Linux 系统概论

天津医科大学
生物医学工程与技术学院

2017-2018 学年下学期（春）
2016 级生信班

第四章　常用 Linux 命令

伊现富（Yi Xianfu）

天津医科大学（TIJMU）
生物医学工程与技术学院

2018 年 5 月

教学提纲

教学提纲

命令

- 命令是可执行的程序，使得用户能够要求机器执行某种操作
- 有时候是 shell 的内置功能，如：列出目录的内容
- 有时候是单独的程序，如：设置机器运行时的基本参数的一长串脚本

远程登录

- ssh USERNAME@HOSTNAME

本地桌面

- 开机直接进入：不安装图形界面（只有命令行界面可用）
- 虚拟终端（从图形界面完全切换到命令行界面）：Ctrl + Alt + F[1-6]/F[2-7]
- 终端模拟器（在图形界面中使用命令行）：打开 Terminal/终端
- 从命令行界面切换回图形界面：Ctrl + Alt + F7/F1

远程登录

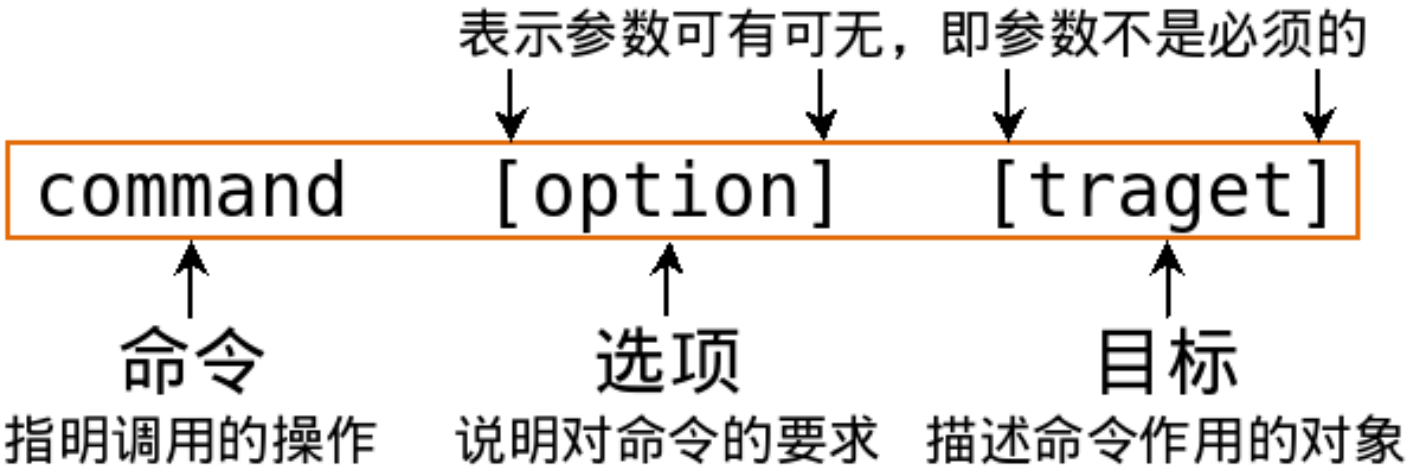
- ssh USERNAME@HOSTNAME

本地桌面

- 开机直接进入：不安装图形界面（只有命令行界面可用）
- 虚拟终端（从图形界面完全切换到命令行界面）：Ctrl + Alt + F[1-6]/F[2-7]
- 终端模拟器（在图形界面中使用命令行）：打开 Terminal/终端
- 从命令行界面切换回图形界面：Ctrl + Alt + F7/F1

教学提纲

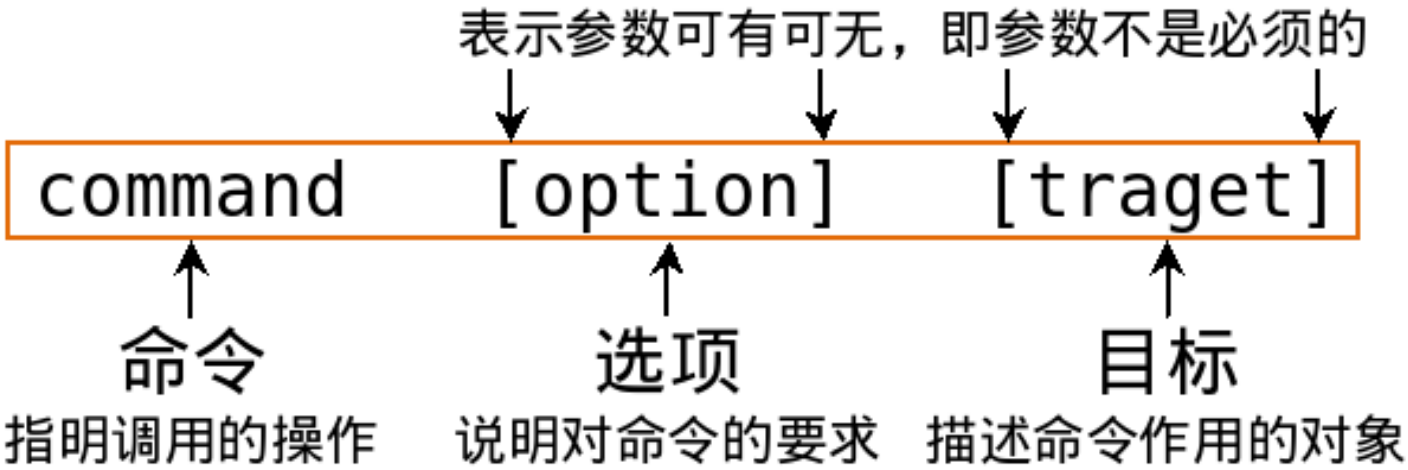
命令剖析



Linux 命令

- Linux 命令 = 命令 + [参数]（command + [argument]）
- 命令参数 = [选项] + [目标]（[option] + [target]）
- 分隔符：空格（Space）

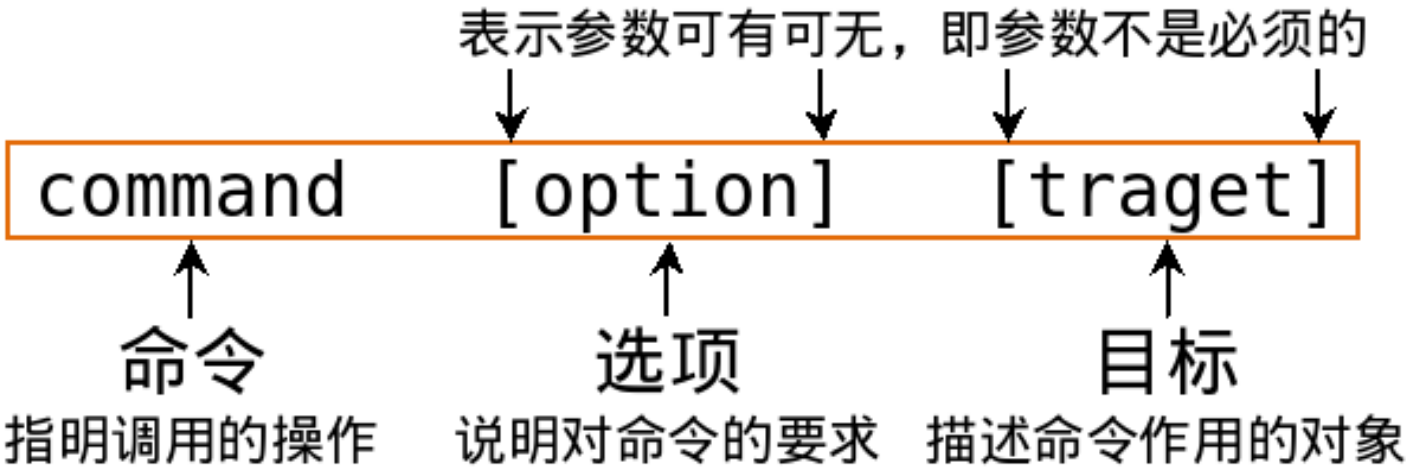
命令剖析



Linux 命令

- Linux 命令 = 命令 + [参数] （command + [argument]）
- 命令参数 = [选项] + [目标] （[option] + [target]）
- 分隔符：空格（Space）

