**快捷键/命令大全**

目录

[**一、 Git** 2](#_Toc59547745)

[**二、 Linux** 5](#_Toc59547746)

[**三、 Mac** 11](#_Toc59547747)

1. **Git**
2. 查看远程仓库：

git remote

git remote -v

1. 将当前分支推送到远程某个分支：

git push origin master

1. 将远程仓库代码下载到本地：

git clone

1. 查看当前分支：

git branch

1. 创建远程分支到本地：

git checkout -b dev origin/dev

1. 将本地分支与远程分支对应起来：

git branch --set-upstream-to=origin/dev dev

git branch --set-upstream-to dev origin/dev

1. 回滚：

git reset --hard HEAD^

git reset --hard HEAD^^

git reset --hard commit id

1. 查看每次的命令：

git reflog

1. 查看提交日志：

git log

git log --pretty=oneline

1. 复制某个提交到当前分支：

git cherry-pick commit id

1. 暂存修改、恢复

git stash 暂存

git stash list 查看暂存列表

git stash apply 恢复暂存但不删除

git stash apply stash@{id} 恢复暂存但不删除

git stash drop 删除暂存

git stash pop 恢复并删除暂存

1. 选中分支：

git checkout branchName

git switch branchName

1. 创建并选中分支：

git checkout -b newBranchName

1. **Linux**

**1、新手基础命令**

**1.1 针对命令的命令**

（1）man 某个命令：查看某个命令的帮助信息：

（2）alias/unalias：为命令创建/删除别名

* alias 别名=’命令 [参数] [对象]’，比如alias rm=’rm -i’即可将rm设置为删除前确认，但是关闭终端之后即失效
* unalias 别名 → 删除别名

（3）type 某个命令：查看某个命令所在目录或别名

* 如果命令不是别名，那么显示其所在的目录，否则则显示其对应的真正命令

（4）whereis：查找命令的绝对路径

**1.2 系统工作命令**

（1）echo：在终端输出字符串或者变量对应的值：

* echo “Hello world” 或者不加引号 → 输出Hello world
* echo $SHELL → 用$提取变量对应的值，输出终端的值

（2）date：显示和设置系统时间和日期

* date → 显示当前时间日期
* date “+%Y-%m-%d %H:%M:%S” → 以yyyy-MM-dd HH:mm:ss格式显示日期
* date “+%j” → 显示当天是当年的第几天
* date –s “20210101 1:30:00” → 将当前时间设置为指定日期

（3）reboot：重启系统

（4）poweroff：关闭系统

（5）wget：在终端下载网络文件，wget [参数] 下载地址

（6）ps：查看系统中的进程状态

* ps –a → 显示所有进程
* ps –u → 显示进程用户及其他详细信息
* ps –x → 显示没有控制终端的进程
* ps –aux/ps aux → a、u、x短格式命令参数之间的合并，且ps命令允许命令不加-号

（7）top：动态监视进程活动和系统负载等信息，按q退出

（8）pidof [参数] [服务名称]：查询某个指定服务进程的PID值

（9）kill [参数] [进程PID]：终止某个指定PID的服务进程

（10）killall [参数] [服务名称]：终止指定服务的全部进程

（11）Ctrl + C：终止终端中正在执行的命令

（12）&：当命令执行时不断输出信息影响后续命令输入，可在执行命令时在末尾添加&，可使命令进入系统后台执行

（13）sudo passwd root → 更改root用户的密码，但是需要输入当前用户的密码

（14）su 用户名 → 切换到另一个用户

**1.3 系统状态检测命令**

（1）ifconfig ([网络设备] [参数])：获取网卡配置以及网络状态等信息

（2）uname：查看系统内核和系统版本信息，搭配-a参数可完整查看当前系统的内核名称、主机名、内核发行版本、节点名、系统时间、硬件名称、硬件平台、处理器类型以及操作系统名称等信息。

要查看当前系统版本的详细信息，需查看redhat-release文件：cat /etc/redhat-release

（3）uptime：查看系统负载信息，包括当前系统时间、系统已运行时间、启用终端数、平均负载值（系统在最近1 min、5 min、15 min内的负载情况，越低越好）

