1 Linux基础入门和帮助
1.1 Linux基础
1.1.1 登录方式
1.1.2 linux用户
1.1.3 linux交互
1.1.4 shell基础
1.1.5 终端样式
1.1.6 linux命令
1.1.7 作用范围
1.1.8 软件安装
1.2 常见信息获取
1.2.1 变量信息
1.2.2 登录信息
1.2.3 系统信息
1.2.4 日期时间
1.2.5 补全功能
1.2.6 历史命令
1.2.7 linux快捷键
1.3 会话管理
1.3.1 会话基础
1.3.2 会话解绑
1.4 输出格式化
1.4.1 echo解读
1.4.2 字体颜色
1.4.3 printf格式化
1.5 自学方法
1.5.1 命令帮助
1.5.2 man手册
1.5.3 软件文档
1.5.4 精简帮助

1 Linux基础入门和帮助

1.1 Linux基础

1.1.1 登录方式

linux登录

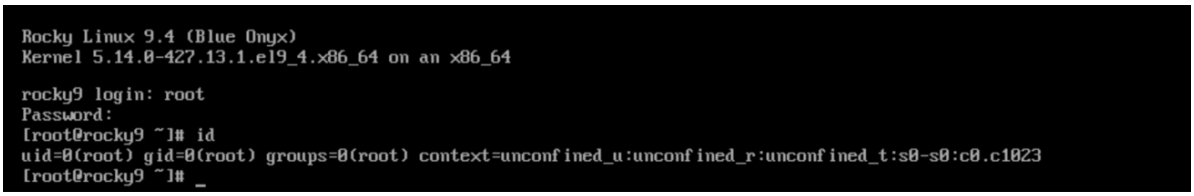
linux登录方式

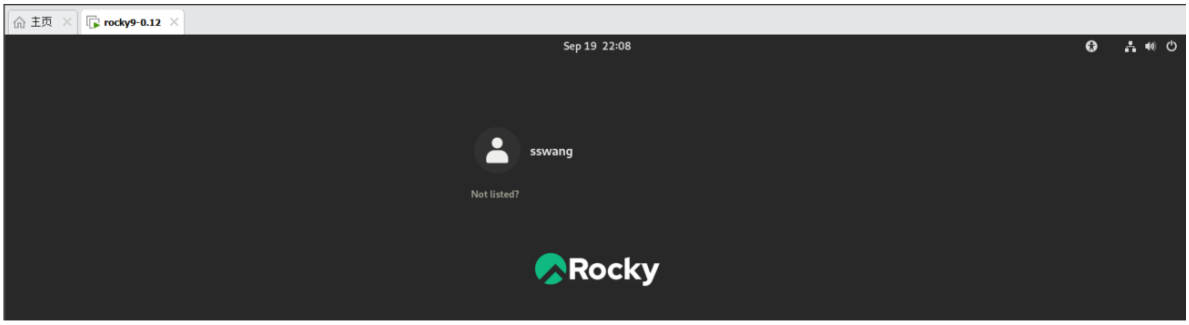
> 文本界面登录：
> 在系统启动后，用户会看到一个文本界面（也称为命令行界面或终端界面），需要输入用户名和密码进行登录。这种方式简单直接，不占用过多系统资源，适合不擅长图形界面操作的用户或需要高效执行命令的场景。
> 适用场景：服务器管理、脚本编程、系统维护等。

图形界面登录：

Linux系统中通常会安装图形化桌面环境（如GNOME、KDE等），用户可以在图形界面下选择用户名并输入密码进行登录。这种方式界面友好，操作直观，支持运行图形化应用程序。

适用场景：日常办公、软件开发、图形设计等需要图形界面支持的工作。





远程登录：

远程登录允许用户从其他主机或终端通过网络连接到Linux系统进行登录操作。

SSH（Secure Shell）：

提供加密的远程登录会话，确保数据传输的安全性。SSH是Linux系统中广泛使用的远程登录协议。

适用场景：远程服务器管理、远程软件开发、安全的数据传输等。

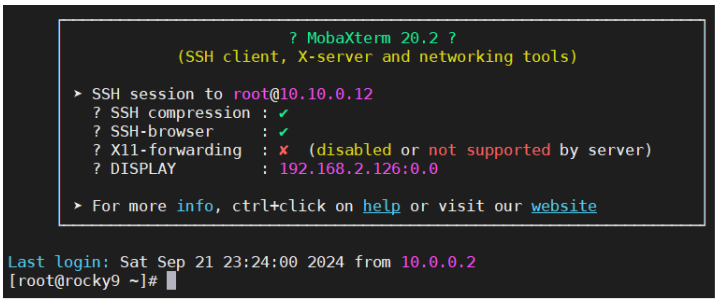
登录方式：使用SSH客户端软件或直接执行 "ssh 用户@地址" 命令进行登录。

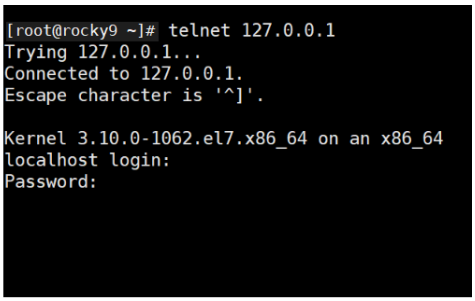
Telnet：

一种较旧的远程登录协议，不提供数据加密，因此安全性较低。在现代Linux系统中已较少使用。

适用场景：由于安全性问题，不推荐在需要保护数据传输安全性的场景中使用。

登录方式：在Linux终端下执行"telnet 地址"命令进行登录。





虚拟终端登录

在Linux系统中，用户可以通过特定的快捷键组合（如Ctrl+Alt+F1~F6）切换到虚拟终端界面进行登录。这种方式适合在图形界面出现问题时快速访问系统命令行。

适用场景：系统维护、故障排查、紧急救援等。

特殊登录方式

单用户模式登录：

在系统出现问题时，为系统管理员提供修复系统的功能。在这种模式下，只有root用户可以登录系统。

多用户登录：

允许多个用户同时登录到系统，共享系统资源。这种方式在多人协作的环境中非常有用。

1.1.2 linux用户

用户基础

简介

> Linux系统是一个多用户、多任务的操作系统，它通过严格的权限管理来保证系统的安全性和稳定性。在Linux系统中，用户是访问和使用系统资源的基本单位，不同类型的用户拥有不同的权限。Linux系统的用户主要分为两大类：root用户和普通用户。

root用户

> Root用户是Linux系统中的超级用户，拥有对系统的完全控制权。它可以执行系统中的任何命令，访问任何文件，以及更改系统配置。在Linux系统中，root用户的用户ID（UID）是0。

> Root用户的权限几乎没有限制，可以对系统进行任何操作，包括安装软件、管理系统服务、更改文件权限和所有权、格式化磁盘等。因此，使用root用户执行操作时应当格外小心，避免误操作导致系统损坏或数据丢失。

> 在Linux系统中，通常需要以root用户身份执行的操作包括系统维护、软件安装、网络配置等。然而，出于安全考虑，并不推荐频繁使用root用户登录系统或执行日常操作。相反，可以通过sudo命令临时获取root权限来执行必要的操作。

普通用户

> 普通用户是Linux系统中除了root用户之外的所有用户。它们的权限受到限制，只能访问和操作自己被授权的文件和目录。普通用户的UID通常大于0，且每个用户都有一个唯一的UID和用户名。

> 普通用户的权限相对较低，无法直接执行系统级别的操作，如安装软件、修改系统文件等。但是，它们可以在自己的用户目录下创建、修改和删除文件，以及执行自己的应用程序。

> 普通用户是Linux系统中使用最广泛的用户类型。它们被用于日常操作、编程开发、数据管理等场景。每个普通用户都应该拥有自己独立的用户目录和文件权限，以确保数据的安全性和隐私性。

管理员用户

> 在Linux系统中，管理员用户通常指的是两类用户：root用户、拥有root能力的普通用户。它们拥有对系统的完全或部分控制权，可以执行想当范围的操作。
> root用户和拥有root能力的普通用户在表现样式上，主要是少了一个额外的 sudo命令。

> 管理员用户负责维护系统的稳定性、安全性和性能。他们需要监控系统资源使用情况、安装和更新软件、配置系统服务、管理用户账户和权限等。

> 为了保护系统安全，管理员用户应遵循一系列安全实践，如使用强密码、限制root用户的远程登录、定期更新系统和软件、使用sudo代替直接登录为root用户等。

用户实践

查看当前用户

```
命令 whoami 用于查看我是谁 -- 也就是说当前登录的是哪个用户
[root@rocky9 ~]# whoami
root

用户 id 用于查看当前登录用户的全面信息，包括归属的用户管理组信息
[root@rocky9 ~]# id
用户id=0(root) 组id=0(root) 组=0(root) 上下文
=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

切换用户

```
命令 su 可以切换当前用户身份到另外一个用户
命令 exit 可以回退到刚才的那个用户身份
```

```
以root用户身份查看当前效果
[root@rocky9 ~]# ls
anaconda-ks.cfg
[root@rocky9 ~]# whoami
root
[root@rocky9 ~]# ls
anaconda-ks.cfg

使用 su命令切换到普通sswang用户，执行相同的命令
[root@rocky9 ~]# su sswang
[sswang@rocky9 root]$ whoami
sswang
[sswang@rocky9 root]$ ls
ls: 无法打开目录 '.': 权限不够

命令 exit 用于回到root用户身份
[sswang@rocky9 root]$ exit
exit
[root@rocky9 ~]# whoami
root
```

1.1.3 linux交互

终端简介

终端设备

```
    Linux系统的用户交互终端设备是用户与Linux操作系统进行交互的重要界面，它允许用户通过命令行或图形用户界面（GUI）与系统进行交互。
```

命令行方式

命令行界面是Linux系统中最基本的用户交互方式，它允许用户通过输入文本命令来执行各种操作，如文件管理、进程控制、系统配置等。在Linux中，命令行界面通常被称为“终端”（Terminal）或“控制台”（Console）。

他们的表现样式主要有如下三种：

物理终端 - 键盘+显示器

虚拟终端 - 操作系统模拟的多用户登录界面（如Ctrl+Alt+F1至F7）

伪终端 - xshell、mobaxterm、powershell等

图形界面

Linux系统还支持图形用户界面（GUI），它为用户提供了更加直观和易用的交互方式。在图形用户界面中，用户可以通过鼠标、键盘和触摸屏等输入设备与系统进行交互。Linux系统中有多种图形用户界面可供选择，如GNOME、KDE、XFCE等。这些图形用户界面通常包含窗口管理器、桌面环境、应用程序菜单等组件，为用户提供了丰富的功能和良好的用户体验。

终端类型

在linux中，每一种终端，在系统里面，都会有一个文件来承载，这些终端所占用的文件主要有：

控制台终端： /dev/console
串行终端：/dev/ttyS数字
虚拟终端：tty：teletypewriters， /dev/tty数字，tty 可有n个，Ctrl+Alt+F#
伪终端：pty：pseudo-tty ， /dev/pts/数字 如：SSH远程连接
图形终端：startx, xwindows

简单实践

查看当前终端类型

```
tty 命令可以查看当前所在终端
[root@rocky9 ~]# tty
/dev/pts/2
```

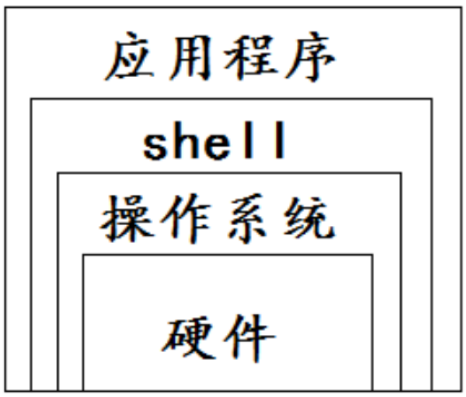
mobaxterm连接　　图形界面连接　　控制台连接

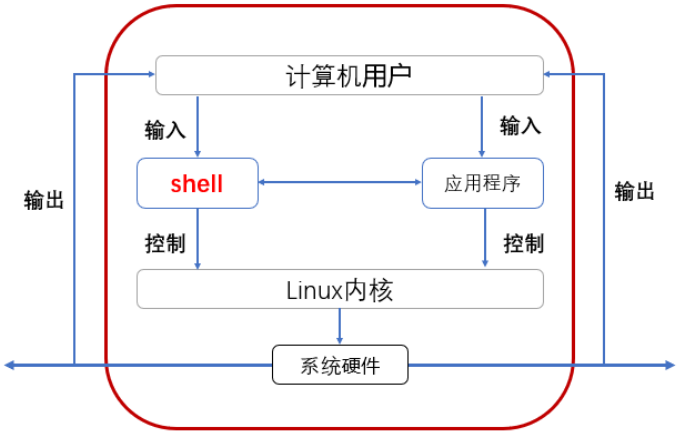
1.1.4 shell基础

Shell简介

shell定义

在计算机科学中，Shell就是一个命令解释器。
shell是位于操作系统和应用程序之间，是他们二者最主要的接口，shell负责把应用程序的输入命令信息解释给操作系统，将操作系统指令处理后的结果解释给应用程序。

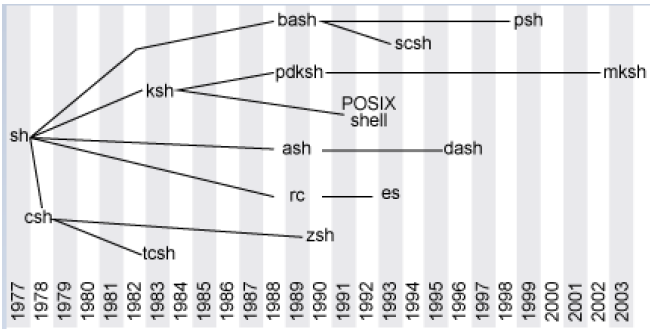




一句话，shell就是在操作系统和应用程序之间的一个命令翻译工具。

shell类型

```
在不同的操作系统上，shell的表现样式是不一样的，按照我的角度，它主要分为两类
图形界面shell
    图形界面shell就是我们常说的桌面
命令行式shell
    windows系统:
        cmd.exe  命令提示字符
    linux系统:
        sh / csh / ksh / bash(默认) / ...
```



shell简单操作

命令介绍

```
查看当前系统的shell
echo $SHELL
cat /etc/shells

更改默认的shell
chsh <用户名> -s <新shell>

安装其他shell
yum install shell名字
```

查看系统的shell

```
查看当前系统的shell类型
[root@rocky9 ~]# echo $SHELL
/bin/bash

查看当前系统环境支持的shell
[root@rocky9 ~]# cat /etc/shells
/bin/sh
/bin/bash
/usr/bin/sh
/usr/bin/bash
```

更改系统shell

```
更改默认的shell
[root@rocky9 ~]# chsh sswang -s /bin/sh
正在更改 sswang 的 shell。
Shell 已更改。
[root@rocky9 ~]# echo $SHELL
/bin/bash
[root@rocky9 ~]# su sswang
sh-5.1$ echo $SHELL
/bin/sh

登出
sh-5.1$ exit
exit
[root@rocky9 ~]#
```

```
安装shell
yum list | grep zsh
yum install -y zsh

查看效果
[root@rocky9 ~]# cat /etc/shells
/bin/sh
/bin/bash
/usr/bin/sh
/usr/bin/bash
/bin/zsh
```

shell实践实践

命令行实践方式

```
描述：
    手工敲击键盘,在shell的命令行输入命令,按Enter后,执行通过键盘输入的命令,然后shell返回并显示命令执行的结果.

