为秒（默认 3 秒）。
- `n COUNT`：设置更新的次数，显示多少次更新后退出。
- `u USER`：仅显示指定用户的进程。
- `p PID`：仅显示指定进程 ID 的进程。
- `b`：以批处理模式运行，适用于脚本输出。
- `i`：忽略 idle 和 zombie 进程。
- `H`：显示线程而不是进程。
- `c`：显示命令行中的完整进程信息。
- `s`：启用安全模式，限制某些交互操作。
- `-help`：显示帮助信息。

常见用法

- **查看系统资源和进程**:

  ```
  top
  ```

  打开 `top`，显示系统的资源使用情况，包括 CPU、内存、交换分区等，以及当前所有进程的状态。默认每 3 秒刷新一次。

- **设置更新频率为 1 秒**:

  ```
  top -d 1
  ```

设置 `top` 命令每 1 秒刷新一次，显示系统的实时状态。

- **查看特定用户的进程**：

  ```
  top -u username
  ```

  显示 `username` 用户的进程。

- **查看特定进程 ID 的进程**：

  ```
  top -p 1234
  ```

  仅显示进程 ID 为 `1234` 的进程。

- **批处理模式下输出系统进程信息**：

  ```
  top -b -n 1
  ```

  以批处理模式运行 `top` 命令，`-n 1` 表示运行一次更新后退出。适用于脚本获取系统资源信息。

- **查看线程而非进程**：

  ```
  top -H
  ```

  显示当前进程的线程而不是进程本身。适用于检查多线程应用程序。

- **退出 `top` 命令**：
  - 在 `top` 界面按 `q` 键退出。

`top` 输出字段说明

- **PID**：进程 ID。
- **USER**：进程所属的用户。
- **PR**：进程的优先级。
- **NI**：进程的 nice 值，影响进程的优先级。
- **VIRT**：进程使用的虚拟内存总量。
- **RES**：进程占用的实际物理内存（常驻内存）。
- **SHR**：进程共享内存的大小。
- S
  ：进程状态。常见的状态包括：
  - `R`：正在运行。
  - `S`：处于休眠状态。
  - `Z`：僵尸状态（已终止但尚未被父进程清理）。
  - `T`：停止状态。
  - `I`：空闲。
- **%CPU**：进程使用的 CPU 百分比。
- **%MEM**：进程使用的物理内存百分比。
- **TIME+**：进程已使用的总 CPU 时间。
- **COMMAND**：进程的命令名称。

使用 `top` 调整排序

- 在 `top` 命令界面按下 `P` 键：按 CPU 使用率降序排列进程。
- 按下 `M` 键：按内存使用量降序排列进程。
- 按下 `T` 键：按运行时间排序。

二十四、df

`df`

- **全称**：disk free (磁盘空间使用情况)
- **中文解释**：该命令用于显示文件系统的磁盘空间使用情况，包括每个挂载点的总空间、已用空间、可用空间和挂载的文件系统类型等信息。

基本语法：

```
df [选项] [文件]
```

- 如果不指定文件，`df` 会显示所有挂载文件系统的空间使用情况。
- 如果指定文件，`df` 会显示该文件所在的文件系统的空间使用情况。

常见参数：

- `h`：以人类可读的格式显示空间（例如：KB, MB, GB）。
- `T`：显示文件系统类型。
- `i`：显示 inode 信息（包括 inode 的使用情况）。
- `l`：仅显示本地文件系统，不包括网络文件系统（如 NFS）。
- `-total`：显示所有文件系统的总计使用情况。
- `t <类型>`：仅显示特定类型的文件系统（如 `ext4`、`xfs`）。
- `x <类型>`：排除特定类型的文件系统。

常见用法：

1. **显示所有文件系统的空间使用情况**：

   ```
   df
   ```

2. **以人类可读的格式显示磁盘空间**：

   ```
   df -h
   ```

3. **显示指定目录所在文件系统的空间使用情况**：

   ```
   df /home
   ```

4. **显示 inode 使用情况**：

   ```
   df -i
   ```

5. **显示文件系统类型和磁盘使用情况**：

```
df -T
```

6. **显示特定类型的文件系统（例如，只显示 `ext4` 文件系统）**：

```
df -t ext4
```

7. **显示所有文件系统的总计使用情况**：

```
df --total
```

二十五、fdisk

`fdisk`

- **全称**：fixed disk partitioning (磁盘分区)
- **中文解释**：`fdisk` 是一个用于磁盘分区的命令行工具，允许用户在硬盘上创建、删除、修改分区。它常用于管理 MBR（Master Boot Record）类型的分区表。

基本语法：

```
fdisk [选项] [磁盘设备]
```

- 例如，`fdisk /dev/sda` 用来管理 `/dev/sda` 这个磁盘的分区。

常见参数：

- `l`：列出系统中所有磁盘的分区表。
- `u`：以扇区为单位显示分区大小（默认显示为柱面和头部）。
- `c`：显示分区信息时以兼容的格式显示（某些操作系统或工具需要这种格式）。
- `s <设备>`：显示指定分区的大小（比如：`fdisk -s /dev/sda1`）。
- `v`：显示版本信息。

常见用法：

1. **列出所有磁盘的分区表**：

```
fdisk -l
```

2. **列出指定磁盘（如 `/dev/sda`）的分区表**：

```
fdisk -l /dev/sda
```

3. **进入交互式模式来管理磁盘分区**：

```
fdisk /dev/sda
```

然后你可以在交互模式下执行不同的操作，例如：

- `m`：显示帮助菜单。
- `p`：打印当前磁盘的分区表。
- `n`：创建一个新的分区。

- `d`：删除一个分区。
- `t`：更改分区的类型。
- `w`：保存并退出。

4. **创建新的分区**：在 `fdisk` 的交互式界面中：
   - 输入 `n`，然后选择分区类型（主分区或扩展分区），指定起始和结束扇区。
5. **删除一个分区**：在 `fdisk` 的交互式界面中：
   - 输入 `d`，然后选择要删除的分区编号。
6. **查看磁盘分区的详细信息**：

```
fdisk -v /dev/sda
```

7. **以扇区单位列出分区表**：

```
fdisk -u -l /dev/sda
```

注意事项：

- `fdisk` 主要用于管理传统的 MBR 分区表。如果你要管理 GPT（GUID Partition Table）分区表，推荐使用 `gdisk` 或 `parted`。
- 对磁盘分区的修改（如创建、删除分区）会影响数据，操作时务必小心，确保已有数据有备份。