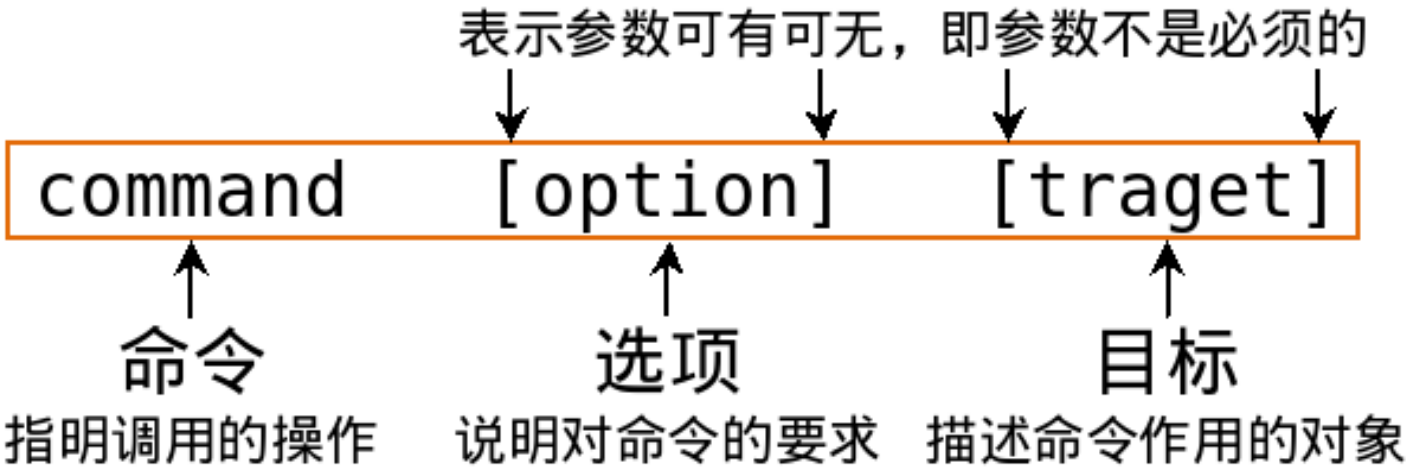
命令剖析



Linux 命令

- Linux 命令 = 命令 + [参数] （command + [argument]）
- 命令参数 = [选项] + [目标] （[option] + [target]）
- 分隔符：空格（Space）

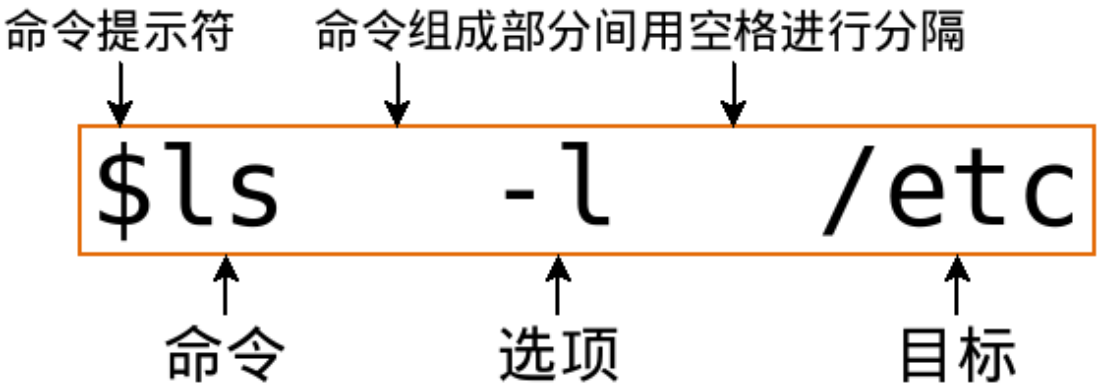
命令剖析



Linux 命令

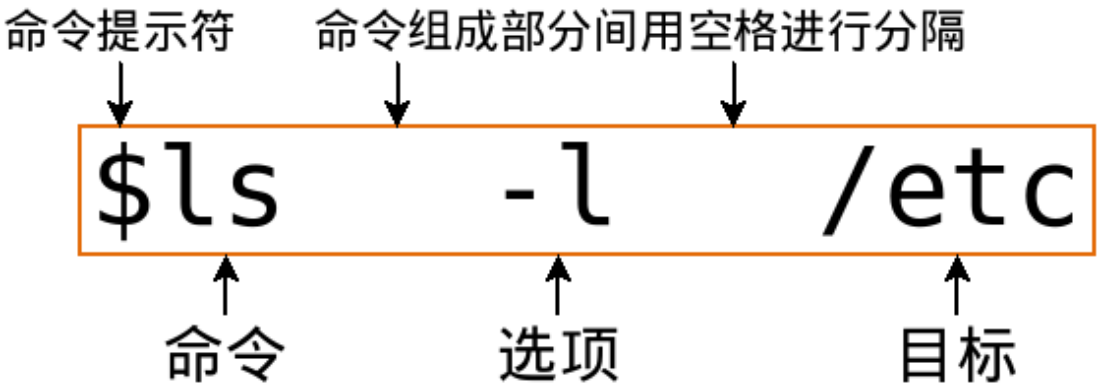
- Linux 命令 = 命令 + [参数] （command + [argument]）
- 命令参数 = [选项] + [目标] （[option] + [target]）
- 分隔符：空格（Space）

