# 运维手册

1. 了解主流服务器系统运维基础知识（用户、用户组、网络权限）
2. 对服务器的基础系统进行运维，整理常见命令
3. 对服务器的内存、CPU、GPU资源占用情况，进程（k、r）进行了解
4. 系统日志，错误情况，宕机等日志了解和分析
5. 使用shell脚本，对内存、CPU、GPU资源输出运维报告

Linux ：coentos\ubantu

Windowssever:2022

目录

[指令介绍 4](#_Toc1965718949)

[基础指令 4](#_Toc715746271)

[1、 ls 4](#_Toc119792627)

[2、 pwd 6](#_Toc4970549)

[3、 cd (change directory) 7](#_Toc1121126062)

[4、 mkdir 7](#_Toc327253273)

[5、 touch 8](#_Toc1287975265)

[6、 cp 8](#_Toc391900857)

[7、 mv 10](#_Toc976094154)

[8、 rm 11](#_Toc165982574)

[9、 vim 12](#_Toc100542413)

[10、 重定向输出 13](#_Toc1004208306)

[11、 cat 13](#_Toc1170789769)

[进阶指令 14](#_Toc1824349326)

[1、 df 14](#_Toc1768867916)

[2、 free 15](#_Toc1461655709)

[3、 head指令 15](#_Toc2146555572)

[4、 tail 16](#_Toc33057999)

[5、 less 17](#_Toc542569716)

[6、 wc指令 17](#_Toc1823383339)

[7、 date 18](#_Toc352801110)

[8、 cal 20](#_Toc568909625)

[9、 clear/ctrl+L 20](#_Toc1749313672)

[10、 管道 20](#_Toc342170577)

[高级指令 21](#_Toc934848952)

[1、 hostname 21](#_Toc1245656124)

[2、 id 21](#_Toc1868006773)

[3、 whoami 一般用于shell脚本 21](#_Toc2142966660)

[4、 ps -ef指令（重点） 22](#_Toc506794539)

[5、 top 22](#_Toc230531563)

[6、 du -sh 23](#_Toc1060958275)

[7、 find 24](#_Toc325029840)

[8、 service 25](#_Toc946277835)

[9、 kill 25](#_Toc1180750902)

[10、 ifconfig 25](#_Toc330000390)

[11、 uptime 26](#_Toc2067403897)

[12、 uname 26](#_Toc1508004175)

[13、 netstat -tnlp 26](#_Toc1617975655)

[14、 man 27](#_Toc311821106)

[15、 ping 27](#_Toc336614682)

[16、 tcpping 28](#_Toc1783958229)

[17、traceroute 28](#_Toc412363519)

[权限管理 29](#_Toc1340822988)

[运行级别 29](#_Toc807264350)

[分类 29](#_Toc89229198)

[原理 30](#_Toc962207256)

[/etc/inittab配置文件详解 30](#_Toc121436411)

[相关命令 31](#_Toc88301122)

[用户（组）管理 31](#_Toc995265255)

[用户权限 31](#_Toc664006127)

[文件权限 32](#_Toc1911684461)

[账户文件和组文件 33](#_Toc1348066366)

[用户文件 33](#_Toc1232915753)

[组文件 33](#_Toc1513514485)

[账户管理 34](#_Toc1690236943)

[网络 34](#_Toc20281057)

[网络地址配置 34](#_Toc611686961)

[获取和修改主机名 37](#_Toc1410760068)

[route命令 37](#_Toc15764069)

[socket统计 ss命令 38](#_Toc1118481500)

[域名解析 nslookup 39](#_Toc1641291632)

[防火墙 40](#_Toc1076722344)

[shell使用 44](#_Toc1443511341)

[基础 44](#_Toc440085819)

[shell变量 45](#_Toc109989598)

[参数传递 48](#_Toc1773511731)

[数组 49](#_Toc360006068)

# 指令介绍

## 基础指令

### ls

-a, --all 不要隐藏以 . 开头的项目

-A, --almost-all 列出除 . 及 .. 以外的所有项目

--author 与 -l 同时使用时，列出每个文件的作者

-b, --escape 以 C 风格的转义序列表示不可打印的字符

--block-size=大小 与 -l 同时使用时，将文件大小以此处给定的大小为

单位进行缩放；例如：“--block-size=M”；

请参考下文的大小格式说明

-B, --ignore-backups 不列出任何以 ~ 字符结束的项目

-c 与 -lt 共同使用时：根据 ctime 排序并显示

ctime（文件状态最后更改的时间）；

与 -l 共同使用时：显示 ctime 并按照名称排序；

其它情况：按照 ctime 排序，最新的最前

-C 每栏由上至下列出项目

--color[=WHEN] 控制是否使用带颜色的输出；WHEN 可以是

“always”（默认缺省值）、“auto”或

“never”；更多信息请见下文

-d, --directory 当遇到目录时列出目录本身而非目录内的文件

-D, --dired 产生适合 Emacs 的 dired 模式使用的结果

-f 不进行排序，-aU 选项生效，禁用 -ls --color

-F, --classify 加上文件类型的指示符号（\*/=@| 其中一个）

--file-type 类似，但不后缀‘\*’字符

--format=关键字 交错 -x，逗号分隔 -m，水平 -x，长 -l，

单栏 -1，详细 -l，垂直 -C

--full-time 类似 -l --time-style=full-iso

-g 类似 -l，但不列出所有者

--group-directories-first

在文件前分组目录；

此选项可与--sort 一起使用，但是一旦使用

--sort=none (-U) 将禁用分组

-G, --no-group 使用长列表格式时，不输出组名

-h, --human-readable 同时使用 -l 或 -s 时，将大小表示为 1K 234M 2G 等

--si 类似，但是使用 1000 的幂，而不是 1024

-H, --dereference-command-line

跟随命令行中列出的符号链接

--dereference-command-line-symlink-to-dir

跟随命令行中列出的、指向目录的符号链接

--hide=模式 隐藏符合 shell <模式> 的项目

（-a 或 -A 使此选项失效）

--hyperlink[=何时] 文件名使用超链接；<何时> 可以是 "always"

（省略时的默认值）、"auto" 或 "never"

--indicator-style=方式 指定在每个项目名称后加上指示符号方式：

none（默认）、slash（-p）、

file-type（--file-type）以及 classify（-F）

-i, --inode 显示每个文件的索引编号（inode 号）

-I, --ignore=模式 不显示任何匹配指定 shell <模式>的项目

-k, --kibibytes 默认使用 1024 字节的块显示磁盘使用量；

只用于 -s 和每个目录的总计

-l 使用较长格式列出信息

-L, --dereference 当显示符号链接的文件信息时，显示符号链接所指示

的对象而并非符号链接本身的信息

-m 所有项目以逗号分隔，并填满整行行宽

-n, --numeric-uid-gid 类似 -l，但列出用户和组的 ID 号

-N, --literal 输出不加引号的项目名称

-o 类似 -l，但不列出有关组的信息

-p, --indicator-style=slash

对目录追加 "/" 作为指示符号

-q, --hide-control-chars 以“?”字符代替无法打印的字符

--show-control-chars 原样显示无法打印的字符（这是默认行为，除非被调用

本程序的名称是“ls”而且是在终端中进行输出）

-Q, --quote-name 将条目名称括上双引号

--quoting-style=方式 使用指定引用的方式显示条目的名称：

literal, locale, shell, shell-always,

shell-escape, shell-escape-always, c, escape

（该选项优先于 QUOTING\_STYLE 环境变量的配置）

-r, --reverse 排序时逆序排列

-R, --recursive 递归地列出子目录

-s, --size 显示给每个文件分配的大小，单位为块

-S sort by file size, largest first

--sort=WORD sort by WORD instead of name: none (-U), size (-S),

time (-t), version (-v), extension (-X)

--time=WORD change the default of using modification times;

access time (-u): atime, access, use;

change time (-c): ctime, status;

birth time: birth, creation;

with -l, WORD determines which time to show;

with --sort=time, sort by WORD (newest first)

--time-style=TIME\_STYLE 使用 -l 时显示的时间/日期格式；请见下面

TIME\_STYLE 的相关内容

-t 按时间排序，最新的最前；参见 --time

-T, --tabsize=列数 指定制表符宽度为 <列数>，而非默认的 8

-u 与 -lt 一起使用：按照访问时间排序，并显示访问时间；

与 -l 一起使用：显示访问时间，并按文件名排序；

其它情况：按照访问时间排序，最新的最前

-U 不进行排序；按照目录顺序列出项目

-v 对文本中的数字（或版本号）进行自然排序

-w, --width=COLS 设置输出宽度为 COLS 对应的值。0 代表无限制

-x 逐行列出项目而不是逐栏列出

-X 根据扩展名按字母顺序排序

-Z, --context 输出每个文件的所有安全上下文信息

-1 每行只列出一个文件。与 -q 或 -b 共用时避免 '\n'

--help 显示此帮助信息并退出

--version 显示版本信息并退出

<大小> 参数是一个整数，后面可以跟一个单位（例如：10K 指 10\*1024）。

可用的单位有 K、M、G、T、P、E、Z、Y（1024 的幂）以及 KB、MB、...（1000 的幂）。

也可以使用二进制前缀：KiB=K、MiB=M，以此类推。

<时间风格> 参数可以是 full-iso、long-iso、iso、locale，或者 +格式。

<格式> 的解析方式同 date(1)。如果 <格式> 是 格式1<换行符>格式2 的话，

则 <格式1> 将应用于时间较久远的文件，<格式2> 将应用于时间较近的文件。

<时间风格> 如果带有 "posix-" 前缀，则它只会在区域设置非 POSIX 时生效。

另外，可以使用 TIME\_STYLE 环境变量设置默认使用的风格。

使用色彩来区分文件类型的功能已被禁用，默认设置和 --color=never 同时禁用了它。

使用 --color=auto 选项，ls 只在标准输出被连至终端时才生成颜色代码。

LS\_COLORS 环境变量可改变此设置，可使用 dircolors 命令来设置。

### pwd

打印当前工作目录的名字。

选项：

-L 打印 $PWD 变量的值，如果它包含了当前的工作目录

-P 打印当前的物理目录，而不包含任何的符号链接

默认情况下，"pwd" 的行为和给定了 "-L" 时一致。

退出状态：

返回 0，除非使用了无效的选项，或者无法读取当前目录。

### cd (change directory)

改变 shell 工作目录。

改变当前目录至 <目录>。默认的 <目录> 是 shell 变量 HOME 的值。

变量 CDPATH 定义了搜索含有 <目录> 的目录的搜索路径。CDPATH 中额外的

目录名称以冒号 (:) 隔开。空的 CDPATH 表示当前目录。如果 <目录> 以

斜杠 (/) 开头，则不会使用 CDPATH。

如果找不到目录，并且 shell 选项 "cdable\_vars" 被设定，则假定参数是一个

变量名。如果该变量有值，则它的值被当作 <目录>。

选项：

-L 强制跟随符号链接: 在处理 ".." 之后解析 <目录> 中的符号链接

-P 使用物理目录结构而不跟随符号链接: 在处理 ".." 之前

解析 <目录> 中的符号链接

-e 如果使用了 -P 选项，但无法成功确定当前工作目录时，以非零状态退出

-@ 在支持扩展属性的系统上，将一个具有扩展属性的文件当作

含有文件属性的目录。

默认情况下，跟随符号链接，正如指定了 "-L" 一样。

".." 的处理方式是，移除路径名中的上一个组成部分，以 <目录> 的开头

或者斜杠为界限。

退出状态：

如果目录改变，或在使用 -P 选项时 $PWD 修改成功，则返回 0；否则返回非零。

### mkdir

用法：mkdir [选项]... 目录...

若 <目录> 不存在，则创建 <目录>。

长选项的必选参数对于短选项也是必选的。

-m, --mode=模式 设置权限模式（类似chmod），而不是 a=rwx 减 umask

-p, --parents 需要时创建目标目录的上层目录，但即使这些目录已存在

也不当作错误处理

-v, --verbose 每次创建新目录都显示信息

-Z 将每个创建的目录的 SELinux 安全上下文设置为默认类型

--context[=上下文] 类似 -Z，但如果指定了 <上下文>，则将 SELinux

或 SMACK 安全上下文设置为 <上下文>

--help 显示此帮助信息并退出

--version 显示版本信息并退出

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/mkdir>

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

### touch

用法：touch [选项]... 文件...