重点：
    逐行输入命令、逐行进行确认执行
```

直接找一个终端界面执行一些可执行的命令即可

```
[root@rocky9 ~]# whoami
root
[root@rocky9 ~]# pwd
/root
[root@rocky9 ~]# date +"%F %T"
2024-09-27 20:14:46
```

文件实现方式

描述：

就是说我们把手工执行的命令a，写到一个脚本文件b中，然后通过执行脚本b，达到执行命令a的效果.

重点：

按照文件内容的顺序执行。

找一个文件，将我们刚才执行成功的命令放到里面。

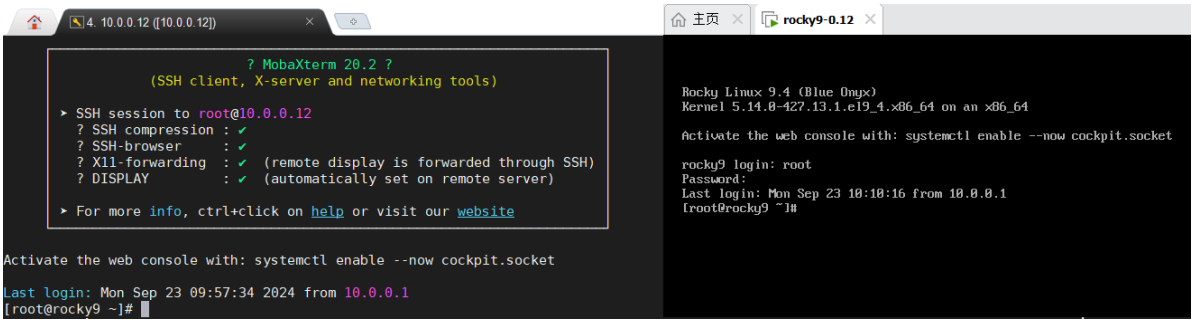
```
[root@rocky9 ~]# cat test
whoami
pwd
date +"%F %T"
[root@rocky9 ~]# /bin/bash test
root
/root
2024-09-27 20:16:20
```

1.1.5 终端样式

终端样式

简介

所谓的终端样式是指，我们在登录linux系统的时候，立刻出现的界面风格。以mobaxterm 和 rocky linux9的console登录为例：



以右侧Rocky linux9的登录console为例，

第1部分：登录之前，

来源于 /etc/redhat-release 文件内容

Rocky Linux ...

来源于 /etc/issue 文件内容

Kernel ....

来源于 /etc/issue.d/cockpit.issue 文件内容

```
        Activate the web ...
第2部分：登录过程
    rocky9 login:
    passwd:
第3部分：登录之后
    来源于 ssh服务登录后的login命令程序设置，通过 man login 来查看
        Last login: ...
    来源于 /etc/motd 文件
        默认为空，除非自己添加
    来源于 /etc/bashrc 文件中的设置
    [root@rocky9 ~]#
```

bashrc 文件设置的信息

```
查看文件内容
[root@rocky9 ~]# vim /etc/bashrc
...
    [ "$PS1" = "\\s-\\v\\$ " ] && PS1="[\u@\h \W]\\$ "
...

PS1 是一个很有意思的变量，是用来定义命令行的提示符的，PS1 可以支持以下这些选项：
    \d：显示日期，格式为"星期 月 日"。
    \H：显示完整的主机名。如默认主机名"localhost.localdomain"。
    \h：显示简写的主机名。如默认主机名"localhost"。
    \t：显示 24 小时制时间，格式为"HH:MM:SS"。
    \T：显示 12 小时制时间，格式为"HH:MM:SS"。
    \A：显示 24 小时制时间，格式为"HH:MM"。
    @：显示 12 小时制时间，格式为"HH:MM am/pm"。
    \u：显示当前用户名。
    \v：显示 Bash 的版本信息。
    \w：显示当前所在目录的完整名称。
    \W：显示当前所在目录的最后一个目录。
    #：执行的第几条命令。
    $：提示符。如果是 root 用户，则会显示提示符为"#"；如果是普通用户，则会显示提示符为"$"。
```

```
查看 PS1 的内容
[root@rocky9 ~]# echo $PS1
[\u@\h \W]\$
```

简单实践

查看不同用户的命令行的提示符格式

```
查看root管理员用户的命令行提示  #
[root@rocky9 ~]# whoami
root
[root@rocky9 ~]#

查看sswang普通用户的命令行提示  $
[root@rocky9 ~]# su sswang
[sswang@rocky9 root]$ cd /home/sswang/
[sswang@rocky9 ~]$ whoami
sswang
[sswang@rocky9 ~]$
```

修改默认的PS1变量

```
[root@rocky9 ~]# PS1='\[\e[1;35m\][\u@\h \w]\\$\[\e[0m\] '
[root@rocky9 ~]$ PS1='\[\e[1;32m\][\u@\h \w]\\$\[\e[0m\] '
[root@rocky9 ~]$ PS1='\[\e[1;34m\][\u@\h \w]\\$\[\e[0m\] '
[root@rocky9 ~]$ PS1='\[\e[1;34m\][\u@\h \w]=\[\e[0m\] '
[root@rocky9 ~]= PS1='\[\e[1;34m\][\u@\h \w]>\[\e[0m\] '
[root@rocky9 ~]> PS1='\[\e[1;34m\][\t\u@\h \w]>\[\e[0m\] '
[11:02:26root@rocky9 ~]> PS1='\[\e[1;34m\][\d\u@\h \w]>\[\e[0m\] '
[一 9月 23root@rocky9 ~]>
```

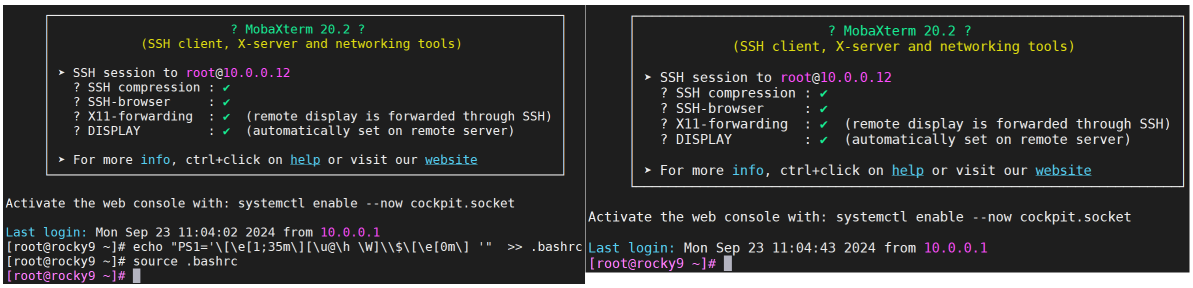
```
Last login: Mon Sep 23 11:00:18 2024 from 10.0.0.1
[root@rocky9 ~]# PS1='\[\e[1;35m\][\u@\h \W]\\$\[\e[0m\] '
[root@rocky9 ~]$ PS1='\[\e[1;32m\][\u@\h \W]\\$\[\e[0m\] '
[root@rocky9 ~]$ PS1='\[\e[1;34m\][\u@\h \W]\\$\[\e[0m\] '
[root@rocky9 ~]$ PS1='\[\e[1;34m\][\u@\h \W]=\[\e[0m\] '
[root@rocky9 ~]= PS1='\[\e[1;34m\][\u@\h \W]>\[\e[0m\] '
[root@rocky9 ~]> PS1='\[\e[1;34m\][\t\u@\h \W]>\[\e[0m\] '
[11:02:26root@rocky9 ~]> PS1='\[\e[1;34m\][\d\u@\h \W]>\[\e[0m\] '
[一 9月 23root@rocky9 ~]>
```

永久设置PS登录的效果

我们可以将PS1的值保存到一个文件里面，从而实现永久设定的效果

```
[root@rocky9 ~]# echo "PS1='\[\e[1;35m\][\u@\h \W]\\$\[\e[0m\] '"  >> .bashrc
[root@rocky9 ~]# source .bashrc
```

然后再开启一个新的终端，就可以看到 命令提示行自动变成我们期望的颜色了。



1.1.6 linux命令

命令简介

命令执行

在命令行输入命令后，按Enter回车，然后命令就会交给linux系统的命令解释器，该解释器交由后端的操作系统的可执行程序或代码，这些程序和代码分析后提交给内核分配资源将其运行起来，执行完毕后，再把结果在屏幕返回给我们。

linux命令

Linux的命令主要有两种类型：

内部命令：

由shell自带的，而且通过某命令形式提供，用户登录后自动加载并常驻内存中

外部命令：

在文件系统路径下有对应的可执行程序文件，当执行命令时才从磁盘加载至内存中,执行完毕后从内存中删除

常见的命令操作

```
    type 确认一个命令是否是内部命令
        type -a <cmd>    : 查看更详细的信息提示
    help 查看内部命令列表
    whereis 查看命令是否存在，若存在显示所有位置。
    which   查看命令是否存在，若存在显示第一个显示的位置。
    man   查看命令帮助

命令的默认存在目录
    $PATH   获取默认命令能够存在的目录

命令简化操作
    alias   为一个命令定义别名，提高命令的操作效率
    unalias 为一个命令取消别名

开机关机命令
    关机：halt、poweroff、shutdown -hnow
    重启：reboot、shutdown -r
```

简单实践

查看内部命令

```
[root@rocky9 ~]# help
GNU bash，版本 5.1.8(1)-release (x86_64-redhat-linux-gnu)
这些 shell 命令是内部定义的。请输入 `help' 以获取一个列表。
输入 `help 名称' 以得到有关函数`名称'的更多信息。
使用 `info bash' 来获得关于 shell 的更多一般性信息。
使用 `man -k' 或 `info' 来获取不在列表中的命令的更多信息。

名称旁边的星号(*)表示该命令被禁用。
...
```

确认命令是否是内部命令

```
查看type命令是否是内部命令
[root@rocky9 ~]# type type
type 是 shell 内建
[root@rocky9 ~]# type -a type
type 是 shell 内建
type 是 /usr/bin/type
```

```
做一个假命令 sswang
[root@rocky9 ~]# cp /usr/bin/type /usr/bin/sswang
[root@rocky9 ~]# type sswang
sswang 是 /usr/bin/sswang
[root@rocky9 ~]# type -a sswang
sswang 是 /usr/bin/sswang
结果显示：
    sswang 不是 一个内部命令
```

查找命令是否存在

```
让命令存在多个位置
[root@rocky9 ~]# cp /usr/bin/sswang /usr/local/bin/sswang

which确认命令是否存在
[root@rocky9 ~]# which sswang
/usr/local/bin/sswang

whereis确认命令是否存在
[root@rocky9 ~]# whereis sswang
sswang: /usr/bin/sswang /usr/local/bin/sswang
```

```
确认未知命令
[root@rocky9 ~]# whereis sswang111
sswang111:
[root@rocky9 ~]# which sswang111
/usr/bin/which: no sswang111 in
(/root/.local/bin:/root/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin)
注意：
    which如果找不到命令，会提示，它在哪些目录下寻找了
    这个目录是 PATH
```

查看命令的目录

```
查看命令默认存在 在 哪些目录下。
[root@rocky9 ~]# echo $PATH
/root/.local/bin:/root/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
```

```
转移一个命令到 $PATH之外的其他目录
[root@rocky9 ~]# cp /usr/local/bin/sswang /tmp/sswang111

查看命令是否存在
[root@rocky9 ~]# whereis sswang111
sswang111:

将/tmp 加入到PATH环境变量中
[root@rocky9 ~]# PATH=$PATH:/tmp

查看效果
[root@rocky9 ~]# whereis sswang111
sswang111: /tmp/sswang111
[root@rocky9 ~]# which sswang111
/tmp/sswang111

将/tmp 移出到PATH环境变量中
[root@rocky9 ~]#
PATH=/root/.local/bin:/root/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bi
n

再次查看sswang111命令
[root@rocky9 ~]# whereis sswang111
sswang111:
```

定制命令别名

```
比如我们要查看当前的日期
[root@rocky9 ~]# date "+%Y-%m-%d %H:%M:%S"
2024-09-23 11:44:07

上面的方式太麻烦了，我自己构造一个命令
[root@rocky9 ~]# alias now='date "+%Y-%m-%d %H:%M:%S"'
[root@rocky9 ~]# now
2024-09-23 11:44:47
```

```
查看所有的命令别名
[root@rocky9 ~]# alias
alias cp='cp -i'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l.='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
alias ls='ls --color=auto'
alias mv='mv -i'
alias now='date "+%Y-%m-%d %H:%M:%S"'
alias rm='rm -i'
alias xzegrep='xzegrep --color=auto'
alias xzfgrep='xzfgrep --color=auto'
alias xzgrep='xzgrep --color=auto'
alias zegrep='zegrep --color=auto'
alias zfgrep='zfgrep --color=auto'
alias zgrep='zgrep --color=auto'

取消命令别名
[root@rocky9 ~]# unalias now

再次确认
[root@rocky9 ~]# now
bash: now: 未找到命令...
```

关机重启

```
立刻关机
[root@rocky9 ~]# shutdown -h now
[root@rocky9 ~]# halt
[root@rocky9 ~]# poweroff