命令剖析 | 实例



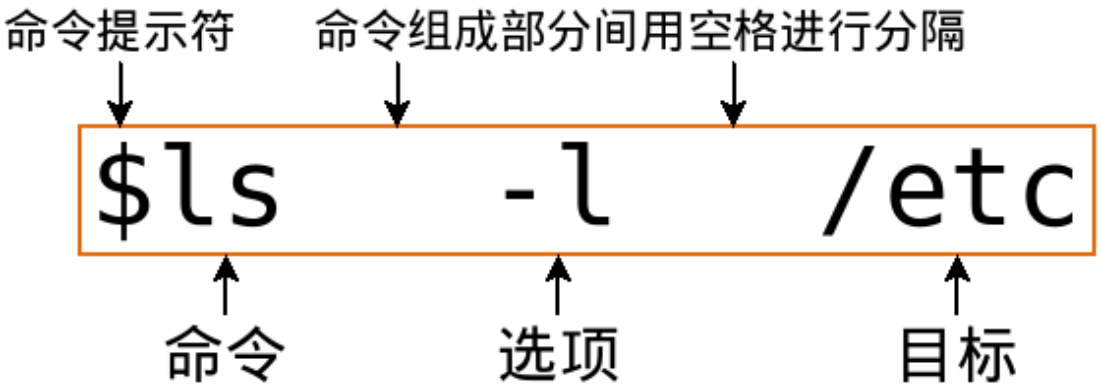
- 大多数命令名与该命令所调用的操作在字面上是一样的（ls: list）
- 单独执行命令时所发生的一切称为命令的默认行为（ls）
- 参数能够影响命令的输出格式以及它的操作（-l /etc）
- 目标为命令提供了所要处理的目标位置（/etc）
- 有些命令要求多个目标（cp source destination）
- 特定的命令还具有自己独有的选项外字母是（-l）
- 选项通常（但不一定）位于一个或两个连字符后（--help）
- 多个选项可以被立即跟在单个连字符后（ls -la）

命令剖析 | 实例



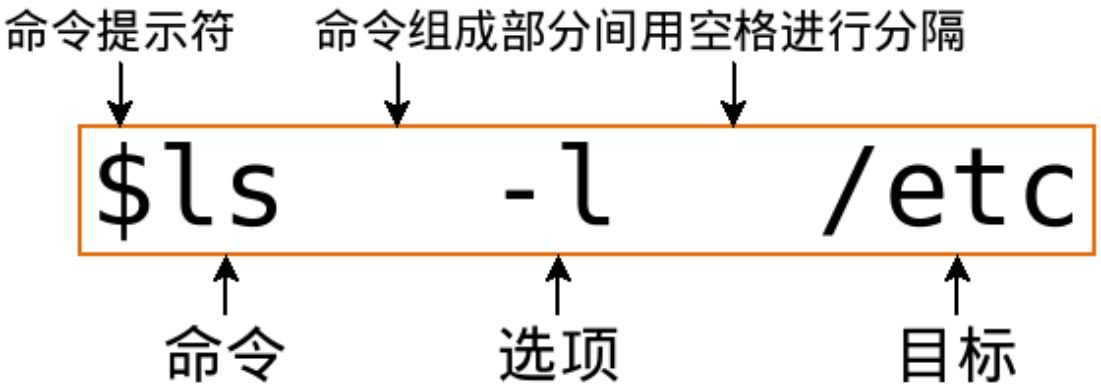
- 大多数命令名与该命令所调用的操作在字面上是一样的（ls: list）
- 单独执行命令时所发生的一切称为命令的默认行为（ls）
- 参数能够影响命令的输出格式以及它的操作（-l /etc）
- 目标为命令提供了所要处理的目标位置（/etc）
- 有些命令要求多个目标（cp source destination）
- 特定的命令通常具有自己独特的选项/开关/标记（-l）
- 选项通常（但不一定）位于一个或两个连字符之后（-a = --all）
- 多个选项可以独立或是合并添加（-a -l = -al）

命令剖析 | 实例



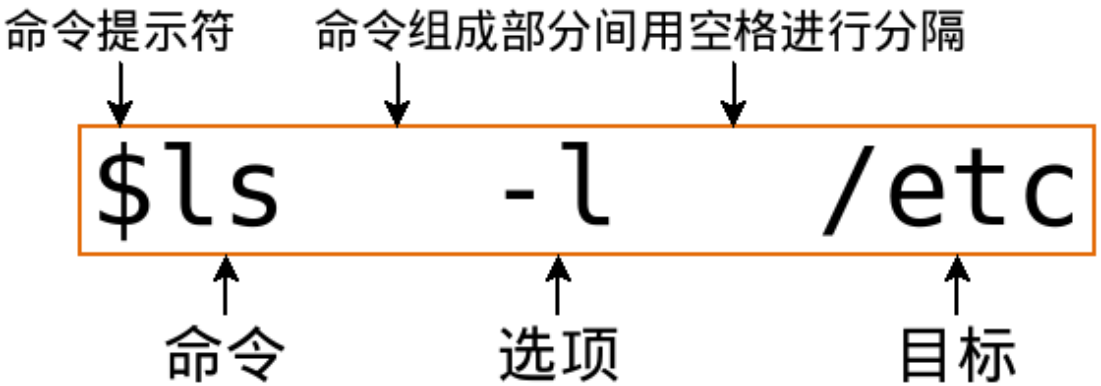
- 大多数命令名与该命令所调用的操作在字面上是一样的（ls: list）
- 单独执行命令时所发生的一切称为命令的默认行为（ls）
- 参数能够影响命令的输出格式以及它的操作（-l /etc）
- 目标为命令提供了所要处理的目标位置（/etc）
- 有些命令要求多个目标（cp source destination）
- 特定的命令通常具有自己独特的选项/开关/标记（-l）
- 选项通常（但不一定）位于一个或两个连字符之后（-a = --all）
- 多个选项可以独立或是合并添加（-a -l = -al）

命令剖析 | 实例



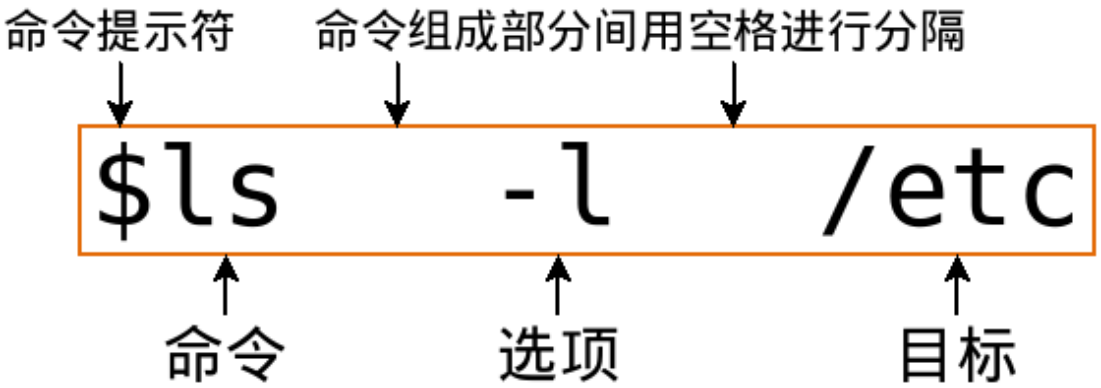
- 大多数命令名与该命令所调用的操作在字面上是一样的（ls: list）
- 单独执行命令时所发生的一切称为命令的默认行为（ls）
- 参数能够影响命令的输出格式以及它的操作（-l /etc）
- 目标为命令提供了所要处理的目标位置（/etc）
- 有些命令要求多个目标（cp source destination）
- 特定的命令通常具有自己独特的选项/开关/标记（-l）
- 选项通常（但不一定）位于一个或两个连字符之后（-a = --all）
- 多个选项可以独立或是合并添加（-a -l = -al）