二十六、free

`free`

- **全称**：free memory (显示内存使用情况)
- **中文解释**：`free` 命令用于显示系统的内存使用情况，包括总内存、已用内存、空闲内存、共享内存、缓存和交换空间的使用情况。它帮助用户了解系统的内存资源的当前状态。

基本语法：

```
free [选项]
```

常见参数：

- `h`：以人类可读的格式显示内存（例如：KB, MB, GB）。
- `b`：以字节为单位显示内存。
- `k`：以千字节（KB）为单位显示内存。
- `m`：以兆字节（MB）为单位显示内存。
- `g`：以千兆字节（GB）为单位显示内存。
- `t`：显示总内存（包括物理内存和交换空间）。
- `s <秒>`：每隔指定的秒数刷新一次内存信息（例如：`free -s 5` 每 5 秒更新一次）。
- `c <次数>`：指定更新次数（例如：`free -s 5 -c 3` 表示每 5 秒更新 3 次）。

常见用法：

1. **显示系统的内存使用情况**：

```
free
```

输出示例：

```
              total        used        free      shared  buff/cache
available
Mem:           7973        2011        1822         142        5140
5583
Swap:          2047           0        2047
```

2. **以人类可读的格式显示内存使用情况**：

```
free -h
```

输出示例：

```
              total        used        free      shared  buff/cache
available
Mem:           7.8G        2.0G        1.8G        142M        5.0G
5.5G
Swap:          2.0G          0B        2.0G
```

3. **显示内存使用情况，并且以 MB 为单位显示**：

```
free -m
```

4. **显示内存的字节级使用情况**：

```
free -b
```

5. **每 5 秒刷新一次内存使用情况**：

```
free -s 5
```

6. **显示内存和交换分区总计的使用情况**：

```
free -t
```

7. **显示内存的详细情况（包括缓存、交换空间等）**：

```
free -h -t
```

输出字段解释：

- **total**：总内存。
- **used**：已用内存（包括缓存和缓冲区）。
- **free**：空闲内存。
- **shared**：与其他进程共享的内存。
- **buff/cache**：被操作系统用作缓存的内存。
- **available**：当前可用的内存（不包括被缓存占用的内存）。

注意事项：

- `free` 显示的是一个瞬时快照，缓存和缓冲区内存可以动态变化，`available` 是实际能用来分配给程序的内存量，比 `free` 更为实际。
- `free` 命令通常用于快速查看内存资源的健康状况，但对于深入分析，可能需要结合其他工具（如 `top` 或 `vmstat`）。

二十七、env

`env`

- **全称**：environment (环境变量)
- **中文解释**：`env` 命令用于显示或设置当前会话的环境变量。它通常用来查看和管理系统中环境变量的值，并且常用于临时修改环境变量以运行某些程序。

基本语法：

```
env [选项] [命令] [参数]
```

- 如果不指定命令，`env` 会列出当前的环境变量。
- 如果指定命令，`env` 会在修改过的环境变量下执行该命令。

常见参数：

- `i`：清空当前环境变量，启动一个干净的环境。
- `u <变量名>`：删除指定的环境变量。
- `v`：显示命令执行时使用的环境变量（会打印变量名和变量值）。
- `0`：以 null 字符分隔输出的环境变量列表（用于脚本处理）。
- `-help`：显示帮助信息。
- `-version`：显示版本信息。

常见用法：

1. **列出当前的环境变量**：

   ```
   env
   ```

2. **列出所有环境变量并且逐行显示**：

   ```
   env -v
   ```

3. **使用特定环境变量执行命令**：

   ```
   env VAR_NAME=value command
   ```

   例如，临时修改 `PATH` 环境变量来运行某个命令：

   ```
   env PATH=/usr/local/bin:$PATH ls
   ```

   这会修改 `PATH` 环境变量并在修改后的环境下执行 `ls` 命令。

4. **清空环境变量并执行命令**：

```
env -i command
```

例如，清空所有环境变量后执行 `bash`，这会启动一个没有任何环境变量的 shell：

```
env -i bash
```

5. **删除指定的环境变量并执行命令**：

```
env -u VAR_NAME command
```

例如，删除 `HOME` 环境变量并执行 `echo` 命令：

```
env -u HOME echo $HOME
```

6. **以 null 字符分隔输出环境变量（适用于脚本处理）**：

```
env -0
```

常见环境变量：

- `PATH`：存放可执行文件的目录路径。
- `HOME`：当前用户的家目录。
- `USER`：当前用户的用户名。
- `SHELL`：当前用户使用的 shell 类型（如 `/bin/bash`）。
- `LANG`：系统语言设置。
- `PWD`：当前工作目录。

注意事项：

- `env` 命令在许多脚本和命令行操作中非常有用，尤其是当需要临时更改环境变量以运行特定的程序时。
- 通过 `env` 可以非常方便地查看当前环境中的变量配置。

二十八、uptime

`uptime`

- **全称**：uptime (系统运行时间)
- **中文解释**：`uptime` 命令用于显示系统的运行时间以及一些系统负载信息。它显示自系统启动以来的运行时间、当前时间、系统负载（1分钟、5分钟、15分钟的平均负载）以及登录系统的用户数量。

基本语法：

```
uptime [选项]
```

常见参数：

- `p`：以更简洁的方式显示系统运行时间（例如：`up 5 days, 2 hours`）。
- `s`：显示系统启动的时间（即系统最后一次重启的时间）。
- `h`：显示帮助信息。

二十九、shutdown

```
shutdown
```

- **全称**：shutdown (关机或重启系统)
- **中文解释**：`shutdown` 命令用于安全地关闭、重启或重新启动 Linux 系统。它可以安排定时关机、重启，并且允许用户发送通知给其他用户或进程。

基本语法：

```
shutdown [选项] [时间] [消息]
```

- **选项**：指定执行操作的类型（关机、重启等）。
- **时间**：指定关机或重启的时间（可以使用绝对时间或相对时间）。
- **消息**：向系统中所有用户广播通知消息。

常见参数：

- `h`：关闭系统并关机（halt）。
- `r`：重新启动系统（reboot）。
- `P`：指定关机时关闭电源（对于支持此操作的硬件）。
- `c`：取消正在进行的关机或重启操作。
- `t <时间>`：设置关机或重启的延迟时间（单位：秒）。
- `f`：强制关机或重启，不进行正常的关闭过程。
- `+<分钟>`：指定延迟的时间（如 `+10` 表示 10 分钟后关机）。
- `<时间>`：指定绝对时间，例如 `22:00` 表示在当天晚上 10 点关机。
- `now`：立即执行关机或重启。

常见用法：

1. **立即关机**：

   ```
   shutdown -h now
   ```

   或者：

   ```
   shutdown now
   ```

   这将立即关闭系统并关机。

2. **立即重启系统**：

   ```
   shutdown -r now
   ```

   这将立即重启系统。

3. **在 10 分钟后关机**：

   ```
   shutdown -h +10
   ```

   这将在 10 分钟后关闭系统。

4. **在 22:00 时关机**：

   ```
   shutdown -h 22:00
   ```

   这将在当天晚上 10 点关机。

5. **取消正在进行的关机或重启操作**：

   ```
   shutdown -c
   ```

   这将取消当前安排的关机或重启。

6. **设置关机并发送通知消息给所有用户**：

   ```
   shutdown -h +5 "The system is going down for maintenance."
   ```

   这将在 5 分钟后关闭系统，并向所有登录的用户广播消息：“The system is going down for maintenance.”