立刻重启
[root@rocky9 ~]# shutdown -r now
[root@rocky9 ~]# reboot
```

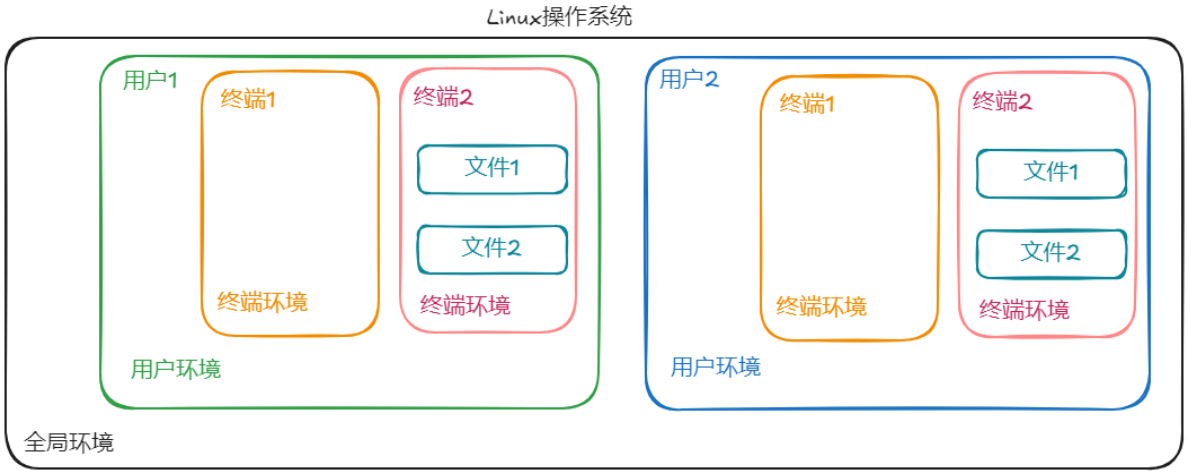
```
延迟关机
[root@rocky9 ~]# shutdown -h 1

撤销关机
[root@rocky9 ~]# shutdown -c
```

1.1.7 作用范围

基础知识

命令的作用范围

```
    linux本身是 多用户多任务的操作系统，所以，一个linux系统的用户，可以同时启动多个程序，也可以多个用户同时登录同一个linux系统。
    而linux为了避免多个用户在操作的时候，互相影响，划分了很多块linux命令的作用区域，从而实现多用户同时登录的时候，操作环境是隔离的。
```



命令作用范围的文件

```
系统级别的环境控制文件 - 默认只有系统启动的时候，才会加载
    /etc/profile
    /etc/profile.d/xxx.sh
系统用户级别的环境控制文件 - 默认开启一个终端登录就会加载
    /etc/bashrc
用户终端级别的环境控制文件    - 切换一个终端用户就会加载
    ~/.bashrc
    ~/.bash
用户终端+文件级别的环境控制文件 - 执行文件的时候才会加载
    /path/to/file.sh

特殊的命令作用范围 - 子shell
    () 代表当前shell范围里面生成一个独立的shell，可以开启一个独立的进程
    $() 代表开启一个独立的进程，执行()范围的可执行命令，然后将信息返回给当前的shell
        注意：如果()范围内的命令执行结果内容，不是可执行命令的话，最好用下面的方法来输出
            echo $(可执行命令)
```

简单实践

定制用户终端文件级别的命令

```
定制文件
[root@rocky9 ~]# cat /tmp/a.sh
now='date "+%Y-%m-%d %H:%M:%S"'
eval $now

执行文件
[root@rocky9 ~]# /bin/bash /tmp/a.sh
2024-09-23 12:05:32

文件外查看效果
[root@rocky9 ~]# now
bash: now: 未找到命令...
```

定制用户终端级别的命令实践

```
终端1
[root@rocky9 ~]# alias now='date "+%Y-%m-%d %H:%M:%S"'
[root@rocky9 ~]# now
2024-09-23 12:06:27

终端2
[root@rocky9 ~]# now
bash: now: 未找到命令...
```

定制用户级别的命令实践

```
终端1撤销别名
[root@rocky9 ~]# unalias now
[root@rocky9 ~]# now
bash: now: 未找到命令...

终端1定制命令别名环境变量
[root@rocky9 ~]# tail -1 .bashrc
alias now='date "+%Y-%m-%d %H:%M:%S"'

终端2查看命令效果
[root@rocky9 ~]# now
2024-09-23 12:08:45
```

```
更换用户查看命令效果
[root@rocky9 ~]# su sswang
[sswang@rocky9 root]$ whoami
sswang
[sswang@rocky9 root]$ now
bash: now: 未找到命令...
```

定制系统级别的命令实践

```
终端1取消命令别名环境变量
[root@rocky9 ~]# tail -1 .bashrc
PS1='\[\e[1;35m\][\u@\h \W]\$\[\e[0m\] '

定制系统级别的命令实践
[root@rocky9 ~]# tail -1 /etc/bashrc
alias now='date "+%Y-%m-%d %H:%M:%S"'
```

```
终端2 root用户测试效果
[root@rocky9 ~]# whoami
root
[root@rocky9 ~]# now
2024-09-23 12:13:24

终端2 切换到普通用户测试效果
[root@rocky9 ~]# su sswang
[sswang@rocky9 root]$ whoami
sswang
[sswang@rocky9 root]$ now
2024-09-23 12:13:43
```

子shell的空间

```
查看当前shell空间的进程id
[root@rocky9 ~]# echo $BASHPID
2057

查看子shell空间的进程id
[root@rocky9 ~]# echo $(echo $BASHPID)
2989
```

```
[root@rocky9 ~]# echo "$BASHPID"; date "+%F %T"
2057
2024-09-24 08:59:45

使用独立的子shell空间执行命令
[root@rocky9 ~]# echo $(echo "$BASHPID"; date "+%F %T")
3019 2024-09-24 08:59:55
```

1.1.8 软件安装

包管理器

简介

```
    Rocky Linux 9 支持的软件管理方式主要基于 RPM（Red Hat Package Manager）包管理器，关于rpm包的管理，在Rocky Linux 9 中可以通过DNF（Dandified YUM）进行包管理，也可以通过yum（Yellowdog Updater, Modified）进行包管理
```

关于 yum

yum（Yellowdog Updater, Modified）是一个在Fedora、RedHat以及许多其他基于RPM的Linux发行版中广泛使用的Shell前端软件包管理器。尽管dnf被设计为yum的下一代替代品，但Rocky Linux 9（以及其他许多现代Linux发行版）通常仍然保留了对yum的支持，以保持向后兼容性。yum允许用户搜索、安装、更新和删除软件包，同时自动处理依赖关系。

关于DNF

dnf（Dandified YUM）是Fedora项目开发的，旨在成为yum的下一代版本。dnf提供了与yum类似的命令行界面，但在性能、依赖解析和用户体验方面有所改进。Rocky Linux 9采用了dnf作为其主要的软件包管理器，因为它能够更快地处理大量的软件包和依赖关系，并且提供了更清晰的错误信息和更友好的用户界面。

关联和区别

在Rocky Linux 9中，用户既可以使用yum进行软件包管理，也可以使用dnf。然而，由于dnf是更现代、更高效的软件包管理器，并且Rocky Linux 9将其作为默认选项，因此建议用户在可能的情况下优先使用dnf。不过，yum的向后兼容性意味着它仍然是一个有用的工具，特别是在处理旧脚本或旧系统时。

命令实践

常见命令

```
更新软件源
    dnf|yum makecache
搜索软件
    dnf|yum search <软件名>
安装软件
    dnf|yum [-y] install <软件名>
删除软件
    dnf|yum [-y] remove <软件名>
```

简单实践

```
更新软件源
[root@rocky9]~# dnf makecache
Rocky Linux 9 - BaseOS          3.9 kB/s | 4.1 kB     00:01
Rocky Linux 9 - AppStream         3.3 kB/s | 4.5 kB     00:01
Rocky Linux 9 - Extras          2.2 kB/s | 2.9 kB     00:01
元数据缓存已建立。
```

```
搜索软件
[root@rocky9]~# dnf search nginx
上次元数据过期检查：0:00:36 前，执行于 2024年09月23日 星期一 09时50分26秒。
================================================ 名称 精准匹配：nginx
================================================
nginx.x86_64 : A high performance web server and reverse proxy server
================================================ 名称 和 概况 匹配：nginx
================================================
nginx-all-modules.noarch : A meta package that installs all available Nginx
modules
nginx-core.x86_64 : nginx minimal core
nginx-filesystem.noarch : The basic directory layout for the Nginx server
nginx-mod-http-image-filter.x86_64 : Nginx HTTP image filter module
nginx-mod-http-perl.x86_64 : Nginx HTTP perl module
```

```
nginx-mod-http-xslt-filter.x86_64 : Nginx XSLT module
nginx-mod-mail.x86_64 : Nginx mail modules
nginx-mod-stream.x86_64 : Nginx stream modules
pcp-pmda-nginx.x86_64 : Performance Co-Pilot (PCP) metrics for the Nginx
Webserver
```

```
安装软件
[root@rocky9]~# dnf install -y nginx
上次元数据过期检查：0:00:57 前，执行于 2024年09月23日 星期一 09时50分26秒。
...
已安装:
  nginx-1:1.20.1-16.el9_4.1.x86_64                              nginx-core-
1:1.20.1-16.el9_4.1.x86_64
  nginx-filesystem-1:1.20.1-16.el9_4.1.noarch                   rocky-logos-httpd-
90.15-2.el9.noarch

完毕!
```

```
删除软件
[root@rocky9]~# dnf remove -y nginx
依赖关系解决。
...
已移除:
  nginx-1:1.20.1-16.el9_4.1.x86_64                              nginx-core-
1:1.20.1-16.el9_4.1.x86_64
  nginx-filesystem-1:1.20.1-16.el9_4.1.noarch                   rocky-logos-httpd-
90.15-2.el9.noarch

完毕!
```

1.2 常见信息获取

1.2.1 变量信息

变量解读

变量

```
在linux系统中，有非常多的变量，我们可以通过变量名来快速获取我们期望获取的数据或者信息。
```

变量格式

```
定义变量
    变量名=值

查看变量
    echo $变量名

查看系统默认的变量
    env
    man bash
```

变量实践

定义变量

```
查看变量名
[root@rocky9 ~]# echo $my_path

定义变量
[root@rocky9 ~]# my_path="/path/to/file"

再次查看变量名
[root@rocky9 ~]# echo $my_path
/path/to/file
```

查看系统变量

```
[root@rocky9 ~]# env
SHELL=/bin/bash
HISTCONTROL=ignoredups
HISTSIZE=1000
HOSTNAME=rocky9
PWD=/root
LOGNAME=root
XDG_SESSION_TYPE=tty
MOTD_SHOWN=pam
HOME=/root
LANG=zh_CN.UTF-8
...
MAIL=/var/spool/mail/root
SSH_TTY=/dev/pts/1
BASH_FUNC_which%%=() {  ( alias;
 eval ${which_declare} ) | /usr/bin/which --tty-only --read-alias --read-
functions --show-tilde --show-dot $@
}
_=/usr/bin/env
```

```
查看常用变量的值
[root@rocky9 ~]# echo $SHELL
/bin/bash
[root@rocky9 ~]# echo $HOME
/root
[root@rocky9 ~]# echo $HOSTNAME
rocky9
[root@rocky9 ~]# echo $PATH
/root/.local/bin:/root/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
```

查看bash内置的变量

```
[root@rocky9 ~]# man bash
...
   Shell Variables
       The following variables are set by the shell:
       ...
```

```
查看当前是shell名字
[root@rocky9 ~]# echo $BASH
/bin/bash
```

```
查看当前bash会话的id
[root@rocky9 ~]# echo $BASHPID
2047

查看当前bash的般本
[root@rocky9 ~]# echo $BASH_VERSION
5.1.8(1)-release

查看当前系统架构信息【GNU风格】
[root@rocky9 ~]# echo $MACHTYPE
x86_64-redhat-linux-gnu

查看当前用户的历史命令文件
[root@rocky9 ~]# echo $HISTFILE
/root/.bash_history

查看当前系统的语言类型
[root@rocky9 ~]# echo $LANG
zh_CN.UTF-8
```

1.2.2 登录信息

命令解读

常见命令解读

```
查看当前登录用户
whoami

显示用户的身份标识信息
id

查看我在哪里
pwd

显示当前系统登录的用户
who

查看当前登录用户的启动程序信息
w

显示上次登录的用户列表信息
last
```

命令实践

查看当前登录用户

```
查看当前登录用户
[root@rocky9 ~]# whoami
root

切换用户后再次查看
[root@rocky9 ~]# su sswang
[sswang@rocky9 root]$ whoami
sswang
```

显示用户的身份标识信息

```
[root@rocky9 ~]# id root
用户id=0(root) 组id=0(root) 组=0(root)

[root@rocky9 ~]# id sswang
用户id=1000(sswang) 组id=1000(sswang) 组=1000(sswang),10(wheel)
```

查看我在哪里

```
查看我在哪里
[root@rocky9 ~]# pwd
/root

切换工作目录后，再次查看
[root@rocky9 ~]# cd /tmp/
[root@rocky9 tmp]# pwd
/tmp
```

显示当前系统登录的用户

```
[root@rocky9 ~]# who
root     pts/0        2024-09-23 12:13 (10.0.0.1)
sswang   seat0        2024-09-23 09:13 (login screen)
sswang   tty2         2024-09-23 09:13 (tty2)
root     pts/2        2024-09-23 11:04 (10.0.0.1)
root     tty4         2024-09-23 10:13
root     pts/3        2024-09-23 10:19 (10.0.0.1)
root     pts/4        2024-09-23 12:26 (10.0.0.1)
```

查看当前登录用户的启动程序信息

```
[root@rocky9 ~]# w
 12:27:16 up  3:18,  7 users,  load average: 0.00, 0.00, 0.00
USER     TTY        LOGIN@   IDLE   JCPU   PCPU WHAT
root     pts/0     12:13    3:39   0.08s  0.00s less
sswang   seat0     09:13    0.00s  0.00s  0.00s /usr/libexec/gdm-wayland-session
--register-session gnome-session
sswang   tty2      09:13    3:18m  0.03s  0.03s /usr/libexec/gnome-session-
binary
root     pts/2     11:04    1:22m  0.02s  0.02s -bash
root     tty4      10:13    2:13m  0.02s  0.02s -bash
root     pts/3     10:19    1:37m  0.14s  0.14s -bash
root     pts/4     12:26    1.00s  0.03s  0.01s w
```

显示当前系统之前登录的用户列表信息

