Simi Documentation / Linux

Author: Craig Brown
Version: 1.0.0
Date: Feb 2, 2025
Source: https://github.com/saidake/simi/tree/master/docs/pdf

Linux / Concept

System Version

linux 发行版	基于社区开发的 debian、archlinux，Mint　Fedora，基于商业开发的 Red Hat Enterprise Linux、SUSE、Oracle Linux 等
linux 内核版	android，intel
宏观操作系统概念	管理计算机系统资源，向下封装硬件，向上提供操作接口　【系统编程 - 系统接口调用】
Unix　操作系统（linux 兼容 Unix）	不同于 linux，大型服务器，收费
POSIX 标准	定义了操作系统应该为应用程序提供的接口标准
磁盘分区方案	swap　Logical　2G　　/boot　Logical　300M　　/　Primary　RemainingSpace

System Directory

System

/bin　　所有用户都可用的 系统命令程序
/sbin　　超级用户才能使用的 系统命令程序【用户，启动，网络，防火墙】
/tmp　　所有程序的临时数据存放位置（可以随时删除）
/boot　　系统启动配置文件　　/boot/vmlinuz 内核文件　/boot/grub 系统启动选项设置
/dev　　硬件，设备配置文件
/etc　　系统和程序配置文件
/lib　　软件和系统 公用库
　/media　　自动挂载的设备目录（U 盘，光盘目录）
　/mnt　　手动挂载的临时设备目录
/opt　　用户机的 额外安装软件默认目录，默认为空
/var　　常常修改的文件

User

/home　　用户目录/home/icons　　鼠标，图标主题
/root　　root 用户目录
/usr　　所有用户都可使用的软件的安装位置
/usr/include　头文件
/usr/local　　用户级的程序目录，可以理解为 C:/Progrem Files/ 。用户自己编译的软件默认会安装到这个目录下。
/usr/local/src　　用户级的源码目录。
/usr/local/lib　　用户公用库
/usr/local/bin　　存放所有用户都可用的与本地机器无关的程序
/usr/local/sbin　　存放超级用户才能使用的与本地机器无关的程序
　/usr/software　　安装系统环境（手动添加）
　/usr/lib　　可理解为 C:/Windows/System32 。
　/usr/src　　系统级的源码目录。
　/usr/lib　　系统公用库
　/usr/bin　　存放所有用户都可用的应用程序

Boot

内核的引导：当计算机打开电源后，首先是 BIOS 开机自检，按照 BIOS 中设置的启动设备（通常是硬盘）来启动。
　　操作系统接管硬件以后，首先读入 /boot 目录下的内核文件。
运行 init：许多程序需要开机启动。它们在 Windows 叫做"服务"（service），在 Linux 就叫做"守护进程"（daemon）

init 进程会运行这些开机启动的程序

不同的场合需要启动不同的程序，比如用作服务器时，需要启动 Apache，用作桌面就不需要，

Linux 允许为不同的场合，分配不同的开机启动程序，这就叫做"运行级别"（runlevel）。也就是说，启动时根据"运行级别"，确定要运行哪些程序。

运行级别 0：系统停机状态，系统默认运行级别不能设为 0，否则不能正常启动

运行级别 1：单用户工作状态，root 权限，用于系统维护，禁止远程登陆

运行级别 2：多用户状态(没有 NFS)

运行级别 3：完全的多用户状态(有 NFS)，登陆后进入控制台命令行模式

运行级别 4：系统未使用，保留

运行级别 5：X11 控制台，登陆后进入图形 GUI 模式

运行级别 6：系统正常关闭并重启，默认运行级别不能设为 6，否则不能正常启动系统初始化

系统初始化： /etc/init.d 存放系统服务启动脚本，用 service 命令可以执行 init.d 目录中相应服务的脚本 // service resin start 可启动/etc/init.d/resin 脚本

/etc/init.d 是只想/etc/rc.d/init.d 的软链接

/etc/rc.d/rc.local rpm 系统给用户自己配置启动项（chmod +x /etc/rc.d/rc.local 需要添加执行权限）

/etc/rc.d/rc.sysinit

在 init 的配置文件中有这么一行： si::sysinit:/etc/rc.d/rc.sysinit 它调用执行了/etc/rc.d/rc.sysinit，

而 rc.sysinit 是一个 bash shell 的脚本，它主要是完成一些系统初始化的工作，rc.sysinit 是每一个运行级别都要首先运行的重要脚本。

它主要完成的工作有：激活交换分区，检查磁盘，加载硬件模块以及其它一些需要优先执行任务。

/etc/rc.d/rc

l5:5:wait:/etc/rc.d/rc 5

这一行表示以 5 为参数运行/etc/rc.d/rc，/etc/rc.d/rc 是一个 Shell 脚本，它接受 5 作为参数，去执行/etc/rc.d/rc5.d/目录下的所有的 rc 启动脚本，

/etc/rc.d/rc5.d/目录中的这些启动脚本实际上都是一些连接文件，而不是真正的 rc 启动脚本，真正的 rc 启动脚本实际上都是放在/etc/rc.d/init.d/目录下。

而这些 rc 启动脚本有着类似的用法，它们一般能接受 start、stop、restart、status 等参数。

/etc/rc.d/rc5.d/

目录下的 rc 启动脚本通常是 K 或 S 开头的连接文件，对于以 S 开头的启动脚本，将以 start 参数来运行。

而如果发现存在相应的脚本也存在 K 打头的连接，而且已经处于运行态了(以/var/lock/subsys/下的文件作为标志)，

则首先以 stop 为参数停止这些已经启动了的守护进程，然后再重新运行。

这样做是为了保证是当 init 改变运行级别时，所有相关的守护进程都将重启。

至于在每个运行级中将运行哪些守护进程，用户可以通过 chkconfig 或 setup 中的"System Services"来自行设定。

/etc/profile.d/

目录下的所有 sh 脚本开机都会自动执行

建立终端 rc 执行完毕后，返回 init。这时基本系统环境已经设置好了，各种守护进程也已经启动了。

init 接下来会打开 6 个终端，以便用户登录系统。在/etc/inittab 中的以下 6 行就是定义了 6 个终端：

/etc/inittab

1:2345:respawn:/sbin/mingetty tty1

2:2345:respawn:/sbin/mingetty tty2

3:2345:respawn:/sbin/mingetty tty3

4:2345:respawn:/sbin/mingetty tty4

5:2345:respawn:/sbin/mingetty tty5

6:2345:respawn:/sbin/mingetty tty6

从上面可以看出在 2、3、4、5 的运行级别中都将以 respawn 方式运行 mingetty 程序，mingetty 程序能打开终端、设置模式。

同时它会显示一个文本登录界面，这个界面就是我们经常看到的登录界面，在这个登录界面中会提示用户输入用户名，

而用户输入的用户将作为参数传给 login 程序来验证用户的身份。

用户登录系统　命令行登录，ssh 登录，图形界面登录

Linux 的账号验证程序是 login，login 会接收 mingetty 传来的用户名作为用户名参数。

然后 login 会对用户名进行分析：如果用户名不是 root，且存在 /etc/nologin 文件，login 将输出 nologin 文件的内容，然后退出。

这通常用来系统维护时防止非 root 用户登录。只有/etc/securetty 中登记了的终端才允许 root 用户登录，如果不存在这个文件，则 root 用户可以在任何终端上登录。

/etc/usertty　文件用于对用户作出附加访问限制，如果不存在这个文件，则没有其他限制。

/etc/securetty　终端安全配置

/etc/nologin　只要存在这个文件，会禁用非 root 用户，文件中的内容将会显示给用户，会一闪而过

Login

agetty　用于打开一个 tty 端口，提示登录名并调用 /bin/login 命令。agetty 一般由 init 调用【-i 在显示登陆提示符前不显示 /etc/issue 文件的内容 】

-f : do not perform authentication【login 参数】

-o, --login-options <opts> : options that are passed to login

-n, --skip-login : do not prompt for login

-a, --auto-login <user> : login the specified user automatically

/var/run/utmp　系统状态文件，记录着现在登录的用户。

/etc/issue　登录前打印其中的内容

/dev/console　报告错误的地方

/etc/inittab　init 的配置文件

File Type

普通文件　-rw-r--r--

目录文件　drwxr-xr-x

设备文件　crw-------　（char 设备）　brw-rw----　（block 设备）

链接文件　lrwxrwxrwx　（快捷方式）

管道文件　s

r 读-无法查看　w 写-无法修改　x 可执行-进入目录　- 无权限　（-rwxr-xr-x　文件所有者　用户组　其他用户）

软链接　链接普通文件或目录 改变同步　（软链接不占空间，源文件删除则失效，等同 windows）

硬链接　链接普通文件　改变同步　（两个文件都占空间，修改同步的两个不同文件，删除互不影响）

VIM

vi xxpath +5　文件不存在就创建，vim 打开文件会创建.swp 隐藏文件【+5 快速到达行号 】

esc 默认命令行模式　:q! 退出不保存　:wq 退出保存　:cq　带一个错误码退出　:set nu　显示行号　v 字符选择可复制粘贴 y p

ctrl PageDown　ctrl PageUp　Ctrl f 向前翻屏　Ctrl b 向后翻屏

x　删除光标后一个字符　X　删除光标前一个字符　dd　删除行　d27d 删除 27 行　gg　到文件开头

h　左移光标　l 右移光标　j 光标下移一行　k 光标上移一行

o　插入新行（进入插入模式）

/xxx　正向查找　n 下一次匹配　N 上一次匹配

?xxx　反向查找

I 插入模式

v 视图模式

G 选择所有

ctrl v 多选模式

LVS

LVS 由 2 部分程序组成，包括 ipvs 和 ipvsadm。

ipvs(ip virtual server)： 一段代码工作在内核空间，叫 ipvs (linux 模块)，是真正生效实现调度的代码。

ipvsadm： 另外一段是工作在用户空间，叫 ipvsadm，负责为 ipvs 内核框架编写规则，定义谁是集群服务，而谁是后端真实的服务器(Real Server)。