（4）free：显示当前系统内存使用量，加-m单位为MB，加-h参数会以GB单位显示

（5）who：查看当前登录本机的用户及其开启的终端信息

（6）last：查看所有系统的登录记录

（7）history：显示历史执行命令，能显示最近执行的1000条命令，列出的命令前有“编码数字”，可以通过“！编码数字”来执行数字对应的命令。显示条数可以通过修改/etc/profile文件中的HISTSIZE值来自定义。历史命令会保存到~/.bash\_history文件中，要清除历史命令记录可加-c参数：history –c。

（8）sosreport：用于收集系统运行状态和配置信息并输出诊断文档

**1.4 工作目录切换命令**

（1）pwd：显示当前所处的工作目录

（2）cd：切换工作路径

* cd /etc → 切换到/etc目录
* cd - → 切换到上一次所在目录
* cd .. → 切换到上级目录
* cd ~ → 切换到home目录
* cd ~usename → 切换到其他用户的home目录

（3）ls [选项] [文件]：查看目录中的文件信息

* ls -a → 查看全部文件，包括隐藏文件
* ls -l → 查看文件的属性、大小等详细信息，同样加-h可以用MB/GB单位显示大小
* ls -al → 组合了-a和-l选项，查看全部文件的属性和大小等信息
* ls -ld 目录 → 要想单独查看目录的属性信息而不是目录内的全部文件的属性信息，需要额外加个-d

**1.5 文本文件编辑命令**

（1）cat：查看内容较少的纯文本文件，加-n参数可以显示行号

（2）more：查看内容较多的纯文本文件，空格或回车键来向下翻页，按q退出

（3）head -n [N] [文件]：查看纯文本文档的前N行

（4）tail -n [N] [文件]：查看纯文本文件的后N行

tail -f [文件]：持续刷新一个文件的内容，可用于实时查看最新日志文件。每次刷新都会再次获得文件的全部内容

（5）tr [原始字符] [目标字符]：用于替换文本文件中的字符，可使用正则表达式

可以使用cat命令读取文本内容，再使用管道符|把读取到的文本内容传递给tr命令进行替换操作：cat 文件 | tr [a-z] [A-Z] → 把文件中的小写字母全转为大写，但其实不会修改源文件，只是终端输出时全为大写

（6）wc [参数] 文本：统计指定文本的行数、字数、字节数

* wc -l 文本 → 只显示行数
* wc -w 文本 → 只显示单词数
* wc -c 文本 → 只显示字节数
* （短格式的参数可以组合使用哦）

（7）stat：查看文件的存储信息和时间状态

* Access：文件访问时间atime
* Modify：文件内容修改时间mtime
* Change：文件权限或属性更改时间ctime

（8）cut [参数] 文本：按“列”提取文本字符，-d参数指定分隔符，-f参数指定看哪一列

* cut -d: -f1 /etc/passwd → 以”:”为分隔符，查看第一列
* cut -d : -f 2 /etc/passwd → -d、-f和其后面的参数之间可以有空格

（9）diff [参数] 文本1 文本2：比较两个文本文件的差异

* diff --brief 文本1 文本2 → 确认两个文件是否不同，如果内容相同则什么也不输出
* diff -c 文本1 文本2 → 详细比较两个文件的差异

**1.6 文件目录管理命令**

（1）touch [参数] 文件：创建空白文件，或者设置文件的访问、内容修改、属性更改时间

* touch 文件名 → 直接创建空白文件
* -a：仅修改“读取时间”atime
* -m：仅修改“修改时间”mtime
* -d：同时修改atime和mtime

（2）mkdir：创建空白目录，使用-p参数可以创建具有嵌套结构的目录

* mkdir dir1 → 创建空白目录dir1，如果dir1已存在则报错
* mkdir -p dir1/dir2/dir3 → 在当前目录下创建嵌套目录dir1/dir2/dir3
* mkdir -p /home/linuxprobe/dir1/dir2 → 创建嵌套目录/home/linuxprobe/dir1/dir2