```
[root@rocky9 ~]# last
root     pts/0        10.0.0.1         Mon Sep 23 12:28   still logged in
root     pts/4        10.0.0.1         Mon Sep 23 12:26 - 12:28  (00:01)
root     pts/0        10.0.0.1         Mon Sep 23 12:13 - 12:28  (00:14)
解析
    第一列：用户名
    第二列：登录终端
    第三列：登录ip
    第四列：登录日期
    第五列：登录持续时间段
    第六列：登录持续时间(单位是分钟)
```

```
注销登录
[root@rocky9 ~]# exit

再次登录
[root@rocky9 ~]# last
root     pts/0        10.0.0.1         Mon Sep 23 12:29   still logged in
root     pts/0        10.0.0.1         Mon Sep 23 12:28 - 12:29  (00:01)
root     pts/4        10.0.0.1         Mon Sep 23 12:26 - 12:28  (00:01)
root     pts/0        10.0.0.1         Mon Sep 23 12:13 - 12:28  (00:14)
```

1.2.3 系统信息

命令解读

常见硬件信息获取

```
查看cpu信息
    命令：lscpu
    文件：/proc/cpuinfo

查看mem信息
    命令：free
    文件：/proc/meminfo

查看磁盘分区信息
    命令：lsblk、df
    文件：/proc/partitions

查看磁盘设备信息
    命令：blkid
```

常见系统信息获取

查看系统架构
    命令：arch

查看内核信息
    命令：uname -r

查看操作系统发行版本
    文件：/etc/redhat-release[Rocky系统]、/etc/os-release、/etc/lsb-release[Ubuntu系统]

命令实践

查看cpu信息

```
[root@rocky9 ~]# lscpu
架构：                    x86_64
  CPU 运行模式：          32-bit, 64-bit
  Address sizes:          45 bits physical, 48 bits virtual
  字节序：                Little Endian
CPU:                      4
  在线 CPU 列表：         0-3
厂商 ID：                 GenuineIntel
  BIOS Vendor ID:         GenuineIntel
  型号名称：              Intel(R) Core(TM) i7-6820HQ CPU @ 2.70GHz
    BIOS Model name:      Intel(R) Core(TM) i7-6820HQ CPU @ 2.70GHz
    CPU 系列：            6
    型号：                94
    每个核的线程数：      1
    每个座的核数：        2
    座：                  2
    步进：                3
    BogoMIPS:             5424.00
    标记：                fpu ... arch_capabilities
Virtualization features:
  虚拟化：                VT-x
  超管理器厂商：          VMware
  虚拟化类型：            完全
Caches (sum of all):
  L1d:                    128 KiB (4 instances)
  L1i:                    128 KiB (4 instances)
  L2:                     1 MiB (4 instances)
  L3:                     16 MiB (2 instances)
NUMA:
  NUMA 节点：             1
  NUMA 节点0 CPU：        0-3
Vulnerabilities:
  Gather data sampling:   Unknown: Dependent on hypervisor status
  Itlb multihit:          KVM: Mitigation: VMX disabled
  L1tf:                   Mitigation; PTE Inversion; VMX flush not necessary, SMT disabled
  Mds:                    Mitigation; Clear CPU buffers; SMT Host state unknown
  Meltdown:               Mitigation; PTI
  Mmio stale data:        Mitigation; Clear CPU buffers; SMT Host state unknown
  Retbleed:               Mitigation; IBRS
  Spec rstack overflow:   Not affected
  Spec store bypass:      Mitigation; Speculative Store Bypass disabled via prctl
```

```
  Spectre v1:           Mitigation; usercopy/swapgs barriers and __user pointer
sanitization
  Spectre v2:           Mitigation; IBRS, IBPB conditional, STIBP disabled, RSB
filling, PBRSB-eIBRS Not affected
  Srbds:                Unknown: Dependent on hypervisor status
  Tsx async abort:      Not affected
```

```
[root@rocky9 ~]# cat /proc/cpuinfo
processor       : 0
...
address sizes   : 45 bits physical, 48 bits virtual
power management:

processor       : 1
...
address sizes   : 45 bits physical, 48 bits virtual
power management:

processor       : 2
...
address sizes   : 45 bits physical, 48 bits virtual
power management:

processor       : 3
...
address sizes   : 45 bits physical, 48 bits virtual
power management:
```

查看mem信息

```
[root@rocky9 ~]# free
               total        used        free      shared  buff/cache   available
Mem:         7837292     1403456     4669452       22244     2048668     6433836
Swap:        8232956           0     8232956

[root@rocky9 ~]# cat /proc/meminfo
MemTotal:        7837292 kB
MemFree:         4668476 kB
MemAvailable:    6432884 kB
Buffers:            3960 kB
Cached:          1961848 kB
SwapCached:            0 kB
...
DirectMap4k:      237376 kB
DirectMap2M:     6053888 kB
DirectMap1G:     4194304 kB
```

查看磁盘分区信息

```
[root@rocky9 ~]# lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sr0          11:0    1  10.2G  0 rom  /run/media/sswang/Rocky-9-4-x86_64-dvd
nvme0n1     259:0    0   200G  0 disk
├─nvme0n1p1 259:1    0     1G  0 part /boot
└─nvme0n1p2 259:2    0   199G  0 part
  ├─rl-root 253:0    0    70G  0 lvm  /
```

```
  ├─rl-swap 253:1    0   7.9G  0 lvm  [SWAP]
  └─rl-home 253:2    0 121.1G  0 lvm  /home

[root@rocky9 ~]# cat /proc/partitions
major minor  #blocks  name

 259        0  209715200 nvme0n1
 259        1    1048576 nvme0n1p1
 259        2  208665600 nvme0n1p2
  11        0   10660544 sr0
 253        0   73400320 dm-0
 253        1    8232960 dm-1
 253        2  127029248 dm-2
```

查看磁盘和目录关系信息

```
[root@rocky9 ~]# blkid
/dev/mapper/rl-swap: UUID="d9e6a8b2-93ad-4110-afd7-3ab92ca21185" TYPE="swap"
/dev/nvme0n1p1: UUID="5ab84930-27a4-418f-bb0a-67324618924a" TYPE="xfs"
PARTUUID="d125fc5c-01"
/dev/nvme0n1p2: UUID="h2Typm-Uskf-T35A-yR8f-8Mgx-AFMm-hvmkbh" TYPE="LVM2_member"
PARTUUID="d125fc5c-02"
/dev/sr0: UUID="2024-05-05-01-12-25-00" LABEL="Rocky-9-4-x86_64-dvd"
TYPE="iso9660" PTUUID="849e8820" PTTYPE="dos"
/dev/mapper/rl-home: UUID="b654e6c1-e957-4b32-ba0f-cbda8e535cb4" TYPE="xfs"
/dev/mapper/rl-root: UUID="9a04a979-9e4e-4d6b-969a-b74f185f2fa3" TYPE="xfs"

[root@rocky9 ~]# df
文件系统                 1K-块      已用       可用 已用% 挂载点
devtmpfs                 4096         0       4096    0% /dev
tmpfs                 3918644         0    3918644    0% /dev/shm
tmpfs                 1567460      9920    1557540    1% /run
/dev/mapper/rl-root  73334784   5786640   67548144    8% /
/dev/mapper/rl-home 126963712    924112  126039600    1% /home
/dev/nvme0n1p1         983040    310544     672496   32% /boot
tmpfs                  783728        36     783692    1% /run/user/0
tmpfs                  783728       100     783628    1% /run/user/1000
/dev/sr0             10660236  10660236          0  100% /run/media/sswang/Rocky-
9-4-x86_64-dvd
```

查看系统架构信息

```
[root@rocky9 ~]# arch
x86_64
```

查看内核信息

```
获取完整信息
[root@rocky9 ~]# uname -a
Linux rocky9 5.14.0-427.13.1.el9_4.x86_64 #1 SMP PREEMPT_DYNAMIC Wed May 1
19:11:28 UTC 2024 x86_64 x86_64 x86_64 GNU/Linux

获取拆解信息
[root@rocky9 ~]# uname -s
```

```
Linux
[root@rocky9 ~]# uname -n
rocky9
[root@rocky9 ~]# uname -r
5.14.0-427.13.1.el9_4.x86_64
[root@rocky9 ~]# uname -v
#1 SMP PREEMPT_DYNAMIC Wed May 1 19:11:28 UTC 2024

[root@rocky9 ~]# uname -m
x86_64
[root@rocky9 ~]# uname -p
x86_64
[root@rocky9 ~]# uname -i
x86_64
[root@rocky9 ~]# uname -o
GNU/Linux
```

查看操作系统发行版本信息

```
[root@rocky9 ~]# cat /etc/redhat-release
Rocky Linux release 9.4 (Blue Onyx)

[root@rocky9 ~]# cat /etc/os-release
NAME="Rocky Linux"
VERSION="9.4 (Blue Onyx)"
ID="rocky"
ID_LIKE="rhel centos fedora"
VERSION_ID="9.4"
PLATFORM_ID="platform:el9"
PRETTY_NAME="Rocky Linux 9.4 (Blue Onyx)"
ANSI_COLOR="0;32"
LOGO="fedora-logo-icon"
CPE_NAME="cpe:/o:rocky:rocky:9::baseos"
HOME_URL="https://rockylinux.org/"
BUG_REPORT_URL="https://bugs.rockylinux.org/"
SUPPORT_END="2032-05-31"
ROCKY_SUPPORT_PRODUCT="Rocky-Linux-9"
ROCKY_SUPPORT_PRODUCT_VERSION="9.4"
REDHAT_SUPPORT_PRODUCT="Rocky Linux"
REDHAT_SUPPORT_PRODUCT_VERSION="9.4"
```

1.2.4 日期时间

命令解析

linux时间

```
在linux中，主要有两种时间
    系统时钟：由Linux内核通过CPU的工作频率进行的
    硬件时钟：主板
```

时间相关的命令

```
显示和设置系统时间
    date

显示和设置硬件时间
    clock，hwclock

查看时区的问题
    文件：/etc/localtime、/etc/timezone
    命令：timedatectl

日历信息查看
    命令：cal -y
```

简单实践

查看日期和时间

```
查看日期和时间
[root@rocky9 ~]# date
2024年 09月 23日 星期一 14:39:26 CST
[root@rocky9 ~]# date +%F
2024-09-23
[root@rocky9 ~]# date +%T
14:39:33
[root@rocky9 ~]# date +"%Y-%m-%d %H:%M:%S"
2024-09-23 14:41:23
[root@rocky9 ~]# date +%s
1727073592
```

```
定制日期显示的格式
[root@rocky9 ~]# date -d "2024-09-23 14:40:00" +"%Y-%m-%d %H:%M:%S"
2024-09-23 14:40:00
[root@rocky9 ~]# date -d "2024-09-23 14:40:00" +"%H:%M:%S %Y-%m-%d"
14:40:00 2024-09-23
```

```
设定日期和时间
[root@rocky9 ~]# date
2024年 09月 23日 星期一 14:43:10 CST
[root@rocky9 ~]# date -s "2024-09-23 11:40:00"
2024年 09月 23日 星期一 11:40:00 CST
```

```
计算指定日期一个月后的日期
[root@rocky9 ~]# date -d "2024-09-23 + 1 month" +"%Y-%m-%d"
2024-10-23
```

查看硬件时间

```
clock 和 hwclock 其实是同一个文件
[root@rocky9 ~]# ls /usr/sbin/clock
/usr/sbin/clock
[root@rocky9 ~]# ls /usr/sbin/clock  -l
lrwxrwxrwx. 1 root root 7  4月 21 02:24 /usr/sbin/clock -> hwclock
```

```
查看修改后的系统时间
[root@rocky9 ~]# date
2024年 09月 23日 星期一 11:43:38 CST

查看硬件时间
[root@rocky9 ~]# hwclock
2024-09-23 14:47:05.908831+08:00
[root@rocky9 ~]# clock
2024-09-23 14:47:19.912026+08:00
```

时区信息实践

```
查看系统支持的时区
[root@rocky9 ~]# timedatectl list-timezones
Africa/Abidjan
Africa/Accra
...