将每个 <文件> 的访问时间和修改时间更改为当前时间。

除非提供了 -c 或 -h 选项，否则不存在的 <文件> 将会被创建，内容为空。

如果 <文件> 为 - 则特殊处理，程序将更改与标准输出相关联的文件的时间。

长选项的必选参数对于短选项也是必选的。

-a 只更改访问时间

-c, --no-create 不创建任何文件

-d, --date=字符串 使用 <字符串> 表示的时间，而非当前时间

-f （被忽略）

-h, --no-dereference 影响符号链接本身，而非符号链接的目标

（当系统支持更改符号链接的时间戳时，此选项才有用）

-m 只更改修改时间

-r, --reference=文件 使用 <文件> 的时间戳，而非当前时间

-t 时间戳 使用格式为 [[CC]YY]MMDDhhmm[.ss] 的时间戳，而非当前时间

--time=关键字 修改指定类型的时间：

若 <关键字> 是 access、atime 或 use：与 -a 等价

若 <关键字> 是 modify 或 mtime：与 -m 等价

--help 显示此帮助信息并退出

--version 显示版本信息并退出

请注意，-d 和 -t 选项所接受的时间/日期格式是不同的。

### cp

用法：cp [选项]... [-T] 源 目标

　或：cp [选项]... 源... 目录

　或：cp [选项]... -t 目录 源...

将 <源> 复制至 <目标>，或将多个 <源> 复制至 <目录>。

长选项的必选参数对于短选项也是必选的。

-a, --archive 等于 -dR --preserve=all

--attributes-only 仅复制属性而不复制文件数据

--backup[=控制] 为每个已存在的目标文件创建备份

-b 类似 --backup 但不接受参数

--copy-contents 在递归操作时复制特殊文件的内容

-d 等于 --no-dereference --preserve=links

-f, --force 如果已存在的目标文件无法打开，则将其删除并重试

（该选项在与 -n 选项同时使用时将被忽略）

-i, --interactive 覆盖前询问（使前面的 -n 选项失效）

-H 跟随命令行里的 <源> 中的符号链接

-l, --link 硬链接文件以代替复制

-L, --dereference 总是跟随 <源> 里的符号链接

-n, --no-clobber 不要覆盖已存在的文件(使前面的 -i 选项失效)

-P, --no-dereference 不跟随源文件中的符号链接

-p 等于 --preserve=mode,ownership,timestamps

--preserve[=属性列表] 如果可能的话，保留指定的属性（默认：

mode,ownership,timestamps）

其他属性：context、links、xattr、all

--no-preserve=属性列表 不保留指定的属性

--parents 在 <目录> 下使用完整的源文件路径

-R, -r, --recursive 递归地复制目录

--reflink[=何时] 控制克隆/CoW 副本。见下文

--remove-destination 尝试打开目标文件前先删除已存在的目标

文件（和 --force 选项有区别）

--sparse=何时 控制创建稀疏文件的方式。见下文

--strip-trailing-slashes 删除参数中所有 <源> 末尾的斜杠

-s, --symbolic-link 只创建符号链接而不复制文件

-S, --suffix=后缀 自行指定备份文件的后缀

-t, --target-directory=目录 将所有 <源> 参数复制到 <目录>

-T, --no-target-directory 将 <目标> 视为普通文件

-u, --update 只在 <源> 比目标文件新，或目标文件

不存在时才进行复制

-v, --verbose 显示详细步骤

-x, --one-file-system 不跨越文件系统进行操作

-Z 设置目标文件的 SELinux 安全上下文为默认类型

--context[=上下文] 类似 -Z，但如果指定了 <上下文>，则将 SELinux

或 SMACK 安全上下文设置为 <上下文>

--help 显示此帮助信息并退出

--version 显示版本信息并退出

默认情况下，将通过简单的启发式算法检测稀疏的 <源> 文件，并将对应的 <目标>

文件设为稀疏。这也是 --sparse=auto 的行为。若使用 --sparse=always，则只要

<源> 文件包含了足够长的零字节序列，都会将 <目标> 文件创建为稀疏文件。

使用 --sparse=never 可以禁止创建稀疏文件。

当指定了 --reflink[=always] 参数时，将进行轻量级复制，此时数据块只在被修改时

进行复制。如果因无法进行这种复制而导致复制失败，或者指定了 --reflink=auto，

程序将会回退到标准复制操作。使用 --reflink=never 可以确保永远进行标准复制。

备份文件的后缀为 "~"，除非使用了 --suffix 选项或 SIMPLE\_BACKUP\_SUFFIX 环境变量

指定了其他后缀。版本控制的方式可通过 --backup 选项或 VERSION\_CONTROL 环境变量

来选择。以下是可用的设置值：

none, off 不进行备份（即使使用了 --backup 选项）

numbered, t 使用带有数字编号的备份文件进行备份

existing, nil 若带数字的备份文件已经存在则使用 numbered，否则使用 simple

simple, never 总是使用简单方式进行备份

有一个特殊情况：如果同时指定 --force 和 --backup 选项，而 <源> 和 <目标>

是同一个已存在的普通文件的话，cp 会对 <源> 进行备份。

### mv

用法：mv [选项]... [-T] 源 目标

　或：mv [选项]... 源... 目录

　或：mv [选项]... -t 目录 源...

将 <源> 重命名为 <目标>，或将 <源> 移动至 <目录>。

长选项的必选参数对于短选项也是必选的。

--backup[=CONTROL] 为每个已存在的目标文件创建备份

-b 类似--backup 但不接受参数

-f, --force 覆盖前不询问

-i, --interactive 覆盖前询问

-n, --no-clobber 不覆盖已存在文件

如果您指定了-i、-f、-n 中的多个，仅最后一个生效。

--strip-trailing-slashes 去掉每个源文件参数尾部的斜线

-S, --suffix=SUFFIX 替换常用的备份文件后缀

-t, --target-directory=目录 将所有 <源> 移动至 <目录> 中

-T, --no-target-directory 将 <目标> 视为普通文件

-u, --update 仅在 <源> 比目标文件更新，或者目标文件

不存在时，才进行移动操作

-v, --verbose 显示详细步骤

-Z, --context 将目标文件的 SELinux 安全上下文设置为默认类型

--help 显示此帮助信息并退出

--version 显示版本信息并退出

备份文件的后缀为 "~"，除非使用了 --suffix 选项或 SIMPLE\_BACKUP\_SUFFIX 环境变量

指定了其他后缀。版本控制的方式可通过 --backup 选项或 VERSION\_CONTROL 环境变量

来选择。以下是可用的设置值：

none, off 不进行备份（即使使用了 --backup 选项）

numbered, t 使用带有数字编号的备份文件进行备份

existing, nil 若带数字的备份文件已经存在则使用 numbered，否则使用 simple

simple, never 总是使用简单方式进行备份

### rm

用法：rm [选项]... [文件]...

删除 (unlink) 一个或多个 <文件>。

-f, --force 忽略不存在的文件和参数，且从不询问

-i 每次删除前询问

-I 在删除超过三个文件或者递归删除前询问一次；此选项比 -i

提示次数更少，但仍可以避免大多数错误的发生

--interactive[=何时] 根据 <何时> 的值进行询问：never、once（同 -I）或者

always（同 -i）；如果省略 <何时>，则默认为 always

--one-file-system 递归删除目录时，跳过所有和该目录所对应的命令行参

数不在同一个文件系统上的目录

--no-preserve-root 不要对 "/" 特殊处理

--preserve-root[=all] 不要删除 "/"（默认行为）；

如添加了 "all" 参数，将拒绝处理与其父目录位于

不同设备上的命令行参数

-r, -R, --recursive 递归删除目录及其内容

-d, --dir 删除空目录

-v, --verbose 显示详细步骤

--help 显示此帮助信息并退出

--version 显示版本信息并退出

默认情况下，rm 不会删除目录。使用 --recursive（-r 或 -R）选项可删除每个给定

的目录，以及其中的全部内容。

要删除文件名第一个字符为 "-" 的文件（例如 "-foo"），请使用

以下命令之一：

rm -- -foo

rm ./-foo

请注意，如果使用rm 来删除文件，通常仍可以将该文件恢复原状。如果想保证

该文件的内容无法还原，请考虑使用shred。

### vim

用法: vim [参数] [文件 ..] 编辑指定的文件

或: vim [参数] - 从标准输入(stdin)读取文本

或: vim [参数] -t tag 编辑 tag 定义处的文件

或: vim [参数] -q [errorfile] 编辑第一个出错处的文件

参数:

-- 在这以后只有文件名

-v Vi 模式 (同 "vi")

-e Ex 模式 (同 "ex")

-E 改进的 Ex 模式

-s 安静(批处理)模式 (只能与 "ex" 一起使用)

-d Diff 模式 (同 "vimdiff")

-y 容易模式 (同 "evim"，无模式)

-R 只读模式 (同 "view")

-Z 限制模式 (同 "rvim")

-m 不可修改(写入文件)

-M 文本不可修改

-b 二进制模式

-l Lisp 模式

-C 兼容传统的 Vi: 'compatible'

-N 不完全兼容传统的 Vi: 'nocompatible'

-V[N][文件名] 显示详细信息 [级别 N] [记录消息到指定的文件名]

-D 调试模式

-n 不使用交换文件，只使用内存

-r 列出交换文件并退出

-r (跟文件名) 恢复崩溃的会话

-L 同 -r

-A 以 Arabic 模式启动

-H 以 Hebrew 模式启动

-T <terminal> 设定终端类型为 <terminal>

--not-a-term 跳过输入/输出不是终端的警告

--ttyfail 如果输入或输出不是终端就退出

-u <vimrc> 使用 <vimrc> 替代任何 .vimrc

--noplugin 不加载 plugin 脚本

-p[N] 打开 N 个标签页 (默认值: 每个文件一个)

-o[N] 打开 N 个窗口 (默认值: 每个文件一个)

-O[N] 同 -o 但垂直分割

+ 启动后跳到文件末尾

+<lnum> 启动后跳到第 <lnum> 行

--cmd <command> 加载任何 vimrc 文件前执行 <command>

-c <command> 加载第一个文件后执行 <command>

-S <session> 加载第一个文件后执行文件 <session>

-s <scriptin> 从文件 <scriptin> 读入正常模式的命令

-w <scriptout> 将所有输入的命令追加到文件 <scriptout>

-W <scriptout> 将所有输入的命令写入到文件 <scriptout>

-x 编辑加密的文件

--startuptime <文件> 将启动计时信息写入 <文件>

-i <viminfo> 使用 <viminfo> 取代 .viminfo

--clean 'nocompatible', Vim 默认选项, 不加载插件, 不读写 viminfo

-h 或 --help 打印帮助(本信息)并退出

--version 打印版本信息并退出

### 重定向输出

将结果保存到文件中进行后续的分析/统计

> 覆盖输出

>> 追加输出

### cat

用法：cat [选项]... [文件]...

连接一个或多个 <文件> 并输出到标准输出。

如果没有指定 <文件>，或者 <文件> 为 "-"，则从标准输入读取。

-A, --show-all 等价于 -vET

-b, --number-nonblank 对非空输出行编号，使 -n 失效

-e 等价于 -vE

-E, --show-ends 在每行行末显示 "$"

-n, --number 对输出的所有行编号

-s, --squeeze-blank 不重复输出空行

-t 等价于 -vT

-T, --show-tabs 将 TAB 显示为 ^I

-u （被忽略）

-v, --show-nonprinting 使用 ^ 和 M- 表示法，LFD 和 TAB 字符除外

--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。

## 进阶指令

### df

显示每个 <文件> 所在的文件系统的信息，默认显示所有文件系统。

长选项的必选参数对于短选项也是必选的。

-a, --all 包含伪文件系统，以及重复的和无法访问的文件系统

-B, --block-size=大小 打印大小前将其除以 <大小>；例如，"-BM" 将以

1,048,576 字节为单位显示大小。

参见下方的 <大小> 格式

-h, --human-readable 以 1024 进制的单位显示大小（例如：1023M）

-H, --si 以 1000 进制的单位显示大小（例如：1.1G）

-i, --inodes 显示 inode 信息而非块使用量

-k 等于 --block-size=1K

-l, --local 只列出本地的文件系统

--no-sync 获取使用量信息前不调用 sync（默认）

--output[=字段列表] 使用 <字段列表> 定义的输出格式，

若省略 <字段列表>，则输出所有字段。

-P, --portability 使用 POSIX 输出格式

--sync 获取使用量信息前调用 sync

--total 省略所有对可用空间无显著影响的项，并生成总计值

-t, --type=类型 只显示指定 <类型> 的文件系统的信息

-T, --print-type 显示文件系统类型

-x, --exclude-type=类型 只显示不是指定 <类型> 的文件系统的信息

-v （被忽略）

--help 显示此帮助信息并退出

--version 显示版本信息并退出

所显示的数值的单位是 --block-size 选项的参数、DF\_BLOCK\_SIZE、BLOCK\_SIZE

和 BLOCKSIZE 环境变量的值中，第一个可用的 <大小>。

否则，默认单位是 1024 字节（如果 POSIXLY\_CORRECT 被设定，则为 512 字节）。

<大小> 参数是一个整数，后面可以跟一个单位（例如：10K 指 10\*1024）。

可用的单位有 K、M、G、T、P、E、Z、Y（1024 的幂）以及 KB、MB、...（1000 的幂）。

也可以使用二进制前缀：KiB=K、MiB=M，以此类推。

"字段列表" 是以逗号分隔的列表，指示需要包含在内的列。有效的字段名称包括：

"source"、"fstype"、"itotal"、"iused"、"iavail"、"ipcent"、"size"、

"used"、"avail"、"pcent"、"file" 和 "target"（请参考 info 信息页）。

### free

用法：查看内存使用情况

free [选项]

选项：

-b, --bytes 以字节显示输出

--kilo 使用千字节显示输出

--mega 使用兆字节显示输出

--giga 使用吉字节显示输出

--tera 使用太字节显示输出

--peta show output in petabytes

-k, --kibi show output in kibibytes

-m, --mebi show output in mebibytes

-g, --gibi show output in gibibytes

--tebi show output in tebibytes

--pebi show output in pebibytes

-h, --human 显示人类可读的输出

--si 使用 1000 的倍数而非使用 1024

-l, --lohi show detailed low and high memory statistics

-t, --total show total for RAM + swap

-s N, --seconds N repeat printing every N seconds

-c N, --count N repeat printing N times, then exit

-w, --wide 宽版输出

--help 显示此帮助信息并退出

-V, --version 显示程序版本然后离开

欲了解更多详细信息，请参见 free(1)。

### head指令

用法：head [选项]... [文件]...