相关术语

VG Volume Group，卷组

DS Director Server，指的是前端负载均衡器节点

RS Real Server，后端真实的工作服务器

VIP 向外部直接面向用户请求，作为用户请求的目标的 IP 地址

DIP Director Server IP，主要用于和内部主机通讯的 IP 地址

RIP Real Server IP，后端服务器的 IP 地址

CIP Client IP，访问客户端的 IP 地址

工作模式（性能比较：DR>TUN>NAT>FULLNAT）

DR 直接路由模式

tun 隧道模式

nat 路由转发模式

fullnat

负载均衡十种算法

轮循调度 rr

均等地对待每一台服务器，不管服务器上的实际连接数和系统负载

加权轮调 wrr

调度器可以自动问询真实服务器的负载情况，并动态调整权值

最少链接 lc

动态地将网络请求调度到已建立的连接数最少的服务器上

如果集群真实的服务器具有相近的系统性能，采用该算法可以较好的实现负载均衡

加权最少链接 wlc

调度器可以自动问询真实服务器的负载情况，并动态调整权值

带权重的谁不干活就给谁分配，机器配置好的权重高

基于局部性的最少连接调度算法 lblc

这个算法是请求数据包的目标 IP 地址的一种调度算法，该算法先根据请求的目标 IP 地址寻找最近的该目标 IP 地址所有使用的服务器，如果这台服务器依然可用，并且有能力处理该请求，调度器会尽量选择相同的服务器，否则会继续选择其它可行的服务器

复杂的基于局部性最少的连接算法 lblcr

记录的不是要给目标 IP 与一台服务器之间的连接记录，它会维护一个目标 IP 到一组服务器之间的映射关系，防止单点服

务器负载过高。

目标地址散列调度算法 dh

该算法是根据目标 IP 地址通过散列函数将目标 IP 与服务器建立映射关系，出现服务器不可用或负载过高的情况下，发往该目标 IP 的请求会固定发给该服务器。

(源地址散列调度算法 sh

与目标地址散列调度算法类似，但它是根据源地址散列算法进行静态分配固定的服务器资源。

最少期望延迟 sed

不考虑非活动链接，谁的权重大，优先选择权重大的服务器来接收请求，但权重大的机器会比较忙

永不排队 nq

无需队列，如果有 realserver 的连接数为 0 就直接分配过去

Linux / Usage

系统信息

/proc/version 操作系统版本信息

// Linux version 4.19.0-6.el7.ucloud.x86_64 (root@2d6b517b2939) (gcc version 4.8.5 20150623 (Red Hat 4.8.5-16) (GCC)) #1 SMP Wed Feb 12 07:32:16 UTC 2020

/etc/issue /etc/centos-release 操作系统发行信息

//CentOS Linux release 7.6.1810 (Core)

/etc/security/limits.conf 设置系统可并发打开的文件数量为 65535

/etc/default/grub GRUB 启动菜单设置

GRUB_TIMEOUT=0 GRUB_HIDDEN_TIMEOUT=0 GRUB_HIDDEN_TIMEOUT_QUIET=true

grub2-mkconfig -o /boot/grub2/grub.cfg 生成有效的配置文件

/etc/ld.so.conf.d/local.conf 模块共享目录配置文件 //EX：/usr/local/lib sudo ldconfig -v 重新加载共享模块

网卡配置

配置目录：

/etc/**network**/interfaces

/etc/**sysconfig/network-scripts**/ifcfg-eth0

DEVICE=eth0 网卡名

HWADDR=08:00:27:C7:1B:22 给数据包添加自定义的 MAC 地址

TYPE=Ethernet 网络类型

UUID=1ebxxxxxxxxxxxxxxxxxx 每个网卡的唯一识别码

ONBOOT=yes 系统启动时激活网卡

BOOTPROTO=static IP 类型【static 静态 IP dhcp 自动设置不重复的 IP none 无，不指定】

IPADDR=192.168.2.2 IP

NETMASK=255.255.255.0 子网掩码

GATEWAY=192.168.2.1 网关

开放端口

vi /etc/sysconfig/iptables 开放端口设置（增强型 selinux，可配置：/etc/selinux/config 当特权端口< 1024 的时候，须要 root 访问权限）

-A INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT #允许 80 端口通过防火墙

-A INPUT -m state --state NEW -m tcp -p tcp --dport 3306 -j ACCEPT #允许 3306 端口通过防火墙)

service iptables restart 重启服务，应用规则（等同/etc/init.d/iptables restart）

-A INPUT -J state --state NEW -m tcp -p tcp --dport 21 -j ACCEPT

uhttpd

/etc/config/uhttpd > /etc/init.d/uhttpd restart 重启 uhttpd

config uhttpd 'main'

list listen_http '0.0.0.0:80' 监听端口为 80，协议为 http，使用 ipv4

list listen_http '[::]:80' 监听端口为 80，协议为 http，使用 ipv6

list listen_https '0.0.0.0:81' 监听端口为 80，协议为 https，使用 ipv4

option home '/root/webroot' 网站根目录

option max_requests '3' 最大请求

option max_connections '100' 最大 TCP 链接

mv /overlay/upper /tmp/backup/

sshd/sftp

安装:

sudo apt-get install sshd

sudo apt-get install openssh-server

/etc/ssh/sshd_config >

PermitRootLogin yes root 用户登录

Port 22 默认 ssh 端口，生产环境中建议改成五位数的端口

#AddressFamily any 地址家族，any 表示同时监听 ipv4 和 ipv6 地址

#ListenAddress 0.0.0.0 监听本机所有 ipv4 地址

#ListenAddress :: 监听本机所有 ipv6 地址

HostKey /etc/ssh/ssh_host_rsa_key ssh 所使用的 RSA 私钥路径

#HostKey /etc/ssh/ssh_host_dsa_key

HostKey /etc/ssh/ssh_host_ecdsa_key ssh 所使用的 ECDSA 私钥路径

HostKey /etc/ssh/ssh_host_ed25519_key ssh 所使用的 ED25519 私钥路径

Subsystem sftp /usr/libexec/openssh/sftp-server <==配置一个外部子系统 sftp 及其路径

偏好配置

/sdk/logs/xxx 程序日志目录

/sdk/cfg/xxx 程序配置目录

Linux / Configuration

Boot

/etc/systemd/system/getty.target.wants

This directory contains symbolic links to services related to terminal login (getty) that systemd manages.

It's used to configure which getty services are started by default.

Password-free login settings

ExecStart=-/sbin/agetty -o '-p -f root' -n -a root --noclear %I $TERM

/etc/inputrc

The configuration file for the readline library, which controls keybindings and input behavior for command-line interfaces like bash.

set bell-style none Remove beeping sound

Profile

/etc/profile.d/

A directory containing shell scripts that are sourced during login (these scripts will be executed when the system boots or connects to a remote server) .

These scripts can set environment variables or configure the shell environment for all users.

/etc/profile

A system-wide shell initialization script executed during login.

It sets up environment variables and startup configurations for all users.

Non-Login Shell

The /etc/profile file may not be sourced when the terminal is opened because the terminal session is starting as a non-login shell rather than a login shell.

Priority

~/.bash_profile > /etc/profile

(When ssh logs in to a non-interactive terminal, it cannot access /etc/profile, so the startup command needs to be added to .bashrc)

export JAVA_HOME=/root/up/jdk1.8.0_64 ($ will identify other exposed environment variables, which makes them accessible to other programs or environment variables.)

export PATH=$PATH:/home/a/s:$JAVA_HOME/bin

source /etc/profile Make the environment variables take effect immediately without logging out and back in

echo $PATH echo $JAVA_HOME View environment variables and custom environment variables.

~/.bash_profile or ~/.bashrc

User-specific configuration files for bash. ~/.bash_profile is executed at login, while ~/.bashrc is run during the startup of non-login interactive shells.

Network

/etc/resolv.conf

This file specifies the system's DNS settings, including nameserver addresses. It's used for domain name resolution.

nameserver 8.8.8.8

nameserver 8.8.4.4

/etc/hosts

A network interface configuration file, used to define settings such as IP address, gateway, and DNS for the specific network interface (e.g., ens33).

172.16.186.110 master.openshift.com master (The third parameter is a URL abbrevation, which can be used to access the corresponding address.)

172.16.186.111 node1.openshift.com node1

172.16.186.112 node2.openshift.com node2

/etc/sysconfig/network-scripts/ifcfg-ens33

A network interface configuration file, used to define settings such as IP address, gateway, and DNS for the specific network interface (e.g., ens33).

User

/etc/group

Contains information about the system's user groups, listing group names, IDs, and member users.

group_name:x:GID:members

group_name

The name of the group.

x

Placeholder for the password (stored in /etc/shadow).

GID

Group ID, identifying the user’s primary group.

members

A comma-separated list of users who belong to the group (optional).

```
root:x:0:
wheel:x:10:user1,user2
```

(When creating a new user, a user group with the same name as the user name is created by default)

/etc/passwd

A file that stores user account information, including usernames, user IDs, home directories, and shells. It is essential for user authentication.

username:x:UID:GID:comment:home_directory:shell

username

The user's login name.

x

Placeholder for the password (stored in /etc/shadow).