查看当前时区信息
[root@rocky9 ~]# timedatectl status
               Local time: 一 2024-09-23 11:46:53 CST
           Universal time: 一 2024-09-23 03:46:53 UTC
                 RTC time: 一 2024-09-23 06:49:56
                Time zone: Asia/Shanghai (CST, +0800)
System clock synchronized: no
              NTP service: active
          RTC in local TZ: no

[root@rocky9 ~]# ll /etc/localtime
lrwxrwxrwx. 1 root root 35  9月 23 06:07 /etc/localtime ->
../usr/share/zoneinfo/Asia/Shanghai
```

```
修改时区
[root@rocky9 ~]# timedatectl set-timezone Africa/Ceuta

查看修改后时区信息
[root@rocky9 ~]# timedatectl status
               Local time: 一 2024-09-23 05:47:45 CEST
           Universal time: 一 2024-09-23 03:47:45 UTC
                 RTC time: 一 2024-09-23 06:50:44
                Time zone: Africa/Ceuta (CEST, +0200)
System clock synchronized: no
              NTP service: active
          RTC in local TZ: no

[root@rocky9 ~]# ll /etc/localtime
lrwxrwxrwx. 1 root root 34  9月 23 05:47 /etc/localtime ->
../usr/share/zoneinfo/Africa/Ceuta

查看时间
[root@rocky9 ~]# date
2024年 09月 23日 星期一 05:48:26 CEST
```

查看日历

```
```

查看当前月

```
[root@rocky9 ~]# cal -1
      九月 2024
一 二 三 四 五 六 日
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30
```

查看三个月

```
[root@rocky9 ~]# cal -3
      八月 2024              九月 2024              十月 2024
一 二 三 四 五 六 日  一 二 三 四 五 六 日  一 二 三 四 五 六 日
          1  2  3  4                      1       1  2  3  4  5  6
 5  6  7  8  9 10 11   2  3  4  5  6  7  8   7  8  9 10 11 12 13
12 13 14 15 16 17 18   9 10 11 12 13 14 15  14 15 16 17 18 19 20
19 20 21 22 23 24 25  16 17 18 19 20 21 22  21 22 23 24 25 26 27
26 27 28 29 30 31     23 24 25 26 27 28 29  28 29 30 31
                      30
```

查看一年

```
[root@rocky9 ~]# cal -y
                               2024

        一月                   二月                   三月
一 二 三 四 五 六 日   一 二 三 四 五 六 日   一 二 三 四 五 六 日
 1  2  3  4  5  6  7            1  2  3  4                1  2  3
 8  9 10 11 12 13 14    5  6  7  8  9 10 11    4  5  6  7  8  9 10
15 16 17 18 19 20 21   12 13 14 15 16 17 18   11 12 13 14 15 16 17
22 23 24 25 26 27 28   19 20 21 22 23 24 25   18 19 20 21 22 23 24
29 30 31               26 27 28 29            25 26 27 28 29 30 31

        四月                   五月                   六月
一 二 三 四 五 六 日   一 二 三 四 五 六 日   一 二 三 四 五 六 日
 1  2  3  4  5  6  7         1  2  3  4  5                   1  2
 8  9 10 11 12 13 14    6  7  8  9 10 11 12    3  4  5  6  7  8  9
15 16 17 18 19 20 21   13 14 15 16 17 18 19   10 11 12 13 14 15 16
22 23 24 25 26 27 28   20 21 22 23 24 25 26   17 18 19 20 21 22 23
29 30                  27 28 29 30 31         24 25 26 27 28 29 30

        七月                   八月                   九月
一 二 三 四 五 六 日   一 二 三 四 五 六 日   一 二 三 四 五 六 日
 1  2  3  4  5  6  7            1  2  3  4                      1
 8  9 10 11 12 13 14    5  6  7  8  9 10 11    2  3  4  5  6  7  8
15 16 17 18 19 20 21   12 13 14 15 16 17 18    9 10 11 12 13 14 15
22 23 24 25 26 27 28   19 20 21 22 23 24 25   16 17 18 19 20 21 22
29 30 31               26 27 28 29 30 31      23 24 25 26 27 28 29
                                              30
        十月                   十一月                 十二月
一 二 三 四 五 六 日   一 二 三 四 五 六 日   一 二 三 四 五 六 日
    1  2  3  4  5  6                  1  2  3                      1
 7  8  9 10 11 12 13    4  5  6  7  8  9 10    2  3  4  5  6  7  8
14 15 16 17 18 19 20   11 12 13 14 15 16 17    9 10 11 12 13 14 15
21 22 23 24 25 26 27   18 19 20 21 22 23 24   16 17 18 19 20 21 22
28 29 30 31            25 26 27 28 29 30      23 24 25 26 27 28 29
```

```
                                                30 31
```

1.2.5 补全功能

基础知识

功能需求

```
    在linux系统环境里面，我们管理各种应用都是通过命令来实现的，但是有很多时候，命令太长记不住，我们只记住前面的1-2个字母，后面的内容不知道。
    对于linux来说，它提供了命令补全的能力，也就是说，我们可以通过 Tab键的方式，将我们要敲出来的命令在命令行自动补全。
    - 用户给定的字符串只有一条惟一对应的命令，直接补全，否则，再次Tab会给出列表
```

补全功能

```
    在linux系统里面，关于Tab键补全的功能，主要体现在两个方面：
    - 路径的补全
    - 命令的补全
```

简单实践

路径补全

```
单Tab键自动补全文件路径
[root@rocky9 ~]# ls /etc/hostn^C
[root@rocky9 ~]# ls /etc/hostname

双Tab键查看满足要求的目录
[root@rocky9 ~]# ls /etc/de
debuginfod/ default/    depmod.d/
```

命令补全

```
单Tab键自动补全命令
[root@rocky9 ~]# vms^C
[root@rocky9 ~]# vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0      0 6255628   2896 894968    0    0    21    16   23   33  0  0 100  0
 0

双Tab键查看满足要求的命令
[root@rocky9 ~]# time^C
[root@rocky9 ~]# time
time         timedatectl  timeout      times
```

1.2.6 历史命令

基础知识

历史命令

```
    在linux系统中，当执行命令后，系统默认会在内存记录执行过的命令,当用户正常退出时，会将内存的命令历史存放对应历史文件中，默认是 ~/.bash_history。
    登录shell时，会读取命令历史文件中记录下的命令加载到内存中，登录进shell后新执行的命令只会记录在内存的缓存区中；这些命令会用户正常退出时“追加”至命令历史文件中。
    利用命令历史。可以用它来重复执行命令，提高输入效率
```

history命令解析

```
[root@rocky9 ~]# history --help
history: history [-c] [-d 偏移量] [n] 或 history -anrw [文件名] 或 history -ps 参数 [参数...]
    显示或操纵历史列表。

    带行号显示历史列表，将每个被修改的条目加上 `*' 前缀。
    参数 N 会仅列出最后的 N 个条目。

    选项：
      -c         删除所有条目从而清空历史列表。
      -d 偏移量  从指定位置删除历史列表。负偏移量将从历史条目末尾开始计数
      -a         将当前会话的历史行追加到历史文件中
      -n         从历史文件中读取所有未被读取的行,并且将它们附加到历史列表
      -r         读取历史文件并将内容追加到历史列表中
      -w         将当前历史写入到历史文件中
      -p         对每一个 ARG 参数展开历史并显示结果，而不存储到历史列表中
      -s         以单条记录追加 ARG 到历史列表中

    如果给定了 FILENAME 文件名，则它将被作为历史文件。否则
    如果 $HISTFILE 变量有值的话使用之，不然使用 ~/.bash_history 文件。

    如果 $HISTTIMEFORMAT 变量被设定并且不为空，它的值会被用于
    strftime(3) 的格式字符串来打印与每一个显示的历史条目想关联的
    时间戳，否则不打印时间戳。

    退出状态：
    返回成功，除非使用了无效的选项或者发生错误。
```

与history相关的环境变量

```
HISTSIZE：命令历史记录的条数
HISTFILE：指定历史文件，默认为~/.bash_history
HISTFILESIZE：命令历史文件记录历史的条数
HISTTIMEFORMAT="%F %T `whoami` "    显示时间和用户
HISTIGNORE="str1:str2*:…" 忽略str1命令，str2开头的历史
HISTCONTROL：控制命令历史的记录方式
```

使用历史命令提高执行效率

```
重复执行上一条命令
    - 重复前一个命令使用上方向键，并回车执行              *****
    - 按 !! 并回车执行
    - 输入!-1 并回车执行
    - 按 Ctrl+p 并回车执行

重复执行之前的命令
    - 按 !n 执行history命令列表中的第n编号的命令         *****
```

```
    - 按 !-n 执行history命令列表中的倒数第n个命令
    - 按 !string 重复前一个以“string”开头的命令        *****
    - 按 !?string 重复前一个包含string的命令
    - 按 !string:p 仅打印命令历史，而不执行
    - 按 !$:p 打印输出 !$ （上一条命令的最后一个参数）的内容
    - 按 !*:p 打印输出 !*（上一条命令的所有参数）的内容
    - 按 ^string 删除上一条命令中的第一个string
    - 按 ^string1^string2 将上一条命令中的第一个string1替换为string2
    - 按 !:gs/string1/string2 将上一条命令中所有的string1都替换为 string2使用up（向上）
和down（向下）键来上下浏览从前输入的命令
    - 按 ctrl-r来在命令历史中搜索命令
      （reverse-i-search）`':
    - 按 Ctrl+g：从历史搜索模式退出
```

简单实践

历史命令查看和清理

```
查看历史命令
[root@rocky9 ~]# history
    1  tty
    2  id
    3  whoami
    ...
    590  exit
    591  history

清理历史命令
[root@rocky9 ~]# history -c

再次查看历史命令
[root@rocky9 ~]# history
    1  history
```

历史命令调用

```
多执行几条命令后，查看历史命令
[root@rocky9 ~]# history
    1  history
    2  h
    3  info
    4  ls /etc/
    5  ls /tmp/
    6  echo nihao
    7  history

执行上一条命令
[root@rocky9 ~]# !!
history
    1  history
    2  h
    3  info
    4  ls /etc/
    5  ls /tmp/
    6  echo nihao
```

```
    7  history
[root@rocky9 ~]# !-1
history
    1  history
    2  h
    3  info
    4  ls /etc/
    5  ls /tmp/
    6  echo nihao
    7  history
[root@rocky9 ~]# history            # 执行 Ctrl + p
    1  history
    2  h
    3  info
    4  ls /etc/
    5  ls /tmp/
    6  echo nihao
    7  history
```

```
调用第6条历史命令并执行
[root@rocky9 ~]# !-2                # 使用 !-倒数数字
echo nihao
nihao
[root@rocky9 ~]# !6                 # 使用 !编号数字
echo nihao
nihao
```

关键字检索历史命令执行

```
命令行现状
[root@rocky9 ~]#

按 Ctrl + R
(reverse-i-search)`':

然后输入关键字
(reverse-i-search)`echo': echo nihao

按Enter执行
[root@rocky9 ~]# echo nihao
nihao
```

1.2.7 linux快捷键

shell快捷键

快捷键符号

```
命令执行：
    !!                  执行上一条命令
    !num                执行历史命令中的第num行命令
    Ctrl 关键字          执行内容匹配的命令
```

```
命令行切换
    Ctrl + A            光标迅速回到行首
    Ctrl + E            光标迅速回到行尾
    Ctrl + k            删除光标到行尾内容
    Ctrl + u            删除光标到行首内容
    Ctrl + y            粘贴删除的内容
    Ctrl + c            临时终止命令行命令
    Esc + b             移动到当前单词的开头
    Esc + f             移动到当前单词的结尾
```

1.3 会话管理

1.3.1 会话基础

会话基础

会话简介

```
    命令行的典型使用方式是，打开一个终端窗口（terminal window），在里面输入命令。用户与计算机的这种临时的交互，称为一次"会话"（session）
    会话的一个重要特点是，窗口与其中启动的进程是连在一起的。打开窗口，会话开始；关闭窗口，会话结束，会话内部的进程也会随之终止，不管有没有运行完
```

相关命令

```
查看程序进程信息
    ps aux
    pstree -p

过滤相关信息
    执行命令 | grep 关键字
```

会话实践

查看进程信息

```
查看当前ssh连接的进程
ps aux | grep ssh
```

```
[root@rocky9 ~]# ps aux | grep ssh
root        1013  0.0  0.1  15800  9088 ?        Ss   09:10   0:00 sshd: /usr/sbin/sshd -D [listener]
 0 of 10-100 startups
root        2001  0.0  0.1  19476 11904 ?        Ss   09:10   0:00 sshd: root [priv]
root        2038  0.0  0.0  19776  7108 ?        S    09:10   0:00 sshd: root@pts/0
root        2124  0.0  0.1  19476 12032 ?        Ss   09:17   0:00 sshd: root [priv]
root        2131  0.0  0.0  19780  7120 ?        S    09:17   0:00 sshd: root@pts/1
root        2543  0.0  0.0 221812  2304 pts/1    S+   09:38   0:00 grep --color=auto ssh
[root@rocky9 ~]#
```

```
查看当前ssh程序所关联的进程信息
[root@rocky9 ~]# pstree -p
```

```
[root@rocky9 ~]# pstree -p
systemd(1)─┬─ModemManager(958)─┬─{ModemManager}(973)
           │                   ├─{ModemManager}(979)
           │                   └─{ModemManager}(983)
           ├─sshd(1013)─┬─sshd(2001)───sshd(2038)───bash(2047)
           │            └─sshd(2124)───sshd(2131)───bash(2143)───pstree(2545)
```

查看终端1的 bash进程id

```
[root@rocky9 ~]# echo $BASHPID
2047
```

结果显示：

终端1的会话处于 sshd(2001)的最后一个位置

终端1开启多个程序(提示 & 符号，把程序放到后台)

终端1 通过 sleep命令 + & 开启多个程序

```
[root@rocky9 ~]# sleep 100 &
[1] 2548
[root@rocky9 ~]# sleep 1000 &
[2] 2549
[root@rocky9 ~]# sleep 10000 &
[3] 2550
[root@rocky9 ~]# sleep 100000 &
[4] 2551
```

终端2查看终端1的效果

```
[root@rocky9 ~]# pstree -p
```

```
[root@rocky9 ~]# pstree -p
systemd(1)─┬─ModemManager(958)─┬─{ModemManager}(973)
           │                   ├─{ModemManager}(979)
           │                   └─{ModemManager}(983)
           ├─sshd(1013)─┬─sshd(2001)───sshd(2038)───bash(2047)─┬─sleep(2548)
           │            │                                      ├─sleep(2549)
           │            │                                      ├─sleep(2550)
           │            │                                      └─sleep(2551)
           │            └─sshd(2124)───sshd(2131)───bash(2143)───pstree(2553)
```

我们把 终端1，立刻关闭，然后，在终端2查看bash会话效果

```
[root@rocky9 ~]# pstree -p
```

```
[root@rocky9 ~]# pstree -p
systemd(1)─┬─ModemManager(958)─┬─{ModemManager}(973)
           │                   ├─{ModemManager}(979)
           │                   └─{ModemManager}(983)
           ├─sshd(1013)───sshd(2124)───sshd(2131)───bash(2143)───pstree(2566)
```

结果显示：

终端1的窗口一旦关闭后，该终端上面的所有关联会话都会立刻消失。

1.3.2 会话解绑

基础知识

会话需求

一个典型的例子就是，SSH 登录远程计算机，打开一个远程窗口执行命令。这时，网络突然断线，再次登录的时候，是找不回上一次执行的命令的。因为上一次 SSH 会话已经终止了，里面的进程也随之消失了。为了解决这个问题，会话与窗口可以"解绑"：窗口关闭时，会话并不终止，而是继续运行，等到以后需要的时候，再让会话"绑定"其他窗口。

终端复用器

```
终端复用器软件就是会话与窗口的"解绑"工具，将它们彻底分离。
（1） 它允许在单个窗口中，同时访问多个会话。这对于同时运行多个命令行程序很有用。
（2） 它可以让新窗口"接入"已经存在的会话。
（3） 它允许每个会话有多个连接窗口，因此可以多人实时共享会话。
（4） 它还支持窗口任意的垂直和水平拆分。
```

```
类似的终端复用器还有Screen，Tmux
```

软件部署

```
centos7系统：
    yum install -y screen
    yum install -y tmux

Centos8|rocky8|rocky9系统：
    dnf install -y epel-release
    dnf install -y screen
    dnf install -y tmux
```

screen

基础命令