将每个 <文件> 的前 10 行输出到标准输出。

如果指定了多个 <文件>，在输出每个文件之前先输出文件名作为头部。

如果没有指定 <文件>，或者 <文件> 为 "-"，则从标准输入读取。

长选项的必选参数对于短选项也是必选的。

-c, --bytes=[-]数字 显示每个文件的前 <数字> 字节内容；

如果数字前附加 "-" 字符，则显示每个文件

除了最后 <数字> 字节以外的全部内容

-n, --lines=[-]数字 显示每个文件的前 <数字> 行内容而非前 10 行内容；

如果数字前附加 "-" 字符，则显示每个文件

除了最后 <数字> 行以外的全部内容

-q, --quiet, --silent 不打印文件名作为头部

-v, --verbose 总是打印文件名作为头部

-z, --zero-terminated 以 NUL 空字符而非换行符作为行分隔符

--help 显示此帮助信息并退出

--version 显示版本信息并退出

所给定的数字 K 后面可以附加乘数后缀：

b 512, kB 1000, K 1024, MB 1000\*1000, M 1024\*1024,

GB 1000\*1000\*1000, G 1024\*1024\*1024, 对于 T, P, E, Z, Y 同样适用。

也可以使用二进制前缀，如：KiB=K，MiB=M 等等。

### tail

用法：tail [选项]... [文件]...

将每个 <文件> 的最后 10 行输出到标准输出。

如果指定了多个 <文件>，程序会在每个文件的开头添加文件名作为头部。

如果没有指定 <文件>，或者 <文件> 为 "-"，则从标准输入读取。

长选项的必选参数对于短选项也是必选的。

-c, --bytes=[+]数字 输出最后 <数字> 个字节；或者使用 -c +数字 以输出

每个文件第 <数字> 个字节起的全部内容

-f, --follow[={name|descriptor}]

随文件增长即时输出新增数据；

若未指定选项参数，则默认使用 "descriptor"

-F 同 --follow=name --retry

-n, --lines=[+]数字 输出最后 <数字> 行，而不是默认的最后 10 行；或者

使用 -n +数字 输出每个文件第 <数字> 行起的全部内容

--max-unchanged-stats=N

和 --follow=name 同时使用时, 如果一个 <文件> 在 N 次

（默认为 5 次）迭代后没有改变大小，则重新打开它，

以确认它是否已被删除或重命名（对于轮转 (rotated)

日志文件而言，这种情况很常见）；

如果内核支持 inotify，则此选项通常没有用处

--pid=PID 和 -f 同时使用时，在进程号为 PID 的进程结束后终止执行

-q, --quiet, --silent 不输出含有文件名的头

--retry 即使目标文件无法访问，仍然不断尝试打开

-s, --sleep-interval=N 和 -f 同时使用时，在两次迭代之间睡眠约 N 秒

（默认 1.0 秒）

内核支持 inotify 且使用了 --pid=P 时，每 N 秒

至少检查一次进程 P

-v, --verbose 总是输出含有文件名的头

-z, --zero-terminated 以 NUL 空字符而非换行符作为行分隔符

--help 显示此帮助信息并退出

--version 显示版本信息并退出

<数字> 后面可以加上乘数后缀：

b 512、kB 1000、K 1024、MB 1000\*1000、M 1024\*1024、

GB 1000\*1000\*1000、G 1024\*1024\*1024，T、P、E、Z、Y 以此类推。

也可以使用二进制前缀：KiB=K、MiB=M，以此类推。

### less

作用：查看文件，以较少的内容进行输入，按下辅助功能键查看更多

### wc指令

用法：wc [选项]... [文件]...

　或：wc [选项]... --files0-from=F

打印对每个给定文件的新行、单词和字节计数的结果；如果指定了超过

一个文件，则还会打印一个总行数。此处的一个单词是指由空白字符分隔的

长度大于零的字符序列。

如果没有指定 <文件>，或者 <文件> 为 "-"，则从标准输入读取。

可使用以下选项选择输出哪些种类的计数。输出总是遵循以下顺序：

换行符数、单词数、字符数、字节数、最长行的长度。

-c, --bytes 输出字节数

-m, --chars 输出字符数

-l, --lines 输出换行符数

--files0-from=F 从文件 F 中读取以 NUL 空字符分隔的文件名作为

输入文件的名称；

如果 F 是 "-"，则从标准输入读取文件名

-L, --max-line-length 输出最长行的长度

-w, --words 输出单词数

--help 显示此帮助信息并退出

--version 显示版本信息并退出

### date

用法：date [选项]... [+格式]

　或：date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]

以给定<格式>字符串的形式显示当前时间，或者设置系统日期。

长选项的必选参数对于短选项也是必选的。

-d, --date=字符串 显示指定 <字符串> 所描述的时间，而不是 "now"

--debug 对日期的解析过程进行注解，

对不规范的用法进行警告，并输出到标准错误

-f, --file=日期文件 类似 --date；对 <日期文件> 的每一行执行一次

-I[格式], --iso-8601[=格式] 以 ISO 8601 格式输出日期/时间。

<格式>="date" 时只输出日期（默认行为），

当其为 "hours"、"minutes"、"seconds" 或 "ns"

时则按照所指示的精确度显示日期和时间。

示例：2006-08-14T02:34:56-06:00

-R, --rfc-email 以 RFC 5322 格式输出日期和时间。

例如：Mon, 14 Aug 2006 02:34:56 -0600

--rfc-3339=格式 以 RFC 3339 格式输出日期/时间。

<格式>="date"、"seconds" 或 "ns"，

按照所指示的精确度显示日期和时间。

例如：2006-08-14 02:34:56-06:00

-r, --reference=文件 显示 <文件> 的最后修改时间

-s, --set=字符串 将时间设置为 <字符串> 所描述的时间

-u, --utc, --universal 按照协调世界时 (UTC) 显示或设置时间

--help 显示此帮助信息并退出

--version 显示版本信息并退出

<格式> 控制输出格式。支持解释的序列有：

%% 字面上的百分号 %

%a 当前区域设置的星期几的缩写（例如：日）

%A 当前区域设置的星期几的全称（例如：星期日）

%b 当前区域设置的月份缩写（例如：1月）

%B 当前区域设置的月份全称（例如：一月）

%c 当前区域设置的日期和时间（例如：2005年03月03日 星期四 23时05分25秒）

%C 世纪（指年份的最高两位）；类似 %Y，但省略最后两位数（例如：20）

%d 日（例如：01）

%D 日期；等于 %m/%d/%y

%e 日，以空格填充；等于 %\_d

%F 完整的日期；等于 %+4Y-%m-%d

%g ISO 周数年的最后两位（参见 %G）

%G ISO 周数年（参见 %V）；一般只和 %V 一起使用

%h 等于 %b

%H 小时 (00..23)

%I 小时 (01..12)

%j 一年中的第几日 (001..366)

%k 小时，以空格填充 ( 0..23)；等于 %\_H

%l 小时，以空格填充 ( 1..12)；等于 %\_I

%m 月 (01..12)

%M 分钟 (00..59)

%n 换行

%N 纳秒（000000000..999999999）

%p 当前地区时间设置中 AM 或 PM 的等效值（“上午”或“下午”），未知则为空

%P 类似 %p，但使用小写

%q 一年中的季度（1..4）

%r 当前地区时间中十二小时制钟表时间（例如，下午 11时11分04秒）

%R 24 小时制的时间和分钟；和 %H:%M 相同

%s 1970-01-01 00:00:00 UTC 以来的秒数

%S 秒 (00..60)

%t 制表符

%T 时间；等于 %H:%M:%S

%u 星期几 (1..7)；1 代表星期一

%U 一年中的第几周，以周日为每周第一天 (00..53)

%V ISO 周数，以周一为每周第一天 (01..53)

%w 星期几 (0..6)；0 代表星期日

%W 一年中的第几周，以周一为每周第一天 (00..53)

%x 当前区域设置的日期表示法（例如：1999年12月31日）

%X 当前区域设置的时间表示法（例如：23时13分48秒）

%y 年的最后两位 (00..99)

%Y 年

%z +hhmm 数字时区（例如：-0400）

%:z +hh:mm 数字时区（例如：-04:00）

%::z +hh:mm:ss 数字时区（例如：-04:00:00）

%:::z 数字时区，精度上有必要时加 ":"（例如：-04、+05:30）

%Z 字母时区缩写（例如：EDT）

默认情况下，date 输出的数字字段以 0 填充。

以下可选的标志可以跟在 "%" 后：

- （连字符）不填充该字段

\_ （下划线）以空格填充

0 （数字 0）以 0 填充

+ 以 0 填充，并且在超过四位数的未来年份之前添加 "+" 符号

^ 如果可能，使用大写字母

# 如果可能，使用相反的大小写

在任何标志之后可以有一个可选的字段宽度，它是一个十进制数字；

然后是一个可选的修饰符，它可以是

E，表示如果可用，使用区域设置的另一种表示法，或者

O，表示如果可用，使用区域设置的另一套数字符号。

示例：

将 epoch（1970-01-01 UTC）以来的秒数转换为日期

$ date --date='@2147483647'

显示美国东海岸的当前时间（请使用 tzselect(1) 来查询 TZ 的具体数值）

$ TZ='America/Los\_Angeles' date

显示美国东海岸下一个星期五当地时间早上九点

$ date --date='TZ="America/Los\_Angeles" 09:00 next Fri'

### cal

日历

### clear/ctrl+L

clear: invalid option -- '-'

Usage: clear [options]

Options:

-T TERM use this instead of $TERM

-V print curses-version

-x do not try to clear scrollback

### 管道

作用：过滤、特殊、扩展处理

# ls / | grep "y"

# ls | wc -l

## 高级指令

### hostname

查看/修改 主机名

### id

用法: id [选项]... [用户]...

输出指定<用户>的用户和用户组信息，

或（当没有指定用户信息时）默认使用当前用户信息 。

-a 忽略，仅为兼容其他版本而设计

-Z, --context 仅打印当前进程的安全上下文

-g, --group 仅打印有效组 ID

-G, --groups 仅打印所有的组 ID

-n, --name 打印名称而非数字，与 -ugG 一起使用

-r, --real 打印真实 ID 而非有效 ID，与 -ugG 一起使用

-u, --user 仅打印有效用户 ID

-z, --zero 使用 NUL 字符而非空格来分隔各个条目，在默认格式下不可使用

--help 显示此帮助信息并退出

--version 显示版本信息并退出

如果不附带任何 <选项>，程序会显示一组已识别到的、较为实用的信息。

### whoami 一般用于shell脚本

用法：whoami [选项]...

显示与当前的有效用户 ID 相关联的用户名。

与 id -un 相同。

--help 显示此帮助信息并退出

--version 显示版本信息并退出

### ps -ef指令（重点）

用法：

ps [选项]

基本选项：

-A, -e all processes

-a all with tty, except session leaders

a all with tty, including other users

-d all except session leaders

-N, --deselect negate selection

r only running processes

T all processes on this terminal

x processes without controlling ttys

如需了解更多细节，请阅读 ps(1)。

UID：用户ID

PID：进程ID

PPID：父进程ID

C：CPU占用率

STIME：进行的启动时间

TTY：终端设备，发起该进程的识别号

TIME：进程执行的时间

CMD：进程名称或者对应的路径

### top

查看服务器的进程占用的资源

procps-ng 3.3.17

用法：

top -hv | -bcEeHiOSs1 -d secs -n max -u|U user -p pid(s) -o field -w [cols]

进入：top

推出：q

VIRT：虚拟内存、RES：常驻内存、SHR：共享内存。

%CPU：使用率、%MEM：表示内存的占用百分比

COMMAND：进程的名称或者路径

1. M、1

### du -sh

用法：du [选项]... [文件]...