UID

User ID, a unique number for the user.

GID

Group ID, identifying the user’s primary group.

comment

Optional user information (e.g., full name).

home_directory

The path to the user's home directory.

shell

The default shell for the user (e.g., /bin/bash).

```
root:x:0:0:root:/root:/bin/bash
john:x:1001:1001:John Doe:/home/john:/bin/bash
mysql:x:27:27:MySQL Server:/var/lib/mysql:/bin/false
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
```

Permanently change the hostname:

/etc/sysconfig/network HOSTNAME=Banana

/etc/hosts 127.0.0.1 newname

/etc/hostname Banana

Linux / Command

File

touch <path>

文件不存在就新建，并修改文件为当前时间

`touch {1..4}.text` 创建 4 个文件 `{m..n}` 表示数列

rm <path>

删除文件-多项为先后删除

【-r 删除目录或文件 -f 不显示提示】

`sudo rm ./se ./up -rf`

cp <path> <target-path>

拷贝文件（路径有空格需要加引号 "/IntelliJ IDEA.app/Content/bin"）

【-r 拷贝目录到新目录 -v 显示拷贝进度 -f 不显示提示】

mv <path> <path> xxtarpath

移动文件（修改文件名 mv a b）

【-r 移动目录 -v 显示移动进度 -f 不显示提示】

ln <path> xxlnkpath 建立硬链接快捷方式-完全相同

软链接 链接普通文件或目录 改变同步 （软链接不占空间，源文件删除则失效，等同 windows）

硬链接 链接普通文件 改变同步 （两个文件都占空间，修改同步的两个不同文件，删除互不影响）

【-s 软链接快捷方式，前一个链接用绝对路径 ln /a/x.exe name】

dirname <path>

Get the directory portion of a given path (without the file name)

pwd

显示当前路径

cat <path>

查看文件文本内容

tee <path> << XXXEOF > xxx.txt 输入文件内容

【XXXEOF 自定义终止符】

//EX：python xxx.py args1 | tee -a data/log.txt 打印日志的同时还把这些内容保存在日志文件

mkdir <path>

创建目录

【-p 创建多个目录 mkdir ./a/b/c -p】

ls <path>

-r <path>. Check path for a specific directory.

查看目录 （1 root root 子目录或硬链接数量 所有者 用户组 --- 14 Mar 30 18:15 文件大小 修改时间）【-a 显示所有文件 -l 列表查看 -h 可读显示文件大小】

cd <path>

跳转目录 （cd ~ 切换到当前用户目录 cd - 上一次的目录 cd * 切换到第一个）

sed -i "xxx" xxpath

依照脚本的指令来处理、编辑文本文件

【-i 修改内容 -n 安静模式，只列出特殊处理的那一行，和 p 标志一起使用 -e 直接在命令列模式上进行 sed 的动作编辑】

【-f 直接将 sed 的动作写在一个文件内， -f filename 则可以运行 filename 内的 sed 动作

-r sed 的动作支持的是延伸型正规表示法的语法，默认是基础正规表示法语法】

【-i 直接修改读取的文件内容，而不是输出到终端】

^	锚定行的开始	//	/^sed/匹配所有以 sed 开头的行。
$	锚定行的结束	//	/sed$/匹配所有以 sed 结尾的行。
.	匹配一个非换行符的字符	//	/s.d/匹配 s 后接一个任意字符，然后是 d。
*	匹配零或多个字符	//	/*sed/匹配所有模板是一个或多个空格后紧跟 sed 的行。
[]	匹配一个指定范围内的字符	//	/[Ss]ed/匹配 sed 和 Sed。
[^]	匹配一个不在指定范围内的字符	//	/[^A-RT-Z]ed/匹配不包含 A-R 和 T-Z 的一个字母开头，紧跟 ed 的行。
\(..\)	保留匹配的字符	//	s/\(love\)able/\1rs，loveable 被替换成 lovers。
&	保留搜索字符用来替换其他字符	//	s/love/**&**/，love 这成**love**。
\<	锚定单词的开始	//	/\<love/匹配包含以 love 开头的单词的行。

\>　　锚定单词的结束　　//　　/love\>/匹配包含以 love 结尾的单词的行。

x\{m\}　　重复字符 x，m 次　　//　　/0\{5\}/匹配包含 5 个 o 的行。

x\{m,\}　　重复字符 x，至少 m 次　　//　　/o\{5,\}/匹配至少有 5 个 o 的行。

x\{m,n\}　　重复字符 x，至少 m 次，不多于 n 次　　//　　/o\{5,10\}/匹配 5--10 个 o 的行。

删除：d 命令

sed '2d' xxx-　　删除 xxx 文件的第二行

sed '2,$d' xxx　　删除 xxx 文件的第二行到末尾所有行

sed '$d' xxx　　删除 xxx 文件的最后一行

sed '/test/'d xxx　　删除 xxx 文件所有包含 test 的行

替换：s 命令

sed 's/test/mytest/g' xxx　　在整行范围内把 test 替换为 mytest。如果没有 g 标记，则只有每行第一个匹配的 test 被替换成 mytest。

sed -n 's/^test/mytest/p' xxx　　(-n）选项和 p 标志一起使用表示只打印那些发生替换的行。也就是说，如果某一行开头的 test 被替换成 mytest，就打印它。

sed 's/^192.168.0.1/&localhost/' xxx　　& 符号表示替换换字符串中被找到的部份。所有以 192.168.0.1 开头的行都会被替换成它自已加 localhost，变成 192.168.0.1localhost。

sed -n 's/\(love\)able/\1rs/p' xxx　　love 被标记为 1，所有 loveable 会被替换成 lovers，而且替换的行会被打印出来。

sed 's#10#100#g' xxx　　不论什么字符，紧跟着 s 命令的都被认为是新的分隔符，所以，“#”在这里是分隔符，代替了默认的“/”分隔符。表示把所有 10 替换成 100。

选定行的范围：逗号

sed -n '/test/,/check/p' xxx　　所有在模板 test 和 check 所确定的范围内的行都被打印。

sed -n '5,/^test/p' xxx　　打印从第五行开始到第一个包含以 test 开始的行之间的所有行。

sed '/test/,/check/s/$/sed test/' xxx　　对于模板 test 和 check 之间的行，每行的末尾用字符串 sed test 替换。

多点编辑：e 命令

sed -e '1,5d' -e 's/test/check/' xxx　　(-e）选项允许在同一行里执行多条命令。如例子所示，第一条命令删除 1 至 5 行，第二条命令用 check 替换 test。命令的执行顺序对结果有影响。如果两个命令都是替换命令，那么第一个替换命令将影响第二个替换命令的结果。

sed --expression='s/test/check/' --expression='/love/d' xxx　　一个比-e 更好的命令是--expression。它能给 sed 表达式赋值。

从文件读入：r 命令

sed '/test/r file' xxx　　file 里的内容被读进来，显示在与 test 匹配的行后面，如果匹配多行，则 file 的内容将显示在所有匹配行的下面。

写入文件：w 命令

sed -n '/test/w file' xxx　　在 xxx 中所有包含 test 的行都被写入 file 里。

追加命令：a 命令

sed '/^test/a\\

this is a dog' xxx　　'this is a dog' 被追加到以 test 开头的行后面，sed 要求命令 a 后面有一个反斜杠。

插入：i 命令

sed '/test/i\\

new line

----------' xxx

如果 test 被匹配，则把反斜杠后面的文本插入到匹配行的前面。

下一个：n 命令

sed '/test/{ n; s/aa/bb/; }' xxx　　如果 test 被匹配，则移动到匹配行的下一行，替换这一行的 aa，变为 bb，并打印

该行，然后继续。

变形：y 命令

sed '1,10y/abcde/ABCDE/' xxx　　把 1--10 行内所有 abcde 转变为大写，注意，正则表达式元字符不能使用这个命令。

退出：q 命令

sed '10q' xxx　　打印完第 10 行后，退出 sed。

保持和获取：h 命令和 G 命令

sed -e '/test/h' -e '/$/G' xxx　　在 sed 处理文件的时候，每一行都被保存在一个叫模式空间的临时缓冲区中。除非行被删除或者输出被取消，否则所有被处理的行都将打印在屏幕上。接着模式空间被清空，并存入新的一行等待处理。

在这个例子里，匹配 test 的行被找到后，将存入模式空间，h 命令将其复制并存入一个称为保持缓存区的特殊缓冲区内。

第二条语句的意思是，当到达最后一行后，G 命令取出保持缓冲区的行，然后把它放回模式空间中，且追加到现 在已经存在于模式空间中的行的末尾。

在这个例子中就是追加到最后一行。简单来说，任何包含 test 的行都被复制并追加到该文件的末尾。

保持和互换：h 命令和 x 命令

sed -e '/test/h' -e '/check/x' xxx　　互换模式空间和保持缓冲区的内容。也就是把包含 test 与 check 的行互换。

Find File

find path

在 path 内限制 name 和 size 查找文件（硬盘搜索）

-name <file-name>　　xxx 根据文件名

-iname xxx　　Case insensitive, support wildcard.

-iname “foo*.???”