命令剖析 | 实例



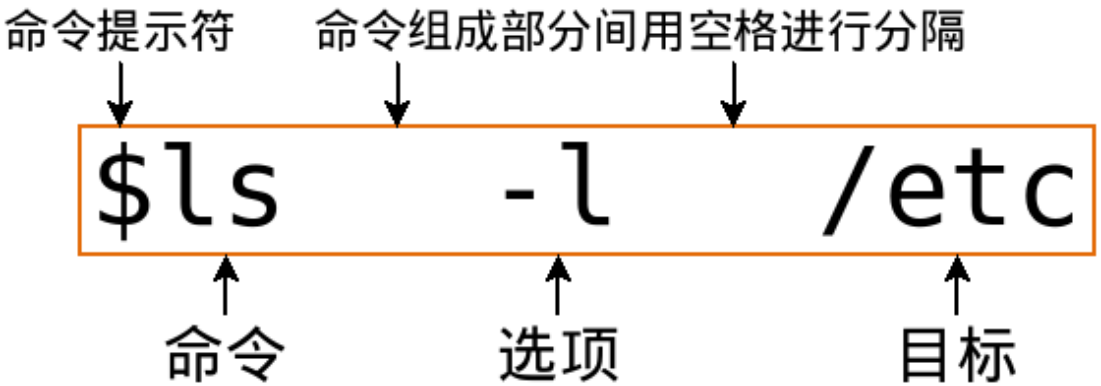
- 大多数命令名与该命令所调用的操作在字面上是一样的（ls: list）
- 单独执行命令时所发生的一切称为命令的默认行为（ls）
- 参数能够影响命令的输出格式以及它的操作（-l /etc)
- 目标为命令提供了所要处理的目标位置（/etc）
- 有些命令要求多个目标（cp source destination）
- 特定的命令通常具有自己独特的选项/开关/标记（-l）
- 选项通常（但不一定）位于一个或两个连字符之后（-a = --all）
- 多个选项可以独立或是合并添加（-a -l = -al）

命令剖析 | 实例

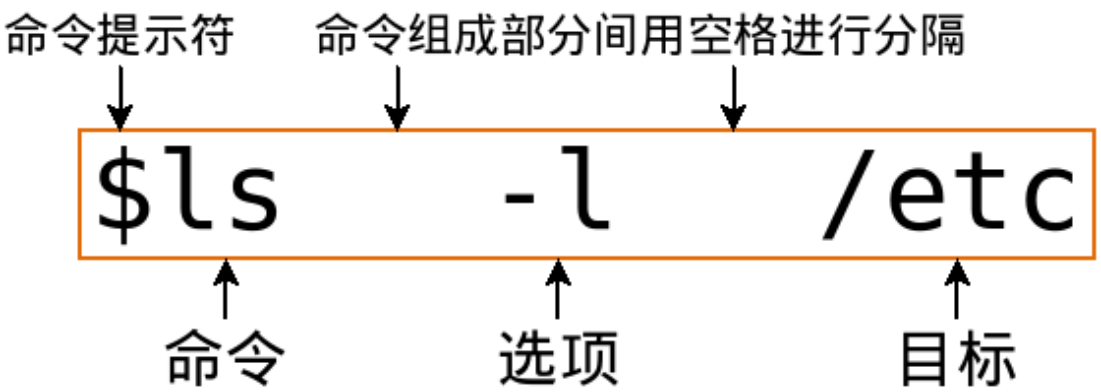


- 大多数命令名与该命令所调用的操作在字面上是一样的（ls: list）
- 单独执行命令时所发生的一切称为命令的默认行为（ls）
- 参数能够影响命令的输出格式以及它的操作（-l /etc)
- 目标为命令提供了所要处理的目标位置（/etc）
- 有些命令要求多个目标（cp source destination）
- 特定的命令通常具有自己独特的选项/开关/标记（-l）
- 选项通常（但不一定）位于一个或两个连字符之后（-a = --all）
- 多个选项可以独立或是合并添加（-a -l = -al）

命令剖析 | 实例

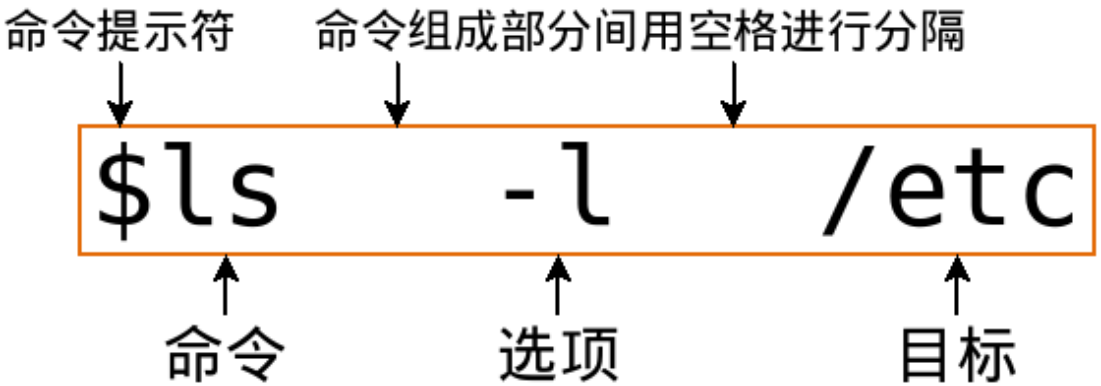


- 大多数命令名与该命令所调用的操作在字面上是一样的（ls: list）
- 单独执行命令时所发生的一切称为命令的默认行为（ls）
- 参数能够影响命令的输出格式以及它的操作（-l /etc)
- 目标为命令提供了所要处理的目标位置（/etc）
- 有些命令要求多个目标（cp source destination）
- 特定的命令通常具有自己独特的选项/开关/标记（-l）
- 选项通常（但不一定）位于一个或两个连字符之后（-a = --all）
- 多个选项可以独立或是合并添加（-a -l = -al）



- 大多数命令名与该命令所调用的操作在字面上是一样的（ls: list）
- 单独执行命令时所发生的一切称为命令的默认行为（ls）
- 参数能够影响命令的输出格式以及它的操作（-l /etc)
- 目标为命令提供了所要处理的目标位置（/etc）
- 有些命令要求多个目标（cp source destination）
- 特定的命令通常具有自己独特的选项/开关/标记（-l）
- 选项通常（但不一定）位于一个或两个连字符之后（-a = --all）
- 多个选项可以独立或是合并添加（-a -l = -al）