　或：du [选项]... --files0-from=F

计算每个文件的磁盘用量，目录则取总用量。

长选项的必选参数对于短选项也是必选的。

-0, --null 每行输出使用 NUL 空字符结尾而非使用换行符

-a, --all 输出所有文件的磁盘用量，不仅仅是目录

--apparent-size 显示表面用量，而并非是磁盘用量；虽然表面用量通常会

小一些，但有时它会因为稀疏文件间的“洞”、内部碎

片、非直接引用的块等原因显得更大一些。

-B, --block-size=大小 打印大小前将其除以 <大小>；例如，"-BM" 将以

1,048,576 字节为单位显示大小。

参见下方的 <大小> 格式

-b, --bytes 等价于 "--apparent-size --block-size=1"

-c, --total 显示总计信息

-D, --dereference-args 仅解引用在命令行中列出的符号链接

-d, --max-depth=N 仅当目录（指定了 --all 时也包括文件）深度小于或等于

<文件> 之下 N 层时才打印目录总大小；

--max-depth=0 等价于 --summarize

--files0-from=F 统计由文件 F 给出的由 NUL 空字符结束的文件名列表

所对应各个文件的信息；如果 F 是 -，则从标准输入

读取文件名列表

-H 等效于 --dereference-args (-D)

-h, --human-readable 以人类可读的格式输出大小（例如，1K 234M 2G）

--inodes 列出 inode 使用信息而非块使用信息

-k 等于 --block-size=1K

-L, --dereference 解引用所有符号链接

-l, --count-links 如果是硬链接，则重复计算其大小

-m 等于 --block-size=1M

-P, --no-dereference 不跟随任何符号链接（默认行为）

-S, --separate-dirs 对于目录，不包括子目录的大小

--si 类似 -h，但使用 1000 的幂，而不是 1024

-s, --summarize 仅对每个命令行参数输出一个总使用量

-t, --threshold=大小 如果 <大小> 为正值，排除比该值更小的项，

如果 <大小> 为负值，排除比该值更大的项

--time 显示目录下的所有文件的修改时间中最晚的那个，

或者所有子目录的修改时间

--time=关键字 显示 <关键字> 对应的时间，而非修改时间：

atime、access、use、ctime 或 status

--time-style=风格 使用指定的 <风格> 显示时间，可能的取值有：

full-iso、long-iso、iso 或 +格式；

<格式> 的解释方式和 "date" 命令相同

-X, --exclude-from=文件 排除和 <文件> 提供的任一模式匹配的文件

--exclude=模式 排除和 <模式> 匹配的文件

-x, --one-file-system 跳过位于不同文件系统上的目录

--help 显示此帮助信息并退出

--version 显示版本信息并退出

所显示的数值的单位是 --block-size 选项的参数、DU\_BLOCK\_SIZE、BLOCK\_SIZE

和 BLOCKSIZE 环境变量的值中，第一个可用的 <大小>。

否则，默认单位是 1024 字节（如果 POSIXLY\_CORRECT 被设定，则为 512 字节）。

<大小> 参数是一个整数，后面可以跟一个单位（例如：10K 指 10\*1024）。

可用的单位有 K、M、G、T、P、E、Z、Y（1024 的幂）以及 KB、MB、...（1000 的幂）。

也可以使用二进制前缀：KiB=K、MiB=M，以此类推。

### find

用法：find [-H] [-L] [-P] [-O级别] [-D 调试选项] [路径...] [表达式]

默认路径为当前目录；默认表达式为 -print

表达式可能由下列成份组成：操作符、选项、测试表达式以及动作：

操作符 (优先级递减；未做任何指定时默认使用 -and):

( EXPR ) ! EXPR -not EXPR EXPR1 -a EXPR2 EXPR1 -and EXPR2

EXPR1 -o EXPR2 EXPR1 -or EXPR2 EXPR1 , EXPR2

位置选项 (总是真): -daystart -follow -regextype

普通选项 (总是真，在其它表达式前指定):

-depth --help -maxdepth LEVELS -mindepth LEVELS -mount -noleaf

--version -xdev -ignore\_readdir\_race -noignore\_readdir\_race

测试(N可以是 +N 或-N 或 N):-amin N -anewer FILE -atime N -cmin

-cnewer 文件 -ctime N -empty -false -fstype 类型 -gid N -group 名称

-ilname 匹配模式 -iname 匹配模式 -inum N -ipath 匹配模式 -iregex 匹配模式

-links N -lname 匹配模式 -mmin N -mtime N -name 匹配模式 -newer 文件

-nouser -nogroup -path PATTERN -perm [-/]MODE -regex PATTERN

-readable -writable -executable

-wholename PATTERN -size N[bcwkMG] -true -type [bcdpflsD] -uid N

-used N -user NAME -xtype [bcdpfls] -context 上下文

actions: -delete -print0 -printf FORMAT -fprintf FILE FORMAT -print

-fprint0 FILE -fprint FILE -ls -fls FILE -prune -quit

-exec COMMAND ; -exec COMMAND {} + -ok COMMAND ;

-execdir COMMAND ; -execdir COMMAND {} + -okdir COMMAND ;

-D 选项有效的参数为：

exec, opt, rates, search, stat, time, tree, all, help

使用 "-D help" 以查看选项描述，或阅读 find(1)

find path -name/type/ string

### service

Usage: service < option > | --status-all | [ service\_name [ command | --full-restart ] ]

service servicename start/stop/restart

### kill

kill: kill [-s 信号说明符 | -n 信号编号 | -信号说明符] pid | 任务说明符 ... 或 kill -l [信号说明符]

向一个任务发送一个信号。

向以 <PID> 或者 <任务说明符> 指定的进程发送一个以 <信号说明符> 或

<信号编号> 指定的信号。如果没有指定 <信号说明符> 和 <信号编号>，

那么假定发送 SIGTERM 信号。

选项：

-s 信号 <信号> 是信号名称

-n 信号 <信号> 是信号编号

-l 列出所有信号名称；如果 "-l" 后有参数，则假定它们是信号编号，

并列出对应的信号名称

-L 同 -l

kill 成为 shell 内建有两个理由：它允许使用任务 ID 代替进程 ID，

并且在您可以创建的进程数达到上限时仍可以杀死进程。

退出状态：

返回成功，除非使用了无效的选项，或者有错误发生。

### ifconfig

查看网卡信息

### uptime

计算机的在线时间

### uname

获取系统信息

uname

uname -a

### netstat -tnlp

查看网络连接状态

usage: netstat [-vWeenNcCF] [<Af>] -r netstat {-V|--version|-h|--help}

netstat [-vWnNcaeol] [<Socket> ...]

netstat { [-vWeenNac] -i | [-cnNe] -M | -s [-6tuw] }

-r, --route 显示路由表

-i, --interfaces display interface table

-g, --groups display multicast group memberships

-s, --statistics display networking statistics (like SNMP)

-M, --masquerade display masqueraded connections

-v, --verbose 显示详细信息

-W, --wide don't truncate IP addresses

-n, --numeric 不解析名称

--numeric-hosts 不解析主机名

--numeric-ports 忽略端口名称

--numeric-users 忽略用户名

-N, --symbolic resolve hardware names

-e, --extend 显示更多信息

-p, --programs display PID/Program name for sockets

-o, --timers display timers

-c, --continuous continuous listing

-l, --listening display listening server sockets

-a, --all display all sockets (default: connected)

-F, --fib display Forwarding Information Base (default)

-C, --cache display routing cache instead of FIB

-Z, --context display SELinux security context for sockets

<Socket>={-t|--tcp} {-u|--udp} {-U|--udplite} {-S|--sctp} {-w|--raw}

{-x|--unix} --ax25 --ipx --netrom

<AF>=Use '-6|-4' or '-A <af>' or '--<af>'；默认： inet

列出所有支持的协议：

inet (DARPA Internet) inet6 (IPv6) ax25 (AMPR AX.25)

netrom (AMPR NET/ROM) ipx (Novell IPX) ddp (Appletalk DDP)

x25 (CCITT X.25)

### man

手册，帮助文档

### ping

ping: 无效的选项 -- -

Usage

ping [options] <destination>

Options:

<destination> dns name or ip address

-a use audible ping

-A use adaptive ping

-B sticky source address

-c <count> stop after <count> replies

-D print timestamps

-d use SO\_DEBUG socket option

-f flood ping

-h print help and exit

-I <interface> either interface name or address

-i <interval> seconds between sending each packet

-L suppress loopback of multicast packets

-l <preload> send <preload> number of packages while waiting replies

-m <mark> tag the packets going out

-M <pmtud opt> define mtu discovery, can be one of <do|dont|want>

-n no dns name resolution

-O report outstanding replies

-p <pattern> contents of padding byte

-q quiet output

-Q <tclass> use quality of service <tclass> bits

-s <size> use <size> as number of data bytes to be sent

-S <size> use <size> as SO\_SNDBUF socket option value

-t <ttl> define time to live

-U print user-to-user latency

-v verbose output

-V print version and exit

-w <deadline> reply wait <deadline> in seconds

-W <timeout> time to wait for response

IPv4 options:

-4 use IPv4

-b allow pinging broadcast

-R record route

-T <timestamp> define timestamp, can be one of <tsonly|tsandaddr|tsprespec>

IPv6 options:

-6 use IPv6

-F <flowlabel> define flow label, default is random

-N <nodeinfo opt> use icmp6 node info query, try <help> as argument

### tcpping

tcpping [-d] [-c] [-C] [-w sec] [-q num] [-x count] ipaddress [port]

-d 在每个响应时间前，打印时间戳

-c 以列表形式显示

-C 输出类似于fping工具中-C选项的结果

-w 等待时间（默认 3）

-r 每N秒重试一次（默认 1）

-x 限定测试总时长 (默认 无限)

实例：测试服务器到大陆TCP是否畅通

在这里，我们要用到百度官网的IP：119.75.217.109 以及他的TCP端口：80

tcping 119.75.217.109 80

### 17、traceroute

-d 使⽤Socket层级的排错功能。

-f 设置第⼀个检测数据包的存活数值TTL的大小。

-F 设置勿离断位。

-g 设置来源路由网关,最多可设置8个。

-i 使⽤指定的⽹络界面送出数据包。

-I 使用ICMP回应取代UDP资料信息。

-m 设置检测数据包的最大存活数值TTL的大小。

-n 直接使⽤IP地址而非主机名称。

-p 设置UDP传输协议的通信端⼝。

-r 忽略普通的Routing Table,直接将数据包送到远端主机上。

-s 设置本地主机送出数据包的IP地址。

-t 设置检测数据包的TOS数值。

-v 详细显表示指令的执行过程。

-w 设置等待远端主机回报的时间。

-x 开启或关实例

# 权限管理

## 运行级别

指系统在不同操作模式下的状态。传统的Linux系统使用了一种基于Inittab和init进程的运行级别系统来管理系统的启动和切换模式。随着Systemd的广泛采用，这种概念有所变化，但运行级别的概念依然存在，只是表现形式有所不同。

### 分类

传统的运行级别（Inittab/init）

在传统的Linux系统中，运行级别由/etc/inittab文件定义，并由init进程（PID 1）管理。以下是传统的运行级别：

0 (S): 停机（shutdown）。这个运行级别用于关机。

1 (S): 单用户模式（single user mode），通常用于系统维护，只有root用户能登录。

2: 多用户模式，没有NFS（Network File System）。在早期，这个级别可能用于没有网络服务的情况，但现在很少使用。

3: 完全多用户模式（multi-user），这是默认的运行级别，系统提供了完整的网络服务。

4: 保留未用，可以由系统管理员自定义。

5: 图形界面多用户模式（multi-user with GUI），在使用X Window图形界面的系统中，这个级别启动图形界面。

6 (S): 重新启动（reboot），这个运行级别用于重启系统。

其中带有S的运行级别表示特殊的启动模式（shutdown或reboot）。

现代Linux系统中的运行级别（Systemd）

现代的Linux发行版大多已经转向了Systemd作为初始化系统。Systemd不再直接使用数字来表示运行级别，而是使用目标（target）的概念来代替。一些常用的目标包括：

systemd.runlevel0.target: 等同于传统的运行级别0，用于关机。

systemd.runlevel1.target: 等同于传统的运行级别1，用于单用户模式。

systemd.runlevel2.target, systemd.runlevel3.target, systemd.runlevel4.target, systemd.runlevel5.target, systemd.runlevel6.target: 分别对应传统的运行级别2、3、4、5和6。

systemd.multi-user.target: 这个目标通常对应传统的运行级别3或5，表示多用户模式。

systemd.rescue.target: 救援模式，当系统无法正常启动时进入的一种简化模式，用于修复系统问题。

在Systemd中，可以通过systemctl isolate 命令来切换到不同的目标。此外，还可以使用systemctl get-default查看当前默认目标，并使用systemctl set-default 设置默认目标。