-size +2M　　根据文件大小---大于+2M 小于-2M 等于 2M】

-exec <command>　　Execute commands on the resulting pathes.

locate path

查找包含 path 的文件路径（搜索文件数据库/var/lib/slocate/slocate.db）

// locate /etho 查找包含/etho 的路径

file path　　显示文件详细信息

more xxpath　　分页查看文件内容

grep -n xxx xxpath　　文本文件中查找关键字 【-10 显示匹配的前后 10 行内容 -A 10 显示匹配的前面 10 行 -B 10 显示匹配的后面 10 行 -n 显示行数 -i 忽略大小写 -o 只输出匹配部分】

wc path　　统计字节数，字数，行数

head -n 3 test.txt　　显示 test.txt 文件的前 3 行

【-n -2 除 2 行外的所有行】

tail -n 2 test2.txt　　显示 test.txt 文件的最后 2 行

【-f 持续输出 -n +2 从 2 行起后面的所有行 -f 持续输出】　　// tail -n +11 yum.conf | head -n 10

tailf 500 test.txt　　等同 tail -f -n 500 test.txt　　yum install util-linux

File Compression

tar -czvf xxx.tar.gz xxtarpath/*　　打包压缩 gz，tgz 文件

tar -xzvf xxx.tar.gz [-C xxtarpath]　　解压解包 gz，tgz 文件（可以直接解压到大目录）

【-c 创建压缩文件 -x 解压文件(默认 xx.tar) -v 显示过程 -f 尾参，使用归档文件 -z 用 gzip 压缩（xx.tar.gz） -j 用 bzip2 压缩（xx.tar.bz2） -J 用 xz 压缩（xx.tar.xz） -C 指定解压到目标目录】

zip -r <file.zip> <directory-path> Compress the entire directory recursively.

zip -j <file.zip> project/src/main/java/App.java project/src/test/java/AppTest.java project/README.md

Using the -j option flattens the directory structure by removing all path information and storing only the file names.

Notice that the directory paths (src/main/java/, src/test/java/, etc.) are not preserved; only the file names are stored.

zip <file.zip> <file-path> Compress a single file.

zip <file.zip> <file-path> <file-path> Compress multiple files.

unzip <zip-file-path> Extract zip file, the target path contains all files extracted from the zip file.

-d <target-path>. Specify a decompression directory.

Find Command

which ls 查看命令所在路径

whereis nginx 查看安装软件的路径

clear 清屏（ctrl + L 快捷键）

history 显示历史命令

source /etc/profile 重新执行刚修改的初始化文件，使之立即生效，而不必注销并重新登录（等同 . /etc/profile）

Network

search localdomain

ifconfig 查看网络信息 （lo 本地回环，yum install net-tools）

ip route show 查看路由表

ip addr 查看 ip 地址

ping IP 检测网络是否连通（xshell 连接前提）

netstat

-apn 查看所有监听端口和服务，sshd 服务端

-nap

-nltp

-a 全部

-n 数字表示目的主机，端口，用户名

-p 显示 PID

-l 显示监听状态

-t 协议 tcp

lsof -i:3306 端口是否开启 【-i:3306 列出符合条件的进程。（4、6、协议、:端口、 @ip ）】

route add default gw IP 修改默认网关

firewall-cmd 防火墙配置

systemctl status firewalld Check service status

firewall-cmd –new-zone=smp

--list-ports 查看开放端口

--permanent 设置永久有效规则，默认情况规则都是临时的

--list-all 列出某个 zone 的所有规则信息

--reload 重新加载防火墙规则

--zone=xxpublic 指定一个 zone

--add-port=3306/tcp 从 zone 中添加允许访问 3306 端口

--remove-port=3306/tcp 从 zone 中移除允许访问 3306 端口

--get-zones显示系统可用的 zone
--get-default-zone 获取默认规则信息
--set-default-zone=xxzone 设置默认 zone
--get-active-zone 显示当前正在使用的 zone

--list-all-zones 显示所有的 zone 信息的所有规矩
--get-zone-of-interface=xxens33 某个网卡接口与哪个 zone 匹配
--add-interface=ens33 将 ens33 网卡与指定的 zone 规则绑定，以后从该接口进入的流量，匹配 zone 中的规则
--remove-interface=ens33 将 ens33 网卡与指定的 zone 规则解除绑定

--zone=xxpublic 指定一个 zone
--new-zone=xxzone 新建 zone
--add-service=ftp 向指定的 zone 中添加允许访问的服务
--remove-service=ftp 从 指定的 zone 中移除允许某个服务的规则
--get-services 显示系统预定义的服务名称
--add-source=6.6.6.7/255.255.255.0 将 源地址与 zone 绑定
--remove-source==6.6.6.7/255.255.255.0 将 源地址与 zone 解除绑定

wget xxxURL 从网络下载一个文件保存到当前目录
-O xxx.repo 将下载的文件输出到另一个文件
-P /root 下载到指定目录

curl xxxURL 从网络下载一个文件保存到当前目录
-o <output-file-path> Export the downloaded file to another file.
-C - This option resumes download which has been stopped due to some reason. This is useful when downloading large files and was interrupted.
curl -C - [URL...]
curl -C - -O ftp://speedtest.tele2.net/1MB.zip
-L 跟随重定向
-k 允许不使用证书到 SSL 站点
-1 使用 TLSv1==SSL

-i 打印出服务器回应的 HTTP 标头。
curl -i https://www.example.com 上面命令收到服务器回应后，先输出服务器回应的标头，然后空一行，再输出网页的源码。

-I 参数向服务器发出 HEAD 请求，然会将服务器返回的 HTTP 标头打印出来。
curl -I https://www.example.com 上面命令输出服务器对 HEAD 请求的回应。

--head 等同于-I
curl --head https://www.example.com

-v 输出通信的整个过程，用于调试。
curl -v https://www.example.com
--trace 也可以用于调试，还会输出原始的二进制数据。
curl --trace - https://www.example.com

crontab [-u username] //省略用户表表示操作当前用户的 crontab

-e (编辑工作表)
-l (列出工作表里的命令)
-r (删除工作作)

/var/spool/cron/ 目录下存放的是每个用户包括 root 的 crontab 任务，每个任务以创建者的名字命名
/etc/crontab 这个文件负责调度各种管理和维护任务。
/etc/cron.d/ 这个目录用来存放任何要执行的 crontab 文件或脚本。
我们还可以把脚本放在/etc/cron.hourly、/etc/cron.daily、/etc/cron.weekly、/etc/cron.monthly 目录中，让它每小时/天/星期、月执行一次。

Transmission

scp <file-path1> <file-path2> <username>@<server-ip>:<target-path>
scp -r <directory-path1> <directory-path2> <username>@<server-ip>:<target-path>
Upload files to a remote server over SSH.
-P <port-number> Specify the SSH port if it is not the default (22).
-r <directory-path> upload entire directories.
-C Enables compression during file transfer.
This can speed up the transfer of large files by reducing their size in transit, especially over slower networks.
-i Specify an Identity File (Private Key)
Specify an identity file (i.e., an SSH private key) for authentication instead of using the default private key file (like ~/.ssh/id_rsa).

```
scp $zip_file_path "$TEST_AMAZON_USER"@"$TEST_AMAZON_IP":~
```

ssh <username>@<server-ip> '<command>'
Connect to a remote server over SSH using the specified username and server IP address.
-p <port-number>
Specify the SSH port if it is not the default (22).
-i <identity-file>
Specify an identity file (private key) for authentication instead of using the default private key file (like ~/.ssh/id_rsa).
-v
Enable verbose mode.
This option provides detailed output about the SSH connection process, which can be helpful for debugging.
-X
Enable X11 forwarding, allowing you to run graphical applications over SSH.
-C
Enable compression. This can speed up the transfer of large amounts of data, especially over slower networks.
-q
Quiet mode. Suppress warning and diagnostic messages.
-s
This tells the remote server to execute bash and read commands from standard input (the script content you're providing).

```
ssh user@example.com 'cd /path/to/directory; ls -la'
ssh user@example.com 'bash /path/to/script.sh'
```

If you have a script on the remote server that you want to execute,

```
ssh username@server_ip 'bash -s' < /path/to/local/script.sh
```

If the script is short or you don't want to copy it, you can execute it directly in the SSH command by enclosing the script in quotes

```
ssh username@server_ip 'bash -l -c "bash -s"' < /path/to/local/script.sh
```

```
ssh "$TEST_AMAZON_USER"@"$TEST_AMAZON_IP" "cd $remote_work_path && bash -l -c 'bash -s' <
\"$invoke_function_bash_path\" && <other_command>"
ssh username@server_ip "bash -c echo $JAVA_HOME"
```

To access environment variables while executing your script, you could structure your SSH command like this.

/path/to/local/script.sh is a local path inseated of a server path

```
ssh user@example.com 'echo "Hello, World!" > /tmp/hello.txt'
```

If your command contains spaces or special characters, make sure to enclose the command in single quotes.

```
ssh -i /path/to/private_key.pem -p 2222 user@example.com 'your_command_here'
```

If you need to specify an identity file (private key) and a non-default port, use the -i and -p options:

sftp remote@192.168.0.101

Login server with sftp

-P <port> Specify server port

rsync datadir root@192.168.11.150:/home/lucas

-av 递归到目录，并详细输出

-e ssh 使用 ssh 作为远程 shell，以便对所有内容进行加密

--exclude='dir*' 排除匹配模式的文件

tftp -g serverIP -r file_path 从服务器上下载文件到当前目录

Process

ps -aux 查看进程信息

-a 显示终端上所有进程

-u 显示进程详细状态

-x 显示没有控制终端的进程

-r 显示正在运行的进程=-

-o pid.pgid,args Change the output format.

-w Ouput wide format

-ww Allows for unlimited width in the output.