（3）cp [选项] 源文件 目标文件：复制文件或目录

当源文件不是目录时：

* 若目标文件是目录，则会把源文件复制到目录中
* 若目标文件为普通文件，则会用源文件的内容覆盖目标文件
* 若目标不存在，则创建一个目标文件后复制
* -p：不仅复制内容，还复制原始文件的属性（修改时间、访问权限等）
* -d：若对象为“链接文件”，则保留“链接文件”的属性
* -r：**当源文件为目录时**必须使用，递归持续复制，目标文件也必须为目录
* -i：若目标文件存在，则询问是否覆盖
* -f：直接覆盖已存在的目标文件而不询问
* -a：相当于-pdr，通常在拷贝目录时使用，可保留链接和文件属性

（4）mv [参数] 源文件 [目标路径|目标文件名]：剪切或重命名文件，在同一目录中对文件进行剪切就可以重命名

（5）rm：删除文件或目录，系统不会询问，加-f可强制删除，-r删除目录，-i可以在删除前确认

（6）dd [参数]：按照指定大小和个数的数据块来复制或转换文件。Linux系统中/dev目录下有个zero的文件/dev/zero，不占用系统存储空间，却可以提供无穷无尽的数据，可以使用它作为dd命令的输入文件，来生成一个指定大小的文件

* if：输入的文件名称
* of：输出的文件名称
* bs：设置每个块的大小
* count：设置要复制的块的个数
* dd if=/dev/zero of=file\_1 count=1 bs=560M

（7）file 文件名：查看文件类型

**1.7 打包压缩与搜索命令**

（1）tar [参数] [文件]：压缩或解压，压缩文件格式主要有.tar/.tar.gz/.tar.bz2

* -c：压缩文件，需指定压缩包名称和后缀（.tar/.tar.gz/.tar.bz2等）
* -x：解压文件，与-c不能同时使用
* -t：查看压缩包内有哪些文件
* -z：用Gzip压缩或解压
* -j：用bzip2压缩或解压
* -v：显示压缩或解压的过程
* -f：压缩或解压的目标文件名，必须放在参数的最后一位
* -p：保留原始的权限和属性
* -P：使用绝对路径来压缩
* -C：指定解压到的目录
* tar -czvf 压缩包名称.tar.gz 要打包的文件/目录 → 打包指定的文件并命名
* tar -xzvf 压缩包名称.tar.gz -C 解压的路径 → 将指定压缩包解压到指定路径

（2）grep [参数] 关键词 文件：在多个文本中执行关键词搜索，并显示具有关键词的行

* -b：将可执行文件（binary）当做文本文件（text）搜索
* -c：仅显示找到的行数
* -i：忽略大小写
* -n：显示行号
* -v：反选——列出没有关键词的行
* grep -n Hello file1 → 查找file1中包含Hello的行
* grep -n “Hello world” file1 → 查找file1中包含”Hello world”的行
* grep -vn “Hello world” file1 file2 → 查找file1和file2中不包含”Hello world”的行

（3）find 查找路径 [参数] [下一步操作]：按照指定条件**查找文件**

* -name：匹配名称
* -perm：匹配权限，mode为完全匹配，-mode为包含即可
* -user：匹配所有者
* -group：匹配所有组
* -mtime -n +n：匹配修改内容的时间，-n指n天以内，+n指n天以前
* -atime -n +n：匹配文件访问的时间
* -ctime -n +n：匹配修改文件权限和属性的时间
* -nouser：匹配无所有者的文件
* -nogroup：匹配无所有组的文件
* -newer f1 !f2：匹配比文件f1新但比文件f2旧的文件
* --type b/d/c/p/l/f：匹配文件类型，块设备/目录/字符设备/管道/链接文件/文本文件
* -size +50KB/-50KB：匹配文件的大小，查找超过50KB/小于50KB的文件
* -prune：忽略某个目录
* -exec 下一步操作命令 {} \; → -exec用于将find命令搜索到的结果交个后面紧跟的命令做进一步处理，{}表示find命令查找出的文件，且必须以\;结尾
* find /etc -name “host\*” → 查找/etc目录中所有名字以host开头的文件
* find / -perm -4000 -print → 在整个系统中搜索权限中包含SUID权限的所有文件，使用-4000即可（啥玩意？？？）
* find / -user linuxprobe -exec cp -a {} /home/linuxprobe/findresults/ \; → 在整个文件系统中查找用户为linuxprobe的文件并复制到/home/linuxprobe/findresults/目录里。-exec参数后面紧跟cp命令，{}表示find查找的结果，并以\;结尾