### 原理

在目录 /etc/rc.d/init.d 下有许多服务器脚本程序，一般称为服务(service)， 在 /etc/rc.d 下有 7 个名为 rcN.d 的目录，其中 N=0-6，对应于系统的 7 个运行级别， rcN.d 目录下，都是一些符号链接文件，这些链接文件都指向 init.d 目录下的 service 脚本文件，这些链接文件的命名规则是 “K+nn+服务名” 或 “S+nn+服务名”，其中 nn 为 2 位数字：

例: rc3.d 目录下的链接文件 S80sendmail 就指向 service 脚本文件 …/init.d/sendmail。系统会根据指定的 runlevel 进入对应的 rcN.d 目录，并按照文件名顺序检索目录下的链接文件

对于以 K 为开头的链接文件，系统将终止对应的服务

对于以 S 为开头的链接文件，系统将启动对应的服务

通过这种方式来实现 "不同的运行级别运行不同的程序和服务

### /etc/inittab配置文件详解

init 的进程号是 1，从这一点就能看出，init 进程是系统所有进程的起点，Linux 在完成核内引导以后，就开始运行 init 程序，init 程序需要读取设置文件/etc/inittab。inittab 是个不可执行的文本文件，他有若干行指令所组成。在 Redhat 系统中，inittab 的内容如下所示:

注：如果改变了 inittab 文件使他立即生效，需要使用一个命令：init q

每一行都有以下格式：

id:runlevel:action:process

1、id 是指入口标识符，他是个字符串，对于 getty 或 mingetty 等其他 login 程序项，需求 id 和 tty 的编号相同，否则 getty 程序将不能正常工作

2、runlevel 是 init 所处于的运行级别的标识，一般使用 0－6 及 S 或 s。0、1、6 运行级别被系统保留：其中 0 作为 shutdown 动作，1 作为重启至单用户模式，6 为重启；S 和 s 意义相同，表示单用户模式，且无需 inittab 文件，因此也不在 inittab 中出现，实际上，进入单用户模式时，init 直接在控制台（/dev/console）上运行/sbin/sulogin。在一般的系统实现中，都使用了 2、3、4、5 几个级别，在 Redhat系统中，2 表示无 NFS 支持的多用户模式，3 表示完全多用户模式（也是最常用的级别），4 保留给用户自定义，5 表示 XDM 图像登录方式。7－ 9 级别也是能使用的，传统的 Unix 系统没有定义这几个级别。runlevel 能是并列的多个值，以匹配多个运行级别，对大多数 action 来说，仅当 runlevel 和当前运行级别匹配成功才会执行。

3、action 是描述其后的 process 的运行方式的。action 可取的值包括：initdefault、sysinit、boot、bootwait等：

* nitdefault是个特别的action值，用于标识缺省的启动级别；当init由核心激活以后，他将读取inittab中的 initdefault 项，取得其中的 runlevel，并作为当前的运行级别。如果没有 inittab 文件，或其中没有initdefault 项， init 将在控制台上请求输入 runlevel。
* sysinit、boot、bootwait 等 action 将在系统启动时无条件运行，而忽略其中的 runlevel。
* 其余的 action（不含 initdefault）都和某个 runlevel 相关。各个 action 的定义在 inittab 的 man 手册中有周详的描述。

4、process 为具体的执行程序。程序后面能带参数。

### 相关命令

查看运行等级

$ runlevel

N 5 //’N’代表先前的 Runlevel; ‘5’代表目前的 Runlevel

切换系统运行等级

# init N //切换到运行级别 N

# init 0 //关机

# init 6 //重启动系统

## 用户（组）管理

### 用户权限

概念：权限=人+事物属性

人:一件事情是否允许被谁“做”，比如:我只能登录我自己的游戏账号，并对其进行操作，我没有权限去登录别人的游戏账号。我没有腾讯VIP，腾讯的部分电影我就无法观看。

事物属性:访问的对象是否有这种“属性“权限，比如:腾讯只能用来看视频，无法用来打游戏。

分类：超级用户、普通用户

超级用户：拥有对系统的完全访问权限，可以执行任何操作而不受限制。且一个Linux只能有一个root用户

普通用户：权限受到限制，只能执行有限的任务。且一个Linux可以有多个普通用户

切换用户：

语法：su [用户名]

功能：切换用户

从root用户切换到普通用户user，直接切换

与超级用户切换到普通用户不同，普通用户切换到超级用户共有两种方式:

su：所处工作目录不发生变换，用户身份发生变化

su -：相当于重新登录超级用户，切换后来到root用户的home目录下

sudo 提权

### 文件权限

文件属性

drwxr-xr-x 连接数 所属用户 所属用户组 文件大小 文件最新修改日期 文件名

文件类型、文件所有者权限、文件所属用户组权限、其他用户的权限

读(r):Read对文件而言，具有读取文件内容的权限;对目录来说，具有浏览该目录的权限

写(w):write对文件而言，具有修改文件内容的权限;对目录来说具有删除移动目录内文件的权限

执行(x):excute对文件而言，具有执行文件的权限;对目录来说，具有进入目录的权限

-:表示不具有该项权限

文件类型

-：普通文件，文件，可执行，归档文件等

d：目录

b：块设备，block，磁盘

c：字符设备，键盘或者显示器

s：网络socket文件

l：连接文件link

修改权限：

访问权限规定了3种访问文件或目录的方式:读(r)、写(w)、执行(x)。u表示文件所有者、g表示同组用户、o表示其它用户、a表示所有用户，+表示增加权限、-表示减掉权限、=表示赋予权限。

chmod命令格式：chmod [ugoa][+-=] 文件名或目录

举例：修改文件sd的权限为“r-xrw-r--”，即将文件 权限设置为“564”

chmod 564 sd

chown命令用来改变某个文件或目录的所有权

chown命令的格式为：chown 用户:组 文件名

## 账户文件和组文件

### 用户文件

passwd（用户账号文件）

文件作用：存储用户的关键信息

文件路径：/etc/passwd

文件内容格式：用户名:密码:用户id:用户组id:注释:用户home目录:解释器shell

shadow文件（用户密码文件）

操作系统将用户的口令加密之后放在另一个文件/etc/shadow中

文件作用：存储用户的密码信息

文件路径：/etc/shadow

每行文件内容分为九个字段，各字段之间用“:”分隔，其格式为：

username: encypted password: number of days: minimum password life: maximum

password life: warning period: disable account: account expiration: reserved

username：该字段是用户的登录名。

encypted password：该字段是已加密的用户口令(即密码)。

number of days ：该字段表示从1970年1月1日到上次修改密码的天数。

minimum password life：该字段表示两次修改口令之间至少要经过的天数。

maximmum password life：该字段表示口令有效期的最大天数，如果是99999则表示密码永不过期。

warning period ：该字段表示在密码失效前，提醒用户密码即将失效的天数。

disable account ：该字段表示密码过期之后，停用该帐号的天数。

account expiration：该字段表示帐号失效、禁止登录的时间。

Reserved：该字段暂未使用，系统保留。

### 组文件

group文件（用户组文件）

文件作用：存储用户组的关键信息

文件路径：/etc/group

文件内容格式：组名:组密码:用户组标识（GID）:组成员列表

组名：是用户组所在系统的标识

组密码：用户组的密码，由于安全原因，相应内容已转到gshadow文件中，在此用“X”表示，一般不设组密码。

用户组标识（GID）：组ID，在数字在系统内必须唯一。

组成员列表：组内的成员，多个成员之间用“:”隔开。

gshadow文件

该文件存放组的加密口令、组管理员等信息，只对root用户可以读取。

文件内容格式：组名称:加密后的组口令(没有就!):组的管理员:组成员列表

## 账户管理

用户：

useradd 用户名

passwd 用户名

userdel kk  //删除用户kk

ls /home    //查看home目录

userdel -r hh //删除hh，同时删除其属主目录

组：

groupadd 用户组名

gpasswd，修改组内成员

gpasswd [选项] 用户组 组名

gpasswd -a kang teacher //-a选项表示将用户添加到组内

gpasswd -d kang teacher //-d选项表示将用户从组内删除

groupdel 用户组名

# 网络

## 网络地址配置

网络地址查看–ifconfig

命令格式：ifconfig 或 ifconfig +网卡名

ifconfig [DEVICE] ##查看网络信息

​第一部分的第一行显示网卡状态信息。

* eth0表示第一块网卡。
* UP代表网卡开启状态。
* RUNNING代表网卡的网线被接上。
* MULTICAST表示支持组播。
* 第二行显示网卡的网络信息。
* inet（IP地址）
* broadcast（广播地址）
* netmask（掩码地址）
* RX表示接收数据包的情况，TX表示发送数据包的情况。
* lo表示主机的回环网卡，是一种特殊的网络接口，不与任何实际设备连接，而是完全由软件实现。与回环地址（127.0.0.0/8 或 ::1/128）不同，回环网卡对系统显示为一块硬件。任何发送到该网卡上的数据都将立刻被同一网卡接收到。

网络配置命令：

在Linux主机中，手工修改网络配置有两种最基本的方法：

1 临时配置：使用命令调整网络参数

　(1)简单、快速，可直接修改运行中的网络参数

　(2)一般只适合在调试网络的过程中使用

　(3) 系统重启以后，所做的修改将会失效

2 固定设置：通过配置文件修改网络参数

　(1) 修改各项网络参数的配置文件

　(2)适合对服务器设置固定参数时使用

　(3) 需要重载网络服务或者重启以后才会生效

　(4) 相对而言复杂一点，但相当于“永久配置”

设置：

ifconfig DEVICE IP netmask NETMASK ##设置ip地址

ifconfig eth0 192.168.168.1/24

禁用：

ifconfig 网络接口 up

ifconfig 网络接口 down

设置虚拟网络接口：

在对服务器进行调试的过程中，有时需临时在同一个网卡上使用一个新的IP地址，但又不能覆盖原有IP地址而导致服务程序不可用

相当于在一个网卡上配置多个IP地址

格式（示例)： ifconfig 网络接口:序号 IP地址

ifconfig ens33:1 11.11.11.11

修改网络配置文件

网络接口的配置文件默认位于目录“/etc/sysconfig/network-scripts/”中，文件名格式为“ifcfg-XXX”，其中“XXX”是网络接口的名称

“ifcfg-ens33”：是第一块以太网卡的配置文件

* TYPE：设置网卡类型，以上表示为以太网
* ONBOOT:设置网络接口是否在Linux系统启动时自动激活
* NETMASK：设置网络接口的子网掩码
* GATEMASK：设置网络接口的默认网关地址
* DNS：设置DNS服务器的IP地址

重启

systemctl restart network

Ubuntu系统进行网络配置

/etc/network/interfaces

操纵步骤：

打开ubuntu的/etc/network/interfaces文件默认的内容如下：

IP命令详解

ip 是个命令， ip 命令的功能很多！基本上它整合了 ifconfig 与 route 这两个命令，不过 ip 的功能更强大。属于iproute2包的一个命令，功能很强大。

-V：显示指令版本信息；

-s：-stats, -statistics输出更详细的信息；可以使用多个-s来显示更多的信息

-f：-family {inet, inet6, link} 强制使用指定的协议族；

-4：-family inet的简写，指定使用的网络层协议是IPv4协议；

-6：-family inet6的简写，指定使用的网络层协议是IPv6协议；

-0：shortcut for -family link.