```
创建一个独立会话：screen -S 会话名
加入screen会话： screen -X 会话名
退出并保存会话：exit
剥离当前screen会话：Ctrl +a,d
显示所有会话：screen -ls
恢复一个会话：screen -r 会话名
```

会话演示

```
查看当前的bash会话id
[root@rocky9 ~]# echo $BASHPID
2143

创建一个会话
[root@rocky9 ~]# screen -S session

在当前会话中执行多个sleep 命令
[root@rocky9 ~]# echo $BASHPID
2785
[root@rocky9 ~]# sleep 1000 &
[1] 2874
[root@rocky9 ~]# sleep 10000 &
[2] 2875

终端2查看进程信息
pstree -p
```

```
[root@rocky9 ~]# pstree -p
systemd(1)─┬─ModemManager(958)─┬─{ModemManager}(973)
           │                   ├─{ModemManager}(979)
           │                   └─{ModemManager}(983)
           ├─sshd(1013)─┬─sshd(2124)───sshd(2131)───bash(2143)───screen(2783)───screen(2784)───bash(2785)─┬─sleep(2874)
           │            │                                                                                 └─sleep(2875)
           │            └─sshd(2806)───sshd(2812)───bash(2825)───pstree(2877)
```

终止终端1的会话，直接关闭窗口即可，再次在终端2检查进程效果

```
终止终端1的会话，直接关闭窗口即可，再次在终端2检查进程效果
结果显示：
    之前的终端1窗口会话，依然存活，只不过是在一个独立的screen进程里面
```

```
[root@rocky9 ~]# pstree -p
systemd(1)─┬─ModemManager(958)─┬─{ModemManager}(973)
           │                   ├─{ModemManager}(979)
           │                   ├─{ModemManager}(983)
           │                   └─{rtkit-daemon}(916)
           ├─screen(2784)───bash(2785)─┬─sleep(2874)
           │                           └─sleep(2875)
           ├─sshd(1013)───sshd(2806)───sshd(2812)───bash(2825)───pstree(2979)
```

加入会话

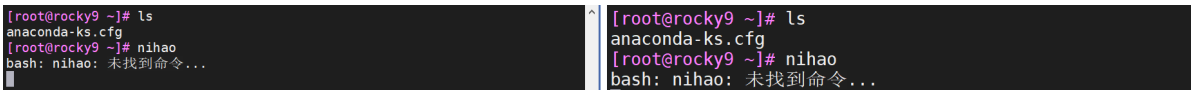
```
开启一个独立的终端
[root@rocky9 ~]# echo $BASHPID
3012
[root@rocky9 ~]# screen -ls
There is a screen on:
        2784.session    (Detached)
1 Socket in /run/screen/S-root.

加入之前的会话
[root@rocky9 ~]# screen -x session

立刻回将之前的终端信息显示出来
[root@rocky9 ~]# sleep 1000 &
[1] 2874
[root@rocky9 ~]# sleep 10000 &
[2] 2875
[root@rocky9 ~]# echo $BASHPID
2785
[root@rocky9 ~]#
```

```
终端2，查看进程效果
```

```
[root@rocky9 ~]# pstree -p
systemd(1)─┬─ModemManager(958)─┬─{ModemManager}(973)
           │                   ├─{ModemManager}(979)
           │                   └─{ModemManager}(983)
           ├─screen(2784)───bash(2785)─┬─sleep(2874)
           │                           └─sleep(2875)
           ├─sshd(1013)─┬─sshd(2806)───sshd(2812)───bash(2825)───pstree(3067)
           │            └─sshd(2987)───sshd(2994)───bash(3012)───screen(3065)
```

会话镜像测试

```
终端1 创建会话
screen -S mirror

终端2 进入会话
screen -x mirror

接下来，两个界面就会实现一个同步的镜像操作
    - A终端执行的信息，在B终端可看
    - B终端执行的信息，在A终端可看

    也就是说，他们可以实现一种类似 实时聊天 的工作讨论效果
```



1.4 输出格式化

1.4.1 echo解读

基础知识

命令简介

```
echo命令的功能是将内容输出到默认显示设备，一般起到一个提示的作用。
```

```
OPTIONS:
-n   不要在最后自动换行
-e   若字符串中出现以下字符，则特别加以处理，而不会将它当成一般文字输出:

转义字符
\a      发出警告声;
\b      删除前一个字符;
\c      最后不加上换行符号;
\f      换行但光标仍旧停留在原来的位置;
\n      换行且光标移至行首;
\r      光标移至行首，但不换行;
\t      插入tab;
\v      与\f相同;
\       插入\字符;
\0nnn   打印nnn(八进制)所代表的ASCII字符;  备注：数字0  不要理解成字母o
\xNN    打印NN(十六进制)所代表的ASCII字符;

--help  显示帮助
--version显示版本信息
```

简单实践

实践1 - 信息的输出

```
echo后边用单引号包围要添加的内容
[root@rocky9 ~]# echo 'hello world' >> /tmp/hello.txt
[root@rocky9 ~]# cat /tmp/hello.txt
hello world
```

实践2 - 引号信息输出

```
通过引号的错开实现引号的输出
[root@rocky9 ~]# echo "I'm a king of the world."
I'm a king of the world.
```

实践3 - 特殊符号的输出

```
使用 -e 选项启用转义字符的解析
[root@rocky9 ~]# echo -e "The 2021 State of DevOps Report\n\t- is here"
The 2021 State of DevOps Report
        - is here
```

实践4 - 内容的拼接

```
使用 -n 选项启用信息输出不换行
[root@rocky9 ~]# echo -n hello;echo world
helloworld
```

1.4.2 字体颜色

基础知识

场景需求

```
echo本质上是将信息内容输出到当前的屏幕终端，如果只是一种颜色的话，可能导致视觉疲劳。所以，一般情况下，我们在显示信息的时候，往往会通过颜色的方式实现特定内容的颜色高亮显示。

echo命令可以修改字体类型，字体背景色以及字体颜色，转义序列\033可以用于改变字体属性。
```

格式解读

```
格式如下：
    echo -e "\033[字背景颜色;文字颜色m字符串\033[0m"
    echo -e "\033[41;36m 显示的内容 \033[0m"
```

颜色分类

色彩	黑	红	绿	黄	蓝	紫	青	灰
字体色	30	31	32	33	34	35	36	37
背景色	40	41	42	43	44	45	46	47

结束控制符

```
最后面控制选项说明
    \033[0m 关闭所有属性          \033[1m 设置高亮度          \033[4m 下划线
    \033[5m 闪烁                  \033[7m 反显                \033[8m 消隐
注意：
     \033 是八进制的ESCAPE字符，我们可以用 \e 来代替
```

简单实践

实践1 - 字体颜色和背景颜色

```
字体颜色示例
echo -e "\033[30m 黑色字 \033[31m 红色字 \033[32m 绿色字 \033[33m 黄色字 \033[0m"
echo -e "\033[34m 蓝色字 \033[35m 紫色字 \033[36m 天蓝字 \033[37m 白色字 \033[0m"

背景颜色示例
echo -e "\033[40;37m 黑底白字 \033[41;37m 红底白字 \033[42;37m 绿底白字 \033[0m"
echo -e "\033[43;37m 黄底白字 \033[44;37m 蓝底白字 \033[45;37m 紫底白字 \033[0m"
echo -e "\033[46;37m 天蓝底白字 \033[47;30m 白底黑字 \033[0m"
```

```
[root@rocky9 ~]# echo -e "\033[30m 黑色字 \033[31m 红色字 \033[32m 绿色字 \033[33m 黄色字 \033[0m"
         红色字  绿色字  黄色字
[root@rocky9 ~]# echo -e "\033[34m 蓝色字 \033[35m 紫色字 \033[36m 天蓝字 \033[37m 白色字 \033[0m"
 蓝色字  紫色字  天蓝字  白色字
[root@rocky9 ~]# echo -e "\033[40;37m 黑底白字 \033[41;37m 红底白字 \033[42;37m 绿底白字 \033[0m"
 黑底白字  红底白字  绿底白字
[root@rocky9 ~]# echo -e "\033[43;37m 黄底白字 \033[44;37m 蓝底白字 \033[45;37m 紫底白字 \033[0m"
 黄底白字  蓝底白字  紫底白字
[root@rocky9 ~]# echo -e "\033[46;37m 天蓝底白字 \033[47;30m 白底黑字 \033[0m"
 天蓝底白字  白底黑字
[root@rocky9 ~]#
```

实践2 - 信息颜色显示

```
定制堡垒机的测试页面脚本
[root@rocky9 ~]# cat simple_jumpserver.sh
#!/bin/bash
# 功能：定制堡垒机的展示页面
# 作者：sswang
# 版本：v0.1
# 联系：sswang.magedu.com

echo -e "\e[31m \t\t 欢迎使用堡垒机"

echo -e "\e[32m
-----------请选择你要登录的远程主机-----------
 1: 10.0.0.14 (nginx)
 2: 10.0.0.15 (tomcat)
 3: 10.0.0.19 (apache)
 q: 使用本地主机
-----------------------------------------------
```

```
"'\033[0m'
echo -e "请输入您要选择的远程主机编号: "
```

```
[root@rocky9 ~]#/bin/bash jumpserver.sh
                欢迎使用堡垒机

-----------请选择你要登录的远程主机-----------
 1: 10.0.0.14 (nginx)
 2: 10.0.0.15 (tomcat)
 3: 10.0.0.19 (apache)
 q: 使用本地主机
-----------------------------------------------

请输入您要选择的远程主机编号:
[root@rocky9 ~]#
```

小结

```
```

1.4.3 printf格式化

学习目标

这一节，我们从 基础知识、简单实践、小结 三个方面来学习。

基础知识

场景需求

```
    虽然我们能够通过 echo的方式实现信息某种程度的格式化输出，但是很多信息的输出偏重于手工的干预，效率太慢。我们需要一种功能更强大、效率更高的格式化手段。
    -- printf
```

printf简介

```
    printf 命令模仿 C 程序库（library）里的 printf() 程序。由于 printf 由 POSIX 标准所定义，因此使用 printf 的脚本比使用 echo 移植性好。
    printf 使用引用文本或空格分隔的参数，外面可以在 printf 中使用格式化字符串，还可以制定字符串的宽度、左右对齐方式等。默认 printf 不会像 echo 自动添加换行符，我们可以手动添加 \n。
```

基本语法

```
查看帮助
[root@rocky9 ~]# help printf
printf: printf [-v var] 格式 [参数]
```

```
格式化替换符
    %s      字符串
    %d,%i   十进制整数
    %f      浮点格式
    %c      ASCII字符，即显示对应参数的第一个字符
    %b      相对应的参数中包含转义字符时，可以使用此替换符进行替换，对应的转义字符会被转义
    %o      八进制值
    %u      不带正负号的十进制值
    %x      十六进制值（a-f）
    %X      十六进制值（A-F）
    %%      表示%本身
```

```
格式化转义符号
    \a      警告字符，通常为ASCII的BEL字符
    \b      后退
    \c      抑制（不显示）输出结果中任何结尾的换行字符
    \f      换页（formfeed）
    \n      换行
    \r      回车（Carriage return）
    \t      水平制表符
    \v      垂直制表符
    \\      一个字面上的反斜杠字符
    \ddd    表示1到3位数八进制值的字符。仅在格式字符串中有效
    \0ddd   表示1到3位的八进制值字符
```

```
格式化字符解释
-       将字段里已格式化的值向左对齐
空格     在正值前置一个空格,在负值前置一个负号
+       总是在数值之前放置一个正号或负号,即便是正值也是
#       下列形式选择其一:
            %o有一个前置的o;                    %x与%X分别前置的0x与0X;
            %e,%E与%f总是在结果中有一个小数点;  %g与%G为没有结尾的零。
0       以零填补输出,而非空白.这仅发生在字段宽度大于转换后的情况
```

简单实践

实践1 - 命令演示

```
普通echo命令演示换行信息和非换行信息
[root@rocky9 ~]# echo "Hello, Shell Base"
Hello, Shell Base
[root@rocky9 ~]# echo -n "Hello, Shell Base"
Hello, Shell Base[root@rocky9 ~]#

printf命令演示换行信息和非换行信息
[root@rocky9 ~]# printf "Hello, Shell Base\n"
Hello, Shell Base
[root@rocky9 ~]# printf "Hello, Shell Base"
Hello, Shell Base[root@rocky9 ~]#
```

实践2 - 替换符号演示

```
基本信息演示 - 单引号和双引号效果一样
[root@rocky9 ~]# printf "姓名:%s, 语文:%d, 数学:%d\n" "张三" 89 98
姓名:张三, 语文:89, 数学:98
[root@rocky9 ~]# printf '姓名:%s, 语文:%d, 数学:%d\n' "张三" 89 98
姓名:张三, 语文:89, 数学:98
注意:
    相关信息会按照顺序依次替换到格式化的替换符位置
```

```
数字格式展示
[root@rocky9 ~]# printf '姓名:%s, 身高:%f米, 体重:%f公斤\n' "张三" 1.7 66
姓名:张三, 身高:1.700000米, 体重:66.000000公斤
[root@rocky9 ~]# printf '姓名:%s, 身高:%.2f米, 体重:%.1f公斤\n' "张三" 1.7 66
姓名:张三, 身高:1.70米, 体重:66.0公斤
注意:
    通过 %.nf，n代表小数点后面的可显示的位数。
```

实践3 - 简单格式化

```
() 用于信息的批量化显示
[root@rocky9 ~]# printf "(%d %s)\n" 1 张三 2 李四 3 王五
(1 张三)
(2 李四)
(3 王五)

printf默认不携带最后的换行
[root@rocky9 ~]# printf "(%d %s)  " 1 张三 2 李四 3 王五
(1 张三)  (2 李四)  (3 王五)  [root@rocky9 ~]#

借助于echo的功能实现换行的效果
[root@rocky9 ~]# printf "(%d %s)  " 1 张三 2 李四 3 王五; echo
(1 张三)  (2 李四)  (3 王五)
```

实践4 - 格式化补充

```
%s字符串格式化
[root@rocky9 ~]# printf '姓名:%-10s语文:%d, 数学:%d\n' "张三" 89 98 "李四" 87 97
姓名:张三      语文:89, 数学:98
姓名:李四      语文:87, 数学:97
注意:
    %-10s，代表信息后面携带10个空格，便于格式化
```