7. **强制重启系统（跳过正常关闭过程）**：

   ```
   shutdown -r -f
   ```

   强制立即重启系统，跳过正常的关闭进程。

三十、reboot

`reboot`

- **全称**：reboot (重启系统)
- **中文解释**：`reboot` 命令用于重新启动 Linux 系统。它可以立即重启计算机，并且类似于 `shutdown` 命令中的 `r` 参数。

基本语法：

```
reboot [选项]
```

常见参数：

- `f`：强制重启，跳过正常的关机过程。
- `i`：显示信息模式，显示系统重新启动的信息。
- `p`：关闭系统并关闭电源（仅适用于支持此功能的硬件）。
- `-help`：显示帮助信息。
- `-version`：显示版本信息。

常见用法：

1. **立即重启系统**：

   ```
   reboot
   ```

   该命令将立即重启系统。

2. **强制重启系统（跳过正常关闭过程）**：

   ```
   reboot -f
   ```

   该命令将强制重启系统，不进行正常的系统关闭程序。

3. **显示重启信息**：

   ```
   reboot -i
   ```

   该命令将显示系统重启的信息。

4. **关闭并重启系统（关闭电源）**：

   ```
   reboot -p
   ```

   该命令将在支持的硬件上关机并关闭电源。

三十一、mount

`mount`

- **全称**：mount (挂载文件系统)
- **中文解释**：`mount` 命令用于将存储设备（如硬盘、U盘、CD/DVD 驱动器等）或远程文件系统（如 NFS、SMB）挂载到文件系统的某个目录。挂载操作使得用户能够访问存储设备中的文件数据。

基本语法：

```
mount [选项] [设备] [挂载点]
```

- **设备**：指定要挂载的设备或文件系统。
- **挂载点**：指定挂载设备的目录路径，设备的内容会被挂载到该目录下。

常见参数：

- `a`：挂载 `/etc/fstab` 文件中所有未挂载的文件系统。
- `t <文件系统类型>`：指定文件系统的类型（如 `ext4`、`ntfs`、`vfat`、`nfs` 等）。
- `o <选项>`：设置挂载选项，多个选项使用逗号分隔（如 `ro`、`rw`、`noexec` 等）。
- `v`：显示详细的挂载过程。
- `r`：以只读模式挂载文件系统。
- `w`：以读写模式挂载文件系统（默认）。
- `-bind`：将目录绑定到另一个目录，允许多个目录共享内容。
- `-remount`：重新挂载文件系统（用于更改挂载选项）。
- `h`：显示帮助信息。

常见用法：

1. **挂载设备到指定目录**：

   ```
   mount /dev/sda1 /mnt
   ```

   这将挂载 `/dev/sda1`（例如第一块硬盘的第一个分区）到 `/mnt` 目录。

2. **挂载指定类型的文件系统**：

   ```
   mount -t ext4 /dev/sda1 /mnt
   ```

   指定挂载的文件系统类型为 `ext4`，并将 `/dev/sda1` 挂载到 `/mnt` 目录。

3. **以只读模式挂载设备**：

   ```
   mount -o ro /dev/sda1 /mnt
   ```

   以只读模式挂载设备 `/dev/sda1` 到 `/mnt` 目录。

4. **挂载一个 ISO 文件（如虚拟光驱）**：

   ```
   mount -o loop /path/to/file.iso /mnt
   ```

   通过 `loop` 设备挂载 ISO 文件到 `/mnt` 目录。

5. **挂载所有文件系统**：

   ```
   mount -a
   ```

   挂载 `/etc/fstab` 文件中所有配置的文件系统。

6. **重新挂载文件系统并更改选项**：

   ```
   mount --remount -o rw /mnt
   ```

   将 `/mnt` 目录重新挂载为读写模式。

7. **将目录绑定到另一个目录**：

   ```
   mount --bind /dir1 /dir2
   ```

   将 `/dir1` 挂载到 `/dir2`，使得这两个目录内容相同。

三十二、unmount

`umount`

- **全称**：umount (卸载文件系统)
- **中文解释**：`umount` 命令用于卸载已挂载的文件系统。卸载操作会将指定的文件系统从当前目录结构中移除，释放相应的资源。执行此命令时，系统会确保所有文件系统上的数据都已写入磁盘，且不再有进程使用该文件系统。

基本语法：

```
umount [选项] [设备|挂载点]
```

- **设备**：指定要卸载的设备（如 `/dev/sda1`）。
- **挂载点**：指定要卸载的挂载点目录（如 `/mnt`）。

常见参数：

- `a`：卸载 `/etc/mtab` 文件中列出的所有文件系统。
- `f`：强制卸载，即使文件系统被占用（慎用，可能会导致数据丢失）。
- `l`：延迟卸载，标记设备为待卸载，待设备不再被使用时才会真正卸载。
- `v`：显示详细的卸载过程。
- `h`：显示帮助信息。

常见用法：

1. **卸载指定设备**：

   ```
   umount /dev/sda1
   ```

   卸载 `/dev/sda1` 设备。

2. **卸载指定挂载点**：

   ```
   umount /mnt
   ```

   卸载挂载在 `/mnt` 目录上的文件系统。

3. **卸载所有挂载的文件系统**：

   ```
   umount -a
   ```

   卸载 `/etc/mtab` 文件中列出的所有文件系统。

4. **强制卸载设备**：

   ```
   umount -f /dev/sda1
   ```

   强制卸载 `/dev/sda1`，即使设备正在被使用。

5. **延迟卸载文件系统**：

   ```
   umount -l /mnt
   ```

   将 `/mnt` 标记为延迟卸载，待其不再被使用时才会实际卸载。

6. **显示详细的卸载过程**：

   ```
   umount -v /dev/sda1
   ```

   卸载 `/dev/sda1` 时显示详细的过程信息。

三十三、su

su

- **全称**：su (Switch User 或者 Substitue User)
- **中文解释**：`su` 命令用于切换到其他用户的身份，默认情况下，它切换到超级用户（root）。该命令可以用于获取另一个用户的权限来执行特定任务。