-o：-oneline，输出信息每条记录输出一行，即使内容较多也不换行显示；

-r：-resolve，显示主机时，不使用IP地址，而使用主机的域名。

OBJECT对象

link ：网卡信息

address：IP地址信息

neighbour：邻居表

route：路由表

rule：IP策略

maddress：多播地址

mourte：组播路由缓存条目

tunnel：IP隧道

查看网络接口信息：

ip addr show ：显示所有网络接口的IP地址和相关信息。

ip link show ：显示所有网络接口的状态信息。

ip addr add [ip/mask] dev [interface] ：为指定网络接口添加IP地址

ip addr change [ip/mask] dev [interface] ：为指定网络接口修改IP地址

ip addr del [ip/mask] dev [interface] ：从指定网络接口删除IP地址

ip link set dev [interface] up/down ：启用或禁用指定的网络接口

ip link set设备接口属性

ip link set dev eth0 promisc on 开启混杂模式

ip link set eth0 up 启用网卡

ip link set eth0 mtu 1500 设置MTU值，即设置最大传输单元

ip link set eth0 name vbird 设置网卡名，需要在网卡down的时候进行设置，意义不是很大

ip link set eth0 address aa:aa:aa:aa:aa:aa 设置MAC地址

## 获取和修改主机名

查看主机名：

hostname

hostname [选项] [参数]

临时：

# 语法

hostname 新主机名

# 示例，修改新主机名为 centos-7

hostname centos-7

永久修改：

systemctl set-hostname 主机名

如果是 CentOS 7 及以上（包括 7）的版本，则需要修改 /etc/hostname 文件。直接填入新主机名即可。

## route命令

route [-nee]

-n ：不要使用通讯协定或主机名称，直接使用 IP 或 port number；

-ee ：使用更详细的资讯来显示

Destination 目标网络或目标主机（本机的数据要发送的目的地：子网或主机），与Genmask组成一个网段。

Gateway 网关(如果是默认网关，网关的地址必须和自己的主机上的其中一块网卡在同一子网)即网关地址。如果没有就显示星号。

Genmask 网络掩码，如果目标网络的的子网掩码为255.255.255.255,说明目标是一台主机；如果子网掩码为’0.0.0.0’ 说明该路由是默认路由。

Flags（旗标）

U (route is up)：该路由是启动的；

H (target is a host)：目标是一部主机 (IP) 而非网段；

G (use gateway)：需要透过外部的主机 (gateway) 来转递封包（该行有gw）；

R (reinstate route for dynamic routing)：使用动态路由时，恢复路由资讯的旗标；

D (dynamically installed by daemon or redirect)：已经由服务或转 port 功能设定为动态路由

M (modified from routing daemon or redirect)：路由已经被修改；

! (reject route)：这个路由将不会被接受(用来抵挡不安全的网域！)

A (installed by addrconf)

C (cache entry)

添加路由：

route add [-net|-host] [网域或主机] netmask [mask] [gw|dev]

route del [-net|-host] [网域或主机] netmask [mask] [gw|dev]

参数：

-net ：表示后面接的路由为一个网域（网段）的路由；

-host ：表示后面接的为连接到单部主机的路由；

netmask ：掩码，决定了网域的大小（配合-net使用，构成一个网段）；

gw ：gateway 的简写，后续接的是 IP （必须和本机的其中一块网卡处于同一网段），与 dev 不同；

dev ：如果只是要指定由哪一块网卡连线出去，则使用这个设定，后面接 eth0了，eth1 等

删除路由：

格式：

route del -net {NETWORK-ADDRESS} netmask {NETMASK} reject

添加、删除默认网关记录

在同一个主机的路由表中只有一条默认网关记录，若同时存在多条默认网关记录，则可能导致该主机的网络连接出现故障。

添加删除默认网关记录时，与添加、删除静态路由记录的命令格式类型，但制定目标网段时，只需简单地使用“ default ”表示即可。

添加默认网关 route add default gw xxx.xx.x.xxx

删除默认网关 route del default gw xxx.xxx.x.xxx

## socket统计 ss命令

ss [option]

-h, --help 帮助信息

-V, --version 程序版本信息

-n, --numeric 不解析服务名称

-r, --resolve 解析主机名

-a, --all 显示所有套接字（sockets）

-l, --listening 显示监听状态的套接字（sockets）

-o, --options 显示计时器信息

-e, --extended 显示详细的套接字（sockets）信息

-m, --memory 显示套接字（socket）的内存使用情况

-p, --processes 显示使用套接字（socket）的进程

-i, --info 显示 TCP内部信息

-s, --summary 显示套接字（socket）使用概况

-4, --ipv4 仅显示IPv4的套接字（sockets）

-6, --ipv6 仅显示IPv6的套接字（sockets）

-0, --packet 显示 PACKET 套接字（socket）

-t, --tcp 仅显示 TCP套接字（sockets）

-u, --udp 仅显示 UCP套接字（sockets）

-d, --dccp 仅显示 DCCP套接字（sockets）

-w, --raw 仅显示 RAW套接字（sockets）

-x, --unix 仅显示 Unix套接字（sockets）

-f, --family=FAMILY 显示 FAMILY类型的套接字（sockets），FAMILY可选，支持 unix, inet, inet6, link, netlink

-A, --query=QUERY, --socket=QUERY

QUERY := {all|inet|tcp|udp|raw|unix|packet|netlink}[,QUERY]

-D, --diag=FILE 将原始TCP套接字（sockets）信息转储到文件

-F, --filter=FILE 从文件中都去过滤器信息

FILTER := [ state TCP-STATE ] [ EXPRESSION ]

## 域名解析 nslookup

DNS英文全称Domain Name System，它是域名系统，在网络中的作用就是维护着一个地址数据库，其中记录了各种主机域名与IP地址的对应关系，以便为客户程序提供正向或反向的地址查询服务，及正向解析与反向解析：

① 正向解析：将指定的域名解析为相应的IP地址

② 反向解析：将指定的IP地址解析为相对应的域名。

yum install -y bind-utils

nslookup internationalsaimoe.com

主服务器配置文件

配置文件：/etc/resolv.conf

①/etc/resolv.conf 文件记录了本机默认使用的DNS服务器的地址信息，对于该文件所做的修改将会立刻

生效。

②Linux系统中最多可以指定3个不同的DNS服务器（超过3个的会被忽略），有限使用第一个DNS服务

器。

③resolv.cong文件中的“searh localdomain”行用来设置默认的搜索域（域名后缀），例如：当访问主机

“localhost”时，就相当于访问“localhost.localdomain ”。

保存主机名与ip地址的映射记录

映射文件路径：/etc/hosts

配置映射关系

编辑文件vim /etc/hosts，在文件末尾添加映射关系，比如：192.168.171.72 dsports-asp

hosts文件和DNS服务器比较

①默认情况下，系统首先从hosts文件查找解析记录

②hosts文件只对当前主机有效

③hosts文件可减少DNS查询过程，从加快访问速度

## 防火墙

防⽕墙是保护机器不受来⾃外部的、不需要的⽹络数据的⼀种⽅式。它允许⽤⼾通过定义⼀组防⽕墙规则来控制主机上的⼊站⽹络流量。这些规则⽤于对进⼊的流量进⾏排序，并可以阻断或允许流量。

firewalld 是⼀个防⽕墙服务守护进程

firewalld 使⽤区和服务的概念来简化流量管理。

----zones 是预定义的规则集。⽹络接⼝和源可以分配给区。允许的流量取决于您计算机连接到的⽹络，并分配了这个⽹络的安全级别。

----防⽕墙服务是预定义的规则，覆盖了允许特定服务进⼊流量的所有必要设置，并在区中应⽤。

服务使⽤⼀个或多个端⼝或地址进⾏⽹络通信。防⽕墙会根据端⼝过滤通讯。要允许服务的⽹络

流量，必须打开其端⼝。 firewalld 会阻⽌未明确设置为打开的端⼝的所有流量。⼀些区

（如可信区）默认允许所有流量。

zones

可以根据⽤⼾对该⽹络中的接⼝和流量设置的信任程度，使⽤ firewalld 来将⽹络划分为

不同的区。⼀个连接只能是⼀个区的⼀部分，但⼀个区可以被⽤来进⾏很多⽹络连接。

已经存在的预定义的区存储在 /usr/lib/firewalld/zones/ 文件中

预定义服务

服务可以是本地端⼝、协议、源端⼝和⽬的地列表，并在启⽤了服务时⾃动载⼊防⽕墙帮助程序

模块列表。使⽤服务可节省⽤⼾时间，因为它们可以完成⼀些任务，如打开端⼝、定义协议、启

⽤数据包转发等等，⽽不必在另外的步骤中设置所有任务。

服务配置选项和通⽤⽂件信息在 firewalld.service(5) ⼿册⻚中进⾏了描述。

可 使 ⽤ 图 形 化 的 firewall-config ⼯ 具 、 firewall-cmd 和 firewall-offline-cmd来添加和删除服务。

或者可以在 /etc/firewalld/services/ 目录中编辑xml 文件，如果⽤⼾没有添加或更改服务，则在 /etc/firewalld/services/ 中找不到相应的xml文件，如果要添加或更改服务，/usr/lib/firewalld/services/文件可以作为参考

开启

systemctl unmask firewalld

systemctl start firewalld

关闭

systemctl stop firewalld

systemctl disable firewalld

unmask： 将不标记 firewalld 服务 ，如果为 mask 无论怎么都不启动防火墙

stop： 停止防火墙，重新启动系统时还是会重新启动防护墙

disable： 在系统重新启动时也不启动防火墙

firewarll-cmd --state 显示状态

systemctl status firewalld.service 更详细的设置

列出默认区所以相关信息

firewall-cmd --list-all

列出指定区域的相关信息

firewall-cmd --list-all --zone=block

查看帮助信息

irewall-cmd --help

firewalld控制流量

firewalld 软件包安装⼤量预定义的服务⽂件，您可以添加更多或⾃定义它们。然后，您可以使⽤这些服务定义为服务打开或关闭端⼝，⽽⽆需了解这些服务使⽤的协议和端⼝号。

使⽤ CLI 禁⽤紧急事件的所有流量

1，要⽴即禁⽤⽹络流量，请切换 panic 模式：

启⽤ panic 模式可停⽌所有⽹络流量。因此，只有当您具有对机器的物理访问权限或使⽤串⾏控制台登录时，才应使⽤它。

firewall-cmd --panic-on

使⽤ CLI 控制预定义服务的流量

控制流量的最简单的⽅法是在 firewalld 中添加预定义的服务，这会打开所有必需的端⼝并根据 服务定义⽂件 修改其他设置。

列出被添加的服务

firewall-cmd --list-services

列出所有的预定义服务

firewall-cmd --get-services

在允许的服务中添加服务

firewall-cmd --add-service=http 默认在默认区域添加，但是重载后该服务就会不见，需要重新添加

永久添加这个服务

firewall-cmd --add-service=http --permanent 重载后也不会不见

打开端口

通过打开端⼝，系统可从外部访问，这代表了安全⻛险。通常，让端⼝保持关闭，且只在某些服务需要时才打开。

列出所有允许的端口

firewall-cmd --list-ports

在允许的端⼝中添加⼀个端⼝，以便为⼊站流量打开这个端⼝

firewall-cmd --add-port=80/tcp 添加80端口，类型为tcp

端⼝类型为 tcp，udp ，sctp，dccp 这个类型必须与⽹络通信的类型匹配

关闭端口

当打开的端⼝不再需要时，在 firewalld 中关闭此端⼝。强烈建议您尽快关闭所有不必要的端⼝，因为端⼝处于打开状态会存在安全隐患。

从允许的端口中删除端口，以便对传⼊的流量关闭

firewall-cmd --remove-port=80/tcp

firewalld 区

zones 代表⼀种更透明管理传⼊流量的概念。这些区域连接到联⽹接⼝或者分配⼀系列源地址。您可以独⽴为每个区管理防⽕墙规则，这样就可以定义复杂的防⽕墙设置并将其应⽤到流量。

列出系统中可用的区域

firewall-cmd --get-zones

查看所有系统中可用区的详细信息：

firewall-cmd --list-all-zones

例如，要允许在区public 中使⽤ SSH 服务

firewall-cmd --add-service=ssh --zone=public

更改默认区

系统管理员在其配置⽂件中为⽹络接⼝分配区域。如果接⼝没有被分配给指定区，它将被分配给默认区。每次重启 firewalld 服务后，firewalld 会加载默认区的设置，并使其处于活动状态。

当系统的前默认区

firewall-cmd --get-default-zone

设置默认区为home区

firewall-cmd --set-default-zone home

[root@bogon ~]# firewall-cmd --get-active-zones

public

interfaces: ens160

使用nmcli命令为连接分配区域

nmcli connection modify ens160 connection.zone public 分配到public区

创建⼀个新区

要使⽤⾃定义区，创建⼀个新的区并使⽤它像预定义区⼀样。新区需要 --permanent 选项，否则命令⽆法正常⼯作。

例：创建⼀个新区testzone

[root@bogon ~]# firewall-cmd --new-zone=testzone --permanent

[root@bogon ~]# firewall-cmd --reload

success

[root@bogon ~]# firewall-cmd --get-zones

block dmz drop external home internal libvirt nm-shared public testzone trusted work

# shell使用

Shell 是一个用 C 语言编写的程序，它是用户使用 Linux 的桥梁。Shell 既是一种命令语言，又是一种程序设计语言。

Shell 是指一种应用程序，这个应用程序提供了一个界面，用户通过这个界面访问操作系统内核的服务。

Ken Thompson 的 sh 是第一种 Unix Shell，Windows Explorer 是一个典型的图形界面 Shell。

## 基础

打开文本编辑器(可以使用 vi/vim 命令来创建文件)，新建一个文件 test.sh，扩展名为 sh（sh代表shell），扩展名并不影响脚本执行，见名知意就好，如果你用 php 写 shell 脚本，扩展名就用 php 好了。

#!/bin/bash

echo "Hello World !"