命令剖析 | 实例

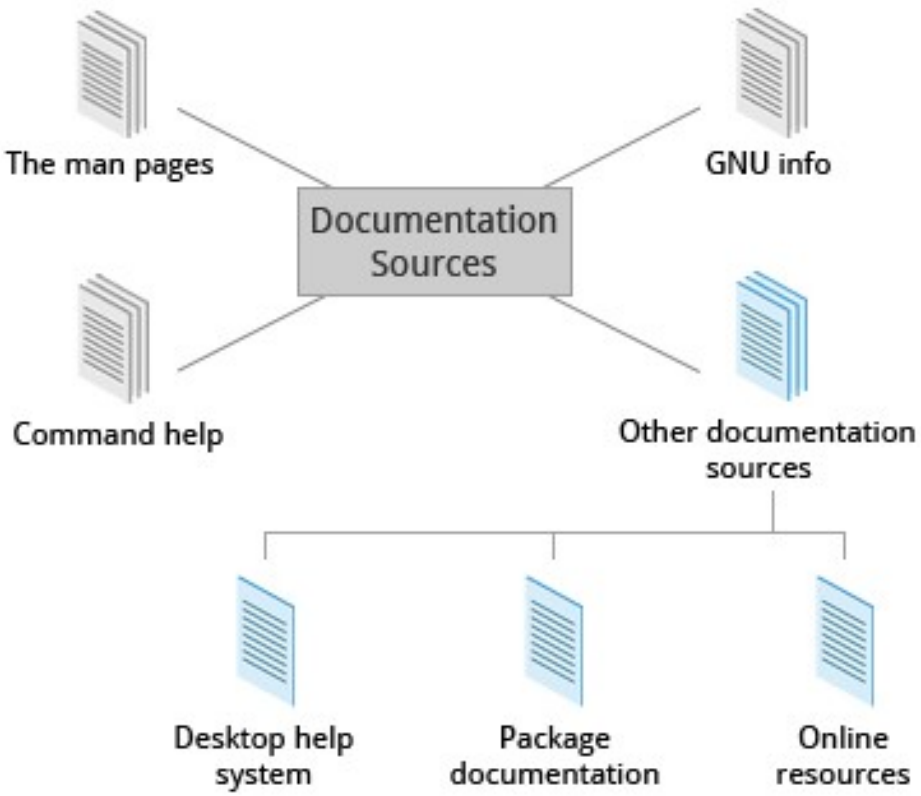


- 大多数命令名与该命令所调用的操作在字面上是一样的（ls: list）
- 单独执行命令时所发生的一切称为命令的默认行为（ls）
- 参数能够影响命令的输出格式以及它的操作（-l /etc)
- 目标为命令提供了所要处理的目标位置（/etc）
- 有些命令要求多个目标（cp source destination）
- 特定的命令通常具有自己独特的选项/开关/标记（-l）
- 选项通常（但不一定）位于一个或两个连字符之后（-a = --all）
- 多个选项可以独立或是合并添加（-a -l = -al）

教学提纲

命令信息 | 文档来源

1. The man pages (short for manual pages)
2. GNU Info
3. The help command and --help option
4. Other Documentation Sources, e.g. https://www.gentoo.org/doc/en/



教学提纲

man

man COMMAND/CONFIG：联机帮助页是了解任何特定命令或配置文件的最佳方法（`man passwd` vs. `man 5 passwd`）

```
apropos(1)                                                    apropos(1)

NAME
       apropos - search the whatis database for strings

SYNOPSIS
       apropos keyword ...

DESCRIPTION
       apropos  searches a set of database files containing short descriptions of
       system commands for keywords and displays the result on the standard  out-
       put.

AUTHOR
       John  W. Eaton was the original author of man.  Zeyd M. Ben-Halim released
       man 1.2, and Andries Brouwer followed up  with  versions  1.3  thru  1.5p.
       Federico Lucifredi <flucifredi@acm.org> is the current maintainer.

SEE ALSO
       whatis(1), man(1).

                                 September 19, 2005                 apropos(1)
~
(END)
```

命令信息 | man | 章节（数字）

章节/数字	内容/含义	补充说明
1	可执行程序或 shell 命令	可由任何人启动
2	系统调用	内核提供的函数
3	库调用	程序库中的函数
4	特殊文件	通常位于/dev
5	文件格式和规范	如/etc/passwd
6	游戏	—
7	杂项 (包括宏包和规范)	如 man(7), groff(7)
8	系统管理命令	通常只针对 root 用户
9	内核例程	非标准，Linux 特有

- 限定章节：`man 3 printf`
- 所有章节：`man -a printf`

命令信息 | man | 章节（数字）

章节/数字	内容/含义	补充说明
1	可执行程序或 shell 命令	可由任何人启动
2	系统调用	内核提供的函数
3	库调用	程序库中的函数
4	特殊文件	通常位于/dev
5	文件格式和规范	如/etc/passwd
6	游戏	—
7	杂项 (包括宏包和规范)	如 man(7), groff(7)
8	系统管理命令	通常只针对 root 用户
9	内核例程	非标准，Linux 特有

- 限定章节：`man 3 printf`
- 所有章节：`man -a printf`

教学提纲

info

info KEYWORD：访问命令的信息帮助页（info page）

```
File: coreutils.info,  Node: ls invocation,  Next: dir invocation,  Up: Directo\
ry listing

`ls': List directory contents
============================

   The `ls' program lists information about files (of any type,
including directories).  Options and file arguments can be intermixed
arbitrarily, as usual.

   For non-option command-line arguments that are directories, by
default `ls' lists the contents of directories, not recursively, and
omitting files with names beginning with `.'.  For other non-option
arguments, by default `ls' lists just the file name.  If no non-option
argument is specified, `ls' operates on the current directory, acting
as if it had been invoked with a single argument of `.'.

   By default, the output is sorted alphabetically, according to the
locale settings in effect.  (1) If standard output is a terminal, the
output is in columns (sorted vertically) and control characters are
output as question marks; otherwise, the output is listed one per line
and control characters are output as-is.
--zz-Info: (coreutils.info.gz)ls invocation, 48 lines --Top----------------------
Welcome to Info version 4.3. Type C-h for help, m for menu item.
```

教学提纲

--help

COMMAND --help：显示命令的帮助信息；CMD -h

help BUILD-IN SHELL CMD：例如 help echo

```
yixf@Yixf-Ubuntu: ~ 13:35:31 $ ls --help
用法：ls [选项]... [文件]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

长选项必须使用的参数对于短选项时也是必需使用的。
  -a, --all                  不隐藏任何以 . 开始的项目
  -A, --almost-all           列出除 . 及 .. 以外的任何项目
      --author               与 -l 同时使用时列出每个文件的作者
  -b, --escape               以八进制溢出序列表示不可打印的字符
      --block-size=SIZE      scale sizes by SIZE before printing them.  E.g.,
                               '--block-size=M' prints sizes in units of
                               1,048,576 bytes.  See SIZE format below.
  -B, --ignore-backups       do not list implied entries ending with ~
  -c                         with -lt: sort by, and show, ctime (time of last
                               modification of file status information)
                               with -l: show ctime and sort by name
                               otherwise: sort by ctime, newest first
  -C                         list entries by columns
      --color[=WHEN]         colorize the output.  WHEN defaults to 'always'
                               or can be 'never' or 'auto'.  More info below
  -d, --directory            list directory entries instead of contents,
                               and do not dereference symbolic links
```

教学提纲

其他命令

- which COMMAND：查找命令
- whereis COMMAND：查找软件包
- whatis COMMAND：获得索引的简单说明信息，相当于 man -f
- apropos KEYWORD：使用关键字来查找相关文件，相当于 man -k
- makewhatis：建立 whatis 和 apropos 搜索使用的数据库

```
josemon@creativemindz:~$ apropos who
at.allow (5)         - determine who can submit jobs via at or batch
at.deny (5)          - determine who can submit jobs via at or batch
bsd-from (1)         - print names of those who have sent mail
from (1)             - print names of those who have sent mail
w (1)                - Show who is logged on and what they are doing.
w.procps (1)         - Show who is logged on and what they are doing.
who (1)              - show who is logged on
whoami (1)           - print effective userid
whois (1)            - client for the whois directory service
josemon@creativemindz:~$
```