**2、管道符、重定向与环境变量**

**2.1 输入输出重定向**

* 标准输入（STDIN，文件描述符为0）：默认从键盘输入，也可从其他命令或文件中输入。
* 标准输出（STDOUT，文件描述符为1）：默认输出到屏幕
* 错误输出（STDERR，文件描述符为2）：默认输出到屏幕

输入重定向是指把文件导入到命令中，输出重定向是指把原本要输出到屏幕的数据写入到指定文件中。

* 输出重定向：
  + 标准输出重定向：
    - 清空写入（覆盖原文件的内容）
    - 追加写入（添加到原文件的后面）
  + 错误输出重定向：
    - 清空写入
    - 追加写入

**2.1.1 输入重定向**

（1）命令 < 文件：将文件作为命令的标准输入

* wc -l < readme.txt 等同于 wc -l readme.txt 等同于 cat readme.txt | wc -l

（2）命令 << 分界符：从标准输入中读入，直到遇见分界符才停止

（3）命令 < 文件1 > 文件2：将文件1作为命令的标准输入

**2.1.2 输出重定向**

（1）命令 > 文件：将标准输出重定向到文件中（覆盖原文件的内容）

* man bash > readme.txt
* echo “Welcome to LinuxProbe.Com” > readme.txt

（2）命令 2> 文件：将错误输出重定向到文件中（覆盖原文件的内容）

* ls -l linuxprobe > file1.txt → 假设linuxprobe存在，则输出到file1.txt中
* ls -l linuxprobe 2> file1.txt → 假设linuxprobe存在，则依然输出到屏幕上

（3）命令 >> 文件：将标准输出重定向到文件中（追加到原文件的内容后面）

* echo “Welcome to LinuxProbe.Com” >> readme.txt

（4）命令 2>> 文件：将错误输出重定向到文件中（追加到原文件的内容后面）

（5）命令 >> 文件 2>%1 或者 命令 &>> 文件：将标准输出与错误输出共同写入到文件中（追加到原有内容的后面）

* 对于重定向中的标准输出模式，可以省略文件描述符1，而错误输出模式的文件描述符2必须要写。
* /dev/null：被称为Linux黑洞的文件，可以把输出信息重定向到这里等同于删除输出信息，可以使窗口保持简洁

**2.2 管道命令符**

* grep -c “/sbin/nologin” /etc/passwd 等同于 grep “/sbin/nologin” /etc/passwd | wc -l → 在用户信息/etc/passwd中查找终端为nologin的用户，该用户不允许登录，仅显示用户数
* man ls | more → 将ls命令的帮助信息传给more命令，用翻页形式查看
* echo “Content Hello world” | mail -s “Subject Linux is funny” linuxprobe

等同于：

mail -s “Subject Linux is funny” linuxprobe@linuxprobe.com << over（标准输入重定向的分界符）

> Content Hello world

> over（分界符）

将主题为”Subject Linux is funny”内容为”Content Hello world”的邮件发送给linuxprobe.com主机上的linuxprobe用户

**2.3 命令行的通配符**

\*匹配零个或多个字符，？匹配单个字符，[0-9]匹配0~9，[123]匹配1或2或3

**2.4 常用的转义字符**

（1）反斜杠（\）：使反斜杠后面的一个变量变为单纯的字符串

（2）单引号（’ ’）：转义其中的所有变量为单纯的字符串

（3）双引号（” “）：保留其中的变量属性，不进行转义

（4）反引号（` `）：执行其中的命令后返回结果，等同于$(命令)或$”命令”