基本语法：

```
su [选项] [用户名]
```

- **用户名**：指定要切换到的目标用户（如果省略，默认切换到 root 用户）。
- **选项**：控制命令行为的选项。

常见参数：

- `` ` 或-login：模拟完整的登录，切换到目标用户时加载目标用户的环境变量（如.bash_profile 或.bashrc`）。
- `c`：在目标用户下执行指定的命令并返回。
- `s <shell>`：指定使用的 shell 程序，而不是默认的 shell。
- `p`：保留当前环境变量不变，仅改变用户身份。
- `-help`：显示帮助信息。

常见用法：

1. **切换到 root 用户**：

   ```
   su
   ```

   以 root 用户身份切换到系统。执行该命令后，系统会提示输入 root 用户的密码。

2. **切换到其他用户**：

   ```
   su username
   ```

   切换到指定的用户 `username`，执行该命令后，系统会要求输入该用户的密码。

3. **模拟完整登录切换到 root 用户**：

   ```
   su -
   ```

   切换到 root 用户并模拟登录环境，加载 root 用户的环境变量，使得所有的登录配置生效。

4. **以 root 用户身份执行单个命令**：

   ```
   su -c "command"
   ```

   以 root 用户身份执行指定的命令。例如：

   ```
   su -c "ls /root"
   ```

   以 root 身份列出 `/root` 目录的内容。

5. **切换到其他用户并指定 shell**：

   ```
   su -s /bin/bash username
   ```

   切换到 `username` 用户，并指定使用 `/bin/bash` 作为 shell。

6. **保留当前环境变量并切换到 root 用户**：

   ```
   su -p
   ```

   切换到 root 用户，但保留当前环境变量不变。

三十四、sudo

sudo

- **全称**：sudo (SuperUser Do)
- **中文解释**：`sudo` 命令允许普通用户以超级用户（root）或其他用户的身份执行命令，前提是该用户在 `/etc/sudoers` 配置文件中具有相应权限。`sudo` 可以帮助管理员授权特定用户执行特权操作，而无需给他们完全的 root 权限。

基本语法：

```
sudo [选项] <命令> [参数]
```

- **命令**：指定要执行的命令。
- **选项**：用于修改 `sudo` 的行为。

常见参数：

- `u <用户名>`：以指定用户身份执行命令，默认是 root 用户。
- `i`：模拟完整的登录环境，执行命令时加载目标用户的环境变量。
- `s`：启动一个新的 shell 会话，以目标用户身份执行。
- `k`：强制清除 `sudo` 缓存的密码，要求下次使用时重新输入密码。
- `v`：验证 `sudo` 会话是否仍然有效，不执行任何命令。
- `l`：列出当前用户的 sudo 权限。
- `-help`：显示帮助信息。
- `-version`：显示 `sudo` 的版本信息。

三十五、who

who

- **全称**：who (显示当前登录系统的用户)
- **中文解释**：`who` 命令用于显示当前登录到系统的用户信息。它可以显示用户的登录名称、终端、登录时间等信息，帮助管理员了解哪些用户正在使用系统。

基本语法：

```
who [选项] [文件]
```

- **文件**：指定一个文件作为输入，通常是 `/var/run/utmp` 文件，默认情况下会读取该文件。
- **选项**：控制输出的格式或内容。

常见参数：

- `a`：显示所有可用的信息，等效于 `b -d -l -p -r -t -T -u` 的组合。
- `b`：显示系统最后启动的时间。
- `d`：显示系统的死锁信息（如果有）。
- `H`：显示列标题。
- `l`：列出所有登录的用户的名称。
- `p`：显示用户的 PID（进程ID）。
- `r`：显示当前运行级别。
- `T`：显示用户的终端类型。
- `u`：显示用户的登录时间和活动状态。
- `-help`：显示帮助信息。

常见用法：

1. **显示当前登录的所有用户**：

   ```
   who
   ```

   显示系统上所有当前登录的用户信息，包括用户名、登录时间、终端、登录来源等。

2. **显示系统最后启动的时间**：

   ```
   who -b
   ```

   显示系统最后一次启动的时间。

3. **显示所有登录的用户（包括没有活动的用户）**：

   ```
   who -a
   ```

   显示所有可用的信息，包括系统启动时间、登录用户、死锁信息等。

4. **显示带有标题的用户信息**：

   ```
   who -H
   ```

   在输出中显示列标题，方便查看各列信息的含义。

5. **显示指定用户的登录信息**：

   ```
   who username
   ```

   显示指定用户（`username`）的登录信息，如果该用户当前登录系统，则会显示相应的信息。

6. **显示当前用户的登录信息**：

```
who am i
```

显示当前用户的登录信息，通常用于显示当前用户在当前终端的活动状态。

7. **显示系统的运行级别**：

```
who -r
```

显示当前系统的运行级别，通常在 `init` 系统下使用。

三十六、ssh

ssh

- **全称**：SSH (Secure Shell)
- **中文解释**：`ssh` 命令用于通过网络安全地登录到远程计算机系统，并执行命令。它通过加密通信保护数据传输的安全，常用于远程管理服务器或在不同计算机之间进行安全通信。

基本语法：

```
ssh [选项] [用户名@]主机 [命令]
```

- **用户名**：指定登录到远程主机的用户名（如果省略，默认使用当前用户）。
- **主机**：目标主机的 IP 地址或域名。
- **命令**：要在远程主机上执行的命令（如果省略，则进入远程主机的 shell）。

常见参数：

- `p <port>`：指定远程主机的端口号，默认为 22。
- `i <文件>`：指定使用的私钥文件进行身份验证。
- `v`：启用详细模式，显示调试信息。
- `X`：启用 X11 转发，允许在远程主机上运行图形界面程序，并将其显示在本地主机上。
- `C`：启用压缩，减少数据传输量，适用于低带宽连接。
- `T`：禁用伪终端分配，适用于不需要交互的命令执行。
- `L <local_port>:<remote_host>:<remote_port>`：本地端口转发，将本地端口转发到远程主机的指定端口。
- `R <remote_port>:<local_host>:<local_port>`：远程端口转发，将远程主机的端口转发到本地指定端口。
- `o <option>`：指定其他 SSH 配置选项，如 `StrictHostKeyChecking=no` 禁止主机密钥检查。
- `A`：启用代理转发，允许使用本地 SSH 代理进行身份验证。