命令信息 | man vs. info vs. --help

man

man 可以显示系统手册页中的内容，这些内容大多数都是对命令的解释信息，这些信息是操作系统文档里面的。如果没有文档，是不会显示这些帮助信息的。一般比 help 出来的要详细。

info

info 是一个基于菜单的超文本系统，由 GNU 项目开发并由 Linux 发布。info 工具包括一些关于 Linux Shell、工具、GNU 项目开发程序的说明文档。

--help

--help 并不是一个独立的工具，而是一个工具选项，可以用来显示一些工具的信息。这些帮助信息是程序的作者加上去的，也就是说，这些信息是程序内部的。一般比 man 出来的要简单。

命令信息 | man vs. info vs. --help

man

man 可以显示系统手册页中的内容，这些内容大多数都是对命令的解释信息，这些信息是操作系统文档里面的。如果没有文档，是不会显示这些帮助信息的。一般比 help 出来的要详细。

info

info 是一个基于菜单的超文本系统，由 GNU 项目开发并由 Linux 发布。info 工具包括一些关于 Linux Shell、工具、GNU 项目开发程序的说明文档。

--help

--help 并不是一个独立的工具，而是一个工具选项，可以用来显示一些工具的信息。这些帮助信息是程序的作者加上去的，也就是说，这些信息是程序内部的。一般比 man 出来的要简单。

命令信息 | man vs. info vs. --help

man

man 可以显示系统手册页中的内容，这些内容大多数都是对命令的解释信息，这些信息是操作系统文档里面的。如果没有文档，是不会显示这些帮助信息的。一般比 help 出来的要详细。

info

info 是一个基于菜单的超文本系统，由 GNU 项目开发并由 Linux 发布。info 工具包括一些关于 Linux Shell、工具、GNU 项目开发程序的说明文档。

--help

--help 并不是一个独立的工具，而是一个工具选项，可以用来显示一些工具的信息。这些帮助信息是程序的作者加上去的，也就是说，这些信息是程序内部的。一般比 man 出来的要简单。

教学提纲

教学提纲

元字符

- 元字符（metacharacter）并不是命令本身的一部分，但它们是 shell 的特性，能够让用户创建复杂的操作。
- 元字符是正则表达式的一个重要方面。
- 最流行的元字符称为通配符（wildcard）。

通配符

元字符

- 元字符（metacharacter）并不是命令本身的一部分，但它们是 shell 的特性，能够让用户创建复杂的操作。
- 元字符是正则表达式的一个重要方面。
- 最流行的元字符称为通配符（wildcard）。

通配符

元字符

- 元字符（metacharacter）并不是命令本身的一部分，但它们是 shell 的特性，能够让用户创建复杂的操作。
- 元字符是正则表达式的一个重要方面。
- 最流行的元字符称为通配符（wildcard）。

通配符

- 通配符是专用字符，能够用于同时匹配多个文件，从而增大一次性找到想要的文件名或目标的可能性。
- 指定工作目录中的所有 word 文件（*.doc）

元字符

- 元字符（metacharacter）并不是命令本身的一部分，但它们是 shell 的特性，能够让用户创建复杂的操作。
- 元字符是正则表达式的一个重要方面。
- 最流行的元字符称为通配符（wildcard）。

通配符

- 通配符是专用字符，能够用于同时匹配多个文件，从而增大一次性找到想要的文件名或目标的可能性。
- 通配符一种命令行的路径扩展（path expansion）功能，只作用在 argument 的 path 上。
- 指定工作目录中的所有 word 文件（*.doc）
- 指定工作目录中的所有以 [或] 为后缀名的文件（*[]）

元字符

- 元字符（metacharacter）并不是命令本身的一部分，但它们是 shell 的特性，能够让用户创建复杂的操作。
- 元字符是正则表达式的一个重要方面。
- 最流行的元字符称为通配符（wildcard）。

通配符

- 通配符是专用字符，能够用于同时匹配多个文件，从而增大一次性找到想要的文件名或目标的可能性。
- 通配符一种命令行的路径扩展（path expansion）功能，只作用在 argument 的 path 上。
- 指定工作目录中的所有 word 文件（*.doc）
- 指定工作目录中的所有以 txt 或 text 为后缀的文本文件（*.t[ex]*）

元字符

- 元字符（metacharacter）并不是命令本身的一部分，但它们是 shell 的特性，能够让用户创建复杂的操作。
- 元字符是正则表达式的一个重要方面。
- 最流行的元字符称为通配符（wildcard）。

通配符

- 通配符是专用字符，能够用于同时匹配多个文件，从而增大一次性找到想要的文件名或目标的可能性。
- 通配符一种命令行的路径扩展（path expansion）功能，只作用在 argument 的 path 上。
- 指定工作目录中的所有 word 文件（*.doc）
- 指定工作目录中的所有以 txt 或 text 为后缀的文本文件（*.t[ex]*）

元字符

- 元字符（metacharacter）并不是命令本身的一部分，但它们是 shell 的特性，能够让用户创建复杂的操作。
- 元字符是正则表达式的一个重要方面。
- 最流行的元字符称为通配符（wildcard）。

通配符

- 通配符是专用字符，能够用于同时匹配多个文件，从而增大一次性找到想要的文件名或目标的可能性。
- 通配符一种命令行的路径扩展（path expansion）功能，只作用在 argument 的 path 上。
- 指定工作目录中的所有 word 文件（*.doc）
- 指定工作目录中的所有以 txt 或 text 为后缀的文本文件（*.t[ex]*）

元字符

- 元字符（metacharacter）并不是命令本身的一部分，但它们是 shell 的特性，能够让用户创建复杂的操作。
- 元字符是正则表达式的一个重要方面。
- 最流行的元字符称为通配符（wildcard）。

通配符

- 通配符是专用字符，能够用于同时匹配多个文件，从而增大一次性找到想要的文件名或目标的可能性。
- 通配符一种命令行的路径扩展（path expansion）功能，只作用在 argument 的 path 上。
- 指定工作目录中的所有 word 文件（*.doc）
- 指定工作目录中的所有以 txt 或 text 为后缀的文本文件（*.t[ex]*）

字符	含义	实例
*	匹配 0 或多个字符	a*b a与b之间可以有任意长度的任意字符，也可以一个也没有，如aabcb, axyzb, a012b, ab。
?	匹配任意一个字符	a?b a与b之间必须也只能有一个字符，可以是任意字符，如aab, abb, acb, a0b。
[list]	匹配 list 中的任意单一字符	a[xyz]b a与b之间必须也只能有一个字符，但只能是 x 或 y 或 z，如：axb, ayb, azb。
[!list]	匹配 除list 中的任意单一字符	a[!0-9]b a与b之间必须也只能有一个字符，但不能是阿拉伯数字，如axb, aab, a-b。
[c1-c2]	匹配 c1-c2 中的任意单一字符 如：[0-9] [a-z]	a[0-9]b 0与9之间必须也只能有一个字符 如a0b, a1b... a9b。
{string1,string2,...}	匹配 sring1 或 string2 (或更多)其一字符串	a{abc,xyz,123}b a与b之间只能是abc或xyz或123这三个字符串之一。