[~] # PRICE=5 定义一个变量并赋值为5

[~] # echo “Price is $PRICE” → 输出Price is 5

[~] # echo ‘Price is $PRICE’ → 输出Price is $PRICE

[~] # echo “Price is $$PRICE” → 输出Price is 2639PRICE，$$表示当前程序的进程ID

[~] # echo “Price is \$$PRICE” → 输出Price is $5

[~] # echo `uname -a` 等同于 uname -a，查看本机的用户、系统版本和内核信息

**2.5 重要的环境变量**

（1）HOME：用户的主目录

（2）SHELL：用户使用的Shell解释器名称

（3）HISTSIZE：输出的历史命令记录条数

（4）HISTFILESIZE：保存的历史命令记录条数

（5）MAIL：邮件保存路径

（6）LANG：系统语言、语系名称

（7）RANDOM：生成一个随机数字

（8）PS1：Bash解释器的提示符

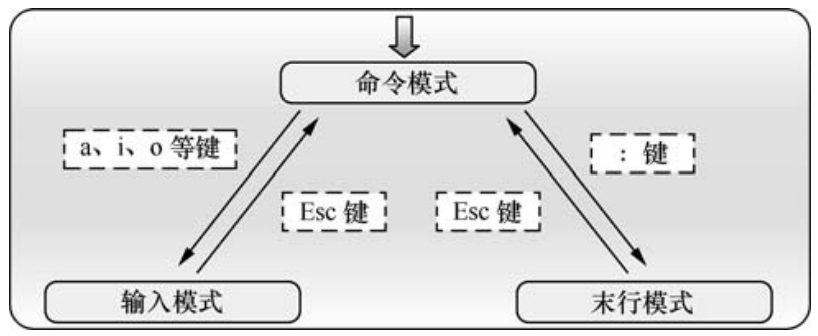
（9）PATH：定义解释器搜索用户执行命令的路径

（10）EDITOR：用户默认的文本编辑器

* export：将某个用户定义的局部变量提升为全局变量，其他用户也可使用。但用户自定义的变量一旦终端关闭即失效

**3、Vim编辑器与Shell命令脚本**

**3.1 Vim编辑器的三种模式**



（1）命令模式：控制光标移动，可对文本进行复制、粘贴、删除和查找等工作

* dd：删除（剪切）光标所在整行
* 5dd：删除（剪切）从光标处开始的5行
* yy：复制光标所在整行
* 5yy：复制从光标处开始的5行
* n：显示搜索命令定位到的下一个字符串
* N：显示搜索命令定位到的上一个字符串
* u：撤销上一个操作
* p：将之前删除（dd）或复制（yy）的数据粘贴到光标后面

（2）输入模式：正常的文本录入，从命令模式到输入模式可以按a、i、o

* a：在光标后面一位切换到输入模式
* i：在光标当前位置切换到输入模式
* o：在光标下面创建一个空行

（3）末行模式：保存或退出文档，以及设置编辑环境

* :w → 保存
* :q → 退出
* :wq → 保存并退出
* :q! → 强制退出（放弃对文档的修改）
* :wq! → 强制保存退出
* :set nu → 显示行号
* :set nonu → 不显示行号
* :命令 → 执行该命令
* :整数 → 跳转到改行
* :s/one/two → 将当前光标所在行的第一个one替换成two
* :s/one/two/g → 将当前光标所在行的所有one替换成two
* :%s/one/two/g → 将全文中的所有one替换成two
* ?字符串 → 在文本中从下至上搜索该字符串
* /字符串 → 在全文中从上至下搜索该字符串