常见用法：

1. **基本远程登录**：

```
ssh username@hostname
```

以 `username` 用户身份登录到远程主机 `hostname`（IP 地址或域名）。系统会提示输入密码进行身份验证。

2. **指定端口号登录**：

```
ssh -p 2222 username@hostname
```

使用端口号 `2222` 连接到远程主机 `hostname`。

3. **使用 SSH 密钥进行身份验证**：

```
ssh -i /path/to/private_key username@hostname
```

使用指定的私钥文件 `/path/to/private_key` 进行身份验证登录到远程主机。

4. **远程执行命令**：

```
ssh username@hostname "command"
```

在远程主机上执行指定的命令。例如：

```
ssh username@hostname "ls /home"
```

在远程主机的 `/home` 目录下列出文件。

5. **启用 X11 转发（图形界面）**：

```
ssh -X username@hostname
```

启用 X11 转发，使得你可以在远程主机上运行图形界面程序并显示在本地主机上。

6. **使用代理转发**：

```
ssh -A username@hostname
```

启用 SSH 代理转发，允许在远程主机上使用本地 SSH 密钥进行身份验证。

7. **本地端口转发**：

```
ssh -L 8080:localhost:80 username@hostname
```

将本地主机的 8080 端口转发到远程主机的 80 端口。访问本地主机的 `localhost:8080` 相当于访问远程主机的 `localhost:80`。

8. **远程端口转发**：

```
ssh -R 9090:localhost:8080 username@hostname
```

将远程主机的 9090 端口转发到本地主机的 8080 端口。访问远程主机的 `localhost:9090` 相当于访问本地主机的 `localhost:8080`。

9. **列出远程主机的 SSH 配置信息**：

```
ssh -v username@hostname
```

启用详细调试信息，帮助分析连接过程中出现的问题。

三十七、useradd

`useradd`

- **全称**：useradd (添加新用户)
- **中文解释**：`useradd` 命令用于在 Linux 系统中创建一个新的用户账户。通过该命令，可以为用户指定用户名、主目录、shell 等基本信息。

基本语法：

```
useradd [选项] 用户名
```

- **用户名**：要创建的用户名。

常见参数：

- `d <目录>`：指定用户的主目录。
- `m`：如果没有指定主目录，则自动为用户创建主目录。
- `s <shell>`：指定用户的默认 shell（如 `/bin/bash`）。
- `g <组名>`：指定用户的初始用户组。
- `G <组1,组2,...>`：指定用户的附加组。
- `p <密码>`：为用户设置加密后的密码（需要预先加密）。
- `e <日期>`：设置用户账户的过期日期，格式为 `YYYY-MM-DD`。
- `f <天数>`：设置密码过期后的宽限天数。
- `u <UID>`：指定用户的 UID。
- `r`：创建一个系统用户。
- `c <注释>`：为用户指定描述信息（例如，用户的全名）。
- `h`：显示帮助信息。

常见用法：

1. **创建一个基本的用户账户**：

   ```
   useradd username
   ```

2. **创建用户并指定主目录**：

   ```
   useradd -d /custom/home/dir username
   ```

3. **创建用户并指定默认 shell**：

   ```
   useradd -s /bin/bash username
   ```

4. **为用户设置初始密码**：

   ```
   useradd -p $(echo "password" | openssl passwd -1 -stdin) username
   ```

5. **创建用户并加入附加组**：

   ```
   useradd -G group1,group2 username
   ```

6. **创建一个用户并指定 UID**：

```
useradd -u 1500 username
```

7. **为用户添加描述信息**：

```
useradd -c "John Doe, Developer" username
```

8. **创建一个系统用户**：

```
useradd -r -s /usr/sbin/nologin username
```

9. **为用户设置账户过期日期**：

```
useradd -e 2024-12-31 username
```

三十八、userdel

`userdel`

- **全称**：userdel (删除用户)
- **中文解释**：`userdel` 命令用于删除系统中的用户账户。通过该命令，可以删除用户账号及其相关的用户信息。如果指定了额外的选项，还可以删除与该用户关联的主目录及其他文件。

基本语法：

```
userdel [选项] 用户名
```

- **用户名**：要删除的用户账户名称。

常见参数：

- `r`：删除用户时，同时删除该用户的主目录及其所有文件。
- `f`：强制删除用户。如果用户当前有正在运行的进程或文件被该用户占用，`f` 可以强制删除用户，即使用户未完全注销。
- `-help`：显示帮助信息。

常见用法：

1. **删除用户账户**：

```
userdel username
```

删除用户 `username`，但是不删除该用户的主目录和文件。

2. **删除用户及其主目录**：

```
userdel -r username
```

删除用户 `username` 并删除该用户的主目录以及所有与该用户相关的文件。

3. **强制删除用户**：

```
userdel -f username
```

强制删除用户 `username`，即使该用户当前有进程在运行或文件被占用。

4. **删除用户但保留主目录**：

```
userdel -r username
```

删除用户 `username` 并保留用户的主目录文件。如果不使用 `-r`，则主目录不会被删除。

三十九、usermod

`usermod`

- **全称**：usermod (修改用户信息)
- **中文解释**：`usermod` 命令用于修改现有用户的属性，例如用户名、用户的主目录、默认 shell 等。管理员可以使用此命令来更新或调整用户的账户设置。

基本语法：

```
usermod [选项] 用户名
```

- **用户名**：要修改的用户账户名称。

常见参数：

- `d <目录>`：修改用户的主目录。
- `m`：将用户的现有主目录内容迁移到新的主目录。
- `s <shell>`：修改用户的默认 shell（如 `/bin/bash`）。
- `g <组名>`：修改用户的主组。
- `G <组1,组2,...>`：修改用户的附加组。
- `u <UID>`：修改用户的用户ID（UID）。
- `e <日期>`：设置用户账户的过期日期，格式为 `YYYY-MM-DD`。
- `f <天数>`：设置密码过期后的宽限天数，超过后用户无法登录。
- `c <注释>`：修改用户的描述信息（如用户全名）。
- `l <新用户名>`：修改用户名。
- `a`：与 `G` 参数一起使用，追加用户到附加组而不删除现有附加组。
- `p <密码>`：设置用户的密码（已加密密码）。
- `L`：锁定用户账户，禁止该用户登录。
- `U`：解锁用户账户，允许该用户登录。
- `-help`：显示帮助信息。