--cols 200 Set the width of the output explicitly

kill 终止进程（PID 进程号）

-9 233 根据 PID 强力终止进程

-9 %1 杀死挂起进程 job

-2 <PID> or -<PGID>

Terminate a process gracefully.

```
ps -o pid.pgid,args
```

Sends a SIGINT, which is the same signal that Ctrl+C sends to a running process.

top 动态显示进程

M 根据内存使用量排序

P 根据 CPU 占有率排序

T 根据进程运行时间排序

U 可用用户名筛选进程

K 杀死进程

q 退出

h 帮助】

pgrep [options] pattern

The pgrep command in Unix-like operating systems is used to search for processes currently running on the system based on specific criteria.

It returns the process IDs (PIDs) of the processes that match the specified conditions.

-l	List the process names along with their PIDs.
-u user	Search for processes owned by a specific user.
-f	Match against the full command line instead of just the process name.
-n	Show only the newest (most recently started) matching process.
-o	Show only the oldest (least recently started) matching process.

nohup command [arguments] &

command	The command or script you want to run.
[arguments]	Any arguments or options for the command.
&	The ampersand at the end puts the command in the background.

The nohup command in Unix-based systems (like Linux and macOS) allows you to run a command or script in the background,

preventing it from being terminated when the user logs out or the terminal is closed.

Find all running nohup tasks

```
ps aux | grep nohup
```

The term "nohup" stands for no hang up.

```
nohup java -jar simi.jar &
```

Redirect Output

If you want to redirect the output to a specific file instead of the default nohup.out file in the current directory:

> mylogfile.log redirects the standard output (stdout) to mylogfile.log.

2>&1 redirects the standard error (stderr) to the same location as the standard output.

> /dev/null Redirects the standard output (stdout) to /dev/null, which is a special device that discards everything written to it (essentially, it’ s like sending the output to a "black hole").

```
nohup python myscript.py > mylogfile.log 2>&1 &
nohup java -jar xx.jar >> sdkjava.log 2>&1 &
nohup python myscript.py > /dev/null 2>&1 &
```

pkill xxname 终止所有名字为 xxname 的进程

jobs 查看后台运行程序 【-l 列出进程 id】

fg jobsnum 后台程序调出前台

Mount Directory

fuser -cu /tv_data #查看挂载文件进程

fuser -mv /dev/sdb #或者查看挂载点进程

fuser -ck /tv_data #结束进程

fuser -mk /dev/sdb #使用挂载点结束进程

Service

Start a Service

systemctl start xxx.service 开启服务

systemctl stop xxx.service　　关闭服务【修改 tomcat/logs 的权限后 tomcat 正常启动。】
systemctl enable xxx.service　　Start the service at boot
systemctl is-enabled xxx.service　　查看是否开机启动
systemctl disable xxx.service　　关闭开机启动
systemctl daemon-reload　　所有服务重新加载配置文件
systemctl status xxx.service　　显示服务状态
systemctl list-unit-files　　显示所有的服务【 --state=failed　disabled　static　enabled　generated　indirect】
man systemd　　用户手册
journalctl -xe　　显示 systemctl 命令执行记录
chkconfig --add xxservice　　添加开机启动服务 xxservice【--list 列表　--del mysqld 删除】

service sshd **start**　　启动服务 【--status-all 显示所有服务状态】
service sshd **status**　　服务状态
service sshd **restart**　　重启服务　　// service network-manager restart　　网络重启（使用最新的网络配置）
service　--status-all　　显示所有服务状态

lvs　显示逻辑卷的信息

systemctl restart NetworkManager　　重启网卡（centos8 已经替换了原来的 network，新版的叫：NetworkManager）

Service File

cd /lib/systemd/system　【默认除 80 端口的其他端口无法访问，需要用 firewall-cmd 命令放行】
touch xxx.service　　service 无法获取系统定义的环境变量（/etc/profile）
tail -f /var/log/messages　日志文件
<service-name>.service
[Unit]
Description=redis　　#描述
After=network.target　#服务后启动

After=syslog.target　　# 描述服务类别，表示本服务需要在 network 服务启动后在启动
Before=xxx.service　　#表示需要在某些服务启动之前启动，After 和 Before 字段只涉及启动顺序，不涉及依赖关系。
Requires=B　　#依赖的服务 B （限制用户后，只有指定用户才能启动服务）
[Service]
Type=forking
　　# 创建子线程，服务命令进程退出后服务启动成功，没有 pid 文件时可以不创建 PIDFile
　　notify 通知
　　simple　执行 ExecStart 指定的命令启动主进程，服务命令立即退出
　　oneshot
　　dbus

ExecStart=/opt/startup.sh start　　#服务运行的具体执行命令，${JAVA_HOME}使用环境变量
ExecStop=/opt/shutdown.sh stop　　#服务停止的执行命令
ExecReload=/opt/shutdown.sh restart　　#服务重启的执行命令
PrivateTmp=true　　# 表示给服务分配独立的临时空间

PIDFile=/data/mysql/data/mysqld.pid

#存放需要保护的进程 PID 文件的位置，systemd 不会对该文件写入数据。

ExecStartPost=/bin/sleep 0.1

在执行可执行文件前等待 0.1 秒

ExecReload=/bin/kill -HUP $MAINPID

ExecStop=/bin/kill -s QUIT $MAINPID

KillMod=process

定义 systemd 如何停止服务

control-group 当前控制组里所有的子进程都会被杀掉

process 只杀主进程

none 没有进程会被杀掉，只是执行服务的 stop 命令

mixed 主进程将收到 SIGTERM(终止进程)信号，子进程将收到 SIGKILL（无条件终止）信号

ExecStartPre= #启动服务之前执行的命令

ExecStartPost= #启动服务之后执行的命令

ExecStopPost= #停止服务之后执行的命令

Environment="JAVA_HOME=/root/jdk1.8" #设置服务启动的环境变量

Environment="MAVEN_HOME=/root/maven1.8" #设置服务启动的环境变量

User=labuladuo

Group=labuladuo

Restart=no

定义服务进程退出后，systemd 的重启方式，默认是不重启

on-failure 非正常退出时，重启，包括信号终止，和超时

always 不管什么退出原因，都会重启

on-watchdog 超时退出时，才会重启

on-success 只有正常退出时（退出状态码为 0），才会重启】

on-abnaomal 只有信号终止或超时，才会重启

on-abort 只有在收到没有捕捉到信号终止时，才会重启

RestartSec=10 #表示 systemd 重启服务之前，需要等待的秒数

TimeoutSec=0 Disable service start and stop timeout logic of systemd for mysqld service.

WorkingDirectory=/usr/ddns/ #工作目录

RestartPreventExitStatus=1

LimitNOFILE = 65535 # Sets open_files_limit

[Install]

WantedBy=multi-user.target # 系统以该形式运行时，服务方可启动

User

Check User

whoami 当前用户名

who 之前登录用户

w 之前登录用户详细信息

last 用户登录情况

su [-] [root]　切换用户　（默认切换到 root 用户，　-　同时切换到所在目录/kpent）
sudo command　管理员权限执行命令　（root 权限第一次输入密码后可不再输入密码）　// sudo -i -u postgres　【-i 获取一个登录外壳，外壳将读取特定于登录的资源文件，例如.profile 或.login】
id oracle　查看用户信息
exit　退出当前用户

useradd sdk　添加用户-权限限制无法登录管理员（默认创建一个同名用户组，和主目录）【-d /home/xxx 指定创建的用户目录 -g xxxg 指定用户组，可多个 -m 主目录不存在此用户目录时自动建立用户目录】
passwd sdk　给 xxx 用户设置密码（会提示输入密码）
userdel sdk　删除用户（单用户组，会连带删除用户组）　【-r 同时删除用户目录 -f 强制删除用户】
groupadd sdktrust　添加用户组
groupdel sdktrust　删除用户组（里面还有用户时不能删除）
usermod sdk [-G gname]　修改用户所在组 【-G 添加到多个组中 -a 不离开之前组】

Permission

chmod [u g o a][+ - =][rwx] path -R　修改文件权限,同一个组的权限为 g 权限　**chmod** [u g o a][+ - =][rwx] <path> -R　修改文件权限，同一个组的权限为 g 权限
u 文件所有者 g 用户组 o 其他用户 a 三者皆是　+增加权限 -撤销权限 =设定权限

chmod 775 path –R　三组权限用十进制表示
-R 修改目录下所有文件权限　421 421 421 rwx rwx rwx　文件所有者　用户组　其他用户

chown [OPTIONS] USER[:GROUP] FILE
change the ownership of files and directories.
USER　the new owner of the file.
GROUP　the new group owner (optional).
FILE　the path to the file or directory.

```
chown :vmshare  /root/vm
chown root  /root/vm
```

-R To change the ownership of a directory and all its contents recursively

```
sudo chown -R username:groupname /path/to/directory
```

chgrp <usergroup> <group path>　修改文件用户组
【-R 递归目录设置权限】

umask 543　When we make a new directory, the permissions will be calculated as
(full permissions for directory) – (umask value)