```
%d数字格式补全
[root@rocky9 ~]# printf "%5d %s\n" 1 张三
    1 张三
[root@rocky9 ~]# printf "%05d %s\n" 1 张三
00001 张三
[root@rocky9 ~]# printf "%05d %s\n" 100 张四
00100 张四
注意:
    0的作用就是不用空格补全，而是用0填充，实现格式化
```

实践5 - 环境变量的使用

```
定制环境变量
[root@rocky9 ~]# VAR1=hello; VAR2=shell

颜色显示
[root@rocky9 ~]# printf "\033[32m%s \033[31m%s\033[0m\n" $VAR1 $VAR2
hello shell
```

信息显示

定制java环境部署脚本

```
[root@rocky9 ~]# cat /tmp/java_info.sh
#!/bin/bash
# 功能: java环境的信息提示
# 版本: v0.1
# 作者: sswang
# 联系: sswang.magedu.com

# 定制基本变量信息
RED="echo -e \E[1;31m"
END="\E[0m"
java_version="11.0.24"
runtime_version="Red_Hat-11.0.24.0.8-2"
jvm_type="mixed"

# 信息显示
$RED---------$JAVA_NAME软件运行情况---------$END
printf "\033[32m%s:\033[33m%s\033[0m\n" "java软件版本" $java_version
printf "\033[32m%s:\033[33m%s\033[0m\n" "java运行时环境版本" $runtime_version
printf "\033[32m%s:\033[33m%s\033[0m\n" "jvm运行模式" $jvm_type
$RED---------------------------------$END
```

```
[root@rocky9 ~]# /bin/bash /tmp/java_info.sh
---------软件运行情况---------
java软件版本:11.0.24
java运行时环境版本:Red_Hat-11.0.24.0.8-2
jvm运行模式:mixed
---------------------------------
[root@rocky9 ~]#
```

1.5 自学方法

1.5.1 命令帮助

基础知识

在Linux系统中，当你遇到不熟悉的命令时，有几种常用方法可以帮助你找到所需的信息、学习命令的用法，甚至快速解决问题。

```
使用whatis命令：
    使用whatis命令来借助linux内部的命令数据库文档，来显示命令的简单描述
    生成数据库的方法：
        Centos7之前使用 makewhatis，Centos7之后使用mandb
使用man命令：
    man是manual（手册）的缩写，它是Linux系统中获取命令和程序使用说明的标准方式。通过在终端中输入man 命令名（例如，man ls），你可以访问该命令的手册页，里面详细说明了命令的用法、选项、示例等。
```

使用--help选项：
    大多数Linux命令都支持--help选项，这个选项会显示命令的基本用法和可用的选项。通过在命令后加上--help（例如，ls --help），你可以快速获取到关于该命令的简要帮助信息。

使用info命令（可选）：
    类似于man命令，info也是用于查看命令和程序的使用说明的。不过，info命令提供的信息通常以更结构化的方式呈现，可能包含更多背景和示例。但是，并非所有Linux发行版都默认安装了info页面，而且它的使用不如man命令那么普遍。
    格式化显示：info '(coreutils) 命令 invocation'
    注意：我们可以将 info的帮助手册，当成一个web网页，看到超链接的样式，就直接跳转到对应的页面内容。

利用Shell的自动补全和提示功能：
    在输入命令时，许多Shell（如Bash）都提供了自动补全和提示功能。这意味着当你开始输入一个命令时，Shell会尝试预测你可能要输入的命令，并为你提供补全选项。这虽然不直接解释命令的用法，但可以帮助你发现你可能不知道的命令或选项。

其他的方法：

搜索互联网：
    使用Google、百度等搜索引擎搜索你遇到的命令或问题是一个高效的方法。搜索时，最好将Linux系统的发行版（如Ubuntu、CentOS等）或特定场景（如shell脚本）也考虑在内，以获取更精确的结果。

参考Linux文档和教程：
    有很多高质量的Linux文档、教程和论坛（如Linux社区网站、Stack Overflow等）提供了大量的学习资源。通过阅读和参与这些资源，你可以逐渐提高自己的Linux技能和知识。

询问他人：
    如果你在某个社区或组织中工作，不妨向同事或同行询问。他们可能遇到过类似的问题，或者能给出你需要的帮助。

简单实践

使用whatis来获取命令的帮助信息

```
生成数据库
[root@rocky9 ~]# mandb
正在删除 /usr/share/man/overrides 里的旧数据库条目...
正在处理 /usr/share/man/overrides 下的手册页...
...
0 个 man 子目录包含更新的手册页。
添加了 0 个手册页。
添加了 0 个孤立 cat 页面。
删除了 0 条旧数据库条目。

查看命令的帮助简介
[root@rocky9 ~]# whatis ls
ls (1)               - list directory contents
ls (1p)              - list directory contents
[root@rocky9 ~]# whatis vmstat
vmstat (8)           - Report virtual memory statistics
[root@rocky9 ~]# whatis top
top (1)              - display Linux processes
```

通过 --help 参数来获取命令帮助信息

```
[root@rocky9 ~]# type --help
type: type [-afptP] 名称 [名称 ...]
    显示命令类型的信息。

    对于每一个 NAME 名称，指示如果作为命令它将如何被解释。

    选项：
      -a        显示所有包含名称为 NAME 的可执行文件的位置；
                包括别名、内建和函数。仅当 `-p' 选项没有使用时
      -f        抑制 shell 函数查询
      -P        为每个 NAME 名称惊醒 PATH 路径搜索，即使它是别名、
                内建或函数，并且返回将被执行的磁盘上文件的名称。
      -p        返回将被执行的磁盘上文件的名称，或者当 `type -t NAME'
        不返回 `file' 时，不返回任何值。
      -t        返回下列词中的任何一个 `alias'、`keyword'、
        `function'、`builtin'、`file' 或者 `'，相应地如果 NAME 是
        一个别名、shell 保留字、shell 函数、shell 内建、
        磁盘文件或没有找到。
    ...

--help显示信息中的相关符号解析
    [] 表示可选项
    CAPS或 <> 表示变化的数据
    ... 表示一个列表
    x |y| z 的意思是“ x 或 y 或 z ”
    -abc的 意思是 -a -b -c
    { }  表示分组
```

```
[root@rocky9 ~]# cat --help
用法：cat [选项]... [文件]...
连接所有指定文件并将结果写到标准输出。

如果没有指定文件，或者文件为"-"，则从标准输入读取。

  -A, --show-all            等效于 -vET
  -b, --number-nonblank     对非空输出行编号，同时取消 -n 选项效果
  -e                        等效于 -vE
  -E, --show-ends           在每行结束处显示"$"
  -n, --number              对输出的所有行编号
  -s, --squeeze-blank       不输出多行空行
  -t                        与 -vT 等效
  -T, --show-tabs           将跳格字符显示为^I
  -u                        (被忽略)
  -v, --show-nonprinting    使用^ 和M- 引用，除了LFD和 TAB 之外
      --help            显示此帮助信息并退出
      --version         显示版本信息并退出

示例：
  cat f - g  先输出f 的内容，然后输出标准输入的内容，最后输出g 的内容。
  cat        将标准输入的内容复制到标准输出。

GNU coreutils 在线帮助：<https://www.gnu.org/software/coreutils/>
请向 <http://translationproject.org/team/zh_CN.html> 报告任何翻译错误
完整文档 <https://www.gnu.org/software/coreutils/cat>
```

或者在本地使用：info '(coreutils) cat invocation'

通过 man 命令来获取命令帮助信息

```
[root@rocky9 ~]# man type
```

```
BASH_BUILTINS(1)                    General Commands Manual                   BASH_BUILTINS(1)

NAME
       bash,  :, ., [, alias, bg, bind, break, builtin, caller, cd, command, compgen, complete, com-
       popt, continue, declare, dirs, disown, echo, enable, eval, exec, exit, export, false, fc, fg,
       getopts,  hash,  help, history, jobs, kill, let, local, logout, mapfile, popd, printf, pushd,
       pwd, read, readonly, return, set, shift, shopt, source, suspend,  test,  times,  trap,  true,
       type, typeset, ulimit, umask, unalias, unset, wait - bash built-in commands, see bash(1)

BASH BUILTIN COMMANDS
```

通过 info 命令来获取命令帮助信息

```
[root@rocky9 ~]# info type
```

```
BASH_BUILTINS(1)                                  General Commands Manual                                  BASH_BUILTINS(1)

NAME
       bash, :, ., [, alias, bg, bind, break, builtin, caller, cd, command, compgen, complete, compopt, continue, declare, dirs, disown, echo, enable,
       eval, exec, exit, export, false, fc, fg, getopts, hash, help, history, jobs, kill, let, local, logout, mapfile, popd, printf, pushd, pwd, read,
       readonly, return, set, shift, shopt, source, suspend, test, times, trap, true, type, typeset, ulimit, umask, unalias, unset, wait - bash built-
       in commands, see bash(1)

BASH BUILTIN COMMANDS
       Unless otherwise noted, each builtin command documented in this section as accepting options preceded by - accepts -- to signify the end of the
       options.   The  :, true, false, and test/[ builtins do not accept options and do not treat -- specially.  The exit, logout, return, break, con-
       tinue, let, and shift builtins accept and process arguments beginning with - without requiring --.  Other builtins that  accept  arguments  but
       are not specified as accepting options interpret arguments beginning with - as invalid options and require -- to prevent this interpretation.
       : [arguments]
              No effect; the command does nothing beyond expanding arguments and performing any specified redirections.  The return status is zero.

        .  filename [arguments]
       source filename [arguments]
              Read  and  execute  commands from filename in the current shell environment and return the exit status of the last command executed from
              filename.  If filename does not contain a slash, filenames in PATH are used  to  find  the  directory  containing  filename.   The  file
              searched  for  in PATH need not be executable.  When bash is not in posix mode, the current directory is searched if no file is found in
              PATH.  If the sourcepath option to the shopt builtin command is turned off, the PATH is not searched.  If any  arguments  are  supplied,
              they  become  the positional parameters when filename is executed.  Otherwise the positional parameters are unchanged.  If the -T option
              is enabled, source inherits any trap on DEBUG; if it is not, any DEBUG trap string is saved and restored around the call to source,  and
              source  unsets  the  DEBUG  trap while it executes.  If -T is not set, and the sourced file changes the DEBUG trap, the new value is re-
              tained when source completes.  The return status is the status of the last command exited within the script (0 if no commands  are  exe-
              cuted), and false if filename is not found or cannot be read.

       alias [-p] [name[=value] ...]
              Alias  with  no  arguments or with the -p option prints the list of aliases in the form alias name=value on standard output.  When argu-
              ments are supplied, an alias is defined for each name whose value is given.  A trailing space in  value  causes  the  next  word  to  be
              checked  for alias substitution when the alias is expanded.  For each name in the argument list for which no value is supplied, the name
              and value of the alias is printed.  Alias returns true unless a name is given for which no alias has been defined.

-----Info: (*manpages*)type, 1338 lines --Top-----------------------------------------------------------------------------------------------------------
No menu item 'type' in node '(dir)Top'
```

```
[root@rocky9 ~]# info '(coreutils) cat invocation'
```

```
Next: tac invocation,  Up: Output of entire files

3.1 'cat': Concatenate and write files
======================================

'cat' copies each FILE ('-' means standard input), or standard input if
none are given, to standard output.  Synopsis:

     cat [OPTION] [FILE]...

   The program accepts the following options.  Also see *note Common
options::.

'-A'
'--show-all'
     Equivalent to '-vET'.

'-b'
'--number-nonblank'
     Number all nonempty output lines, starting with 1.

'-e'
     Equivalent to '-vE'.

'-E'
'--show-ends'
     Display a '$' after the end of each line.

'-n'
'--number'
     Number all output lines, starting with 1.  This option is ignored
     if '-b' is in effect.

'-s'
-----Info: (coreutils)cat invocation, 72 lines --Top----------------------------------------------------------------------------------------------------
Welcome to Info version 6.7.  Type H for help, h for tutorial.
```

1.5.2 man手册

基础知识

man介绍

```
    Linux中的man手册是Linux系统中的一个重要工具，它为用户提供了访问内置手册页的途径，这些手册页涵盖了可执行程序、系统调用、库函数等多个方面的内容。
    man手册页通常包括NAME、SYNOPSIS、DESCRIPTION等段落，详细说明命令或函数的使用、选项和返回值。用户可以通过man命令查询特定章节的内容，获取Linux系统的相关知识，无需联网即可轻松获取信息。
    linux系统的man手册位于 /usr/share/man 目录下
```

```
参考资料：
    https://man7.org/linux/man-pages/index.html
    https://man7.org/linux/man-pages/dir_all_alphabetic.html
```

man手册组成

```
Linux的标准man手册通常由多个章节组成，每个章节针对不同的内容类型进行分类。常见的章节包括：
    1：标准命令（Standard commands），如ls、cat等。
    2：系统调用（System calls），如read、write等。
    3：库函数（Library functions），涵盖C标准函数库等。
    4：特殊文件（Special devices），通常是/dev中的设备文件和驱动程序。
    5：文件格式和约定（File formats and conventions）。
    6：游戏和娱乐（Games and toys）。
    7：杂项（Miscellaneous），包括各种工具和宏包。
    8：系统管理命令（Administrative Commands）和守护进程。
```

man操作

```
最有用的操作：
    /KEYWORD     根据KEYWORD关键字，从当前位置向后搜索；不区分字符大小写
    ?KEYWORD     根据KEYWORD关键字，从当前位置向前搜索；不区分字符大小写
    q：退出

作用不大的操作：
    #：跳转至第#行
    1G：回到文件首部
    G：翻至文件尾部
    space，^v，^f，^F：向文件尾翻屏
    b，^b：向文件首部翻屏
    d，^d：向文件尾部翻半屏
    u，^u：向文件首部翻半屏
    RETURN，^N，e，^E or j or ^J：向文件尾部翻一行
    y or ^Y or ^P or k or ^K：向文件首部翻一行
```

man命令参数