常见用法：

1. **修改用户的主目录**：

```
usermod -d /new/home/directory username
```

修改用户 `username` 的主目录为 `/new/home/directory`。

2. **移动用户的主目录并修改**：

```
usermod -d /new/home/directory -m username
```

修改用户 `username` 的主目录为 `/new/home/directory` 并将旧主目录的内容迁移到新目录。

3. **修改用户的默认 shell**:

```
usermod -s /bin/zsh username
```

将用户 `username` 的默认 shell 修改为 `/bin/zsh`。

4. **修改用户的主组**:

```
usermod -g newgroup username
```

将用户 `username` 的主组修改为 `newgroup`。

5. **将用户添加到附加组**:

```
usermod -G group1,group2 username
```

将用户 `username` 添加到 `group1` 和 `group2` 附加组中。

6. **修改用户的用户名**:

```
usermod -l newusername oldusername
```

将用户 `oldusername` 的用户名修改为 `newusername`。

7. **设置用户密码过期时间**:

```
usermod -e 2024-12-31 username
```

设置用户 `username` 的账户在 `2024-12-31` 过期。

8. **锁定用户账户**:

```
usermod -L username
```

锁定用户 `username` 的账户，禁止其登录。

9. **解锁用户账户**:

```
usermod -U username
```

解锁用户 `username` 的账户，允许其重新登录。

10. **修改用户描述信息**:

```
usermod -c "John Doe, Developer" username
```

为用户 `username` 设置描述信息为 "John Doe, Developer"。

四十、groupadd

英文全称

Group Add

中文解释

添加新用户组

基本语法

```
groupadd [选项] 用户组名
```

常见参数

- `g GID`：指定用户组的组 ID (GID)。
- `f`：如果组已存在，静默退出，不报错。
- `r`：创建一个系统用户组（组 ID 小于 1000）。

常见用法

1. 创建一个普通用户组：

   ```
   groupadd developers
   ```

2. 创建一个指定 GID 的用户组：

   ```
   groupadd -g 1001 admins
   ```

3. 创建一个系统用户组：

   ```
   groupadd -r sysgroup
   ```

四十一、groupdel

英文全称

Group Delete

中文解释

删除用户组

基本语法

```
groupdel 用户组名
```

常见参数

无额外参数，仅需指定用户组名。

常见用法

1. 删除一个用户组：

```
groupdel developers
```

2. 强制删除不存在用户的用户组（某些系统可能需手动清理关联）：

```
groupdel admins
```

四十二、groupmod

英文全称

Group Modify

中文解释

修改用户组

基本语法

```
groupmod [选项] 用户组名
```

常见参数

- `g GID`：修改用户组的组 ID (GID)。
- `n 新组名`：修改用户组的名称。
- `o`：允许设置重复的组 ID，与 `g` 一起使用。

常见用法

1. 修改用户组的组 ID：

```
groupmod -g 2001 developers
```

2. 修改用户组的名称：

```
groupmod -n devteam developers
```

3. 修改用户组的组 ID 并允许重复：

```
groupmod -g 1001 -o sharedgroup
```

四十三、passwd

英文全称

Password

中文解释

设置或修改用户密码

基本语法

```
passwd [选项] [用户名]
```

常见参数

- `d`：删除用户的密码（用户登录时无需密码）。
- `e`：立即使密码过期，强制用户下次登录时更改密码。
- `l`：锁定用户密码，禁止用户登录。
- `u`：解锁用户密码。
- `S`：显示用户密码的状态信息。

常见用法

1. 修改当前用户的密码：

   ```
   passwd
   ```

2. 修改指定用户的密码：

   ```
   passwd username
   ```

3. 删除用户的密码：

   ```
   passwd -d username
   ```

4. 锁定用户的密码：

   ```
   passwd -l username
   ```

5. 解锁用户的密码：

   ```
   passwd -u username
   ```

6. 查看用户密码状态：

   ```
   passwd -S username
   ```

四十四、last

英文全称

Last Login

中文解释

显示系统的用户登录记录

基本语法

```
last [选项] [用户名] [终端]
```

常见参数

- `n 数字`：显示最近的 N 条登录记录。
- `a`：将主机名或 IP 地址显示在记录的最后一列。
- `x`：显示系统重启和关机信息。
- `f 文件`：指定一个不同的日志文件（默认使用 `/var/log/wtmp`）。
- `R`：不显示主机名或 IP 地址。

常见用法

1. 显示所有用户的登录记录：

   ```
   last
   ```

2. 查看特定用户的登录记录：

   ```
   last username
   ```

3. 查看最近 5 条登录记录：

   ```
   last -n 5
   ```

4. 显示系统重启和关机记录：

   ```
   last -x
   ```

5. 从指定日志文件中读取登录记录：

   ```
   last -f /path/to/logfile
   ```

四十五、crontab

英文全称

Cron Table

中文解释

定时任务表，用于设置周期性任务

基本语法

```
crontab [选项]
```

常见参数

- `e`：编辑当前用户的 crontab 文件。
- `l`：列出当前用户的 crontab 文件内容。
- `r`：删除当前用户的 crontab 文件。
- `u 用户名`：指定用户管理其 crontab（需要管理员权限）。

crontab 时间字段格式

```
*  *  *  *  *  命令
-  -  -  -  -
|  |  |  |  |
|  |  |  |  +---- 星期几 (0-7, 0 和 7 表示周日)
|  |  |  +------- 月份 (1-12)
|  |  +---------- 日期 (1-31)
|  +------------- 小时 (0-23)
+---------------- 分钟 (0-59)
```