System

Core

uname　显示操作系统内核全部信息
【-a 显示全部 -s 显示内核类型 -n 主机公网 ip -r 显示发布的内核 -v 显示内核版本 -m 显示机器硬件名 -p 显示处理器位数 -i 显示硬件位数 -o 显示操作系统信息 --version 命令版本】
cal [-y]　查看当前日历 【-y 查看年日历】
date　显示时间　【-s "2016-06-18 16:10:00" 修改系统时间 -R 显示时区】 // 修改时区： tzselect
TZ='Asia/Shanghai'; export TZ
clock　时间设置　【--show 显示硬件时钟的时间 --systohc 系统时间写入硬件时钟】
export JAVA_HOME=/usr/local/jdk_1.8　当前 shell 有效的环境变量新增

reboot 重启操作系统

shutdown -r now 重新启动操作系统（会提醒其他用户）

shutdown -h [now 20:25 +10] 立刻关机 指定时间关机 十分钟后关机

init 0 【0 关机 6 重启 3 切换到字符界面-ctrl alt f2 5 切换到图形界面-ctrl alt f7】

du path -h 查看目录所占磁盘空间 【-a 显示目录内所有文件 -s 指定文件目录占用数据块 -b 字节为单位 -h 提高可读性】

df –h 列出文件系统的整体磁盘使用量 【-h 提高可读性】

env 查看环境变量

mount -o loop test.iso /mnt/cdrom 挂载文件到挂载点 // 测试成功：ls /mnt/cdrom

【-t xxx 指定挂载文件类型 auto 自动 iso9660 光盘，msdos DOS fat16 文件系统，vfat Windows 9x fat32 文件系统，ntfs Windows NT ntfs 文件系统，smbfs Mount Windows 文件网络共享】

【-o xxx 挂载方式 loop 用来把一个文件当成硬盘分区挂接上系统 ro 采用只读方式挂接设备 rw 采用读写方式挂接设备 iocharset 指定访问文件系统所用字符集

umount /mnt/cdrom 【-f 强制 -l 懒卸载 】

Installation

apt-get update 获得最新的软件包列表

apt-get install xxx 安装软件（自动从网络中获取）【ssh 服务： sudo apt-get install openssh-server】

apt-get remove - purge xxx 卸载软件并删除配置文件

apt-get update 更新软件源（更新可获取的软件库）

apt-get upgrade 升级所有软件到最高版本

dpkg –i xxx.deb 安装 deb 包 【–l 查看已安装的 deb 包 -P xxx 卸载软件并删除配置文件】

rpm -i xxx.rpm 安装 rpm 包

【-i 安装 -e xxx 卸载 -e --nodeps xxx 强力卸载，提示删除依赖文件 -v 显示安装过程 -h 打印更多的安装信息 -q 查询 rpm 包是否被安装 -qa 查看已安装的 rpm 仓库包 -ql 查看包位置 】

yum install xxx.rpm 安装 rpm 包，基于 rpm，自动安装依赖）【 -y 自动回答所有 yes --elablerepo=xxx 使用额外的仓库安装 --disablerepo=xxx 禁用一个仓库 --allowerasing 允许替换冲突的包 】

epel-release EPEL 存储库

sudo yum install java-17-openjdk (JRE)

sudo yum install java-17-openjdk-devel (JDK)

yum localinstall xxx.rpm -y 本地安装，不同于 rpm，yum 会解决依赖问题

yum update xxx 升级指定软件包（jenkins 更新后 service 文件会初始化）

yum reinstall xxx 重装软件

yum remove xxx 卸载 rpm 包

yum list xxx* 列出所有可安装的软件包

yum list installed 'xxx*' 列出所有已安装的软件包

yum list updates 'xxx*' 列出所有可更新的软件包

yum info updates 列出所有可更新的软件包信息

yum info installed 列出所有已安装的软件包信息

yum info extras 列出所有已安装但不在 Yum Repository 内的软件包信息

yum update -y 添加新的 repo 后刷新仓库

yum search 'xxx*' 查找软件包

yum provides xxx　　查询软件所归属的软件包

yum clean all　　清除仓库缓存（更换仓库 repo 时必须使用）
yum makecache　　缓存软件包信息（提高搜索/安装软件的速度）

yum repolist　　列出所有的 repo
yum module disable mysql　　禁用本地的 mysql 模块（之后才能从自己的 repo 里安装）
yum-config-manager --add-repo=xxxrepo　　配置管理　【--add-repo=http://baidu.repo 添加仓库　--enable mysql80-community　激活已有仓库 --disable mysql80-community　禁用仓库】

找不到包解决办法：　https://centos.pkgs.org/

yum 配置项

安装目录：

/etc/yum.repos.d/*　yum 镜像库
　　enabled=0　　可以禁止 OS 从外部更新已经包含的包
/etc/xxx　　配置目录
/usr/bin　　可执行文件
/usr/lib　　程序使用的动态函数库
/usr/share/doc　　程序使用手册和帮助文档
/usr/share/man　　一些 man page 文件

sudo wget -O /etc/yum.repos.d/CentOS-Base.repo https://mirrors.aliyun.com/repo/Centos-vault-8.5.2111.repo　　启用阿里镜像

touch /etc/yum.repos.d/jenkins.repo　　创建一个镜像文件
vi /etc/yum/pluginconf.d/fastestmirror.conf　　yum 禁用 fastestmirror 插件
rm -f /etc/yum.repos.d/*　　删除镜像库

openssl

openssl genrsa -out private.pem/private.key 2048　生成私钥 这条命令让 openssl 随机生成一份私钥，加密长度是 1024 位。
加密长度是指理论上最大允许”被加密的信息“长度的限制，也就是明文的长度限制。
随着这个参数的增大（比方说 2048），允许的明文长度也会增加，但同时也会造成计算复杂度的极速增长。一般推荐的长度就是 2048 位。
生成私钥文件：C:\Users\PC\rsa_private_key.pem，内容都是标准的 ASCII 字符，开头一行和结尾一行有明显的标记，真正的私钥数据是中间的不规则字符。
-aes256　　使用 aes256 加密，默认 rsa 加密
-out　　输出密钥文件
-passout pass:111111　输出密钥文件 phrase

openssl rsa -in rsa_private_key.pem -pubout -out rsa_public_key.pem　根据私钥生成公钥：C:\Users\PC\rsa_public_key.pem
-in　　输入密钥文件
-out　　输出密钥文件
-pubout　　输出一个公钥
-passin pass:111111　输入密钥文件 phrase
openssl req -new -key pub.key -out prive.crt -subj

"/C=CN/ST=nanning/L=nanning/O=example/OU=it/CN=domain1/CN=domain2"

// openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.crt -days 365 -subj '/CN=My App PISP AISP/C=NL'