形式	匹配项
g*	以g开头的任一文件
b*.txt	以b开头，中间有任意多个字符， 并以.txt结尾的任一文件
Data???	以Data开头，后面跟3个字符的任一文件
[abc]*	以abc中的任一个开头的任一文件
BACKUP.[0-9][0-9][0-9]	以BACKUP.开头，后面紧跟3个数字的任一文件
[[:upper:]]*	以大写字母开头的任一文件
[![:digit:]]*	不以数字开头的任一文件
*[[:lower:]123]	以小写字母或数字1、2、3中 的任一个结尾的任一文件

教学提纲

字符	说明
'(单引号)	又叫硬转义，其内部所有的shell 元字符、通配符都会被关掉。注意，硬转义中不允许出现'(单引号)。
"(双引号)	又叫软转义，其内部只允许出现特定的shell 元字符：$用于参数代换 `用于命令代替
\(反斜杠)	又叫转义，去除其后紧跟的元字符或通配符的特殊意义。

转义符使用

- 使用单引号：`echo '$SHELL'`
- 使用双引号：`echo "$SHELL"`
- 使用反斜杠：`echo \$SHELL`

输出结果

- `$SHELL`
- `/bin/bash`
- `$SHELL`

字符	说明
'(单引号)	又叫硬转义，其内部所有的shell 元字符、通配符都会被关掉。注意，硬转义中不允许出现'(单引号)。
"(双引号)	又叫软转义，其内部只允许出现特定的shell 元字符：$用于参数代换 `用于命令代替
\(反斜杠)	又叫转义，去除其后紧跟的元字符或通配符的特殊意义。

转义符使用

- 使用单引号：`echo '$SHELL'`
- 使用双引号：`echo "$SHELL"`
- 使用反斜杠：`echo \$SHELL`

输出结果

- `$SHELL`
- `/bin/bash`
- `$SHELL`

字符	说明
'(单引号)	又叫硬转义，其内部所有的shell 元字符、通配符都会被关掉。注意，硬转义中不允许出现'(单引号)。
"(双引号)	又叫软转义，其内部只允许出现特定的shell 元字符：$用于参数代换 `用于命令代替
\(反斜杠)	又叫转义，去除其后紧跟的元字符或通配符的特殊意义。

转义符使用

- 使用单引号：`echo '$SHELL'`
- 使用双引号：`echo "$SHELL"`
- 使用反斜杠：`echo \$SHELL`

输出结果

- `$SHELL`
- `/bin/bash`
- `$SHELL`

字符	说明
"(单引号)	又叫硬转义，其内部所有的shell 元字符、通配符都会被关掉。注意，硬转义中不允许出现'(单引号)。
""(双引号)	又叫软转义，其内部只允许出现特定的shell 元字符：$用于参数代换 `用于命令代替
\(反斜杠)	又叫转义，去除其后紧跟的元字符或通配符的特殊意义。

转义符使用

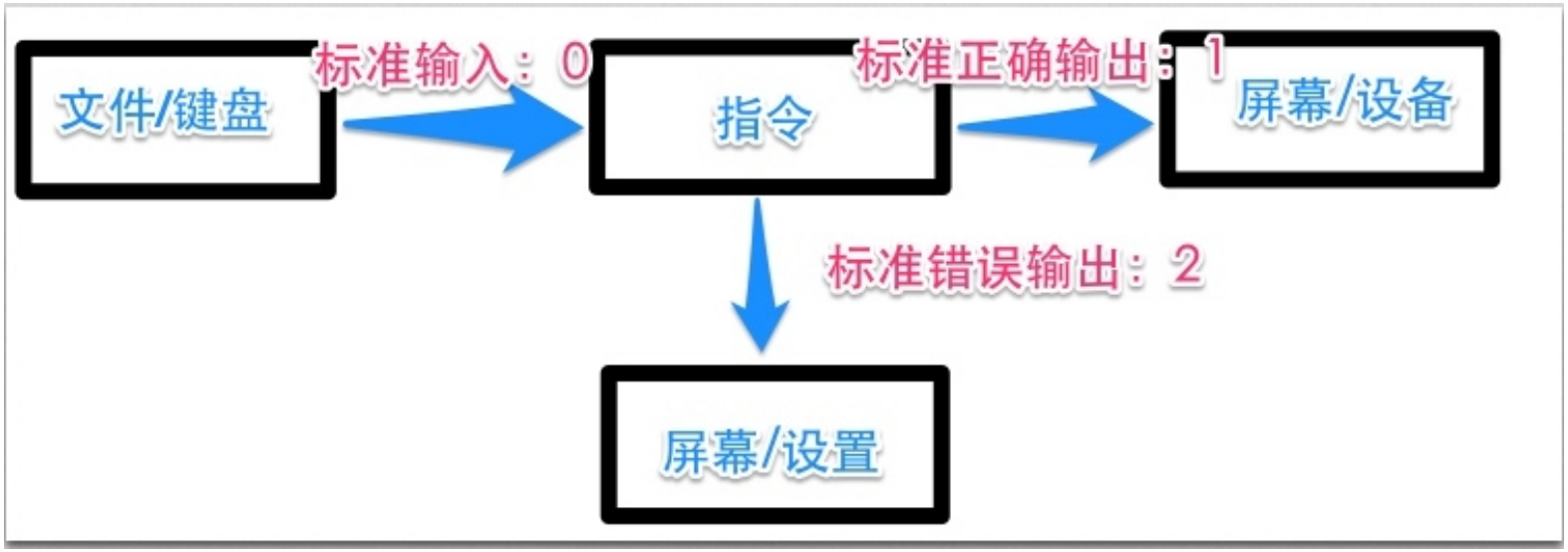
- 使用单引号：`echo '$SHELL'`
- 使用双引号：`echo "$SHELL"`
- 使用反斜杠：`echo \$SHELL`

输出结果

- `$SHELL`
- `/bin/bash`
- `$SHELL`

教学提纲

名称	说明	编号	默认	作用
STDIN(standard input)	标准输入	0	键盘	接收参数或数据
STDOUT(standard output)	标准输出	1	屏幕	输出结果
STDERR(standard error)	标准错误	2	屏幕	输出错误