**#!** 是一个约定的标记，它告诉系统这个脚本需要什么解释器来执行，即使用哪一种 Shell。

echo 命令用于向窗口输出文本。

Shell 编程跟 JavaScript、php 编程一样，只要有一个能编写代码的文本编辑器和一个能解释执行的脚本解释器就可以了。

Linux 的 Shell 种类众多，常见的有：

Bourne Shell（/usr/bin/sh或/bin/sh）

Bourne Again Shell（/bin/bash）

C Shell（/usr/bin/csh）

K Shell（/usr/bin/ksh）

Shell for Root（/sbin/sh）

……

本教程关注的是 Bash，也就是 Bourne Again Shell，由于易用和免费，Bash 在日常工作中被广泛使用。同时，Bash 也是大多数Linux 系统默认的 Shell。

在一般情况下，人们并不区分 Bourne Shell 和 Bourne Again Shell，所以，像 #!/bin/sh，它同样也可以改为 #!/bin/bash。

#! 告诉系统其后路径所指定的程序即是解释此脚本文件的 Shell 程序。

## shell变量

在 Shell 编程中，变量是用于存储数据值的名称。

定义变量时，变量名不加美元符号（$，PHP语言中变量需要），如：

your\_name="runoob"

注意，变量名和等号之间不能有空格，这可能和你熟悉的所有编程语言都不一样。同时，变量名的命名须遵循如下规则：

* 只包含字母、数字和下划线： 变量名可以包含字母（大小写敏感）、数字和下划线 \_，不能包含其他特殊字符。
* 不能以数字开头： 变量名不能以数字开头，但可以包含数字。
* 避免使用 Shell 关键字： 不要使用Shell的关键字（例如 if、then、else、fi、for、while 等）作为变量名，以免引起混淆。
* 使用大写字母表示常量： 习惯上，常量的变量名通常使用大写字母，例如 PI=3.14。
* 避免使用特殊符号： 尽量避免在变量名中使用特殊符号，因为它们可能与 Shell 的语法产生冲突。
* 避免使用空格： 变量名中不应该包含空格，因为空格通常用于分隔命令和参数。

将 /etc 下目录的文件名循环出来。

for file in `ls /etc`

或

for file in $(ls /etc)

使用一个定义过的变量，只要在变量名前面加美元符号即可，如：

your\_name="qinjx"

echo $your\_name

echo ${your\_name}

变量名外面的花括号是可选的，加不加都行，加花括号是为了帮助解释器识别变量的边界，比如下面这种情况：

for skill in Ada Coffe Action Java; do

echo "I am good at ${skill}Script"

done

只读变量

使用 readonly 命令可以将变量定义为只读变量，只读变量的值不能被改变。

下面的例子尝试更改只读变量，结果报错：

#!/bin/bash

myUrl="https://www.google.com"

readonly myUrl

myUrl="https://www.runoob.com"

删除变量

使用 unset 命令可以删除变量。语法：

unset variable\_name

变量被删除后不能再次使用。unset 命令不能删除只读变量。

变量类型

Shell 支持不同类型的变量，其中一些主要的类型包括：

字符串变量： 在 Shell中，变量通常被视为字符串。

my\_string='Hello, World!'

单引号字符串的限制：

* 单引号里的任何字符都会原样输出，单引号字符串中的变量是无效的；
* 单引号字符串中不能出现单独一个的单引号（对单引号使用转义符后也不行），但可成对出现，作为字符串拼接使用。

双引号的优点：

* 双引号里可以有变量
* 双引号里可以出现转义字符

整数变量： 在一些Shell中，你可以使用 declare 或 typeset 命令来声明整数变量。

declare -i my\_integer=42

这样的声明告诉 Shell 将 my\_integer 视为整数，如果尝试将非整数值赋给它，Shell会尝试将其转换为整数。

数组变量： Shell 也支持数组，允许你在一个变量中存储多个值。

数组可以是整数索引数组或关联数组，以下是一个简单的整数索引数组的例子：

my\_array=(1 2 3 4 5)

关联数组

declare -A associative\_array

associative\_array["name"]="John"

associative\_array["age"]=30

环境变量： 这些是由操作系统或用户设置的特殊变量，用于配置 Shell 的行为和影响其执行环境。

echo $PATH

特殊变量： 有一些特殊变量在 Shell 中具有特殊含义，例如 $0 表示脚本的名称，$1, $2, 等表示脚本的参数。

$#表示传递给脚本的参数数量，$? 表示上一个命令的退出状态等。

字符串：

string="abcd"

echo ${#string} # 输出 4

string="abcd"

echo ${#string[0]} # 输出 4

string="runoob is a great site"

echo ${string:1:4} # 输出 unoo

string="runoob is a great site"

echo `expr index "$string" io` # 输出 4

以上脚本中 ` 是反引号，而不是单引号 '

数组：

bash支持一维数组（不支持多维数组），并且没有限定数组的大小。

类似于 C 语言，数组元素的下标由 0 开始编号。获取数组中的元素要利用下标，下标可以是整数或算术表达式，其值应大于或等于 0。

在 Shell 中，用括号来表示数组，数组元素用"空格"符号分割开。定义数组的一般形式为：

数组名=(值1 值2 ... 值n)

array\_name=(

value0

value1

value2

value3

)

array\_name[0]=value0

array\_name[1]=value1

array\_name[n]=valuen

可以不使用连续的下标，而且下标的范围没有限制。

读取数组值：

${数组名[下标]}

读取所有值：

echo ${array\_name[@]}

# 取得数组元素的个数

length=${#array\_name[@]}

# 或者

length=${#array\_name[\*]}

# 取得数组单个元素的长度

length=${#array\_name[n]}

注释：

以 # 开头的行就是注释，会被解释器忽略。

通过每一行加一个 # 号设置多行注释

多行

:<<EOF

注释内容...

注释内容...

注释内容...

EOF

以上例子中，: 是一个空命令，用于执行后面的 Here 文档，<<'EOF' 表示开启 Here 文档，COMMENT 是 Here 文档的标识符，在这两个标识符之间的内容都会被视为注释，不会被执行。

EOF 也可以使用其他符号:

: <<'COMMENT'

这是注释的部分。

可以有多行内容。

COMMENT

:<<'

注释内容...

注释内容...

注释内容...

'

:<<!

注释内容...

注释内容...

注释内容...

!

直接使用 : 号

我们也可以使用了冒号 : 命令，并用单引号 ' 将多行内容括起来。由于冒号是一个空命令，这些内容不会被执行。

格式为：: + 空格 + 单引号。

## 参数传递

我们可以在执行 Shell 脚本时，向脚本传递参数，脚本内获取参数的格式为 $n，n 代表一个数字，1 为执行脚本的第一个参数，2 为执行脚本的第二个参数。

例如可以使用 $1、$2 等来引用传递给脚本的参数，其中 $1 表示第一个参数，$2 表示第二个参数，依此类推。

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

echo "Shell 传递参数实例！";

echo "执行的文件名：$0";

echo "第一个参数为：$1";

echo "第二个参数为：$2";

echo "第三个参数为：$3";

$ chmod +x test.sh

$ ./test.sh 1 2 3

Shell 传递参数实例！

执行的文件名：./test.sh

第一个参数为：1

第二个参数为：2

第三个参数为：3

## 数组

关联数组

declare -A site=(["google"]="www.google.com" ["runoob"]="www.runoob.com" ["taobao"]="www.taobao.com")

declare -A site

site["google"]="www.google.com"

site["runoob"]="www.runoob.com"

site["taobao"]="www.taobao.com"

获取所有元素：

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

my\_array[0]=A

my\_array[1]=B

my\_array[2]=C

my\_array[3]=D

echo "数组的元素为: ${my\_array[\*]}"

echo "数组的元素为: ${my\_array[@]}"

在数组前加一个！可以获取数组的所有键。

declare -A site

site["google"]="www.google.com"

site["runoob"]="www.runoob.com"

site["taobao"]="www.taobao.com"

echo "数组的键为: ${!site[\*]}"

echo "数组的键为: ${!site[@]}"

## 运算符

Shell 和其他编程语言一样，支持多种运算符，包括：

算数运算符

关系运算符

布尔运算符

字符串运算符

文件测试运算符

原生bash不支持简单的数学运算，但是可以通过其他命令来实现，例如 awk 和 expr，expr 最常用。

expr 是一款表达式计算工具，使用它能完成表达式的求值操作。

例如，两个数相加(注意使用的是反引号 ` 而不是单引号 ')：

#!/bin/bash

val=`expr 2 + 2`

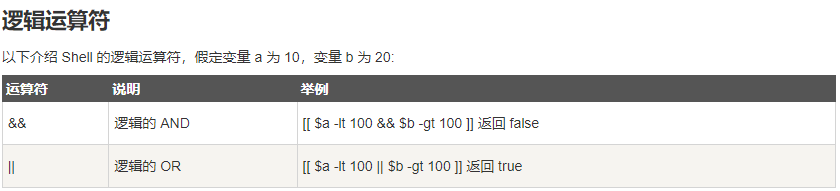
echo "两数之和为 : $val"

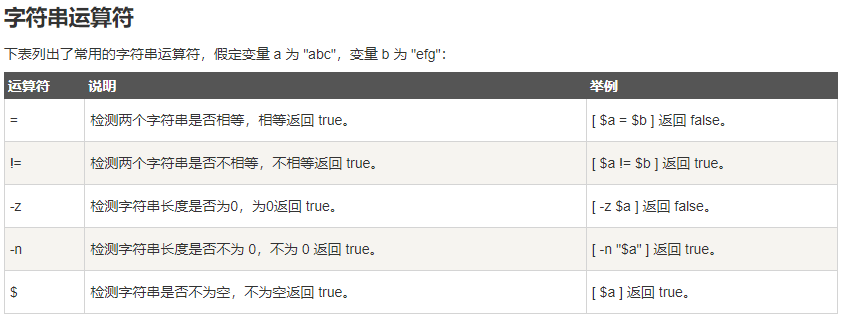
两点注意：

* 表达式和运算符之间要有空格，例如 2+2 是不对的，必须写成 2 + 2，这与我们熟悉的大多数编程语言不一样。
* 完整的表达式要被 ` ` 包含，注意这个字符不是常用的单引号，在 Esc 键下边。











自增自减

使用 let 命令，命令允许对整数进行算术运算。

#!/bin/bash

# 初始化变量

num=5

# 自增

let num++

# 自减

let num--

echo $num

使用$(( ))语法

#!/bin/bash

# 初始化变量

num=5

# 自增

num=$((num + 1))

# 自减

num=$((num - 1))

echo $num

使用expr，在现代脚本中不如 let 和 $(( )) 常用。

#!/bin/bash

# 初始化变量

num=5

# 自增

num=$(expr $num + 1)

# 自减

num=$(expr $num - 1)

echo $num

使用(( ))，与 $(( )) 类似

#!/bin/bash

# 初始化变量

num=5

# 自增

((num++))

# 自减

((num--))

echo $num

## echo命令

echo string

echo "It is a test"

echo It is a test

**echo "\"It is a test\""**

#!/bin/sh

read name

echo "$name It is a test"

echo -e "OK! \n" # -e 开启转义

echo "It is a test"

#!/bin/sh

echo -e "OK! \c" # -e 开启转义 \c 不换行

echo "It is a test"

结果定向：

echo "It is a test" > myfile

显示命令执行结果

echo `date`

## printf命令

rintf 不会像 **echo** 自动添加换行符，我们可以手动添加 **\n**。

printf 命令的语法：

printf format-string [arguments...]