C=FR, OI0.2.5.4.97=NTRFR-794513986, O=YOUSIGN SAS, OU=0002 794513986, CN=YOUSIGN SAS - QUALIFIED SIGNATURE CA

-new 新请求

-x509 X.509 结构的证书请求

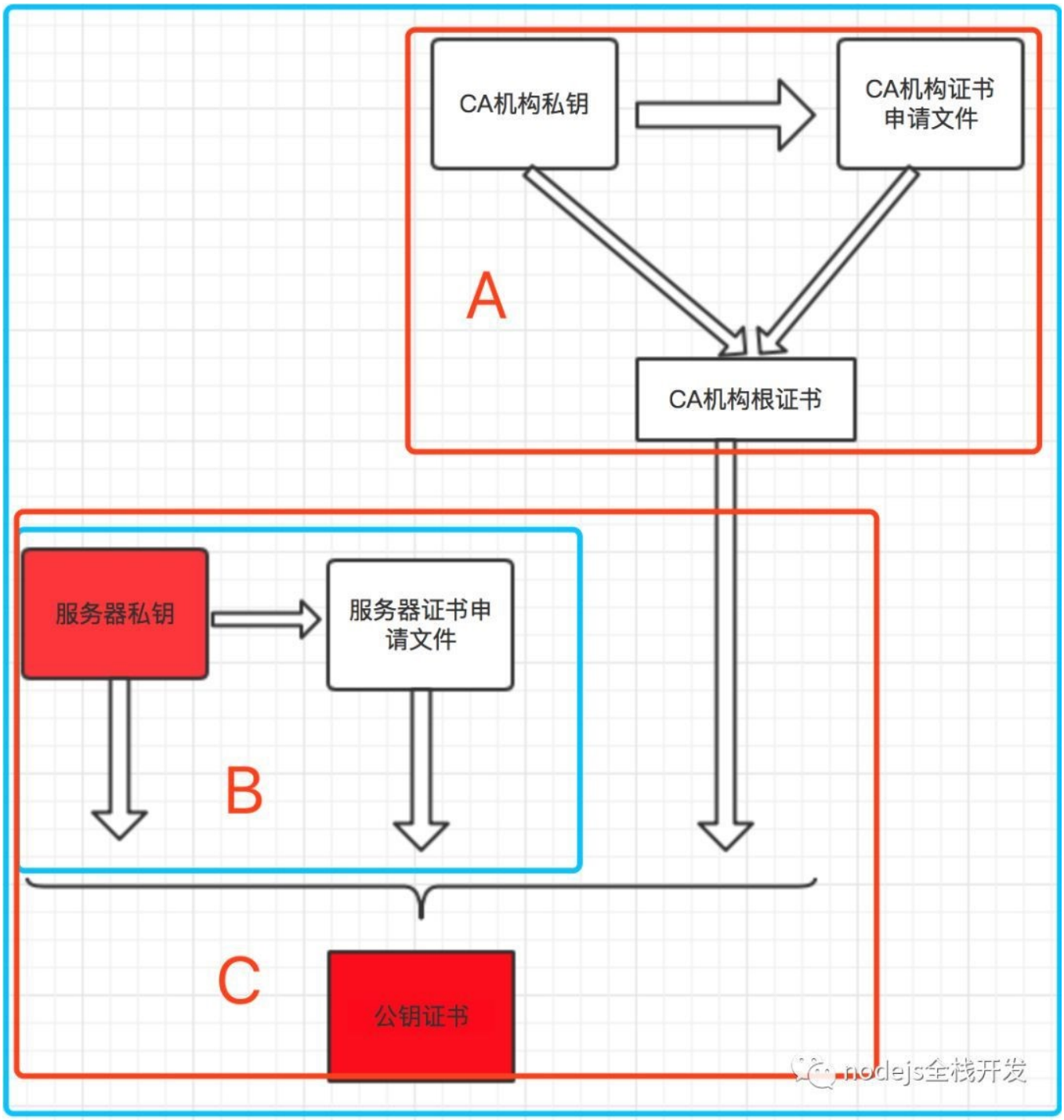
-out xxx.crt 输出的证书文件

-subj 'xxx' 设置或者修改 证书请求的 使用者

-nodes 不加密私钥

-newkey rsa:4096 用参数生成一个新 key

-keyout xxx.pem 新 key 输出的文件名



csr 证书请求文件　　crt 证书文件

CER 是二进制形式的 X.509 证书，DER 编码。

CRT 是二进制 X.509 证书，封装在文本（base-64）编码中。如下两种方式进行尝试转换

openssl x509 -inform DER -in certificate.cer -out certificate. crt

openssl x509 -inform PEM -in certificate.cer -out certificate.crt

根证书验证子证书签名

openssl verify -CAfile root.crt sub.crt>>>> sub.crt OK

1、生成服务器私钥。

openssl genrsa -out server.key 1024

2、根据服务器私钥文件生成证书请求文件，这个文件中会包含申请人的一些信息，所以执行下面这行命令过程中需要用户在命令行输入一些用户信息，随便填写，一路回车即可。

openssl req -new -key server.key -out server.csr

3、生成 CA 机构的私钥，命令和生成服务器私钥一样，只不过这是 CA 的私钥

openssl genrsa -out ca.key 1024

4、生成 CA 机构自己的证书申请文件

openssl req -new -key ca.key -out ca.csr

5、生成自签名证书，CA 机构用自己的私钥和证书申请文件生成自己签名的证书，俗称自签名证书，这里可以理解为根证书。

openssl x509 -req -in ca.csr -signkey ca.key -out ca.crt

6、根据 CA 机构的自签名证书 ca.crt 或者叫根证书生、CA 机构的私钥 ca.key、服务器的证书申请文件 server.csr 生成服务端证书。

openssl x509 -req -CA ca.crt -CAkey ca.key -CAcreateserial -in server.csr -out server.crt

openssl x509 -req -CA ca.cer -CAkey ca.key -CAcreateserial -in server.csr -out server.crt

7、查看 csr 文件信息

openssl req -in test.csr -noout -text

8、查看证书信息

openssl x509 -in test.cer -noout -text

Linux / Bash Script

Core

Bash Commands

make　　在当前目录找到 Makefile 文件，开始编译，

make clean　　清除上次 make 命令产生.o 文件及可执行文件。

make install　　将当前目录编译成功的可执行文件安装到系统目录中，一般为/usr/local/bin 目录。

DESTDIR= /usr/local/python39　指定安装目录（也可以用./configure --prefix=/usr/local/python39）

man [1-9] ls　　Command help　（Number is the selection command type）

ls --help　　Command help

set

-x

When set -x is active, the shell will display each command just before it is executed, preceded by a + sign.
This helps in understanding the flow of the script and tracking the values of variables.

exit 1　　退出脚本

echo "xx"　　输出字符串

-n 不要在最后自动换行

-e 打开反斜杠 ESC 转义

shift 3　　将位置参数向左移动 num 个位置，shift 命令左移。比如 shift 3 表示原来的$4 现在变成$1，原来的$5 现在变成$2 等等，原来的$1、$2、$3 丢弃，$0 不移动。不带参数的 shift 命令相当于 shift 1

eval $xx　　执行一条字符串命令

xargs

It takes the input (either piped from another command or provided as arguments) and converts it into a command-line argument list, often to execute other commands.

```
find . -name "*.tmp" | xargs rm
```

By default, xargs strips leading and trailing whitespace from the input, so piping the output of echo through xargs effectively removes extra spaces.

```
filename="  myfile.txt  "
filename=$(echo "$filename" | xargs)
echo "Filename is: '$filename'"
```

read [options] variable_name

The read command in Bash is used to read a line of input from standard input (typically the keyboard) and assign it to a variable.

-p Allows you to specify a prompt that appears before the input.

-s Makes the input silent, which is useful for sensitive information like passwords.

-t Sets a timeout in seconds. If the user does not enter anything within the specified time, the command will return a non-zero exit status.

-r

Prevent the backslash (\) from being interpreted as an escape character.

When the -r option is specified, backslashes will be treated as literal characters,

which is particularly useful when reading input that might contain backslashes.

perl <pl-file-path>

The perl command in Linux (and other Unix-like operating systems) is used to execute Perl scripts.

Perl is a versatile and powerful scripting language often used for text processing, system administration tasks, web development, and other applications.

-e 'print "Hello, world!\n"' Executing a one-liner script instead of a pl file.

-p Processes each line of input and prints it.

```
perl -p -i -e 's/old_text/new_text/g' file.txt
```

Using Perl for text processing (with regular expressions)

-i Edits files in place (useful for text processing).

-n Processes input line by line but does not print automatically.

-w Enables warnings.

-M Loads modules.

bash <sh-file-path> Execute script file.

-l

This flag makes the shell a login shell, which means it will source the appropriate profile files, allowing it to access any environment variables defined there.

-c

This flag allows you to specify a command to be executed in that shell.

-s

Specify that the script input will be provided via standard input (stdin).

sh <script-file-path> Execute script file.

sh -c ‘ls “myfile.txt”’

sh -c "ls \"myfile.txt\""

sh -c 'eval "ls \"myfile.txt\""'

Configuration

echo -e "\e[31mtest\e[41m" 设置终端 字符和背景颜色

字符颜色 \e[30m 黑色 \e[31m 红色 \e[32m 绿色 \e[33m 淡红色 \e[34m 蓝色 \e[35m 紫色 \e[36m 淡蓝色

\e[37m 灰色
背景色 \e[40m 黑色 \e[41m 红色 \e[42m 绿色 \e[43m 淡红色 \e[44m 蓝色 \e[45m 紫色 \e[46m 淡蓝色 \e[47m 灰色

setfont LatGrkCyr-12x22.psfu.gz 设置字体
字体包 /lib/kbd/consolefonts

Format

#!/bin/bash
The shebang (#!) tells the operating system which interpreter to use to execute the script.
In this case, it specifies that the script should be run using Bash.

#<content>
Comments

yum remove docker \
docker-client
Terminal command line break

String

'<string>'
Single quotes prevent all special characters inside the string from being interpreted or expanded. Everything within single quotes is treated literally.
Variables and command substitutions are not expanded when inside single quotes.

"<string>"
Double quotes allow certain special characters to be interpreted, such as:
Variable expansion ($VAR)
Command substitution ($(command))
Escape sequences (e.g., \n, \t)
However, characters like \ (backslash), $ (dollar sign), and ` (backtick) are still treated specially and will be expanded.

'<string1> \
<string2>'
Multi-line strings

Wildcard

* Matches zero or more occurrences of the given pattern(s).
*.txt matches all text files.
? Matches zero or one occurrence of the given pattern(s).
file?.txt matches file1.txt, file2.txt, etc., but not file10.txt.
+ Matches one or more occurrences of the given pattern(s).
file+ matches file1, file2, fileabc, etc.
@ Matches one of the specified patterns.
@(foo|bar) matches either foo or bar.
! Matches anything that does not match the specified pattern(s).

!(temp*) matches anything except files or directories that start with temp.

[0-9] Matches any single digit from 0 to 9. This is a character class that includes all numeric digits.

[a-b] Matches any single character that is either a or b.

[!abc] or [^abc] Matches any single character not in the specified list (in this case, not a, b, or c). This is used for exclusion.

Operators

<	重定向	//bash -s < test.sh
>	重定向，显示内容覆盖文件，不存在就新建	// ls > test.txt
>>	追加，显示内容追加到文件	// ls >> test.txt
<<	追加，标准输入来自命令行的一对自定义终止符的中间内容	// cat << XXXEOF > xxx.txt
|	管道，左命令输出 为右命令输入	// ls | grep std

Variable

Variable Definition

country="China" Define a variable

没有空格可省引号（ ` 进行运算 $ 识别变量 \ 转义，其他字符正常显示）

连接字符串，windows 的 CRLF 文件会导致字符串覆盖，必须使用 LF（通过 `xx` 和 ${xx} 可以获取命令结果） // s="<string>"

declare -A file_mappings

Declare an associative array, also known as a hash map or dictionary in other programming languages.

```
file_mappings["docker-compose.yml"]="~/docker/"
file_mappings["config.yml"]="~/config/"

remote_path="${file_mappings[$file_name]}"
```

d="<string>" s+=$d or "abc"$i or "abc""efg" Concatenate strings

"abc${country}" Identify variables

`ls -la` Return the command execution result.

echo $country Output content

unset $country Delete a variable

$[(a+b)/c **]** **$((** (a+b)/c **))** $(dirname "$0") $(pgrep -f $APP_JAR_NAME) ${psid:-0}

Return the result of the operation

${parameter:-word}

This syntax is used to evaluate the value of parameter. If parameter is unset or null, it substitutes word instead.

psid This is the name of the variable you're checking.

:- This part of the syntax indicates that you want to provide a default value if psid is either unset (not defined) or null (empty).