```
列出所有帮助
    man -a keyword

搜索man手册所有匹配的页面
    man -k keyword

打印keyword的man帮助文件的路径
    man -w keyword
```

man实践

查看linux系统默认的man手册目录

```
查看整体的帮助手册
[root@rocky9 ~]# ls /usr/share/man/
ca  fr  ko     man2       man3head  man4x  man7   man9x      pt     sv
cs  hu  man0p  man2type   man3p     man5   man7x  mann       pt_BR  tr
da  id  man1   man2x      man3type  man5x  man8   nl         ru     uk
de  it  man1p  man3       man3x     man6   man8x  overrides  sk     zh_CN
es  ja  man1x  man3const  man4      man6x  man9   pl         sr     zh_TW

查看中文的man手册目录
[root@rocky9 ~]# ls /usr/share/man/zh_CN/
man1  man3  man5  man8
```

man环境部署

```
安装man手册
dnf install man-pages
```

man帮助文档

```
查看ls的man帮助文档
[root@rocky9 ~]# man -w ls
/usr/share/man/man1/ls.1.gz

查看ls的man帮助文档格式
[root@rocky9 ~]# file /usr/share/man/man1/ls.1.gz
/usr/share/man/man1/ls.1.gz: gzip compressed data, max compression, from Unix,
original size modulo 2^32 8048

使用man命令解析ls的man帮助手册文件
[root@rocky9 ~]# man /usr/share/man/man1/ls.1.gz
--- 这里与 man ls 的效果一样
```

命令多man手册的解读

```
检索passwd命令的帮助手册都在哪里
[root@rocky9 ~]# whatis passwd
passwd (5)            - password file
passwd (1ossl)        - OpenSSL application commands
passwd (1)            - update user's authentication tokens

确认这三个帮助手册的路径
[root@rocky9 ~]# whereis passwd
passwd: /usr/bin/passwd /etc/passwd /usr/share/man/man5/passwd.5.gz
/usr/share/man/man1/passwd.1ossl.gz /usr/share/man/man1/passwd.1.gz
```

```
第一个是 passwd命令 的man手册
[root@rocky9 ~]# passwd
更改用户 root 的密码 。
新的密码:

第二个是 openssl passwd 命令的 man手册
[root@rocky9 ~]# openssl passwd
Password:

第三个是 /etc/passwd 文件的 man手册
[root@rocky9 ~]# ls /etc/passwd
/etc/passwd
```

```
如果直接 man passwd ，则是查看的是 列表第一的 passwd (5)- password file
```

```
PASSWD(1)                          User utilities                          PASSWD(1)

NAME
       passwd - update user's authentication tokens

SYNOPSIS
       passwd  [-k] [-l] [-u [-f]] [-d] [-e] [-n mindays] [-x maxdays] [-w warndays] [-i inac-
       tivedays] [-S] [--stdin] [-?] [--usage] [username]
```

```
如果要查看具体的passwd的man手册，可以通过如下方式来进行
[root@rocky9 ~]# man 1 passwd
[root@rocky9 ~]# man 5 passwd
[root@rocky9 ~]# man 1ossl passwd
```

同时查看多个man手册

```
[root@rocky9 ~]# man -a passwd
# 默认查看的是 第一个man手册，
# 按Q退出，看到其他的man手册查看方法
--Man-- 下一页: passwd(1ossl) [ 查看 (return) | 跳过 (Ctrl-D) | 退出 (Ctrl-C) ]
# 按Ctrl-D，跳过第二个man手册，直接查看下一个手册
--Man-- 下一页: passwd(5) [ 查看 (return) | 跳过 (Ctrl-D) | 退出 (Ctrl-C) ]
# 按Ctrl-D，跳过第三个man手册，如果后面没有man手册了，就直接退出了
```

1.5.3 软件文档

基础知识

文档介绍

```
    在linux系统中，默认安装的各种软件和命令的帮助文档，默认都是在 /usr/share/doc 目录下。对于后期我们需要安装的一些应用软件，比如nginx、tomcat、kafka等，这些软件在安装的时候，会
    - 在 /usr/share/doc目录下生成各自对应的帮助文档。
    - 在 /usr/share/man目录下生成各自对应的man文档。
```

软件文档

```
对于linux系统里面的后期安装的文档，他们在自己的官方网站上都有相关的帮助手册。
    http://www.github.com
    https://www.kernel.org/doc/html/latest/http://httpd.apache.org
    http://www.nginx.org
    https://mariadb.com/kb/en
    https://dev.mysql.com/doc/
    http://tomcat.apache.org
    https://jenkins.io/zh/doc/
    https://kubernetes.io/docs/home/
    https://docs.openstack.org/train/
    http://www.python.org
    http://php.net
    http://nginx.org/en/docs/
```

```
对于linux系统里面的默认安装应用软件，通过发行版官方的文档光盘或网站可以获得安装指南、部署指南、虚拟化指南等
    http://kbase.redhat.com
    http://www.redhat.com/docs
    http://access.redhat.com
    https://help.ubuntu.com/lts/serverguide/index.html http://tldp.org
```

简单实践

查看默认的命令帮助手册

```
查看系统默认的tar命令的帮助文档
[root@rocky9 ~]# ls /usr/share/doc/tar
AUTHORS  ChangeLog  NEWS  README  THANKS
```

第三方软件帮助手册示例

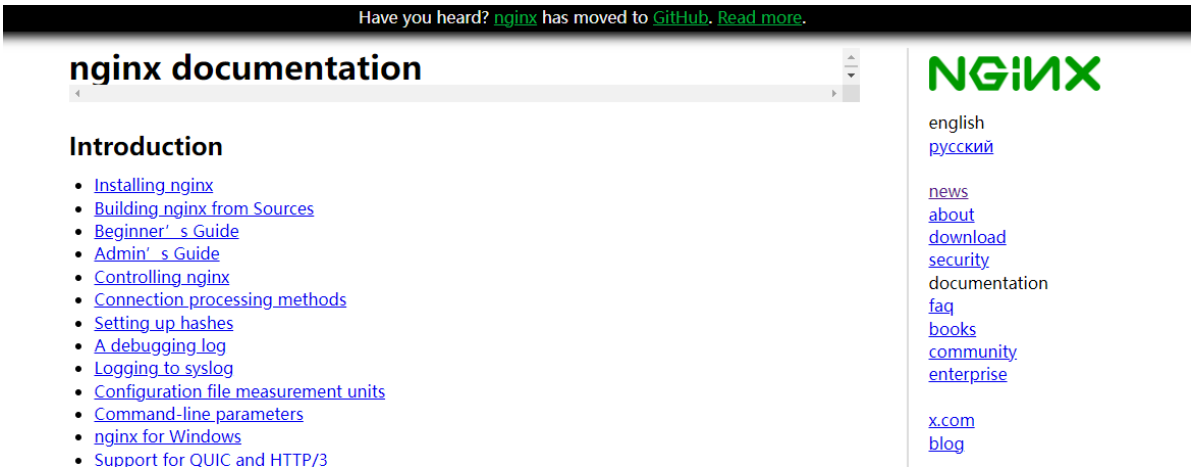
```
目前系统中，没有该文档
[root@rocky9 ~]# ls /usr/share/doc/nginx
ls: 无法访问 '/usr/share/doc/nginx': 没有那个文件或目录
```

```
安装nginx软件
[root@rocky9 ~]# dnf install nginx -y

查看安装的软件
[root@rocky9 ~]# dnf list installed | grep nginx
nginx.x86_64                    1:1.20.1-16.el9_4.1     @appstream
nginx-core.x86_64               1:1.20.1-16.el9_4.1     @appstream
nginx-filesystem.noarch         1:1.20.1-16.el9_4.1     @appstream
```

```
查看软件安装的man手册
```

```
[root@rocky9 ~]# rpm -ql nginx
/usr/bin/nginx-upgrade
/usr/lib/systemd/system/nginx.service
/usr/share/man/man3/nginx.3pm.gz
/usr/share/man/man8/nginx-upgrade.8.gz
/usr/share/man/man8/nginx.8.gz
...

查看软件的帮助文档
[root@rocky9 ~]# find /usr/share/doc -name "*nginx*"
/usr/share/doc/nginx-core
[root@rocky9 ~]# ls /usr/share/doc/nginx-core/
CHANGES   README   README.dynamic
```

查看nginx官网的帮助手册



1.5.4 精简帮助

基础知识

需求

在linux系统中，有很多命令的帮助手册，为了所谓的"美观格式化"，故意将命令的帮助手册，做的又大又长，有用的没用的参数，一股脑全部给你展示出来，让人看起来非常的不舒服。比如tar命令

```
[root@rocky9 ~]# tar --help
用法: tar [选项...] [FILE]...
GNU 'tar' saves many files together into a single tape or disk archive, and can
restore individual files from the archive.

Examples:
  tar -cf archive.tar foo bar  # Create archive.tar from files foo and bar.
  tar -tvf archive.tar         # List all files in archive.tar verbosely.
  tar -xf archive.tar          # Extract all files from archive.tar.

 主操作模式:
  -A, --catenate, --concatenate   追加 tar 文件至归档
  -c, --create                创建一个新归档
      --delete                从归档(非磁带!)中删除
  -d, --diff, --compare       找出归档和文件系统的差异
  -r, --append                追加文件至归档结尾
      --test-label            测试归档卷标并退出
  -t, --list                  列出归档内容
```

```
  -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

 操作修饰符:

      --check-device         当创建增量归档时检查设备号(默认)
  -g, --listed-incremental=FILE   处理新式的 GNU 格式的增量备份
  -G, --incremental          处理老式的 GNU 格式的增量备份
      --hole-detection=TYPE  用于探测holes 的技术
      --ignore-failed-read
                             当遇上不可读文件时不要以非零值退出
      --level=NUMBER         所创建的增量列表归档的输出级别
      --no-check-device      当创建增量归档时不要检查设备号
      --no-seek              归档不可检索
      ...
```

解决

```
    在互联网上，有人将通过 -h|--help|man 获取出来的信息，进行了高度的精简，仅仅显示出来常用的选项，从而让用户查看命令帮助的时候，感觉效率非常高。
    这个软件就是 TLDR(Too Long Didn't Read)
```

```
官方源码：https://github.com/tldr-pages/tldr

注意：
    tldr在使用的时候，依赖于github相关的资源，所以最好提前对于节点主机进行github的主机名记录解析
    185.199.111.133 raw.githubusercontent.com
    在国内访问测试的事后，如果不对网络进行设置的话，有可能会出现一些超时中断的问题。
```

简单实践

环境部署

```
安装依赖环境
[root@rocky9 ~]# dnf install -y python3-pip

安装tldr
[root@rocky9 ~]# pip3 install tldr
Collecting tldr
  Downloading tldr-3.3.0-py3-none-any.whl (12 kB)
Collecting termcolor
  Downloading termcolor-2.4.0-py3-none-any.whl (7.7 kB)
Collecting shtab>=1.3.10
  Downloading shtab-1.7.1-py3-none-any.whl (14 kB)
Collecting colorama
  Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Installing collected packages: termcolor, shtab, colorama, tldr
Successfully installed colorama-0.4.6 shtab-1.7.1 termcolor-2.4.0 tldr-3.3.0
WARNING: Running pip as the 'root' user can result in broken permissions and
conflicting behaviour with the system package manager. It is recommended to use
a virtual environment instead: https://pip.pypa.io/warnings/venv
```

```
配置 tldr
echo '185.199.111.133 raw.githubusercontent.com' >> /etc/hosts
```

测试效果

```
[root@rocky9 ~]# tldr tar

  tar

  归档实用程序。
  通常与压缩方法结合使用，例如 gzip 或 bzip2.
  更多信息：https://www.gnu.org/software/tar.

  - 创建存档并将其写入文件：
    tar cf 目标.tar 文件1 文件2 ...

  - 创建一个 gzip 压缩文件并将其写入文件：
    tar czf target.tar.gz file1 file2 ...

  - 使用相对路径从目录创建一个 gzip 压缩文件：
    tar czf target.tar.gz --directory=path/to/directory .

  - 详细地将（压缩的）存档文件提取到当前目录中：
    tar xvf source.tar[.gz|.bz2|.xz]

  - 将（压缩的）存档文件解压缩到目标目录中：
    tar xf source.tar[.gz|.bz2|.xz] --directory=directory

  - 创建压缩存档并将其写入文件，使用存档后缀确定压缩程序：
    tar caf target.tar.xz file1 file2 ...

  - 详细列出 tar 文件的内容：
    tar tvf source.tar

  - 从存档文件中提取与模式匹配的文件：
    tar xf source.tar --wildcards "*.html"
```

```
[root@rocky9 ~]# tldr nginx

  nginx

  Nginx web server.
  More information: https://nginx.org/en/.

  - Start server with the default configuration file:
    nginx

  - Start server with a custom configuration file:
    nginx -c configuration_file

  - Start server with a prefix for all relative paths in the configuration file:
    nginx -c configuration_file -p prefix/for/relative/paths

  - Test the configuration without affecting the running server:
    nginx -t

  - Reload the configuration by sending a signal with no downtime:
    nginx -s reload
```

```
[root@rocky9 ~]# tldr tar

  tar

  归档实用程序。
  通常与压缩方法结合使用，例如 gzip 或 bzip2.
  更多信息：https://www.gnu.org/software/tar.

  - 创建存档并将其写入文件：
    tar cf 目标.tar 文件1 文件2 ...

  - 创建一个 gzip 压缩文件并将其写入文件：
    tar czf target.tar.gz file1 file2 ...

  - 使用相对路径从目录创建一个 gzip 压缩文件：
    tar czf target.tar.gz --directory=path/to/directory .

  - 详细地将（压缩的）存档文件提取到当前目录中：
    tar xvf source.tar[.gz|.bz2|.xz]

  - 将（压缩的）存档文件解压缩到目标目录中：
    tar xf source.tar[.gz|.bz2|.xz] --directory=directory

  - 创建压缩存档并将其写入文件，使用存档后缀确定压缩程序：
    tar caf target.tar.xz file1 file2 ...

  - 详细列出 tar 文件的内容：
    tar tvf source.tar

  - 从存档文件中提取与模式匹配的文件：
    tar xf source.tar --wildcards "*.html"
```

```
[root@rocky9 ~]# tldr nginx

  nginx

  Nginx web server.
  More information: https://nginx.org/en/.

  - Start server with the default configuration file:
    nginx

  - Start server with a custom configuration file:
    nginx -c configuration_file

  - Start server with a prefix for all relative paths in the configuratio
    nginx -c configuration_file -p prefix/for/relative/paths

  - Test the configuration without affecting the running server:
    nginx -t

  - Reload the configuration by sending a signal with no downtime:
    nginx -s reload
```