$ echo "Hello, Shell"

Hello, Shell

$ printf "Hello, Shell\n"

Hello, Shell

$

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

printf "%-10s %-8s %-4s\n" 姓名 性别 体重kg

printf "%-10s %-8s %-4.2f\n" 郭靖 男 66.1234

printf "%-10s %-8s %-4.2f\n" 杨过 男 48.6543

printf "%-10s %-8s %-4.2f\n" 郭芙 女 47.9876

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

# format-string为双引号

printf "%d %s\n" 1 "abc"

# 单引号与双引号效果一样

printf '%d %s\n' 1 "abc"

# 没有引号也可以输出

printf %s abcdef

# 格式只指定了一个参数，但多出的参数仍然会按照该格式输出，format-string 被重用

printf %s abc def

printf "%s\n" abc def

printf "%s %s %s\n" a b c d e f g h i j

# 如果没有 arguments，那么 %s 用NULL代替，%d 用 0 代替

printf "%s and %d \n"

## test命令

Shell中的 test 命令用于检查某个条件是否成立，它可以进行数值、字符和文件三个方面的测试。

num1=100

num2=100

if test $[num1] -eq $[num2]

then

echo '两个数相等！'

else

echo '两个数不相等！'

fi

#!/bin/bash

a=5

b=6

result=$[a+b] # 注意等号两边不能有空格

echo "result 为： $result"

另外，Shell 还提供了与( -a )、或( -o )、非( ! )三个逻辑操作符用于将测试条件连接起来，其优先级为： **!** 最高， **-a** 次之， **-o** 最低。例如：

## 流程控制

和 Java、PHP 等语言不一样，sh 的流程控制不可为空

if else:

if condition

then

command1

command2

...

commandN

fi

写成一行（适用于终端命令提示符）：

if [ $(ps -ef | grep -c "ssh") -gt 1 ]; then echo "true"; fi

if else 语法格式：

if condition

then

command1

command2

...

commandN

else

command

fi

if else-if else:

if condition1

then

command1

elif condition2

then

command2

else

commandN

fi

if else 的 **[...]** 判断语句中大于使用 **-gt**，小于使用 **-lt**。

如果使用 **((...))** 作为判断语句，大于和小于可以直接使用 **>** 和 **<**。

for 循环

for循环一般格式为：

for var in item1 item2 ... itemN

do

command1

command2

...

commandN

done

for var in item1 item2 ... itemN; do command1; command2… done;

while循环：

while 循环用于不断执行一系列命令，也用于从输入文件中读取数据。其语法格式为：

while condition

do

command

done

until循环：

#!/bin/bash

a=0

until [ ! $a -lt 10 ]

do

echo $a

a=`expr $a + 1`

done

case ... esac:

**case ... esac** 语法格式如下：

case 值 in

模式1)

command1

command2

...

commandN

;;

模式2)

command1

command2

...

commandN

;;

esac

echo '输入 1 到 4 之间的数字:'

echo '你输入的数字为:'

read aNum

case $aNum in

1) echo '你选择了 1'

;;

2) echo '你选择了 2'

;;

3) echo '你选择了 3'

;;

4) echo '你选择了 4'

;;

\*) echo '你没有输入 1 到 4 之间的数字'

;;

esac

跳出循环：

使用两个命令来实现该功能：**break** 和 **continue**。

break 命令

break 命令允许跳出所有循环（终止执行后面的所有循环）。

下面的例子中，脚本进入死循环直至用户输入数字大于5。要跳出这个循环，返回到shell提示符下，需要使用break命令。

#!/bin/bash

while :

do

echo -n "输入 1 到 5 之间的数字:"

read aNum

case $aNum in

1|2|3|4|5) echo "你输入的数字为 $aNum!"

;;

\*) echo "你输入的数字不是 1 到 5 之间的! 游戏结束"

break

;;

esac

done

continue

continue 命令与 break 命令类似，只有一点差别，它不会跳出所有循环，仅仅跳出当前循环。

对上面的例子进行修改：

#!/bin/bash

while :

do

echo -n "输入 1 到 5 之间的数字: "

read aNum

case $aNum in

1|2|3|4|5) echo "你输入的数字为 $aNum!"

;;

\*) echo "你输入的数字不是 1 到 5 之间的!"

continue

echo "游戏结束"

;;

esac

done

## shell函数

定义格式如下：

[ function ] funname [()]

{

action;

[return int;]

}

说明：

* 1、可以带 **function fun()** 定义，也可以直接 **fun()** 定义,不带任何参数。
* 2、参数返回，可以显示加：**return** 返回，如果不加，将以最后一条命令运行结果，作为返回值。 **return** 后跟数值 **n(0-255)**.

下面的例子定义了一个函数并进行调用：

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

demoFun(){

echo "这是我的第一个 shell 函数!"

}

echo "-----函数开始执行-----"

demoFun

echo "-----函数执行完毕-----"

带return

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

funWithReturn(){

echo "这个函数会对输入的两个数字进行相加运算..."

echo "输入第一个数字: "

read aNum

echo "输入第二个数字: "

read anotherNum

echo "两个数字分别为 $aNum 和 $anotherNum !"

return $(($aNum+$anotherNum))

}

funWithReturn

echo "输入的两个数字之和为 $? !"

函数返回值在调用该函数后通过 **$?**来获得。

**注意：**所有函数在使用前必须定义。这意味着必须将函数放在脚本开始部分，直至shell解释器首次发现它时，才可以使用。调用函数仅使用其函数名即可。

**注意：** **return** 语句只能返回一个介于 0 到 255 之间的整数，而两个输入数字的和可能超过这个范围。

要解决这个问题，您可以修改 return 语句，直接使用 echo 输出和而不是使用 return：

函数参数：

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

funWithParam(){

echo "第一个参数为 $1 !"

echo "第二个参数为 $2 !"

echo "第十个参数为 $10 !"

echo "第十个参数为 ${10} !"

echo "第十一个参数为 ${11} !"

echo "参数总数有 $# 个!"

echo "作为一个字符串输出所有参数 $\* !"

}

funWithParam 1 2 3 4 5 6 7 8 9 34 73

注意，$10 不能获取第十个参数，获取第十个参数需要${10}。当n>=10时，需要使用${n}来获取参数。

另外，还有几个特殊字符用来处理参数：



# 资源情况了解

## CPU使用情况

top指令

其中：

%us：表示用户空间程序的cpu使用率（没有通过nice调度）

%sy：表示系统空间的cpu使用率，主要是内核程序。

%ni：表示用户空间且通过nice调度过的程序的cpu使用率。

%id：空闲cpu

%wa：cpu运行时在等待io的时间

%hi：cpu处理硬中断的数量

%si：cpu处理软中断的数量

%st：被虚拟机偷走的cpu

PID - 进程标示号

USER - 进程所有者

PR - 进程优先级

NI - 进程优先级别数值

VIRT - 进程占用的虚拟内存值

RES - 进程占用的物理内存值

SHR - 进程使用的共享内存值

S - 进程的状态，其中S表示休眠，R表示正在运行，Z表示僵死

%CPU - 进程占用的CPU使用率

%MEM - 进程占用的物理内存百分比

TIME＋ - 进程启动后占用的总的CPU时间

Command - 进程启动的启动命令名称

## 内存使用情况

free指令

可以看到上面展示的效果并不适合人类来阅读，因为你看到这个数字是没有直观的反馈的。好吧，作为最人性化设计的linux总会给你留一手的，下面我们看看其他参数：

-b, --bytes ：字节为单位显示

-k, --kilo：以kb为单位显示

-m, --mega：以M为单位显示

-g, --giga：以G为单位显示

-h, --human：自动匹配人类可读并且合适的单位显示

## GPU使用情况

nvidia-smi

间隔查看5秒刷新

watch -n 5 nvidia-smi

gpustat

## 进程信息情况

ps -ef

-a：显示系统中所有进程的信息。

-e：显示所有进程的信息。

-f：显示进程的所有信息。

-l：以长格式显示进程信息。

-r：只显示正在运行的进程。

-u：显示面向用户的格式(包括用户名、CPU 及内存的使用情况等信息)。

-x：显示所有非控制终端上的进程信息。

-p：显示由进程 ID 指定的进程信息。

-t：显示指定终端上的进程信息。

# 服务器日志查看和分析

· 系统日志：/var/log/syslog 或 /var/log/messages

· 内核日志：/var/log/kern.log

· 宕机日志：/var/log/dmesg（内核启动时的日志）

· 特定服务日志：通常在 /var/log/ 下对应服务名的文件夹中

使用 journalctl 分析日志

查看最近系统日志：

journalctl -b

journalctl --since "2024-12-01 00:00:00" --until "2024-12-01 23:59:59"

特定服务

journalctl -u nginx.service

分析宕机原因：

查看

journalctl --list-boots

journalctl -b -1

使用 dmesg 查看内核启动信息和错误：

dmesg | grep -i "error"

#### ****定位常见问题****

* **系统宕机**
  + 检查 /var/log/syslog 或 journalctl，定位 panic 或 oom 关键字。
  + 检查硬件问题：dmesg 中是否有磁盘或内存错误。
* **服务异常**
  + 查看服务专用日志：如 /var/log/nginx/error.log。
  + 检查服务的状态

systemctl status 服务名

**资源不足**

* 查看 /var/log/syslog 中是否有 out of memory 或 high load。
* 使用 top 或 htop 分析资源使用情况。

日志轮替和清理

使用 logrotate 定期清理和压缩日志，避免日志文件过大。 配置文件路径：/etc/logrotate.conf。

日志监控和告警

配置 rsyslog 和 logwatch

rsyslog 用于日志集中管理。

logwatch 定期生成日志摘要。

配置告警工具

Nagios/Zabbix：设置日志告警规则。

Prometheus + Alertmanager：实时监控系统指标，并基于日志触发告警。

设置日志级别

确保日志级别设置合理：

日志级别通常包括：DEBUG, INFO, WARNING, ERROR, CRITICAL。

在生产环境，避免记录过多 DEBUG 级别的日志，减少存储压力。

Linux 宕机案例

问题：系统频繁宕机。

排查步骤：

检查 /var/log/syslog 中是否有 oom-killer。

检查 /var/log/kern.log 是否有硬件相关错误。

使用 journalctl -b -1 检查最近一次重启前的日志。

使用 vmstat 查看内存、IO 压力。

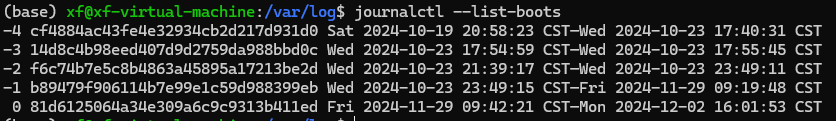
解决方案：

增加内存或优化服务配置。

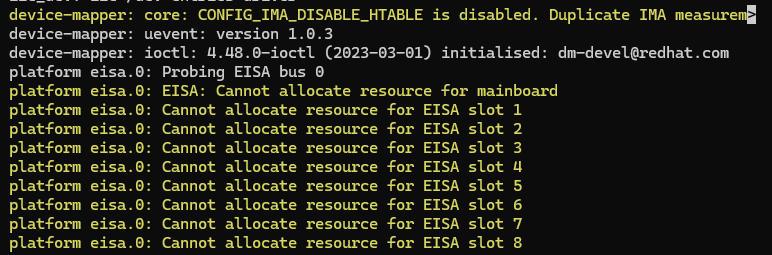
检查硬盘是否损坏。

.1.55问题查找解决方法分析

最近一次系统宕机



问题1：



错误表明系统无法为EISA插槽（一种旧的扩展插槽标准）的第一个位置分配资源。EISA插槽通常用于连接老式的扩展卡，如硬盘控制器或声卡。这个错误可能是由于BIOS设置不正确、硬件冲突、系统资源不足或者硬件故障引起的。

解决方法：

重启计算机并进入BIOS设置，检查EISA插槽相关设置是否正确。

确认是否有其他设备占用了EISA插槽1，如果有，尝试移除或更换这些设备。

如果系统资源充足，尝试更新或更换有问题的硬件。

如果硬件没有问题，尝试更新BIOS/UEFI固件到最新版本。

如果以上步骤无法解决问题，可能需要联系硬件专家进一步诊断或更换有故障的主板。

问题2：

piix4\_smbus 0000:00:07.3: SMBus Host Controller not enabled!

这条错误信息表明系统中的SMBus（System Management Bus）控制器（在这个例子中是PIIX4 chipset中的一个组件）没有被启用。SMBus是一种两线制简化的系统总线，用于系统内部的通信，常用于管理功能，如电池状态、温度监控等。

问题可能出现在BIOS设置中，或者在Linux内核的配置或加载过程中。

解决方法：

进入BIOS设置，查找与SMBus控制器或相关管理功能相关的选项，并确保它们被启用。不同的主板可能会有不同的BIOS设置选项，通常是在“Advanced”或“Integrated Peripherals”等高级设置菜单下。

如果你正在使用Linux系统，并且确定需要SMBus功能，可以尝试更新你的Linux内核，或者确保你的内核配置中包含了对SMBus的支持，并且在启动时没有被禁用。

如果你不需要SMBus控制器提供的功能，你可以选择忽略这条消息，因为它可能只是一个非关键性警告。

如果你是硬件设备的维护者，并且不确定如何操作BIOS设置，可以参考主板的说明书或者联系主板制造商的技术支持。

问题3：