0 This is the default value that will be used if psid is not set or is empty.

APP_HOME = $2 Variable Definition

APP_HOME = $3 Variable assignment

String Operators

字符串切片：

${var:offset:number} 取字符串的最右侧几个字符：${var: -lengh} （冒号后必须有一空白字符）

基于模式取子串：

${var#*word} 其中 word 可以是指定的任意字符；功能：自左而右，查找 var 变量所存储的字符串中，第一次出现的 word，删除字符串开头至第一次出现 word 字符之间的所有字符；

${var##*word}　　删除的是字符串开头至最后一次由 word 指定的字符之间的所有内容　　//EX: ${"/var/log/messages"##*/}　　messages

${var%word*}　　其中 word 可以是指定的任意字符；功能：自右而左，查找 var 变量所存储的字符串中，第一次出现的 word，删除字符串最后一个字符向左至第一次出现 word 字符之间的所有字符 ${"/var/log/messages"%/*}: /var/log

${var%%word*}　　同上，只不过删除字符串最右侧的字符向左至最后一次出现 word 字符之间的所有字符；　//EX: url=http://www.ibm.com:80　${url##*:}　${url%%:*}

查找替换：

${var/pattern/substi}　　查找 var 所表示的字符串中，第一次被 pattern 所匹配到的字符串，以 substi 替换之；

${var//pattern/substi}　　查找 var 所表示的字符串中，所有能被 pattern 所匹配到的字符串，以 substi 替换之；

${var/#pattern/substi}　　查找 var 所表示的字符串中，行首被 pattern 所匹配到的字符串，以 substi 替换之；

${var/%pattern/substi}　　查找 var 所表示的字符串中，行尾被 pattern 所匹配到的字符串，以 substi 替换之；

查找并删除：

${var/pattern}　　查找 var 所表示的字符串中，删除第一次被 pattern 所匹配到的字符串

${var//pattern}

${var/#pattern}

${var/%pattern}

字符大小写转换：

${var^^}　把 var 中的所有小写字母转换为大写；

${var,,}　　把 var 中的所有大写字母转换为小写；

变量赋值：

${var:-value}　　如果 var 为空或未设置，那么返回 value；否则，则返回 var 的值；

${var:=value}　　如果 var 为空或未设置，那么返回 value，并将 value 赋值给 var；否则，则返回 var 的值；

${var:+value}　　如果 var 不空，则返回 value；

${var:?error_info}　　如果 var 为空或未设置，那么返回 error_info；否则，则返回 var 的值；

Built-in Constant

$0　　当前运行脚本的所在路径

$1 ..$3　　文件参数

$#　　传递到脚本的参数个数　　// [$# -ge 1] && shift

$*　　以一个单字符串显示所有向脚本传递的参数。 如"$*"用「"」括起来的情况、以"$1 $2 … $n"的形式输出所有参数。

$@　　与$*相同，但是使用时加引号，并在引号中返回每个参数。 如"$@"用「"」括起来的情况、以"$1" "$2" … "$n" 的形式输出所有参数。

$$　　脚本运行的当前进程 ID 号

$!　　后台运行的最后一个进程的 ID 号

$-　　显示 Shell 使用的当前选项，与 set 命令功能相同。

$?　　执行上一个指令的返回值（显示最后命令的退出状态。0 表示没有错误，其他任何值表明有错误）

$USER　　当前 shell 的执行用户

$SHELL　当前的 shell 程序 //　/bin/bash

Statement

Condition

Conditional Constructs

[] (Single Brackets)

Basic test command for evaluating expressions (usually for strings and file attributes).

Requires spaces around the brackets and within the condition.

```
# String comparison
if [ "$var" == "Hello" ]; then
    echo "Variable is Hello."
fi

# Check if a file exists
if [ -f "/path/to/file.txt" ]; then
    echo "File exists."
fi
```

[[ ]] (Double Brackets)

Enhanced test command that supports advanced string operations and regex matching.

Supports pattern matching (e.g., wildcards) and regex.

More flexible than single brackets; no need for escaping certain characters (e.g., <, >, &).

Does not require quotes for string variables.

You can also use [(single brackets) instead of test:

```
string="Hello World"

# Check if string is empty
if test -z "$string"; then
    echo "The variable string is empty or not set."
else
    echo "The variable string is set to: $basedir"
fi

# String pattern matching
if [[ $string == *"World"* ]]; then
    echo "The string contains 'World'."
fi

# Regex matching
if [[ $string =~ ^Hello ]]; then
    echo "The string starts with 'Hello'."
fi

# String comparison
if [[ $string == "Hello" ]]; then
    echo "var1 is equal to 'Hello'"
fi

# Check if a variable is empty
if [[ -z "$string" ]]; then
    echo "var1 is empty"
else
    echo "var1 is not empty"
fi
```

(( )) (Double Parentheses)

Used for arithmetic evaluation and numeric comparisons.

Allows mathematical operations without needing additional commands.

Does not require spaces around operators.

```
num1=10
num2=5

# Arithmetic operation
result=$((num1 + num2))
echo "The sum is: $result"

# Numeric comparison
```

```
if (( num1 > num2 )); then
    echo "$num1 is greater than $num2."
fi
```

Numeric Comparisons

-eq
: Equal to (e.g., if [$var1 -eq $var2])

-ne
: Not equal to (e.g., if [$var1 -ne $var2])

-gt
: Greater than (e.g., if [$var1 -gt $var2])

-ge
: Greater than or equal to (e.g., if [$var1 -ge $var2])

-lt
: Less than (e.g., if [$var1 -lt $var2])

-le
: Less than or equal to (e.g., if [$var1 -le $var2])

String Comparisons

==
: Equal (e.g., if [$var1 == $var2])

!=
: Not equal (e.g., if [$var1 != $var2])

=~
: Matches a regular expression (e.g., if [[$var1 =~ $var2]])

String Tests

-z string
: Tests if the string is empty (true if empty)

-n string
: Tests if the string is not empty (true if non-empty)

-e file
: Tests if the file exists

-f file
: Tests if the file is a regular file

-d file
: Tests if the specified path is a directory

-r file
: Tests if the file is readable by the current user

-w file
: Tests if the file is writable by the current user

-x file
: Tests if the file is executable by the current user

-s file

Tests if the file exists and is not empty

Logical Operators

&&

Logical AND (e.g., if [condition1] && [condition2])

||

Logical OR (e.g., if [condition1] || [condition2])

Notes

- Use **(( ))** for arithmetic evaluation. This is useful for numeric comparisons without the need for additional operators.
- Use **[[ ]]** for extended test conditions that support regex and improved logical operations.
- When checking for empty strings, always quote the variable to avoid errors with uninitialized variables (e.g., if [-z "$var"]).

[] -只能-ge -le 判断 **(( ))** **[[ ]]** 返回 Boolean 比较

-eq == **-ne** !=

-ge >= **-le** <=

-gt > **-lt** < **-z** xx 是否为空

== 相等则为真 if [$var1 == $var2]

!= 不相等则为真 if [$var1 != $var2]

=~ 前边变量包含后边变量则为真。本质上=~是正则匹配单层中括号不支持正则需要双层中括号 if [[$var1 =~ $var2]]

&& 与

|| 或者

-z xxxstr 测试指定字符是否为空，空着真，非空为假

-n xxstr 测试指定字符串是否为不空，空为假 非空为真

-e FILE 测试文件是否存在

-f file 测试文件是否为普通文件

-d file 测试指定路径是否为目录

-r file 测试文件对当前用户是否可读

-w file Test whether the file is writable to the current user.

-x file Test whether the file is executable for the current user.

-s File 文件存在并且是一个套接字

-z 是否为空 为空则为真（不能对字符串进行判断，字符串应该使用"$3"!=""）

-a 是否不空 （不能对字符串进行判断）

Condition judgment

IF

Error Handling

The if condition does not generate an error; it simply evaluates to true or false based on whether the mount point exists.

Therefore, even if the directory is not a mount point,

the script will continue executing any subsequent commands after the if statement.

```
if mountpoint -q /mnt/hgfs; then
```

```
if [ "$dmin" != "" ] ||  [ "$dmin" != "" ]; then      if test -z "$basedir"; then
    command
elif [ "$dmin" == "" ]; then
    command
```

```
else
    command
fi
```

while

while Starts the loop, and the condition or command after it is checked before each iteration.
do Marks the beginning of the block of commands to be executed during each loop iteration.
done Ends the loop.
< input_source
    This is where the input comes from. It could be a file, another command, or an input stream.
IFS= ensures that leading/trailing spaces are preserved.

```
while IFS= read -r line; do
    echo "Processing line: $line"
done < "file.txt"
```

FOR

for Starts the loop and initializes a variable that will hold the current item from a list. This variable iterates over each item in the specified list.
in Specifies the list of items over which to iterate. This can be a hardcoded list, the output of a command, or a wildcard pattern matching files.
do Marks the beginning of the block of commands that will be executed during each iteration of the loop.
done Ends the loop.

```
for fruit in apple banana cherry; do
    echo "I like $fruit"
done
```

CASE

```
case xxexpression in
    xxxa )
        statement1
        ;;
    xxxb )
        statement2
        ;;
    *)
        statementn
esac
```

Traversal				
for ((i=1; i<=100; i++))	do	command	done	循环
for 变量 in 列表	do	command	done	遍历列表

Function	
function getsum() {	定义函数（有 function 时括号可省，function 可省）
local sum=0	局部变量声明

```
    return $sum
}

getsum 10 20 55 15   #调用函数并传递参数
```