**3.2 Shell脚本**

**3.2.1 脚本声明和注释**

（1）#!/bin/bash：告诉系统使用哪种Shell解释器来执行该脚本

（2）#：注释信息

**3.2.2 脚本执行**

（1）bash example.sh：用bash解释器直接运行Shell脚本

（2）./example.sh：输入完整路径来执行，但是会因为权限不足而不能运行

* 解决方法：chmod u+x example.sh → 为脚本增加执行权限后再输入完整路径执行

**3.2.3 接受用户参数**

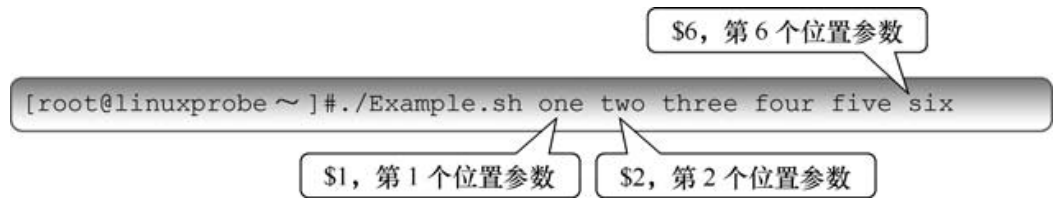
（1）$0：当前Shell脚本程序的名称

（2）$#：总共有几个参数

（3）$\*：所有位置的参数值

（4）$?：上一次命令的执行返回值

（5）$1、$2、$3、...、$N：分别对应第N个位置的参数值



（6）read：在Shell脚本中用来读取用户输入信息，把接收到的用户输入赋值给后面的变量，-p参数用于向用户显示提示信息

* read -p “Enter your score (0-100): “ GRADE → 在Shell脚本中

**3.2.4 条件测试语句**

Shell脚本中的条件测试语句格式为：[ 条件表达式 ]，左右方括号与表达式之间必须有空格，条件成立返回0，否则返回其他随机数字。

按照测试对象划分：

（1）文件测试语句

* -d：测试文件是否为目录
* -e：测试文件是否存在
* -f：测试是否为一般文件
* -r：测试当前用户是否有权限读取
* -w：测试当前用户是否有权限写入
* -x：测试当前用户是否有权限执行
* [ -d /etc/fstab ] → 然后用echo $?显示上一条命令执行后的返回值

（2）逻辑测试语句

* 对测试结果进行逻辑分析
* &&：前面的命令（不管是条件测试语句还是Shell命令）执行成功后才会执行后面的命令

[ -e /dev/cdrom ] && echo “Exist” → 判断文件是否存在，若存在则输出Exist

* ||：前面的命令执行失败后才会执行后面的命令

[ $USER=root ] || echo “not root”

* !：条件测试结果取反

[ ! $USER = root ] && echo “user” || echo “root”

（3）整数比较语句

* -eq：等于
* -ne：不等于
* -gt：大于
* -lt：小于
* -le：小于等于
* -ge：大于等于
* [ 10 -eq 10 ] → echo $?输出0

（4）字符串比较语句

* =：比较字符串内容是否相同
* !=：比较字符串内容是否不相同
* -z：判断字符串内容是否为空，可以判断变量是否为空，进而判断是否定义了该变量

**3.2.5 流程控制语句**

* **if条件语句**

（1）if 条件测试语句

then 命令序列

fi

（2）if 条件测试语句

then 命令序列1

else

命令序列2

fi

（3）if 条件测试语句1 ; then

命令序列1

elif 条件测试语句2 ; then

命令序列2

else

命令序列3

fi

* then需要放到if/elif的下一行，放在同一行时需要在条件测试语句与then之间加分号，否则会报错
* for循环语句

for 变量名 in 取值列表

do

命令序列

done

* while循环语句

while 条件测试语句

do

命令序列

done

* case条件语句

case 变量值 in

模式1）

命令序列1

;;

模式2）

命令序列2

;;

……

\*）

默认命令序列

esac

**3.2.6 计划任务服务程序**

（1）at：一次性计划任务

* at 时间：开始输入一次性定时任务，按Ctrl+D结束编写，交互式的

at 20:45

at> systemctrl restart httpd

at> <EOT> （按Ctrl+D之后显示这个）

等同于

echo “systemctrl restart httpd” | at 20:45，利用管道符来让at接受echo的输出信息，取代交互式

* at -l：查看已设置但未执行的一次性任务
* atrm 任务序号：删除一次性任务

（2）crontab：crond服务，周期性计划任务

* crontab -e：创建、编辑计划任务，任务格式为：分 时 日 月 星期 命令，没有的字段设为\*，“分”字段必须有数值，“日”和“星期”不能同时使用。可以用逗号,分隔多个时间段（如8,9,10表示8,9,10月），用横杠-表示连续的时间（如12-15表示12日-15日），用斜杠/表示执行任务的间隔时间（如/2表示间隔2分钟执行一次任务）。命令必须用绝对路径（可用whereis查询）
* crontab -l：查看当前计划任务，每一行表示一个计划任务
* crontab -r：删除全部计划任务

1. **Mac**

1、

Command：

Option：

H：保留最上层应用隐藏其他

w：关闭当前应用所有窗口

Shift：

Z：重做

G：前往

. ：隐藏/显示隐藏文件

N：打开新窗口

T：打开新标签页

I：显示文件简介

↑：进入上层文件夹

H：隐藏最上层应用

M：最小化

Q：关闭应用

, ：打开应用设置

Delete：删除一行

点击标题：显示路径

Tab：选择打开的应用

Option：拖动复制

Fn：

Delete：删除后一个字符

↑：Page Up

↓：Page Down