常见用法

1. 编辑当前用户的 crontab 文件：

   ```
   crontab -e
   ```

2. 查看当前用户的 crontab 任务：

   ```
   crontab -l
   ```

3. 删除当前用户的 crontab 文件：

   ```
   crontab -r
   ```

4. 添加任务示例：

   每天凌晨 2 点运行备份脚本：

   ```
   0 2 * * * /path/to/backup.sh
   ```

5. 指定用户的 crontab（管理员操作）：

   查看用户名为 `john` 的 crontab：

   ```
   crontab -l -u john
   ```

6. 每小时运行任务：

```
0 * * * * /path/to/task.sh
```

注意

- Cron 服务需处于运行状态，确保启用：

```
sudo systemctl start cron
```

四十六、nohup

英文全称

No Hang Up

中文解释

忽略挂起信号运行命令，使其在用户退出终端后继续运行

基本语法

```
nohup 命令 [参数]
```

常见参数

- **无特定参数**，`nohup` 的作用主要是忽略挂起信号。
- 输出默认保存到 `nohup.out` 文件，若需定向输出，可使用 `>` 明确指定。

常见用法

1. 在后台运行命令并忽略挂起：

```
nohup ./long_running_script.sh &
```

2. 运行命令并将输出保存到指定文件：

```
nohup ./long_running_script.sh > output.log 2>&1 &
```

3. 配合 `&` 后台运行和 `disown` 保持进程独立：

```
nohup python3 script.py &
disown
```

4. 检查后台运行的进程：

```
ps aux | grep long_running_script.sh
```

5. 停止某个 `nohup` 运行的进程：

```
kill -9 <进程ID>
```

注意

- 如果未指定输出文件，`nohup` 会将标准输出和错误输出重定向到当前目录下的 `nohup.out`。
- 常与 `&` 和 `disown` 搭配使用，确保进程不会被终端关闭影响。

四十七、jobs

英文全称

Jobs

中文解释

显示当前终端中的后台任务信息

基本语法

```
jobs [选项]
```

常见参数

- `l`：显示任务的进程 ID（PID）。
- `n`：仅显示状态已更改的任务。
- `p`：仅显示任务的进程 ID（不含其他信息）。
- `r`：仅显示运行中的任务。
- `s`：仅显示已暂停的任务。

常见用法

1. 查看当前终端中的所有后台任务：

   ```
   jobs
   ```

2. 显示任务的进程 ID：

   ```
   jobs -l
   ```

3. 显示运行中的任务：

   ```
   jobs -r
   ```

4. 显示已暂停的任务：

   ```
   jobs -s
   ```

5. 获取任务的进程 ID：

   ```
   jobs -p
   ```

四十八、ps

英文全称

Process Status

中文解释

显示当前系统的进程信息

基本语法

```
ps [选项]
```

常见参数

- `e` 或 `A`：显示所有进程。
- `f`：显示完整格式的信息。
- `u 用户名`：显示指定用户的进程。
- `aux`：显示所有进程（包括其他用户的进程），并包含额外的信息（常用于查看系统资源使用）。
- `p PID`：显示指定 PID 的进程信息。
- `l`：显示长格式的进程列表。

常见用法

1. 显示所有进程：

   ```
   ps -e
   ```

2. 显示当前用户的进程：

   ```
   ps -u username
   ```

3. 显示所有进程并以完整格式显示：

   ```
   ps -ef
   ```

4. 显示指定进程的详细信息：

   ```
   ps -p 1234
   ```

5. 显示所有进程，并包括额外的信息：

   ```
   ps aux
   ```

四十九、kill

英文全称

Kill Process

中文解释

终止指定进程

基本语法

```
kill [选项] PID
```

常见参数

- `9`：强制终止进程，忽略进程的退出请求（发送 `SIGKILL` 信号）。
- `15`：默认信号，终止进程并允许其清理资源（发送 `SIGTERM` 信号）。
- `l`：列出所有信号。
- `s 信号`：指定发送的信号名称或编号。
- `P`：通过进程组来杀死进程。

常见用法

1. 终止指定 PID 的进程：

   ```
   kill 1234
   ```

2. 强制终止进程（忽略退出请求）：

   ```
   kill -9 1234
   ```

3. 列出所有可用的信号：

   ```
   kill -l
   ```

4. 终止进程组中的所有进程：

   ```
   kill -9 -1234
   ```

5. 发送指定信号（如 `SIGSTOP`）：

   ```
   kill -s SIGSTOP 1234
   ```

五十、rpm/yum/apt/apt-get/dpkg

rpm

英文全称

Red Hat Package Manager

中文解释

用于管理 RPM 包（安装、卸载、更新和查询）

基本语法

```
rpm [选项] 包名
```

常见参数

- `i`：安装指定的包。
- `e`：卸载指定的包。
- `U`：升级指定的包。
- `q`：查询包的信息。
- `l`：列出已安装包的文件。
- `v`：显示详细输出。

常见用法

1. 安装 RPM 包：

   ```
   rpm -i package.rpm
   ```

2. 卸载 RPM 包：

   ```
   rpm -e package
   ```

3. 查询已安装的包：

   ```
   rpm -q package
   ```

4. 显示包的文件列表：

   ```
   rpm -ql package
   ```

---

yum

英文全称

Yellowdog Updater, Modified

中文解释

基于 RPM 的包管理工具，广泛用于 CentOS 和 RHEL 系统，支持自动解决依赖关系

基本语法

```
yum [选项] 操作 [包名]
```

常见参数

- `install`：安装指定的包。
- `remove`：卸载指定的包。
- `update`：更新系统包。
- `list`：列出包信息。
- `search`：搜索指定的包。
- `clean`：清理缓存。

常见用法

1. 安装包：

   ```
   yum install package
   ```

2. 卸载包：

   ```
   yum remove package
   ```

3. 更新包：

   ```
   yum update package
   ```

4. 列出可用的包：

   ```
   yum list available
   ```

5. 清理缓存：

   ```
   yum clean all
   ```

---

apt

英文全称

Advanced Package Tool

中文解释

用于基于 Debian 系统的包管理工具，支持安装、升级、卸载和管理包

基本语法

```
apt [选项] 操作
```

常见参数

- **`install`**：安装指定的包。
- **`remove`**：卸载指定的包。
- **`update`**：更新包信息。
- **`upgrade`**：升级已安装的包。
- **`search`**：搜索包。
- **`autoremove`**：卸载不再需要的包。

常见用法

1. 安装包：

   ```
   apt install package
   ```

2. 卸载包：

   ```
   apt remove package
   ```

3. 更新包列表：

   ```
   apt update
   ```

4. 升级所有已安装包：

   ```
   apt upgrade
   ```

5. 搜索包：

   ```
   apt search package
   ```

---

apt-get

英文全称

Advanced Package Tool Get

中文解释

Debian 系统的包管理工具，用于安装、升级、卸载包，功能更丰富

基本语法

```
apt-get [选项] 操作
```

常见参数

- **`install`**：安装指定的包。
- **`remove`**：卸载指定的包。
- **`update`**：更新包信息。
- **`upgrade`**：升级已安装的包。
- **`dist-upgrade`**：升级系统，处理包的依赖变化。
- **`autoremove`**：卸载不再需要的包。

常见用法

1. 安装包：

   ```
   apt-get install package
   ```

2. 卸载包：

   ```
   apt-get remove package
   ```

3. 更新包列表：

   ```
   apt-get update
   ```

4. 升级所有已安装包：

   ```
   apt-get upgrade
   ```

5. 升级系统：

   ```
   apt-get dist-upgrade
   ```

---

dpkg

英文全称

Debian Package

中文解释

用于管理 Debian 包（安装、卸载、查询）

基本语法

```
dpkg [选项] 操作
```

常见参数

- `i`：安装指定的包。
- `r`：卸载指定的包。
- `l`：列出已安装的包。
- `s`：显示包的详细信息。
- `L`：列出已安装包的文件。

常见用法

1. 安装包：

   ```
   dpkg -i package.deb
   ```

2. 卸载包：

   ```
   dpkg -r package
   ```

3. 列出已安装的包：

   ```
   dpkg -l
   ```

4. 显示包的详细信息：

   ```
   dpkg -s package
   ```

5. 列出已安装包的文件：

   ```
   dpkg -L package
   ```

五十一、systemctl

英文全称

System Control

中文解释

用于管理系统服务和管理系统状态的工具，适用于基于 `systemd` 的系统

基本语法

```
systemctl [选项] 操作 [服务名]
```

常见参数

- `start`：启动指定的服务。
- `stop`：停止指定的服务。
- `restart`：重启指定的服务。
- `reload`：重新加载指定的服务配置。
- `status`：查看指定服务的状态。
- `enable`：设置指定服务开机自动启动。
- `disable`：取消指定服务的开机自动启动。
- `is-active`：检查指定服务是否处于活动状态。
- `list-units`：列出所有单元（服务）信息。
- `is-enabled`：检查指定服务是否已设置为开机启动。

常见用法

1. 启动服务：

```
systemctl start service_name
```

2. 停止服务：

```
systemctl stop service_name
```

3. 重启服务：

```
systemctl restart service_name
```

4. 查看服务状态：

```
systemctl status service_name
```

5. 设置服务开机自动启动：

```
systemctl enable service_name
```

6. 取消服务的开机自动启动：

```
systemctl disable service_name
```

7. 检查服务是否正在运行：

```
systemctl is-active service_name
```

8. 列出所有服务：

```
systemctl list-units --type=service
```

五十二、ifconfig

英文全称

Interface Configuration

中文解释

用于显示或配置网络接口的工具

基本语法

```
ifconfig [接口] [选项]
```

常见参数

- `a`：显示所有网络接口（包括未激活的接口）。
- `up`：激活指定的网络接口。
- `down`：禁用指定的网络接口。
- `inet`：显示或配置 IPv4 地址。
- `netmask`：配置子网掩码。
- `broadcast`：配置广播地址。
- `mtu`：设置最大传输单元（MTU）。
- `hw`：设置网络接口的硬件地址（MAC 地址）。
- `add`：添加新的 IP 地址。
- `del`：删除指定的 IP 地址。

常见用法

1. 显示所有网络接口的配置：

   ```
   ifconfig -a
   ```

2. 启动某个网络接口：

   ```
   ifconfig eth0 up
   ```

3. 禁用某个网络接口：

   ```
   ifconfig eth0 down
   ```

4. 查看指定接口的 IP 地址：

   ```
   ifconfig eth0
   ```

5. 为接口配置新的 IP 地址：

   ```
   ifconfig eth0 inet 192.168.1.100 netmask 255.255.255.0
   ```

6. 配置接口的广播地址：

   ```
   ifconfig eth0 broadcast 192.168.1.255
   ```

7. 设置接口的 MTU：

```
ifconfig eth0 mtu 1500
```

五十三、netstat

英文全称

Network Statistics

中文解释

用于显示网络连接、路由表、接口统计信息等网络相关信息的工具

基本语法

```
netstat [选项]
```

常见参数

- `a`：显示所有连接中的套接字，包括监听中的套接字。
- `t`：显示 TCP 连接。
- `u`：显示 UDP 连接。
- `l`：仅显示监听中的套接字。
- `n`：以数字形式显示地址和端口号，避免解析主机名和服务名。
- `p`：显示哪个进程正在使用每个套接字。
- `r`：显示路由表。
- `i`：显示网络接口信息。
- `s`：显示网络统计信息。
- `c`：持续显示输出，每隔一定时间更新一次。

常见用法

1. 显示所有连接和监听套接字：

```
netstat -a
```

2. 显示所有 TCP 连接：

```
netstat -t
```

3. 显示所有 UDP 连接：

```
netstat -u
```

4. 显示监听中的 TCP 和 UDP 套接字：

```
netstat -l
```

5. 显示网络接口统计信息：

```
netstat -i
```

6. 显示进程使用的端口：

```
netstat -p
```

7. 显示路由表：

```
netstat -r
```

8. 显示网络统计信息：

```
netstat -s
```

9. 每隔 5 秒更新一次网络连接状态：

```
netstat -c
```

五十四、iptables

英文全称

IP Tables

中文解释

用于配置 Linux 内核防火墙的工具，主要用于过滤网络流量、设置网络访问规则等

基本语法

```
iptables [选项] [链] [规则]
```

常见参数

- `A`：向指定链添加规则。
- `D`：从指定链删除规则。
- `I`：在指定链的最前面插入规则。
- `L`：列出指定链的所有规则。
- `F`：清除指定链的所有规则。
- `P`：设置链的默认策略（如接受或拒绝）。
- `t`：指定操作的表，如 `filter`、`nat` 等。
- `s`：指定源 IP 地址。
- `d`：指定目标 IP 地址。
- `p`：指定协议（如 `tcp`、`udp`、`icmp`）。
- `-dport`：指定目标端口。
- `j`：指定目标动作（如 `ACCEPT`、`DROP`、`REJECT`）。

常见用法

1. 查看当前防火墙规则：

```
iptables -L
```

2. 添加一条规则，允许来自 `192.168.1.100` 的流量：

```
iptables -A INPUT -s 192.168.1.100 -j ACCEPT
```

3. 删除一条规则：

```
iptables -D INPUT -s 192.168.1.100 -j ACCEPT
```

4. 设置默认策略为拒绝所有进入流量：

```
iptables -P INPUT DROP
```

5. 清除所有规则：

```
iptables -F
```

6. 启用 NAT 转发，进行端口转发：

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination
192.168.1.2:8080
```

7. 设置允许 SSH 连接：

```
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

8. 阻止某个 IP 地址的访问：

```
iptables -A INPUT -s 192.168.1.100 -j DROP
```