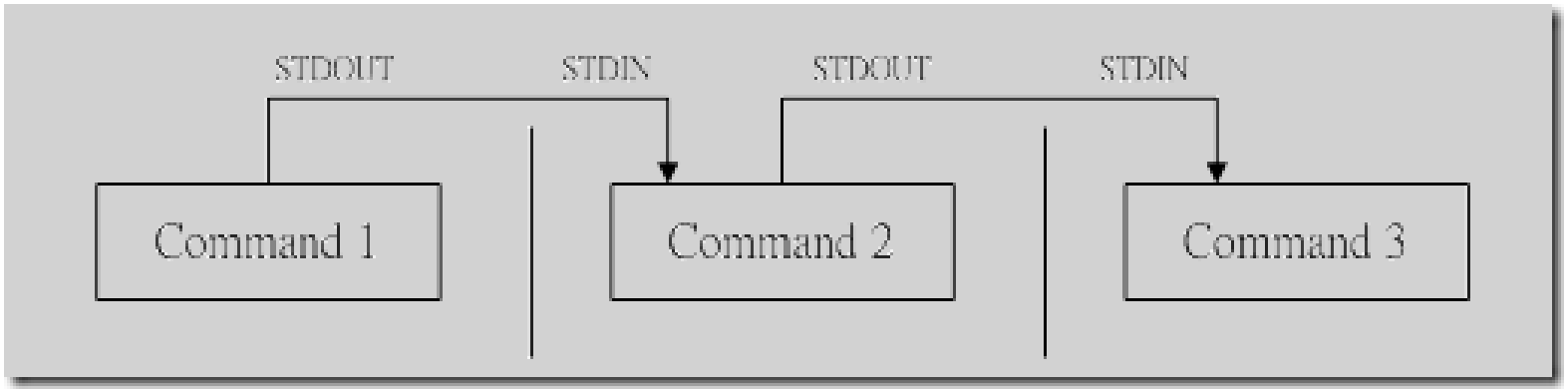


字符	说明	实例
<	重定向 STDIN	`sort < sort.in`
>	将 STDOUT 重定向到文件（覆盖）	`ls > ls.out`
>>	将 STDOUT 重定向到文件（追加）	`date >> date.out`
2>	将 STDERR 重定向到文件（覆盖）	`ls no 2> ls.err`
2>&1	将 STDERR 与 STDOUT 结合	`ls no 2>&1 ls.all`
< >	结合输入输出重定向	`sort <IN >OUT`

教学提纲

管道

- 管道（pipe）是一个操作符，它把输入和输出重定向结合在一起，从而将一个命令的输出立即作为另一个命令的输入。
- 管道用垂直线字符（|）表示。
- `ls -l /etc | less`



教学提纲

命令置换

- 命令置换也是一种将一个命令的输出作为另一个命令的参数的方法。
- 有两种命令置换的方法：
  - 新式写法 [推荐]：`$()`
  - 旧式写法：`` ` `` ` ``

实例

- `ls $(pwd)`
- ``ls `pwd` ``

命令置换

- 命令置换也是一种将一个命令的输出作为另一个命令的参数的方法。
- 有两种命令置换的方法：
  - 新式写法 [推荐]：`$()`
  - 旧式写法：`` ` ` ``

实例

- `ls $(pwd)`
- `ls `pwd``

命令置换

- 命令置换也是一种将一个命令的输出作为另一个命令的参数的方法。
- 有两种命令置换的方法：
  - 新式写法 [推荐]：`$()`
  - 旧式写法：`` ` ` ``

实例

- `ls $(pwd)`
- `ls `pwd``

教学提纲

教学提纲

命令	助记	说明
pwd	Print Work Directory	显示当前目录
ls	LiSt	列出目录的内容
cd	Change Directory	切换目录
mkdir	MaKe DIRectory	创建目录
rmdir	ReMove DIRectory	删除空目录
tree	TREE-like	展示树形的目录结构
du	Disk Usage	显示目录空间占用情况

选项	助记	说明
-a	All	列出目录的所有内容，包括隐藏文件
-i	Inode	列出目录内容，包括 inode 或磁盘索引号
-l	Long	用长格式（大小、权限等信息）列出目录的内容
-R	Recursive	列出目录内容，包括所有的子目录以及它们的内容
-r	Reverse	排序时反向排序
-S	Size	依据文件大小（由大到小）列出目录内容
-t	Time	依据时间标记（上一次的修改时间）列出目录内容

教学提纲

命令	助记	说明
file	—	识别文件类型（二进制、文本等）
echo	—	显示文本、打印信息
cat	conCATenate	显示一个文件
tac	—	逆序显示一个文件
touch	—	创建文件或者修改现有文件的属性
cp	CoPy	复制文件/目录
mv	MoVe	移动文件/目录或重命名文件/目录
rm	ReMove	删除文件
ln	LiNk	创建链接
stat	STATus	查看文件详细时间参数
rsync	Remote SYNChronize	复制/同步/备份文件
dd	Data Definition	转换并复制文件/分区/磁盘(Disk Destroyer，Delete Data)

命令	助记	说明
head	—	显示文件的开始部分
tail	—	显示文件的结尾部分
more	—	从头到尾浏览一个文件（只能向下翻页）
less	—	从开头或结尾开始浏览整个文件（可上下翻页）
wc	Word Count	计算文件行、字（符）数
sort	—	对命令或文件的输出进行排序
uniq	—	报告或删除重复的行
cut	—	以行为处理对象切割数据字段/列
tee	—	将命令的输出发送到多个位置
diff/diff3	—	逐行比较两/三个文本文件，列出其不同之处
patch	—	使用 diff 的结果来完成打补丁的工作
comm	—	对两个有序的文件进行比较
cmp	CoMPare	比较两个文件是否有差异

Command	Usage
echo string > newfile	The specified string is placed in a new file.
echo string >> existingfile	The specified string is appended to the end of an already existing file.
echo $variable	The contents of the specified environment variable are displayed.

-e

- 启用反斜线控制字符（如 \n，\t 等转义字符）的转换
- `echo "line1\nline2"`
- `echo -e "line1\nline2"`

Command	Usage
echo string > newfile	The specified string is placed in a new file.
echo string >> existingfile	The specified string is appended to the end of an already existing file.
echo $variable	The contents of the specified environment variable are displayed.

-e

- 启用反斜线控制字符（如 \n，\t 等转义字符）的转换
- `echo "line1\nline2"`
- `echo -e "line1\nline2"`

选项	助记	说明
-e	—	等同于 `-vE`
-E	End	在每一行的结尾显示一个 `$` 字符
-n	Number	在每一行的开头显示行号
-s	Squeeze	将连续的空行合并成一个空行
-t	Tabs	将非打印字符 tab 显示成 `^I`
-v	—	显示所有的非打印字符

基本用法

- 将多个文件的内容依次输出到屏幕上
- `cat file1 file2 file3`

妙用

- 将多个文件连接成一个更长的新文件
- `cat file1 file2 file3 > newfile`

选项	助记	说明
-e	—	等同于 `-vE`
-E	End	在每一行的结尾显示一个 `$` 字符
-n	Number	在每一行的开头显示行号
-s	Squeeze	将连续的空行合并成一个空行
-t	Tabs	将非打印字符 tab 显示成 `^I`
-v	—	显示所有的非打印字符

基本用法

- 将多个文件的内容依次输出到屏幕上
- `cat file1 file2 file3`

妙用

- 将多个文件连接成一个更长的新文件
- `cat file1 file2 file3 > newfile`

选项	助记	说明
-e	—	等同于 `-vE`
-E	End	在每一行的结尾显示一个 `$` 字符
-n	Number	在每一行的开头显示行号
-s	Squeeze	将连续的空行合并成一个空行
-t	Tabs	将非打印字符 tab 显示成 `^I`
-v	—	显示所有的非打印字符

基本用法

- 将多个文件的内容依次输出到屏幕上
- `cat file1 file2 file3`

妙用

- 将多个文件连接成一个更长的新文件
- `cat file1 file2 file3 > newfile`

常用命令 | 文件操作 | cat

Command	Usage
`cat file1 file2`	Concatenate multiple files and display the output; i.e., the entire content of the first file is followed by that of the second file.
`cat file1 file2 > newfile`	Combine multiple files and save the output into a new file.
`cat file >> existingfile`	Append a file to the end of an existing file.
`cat > file`	Any subsequent lines typed will go into the file until CTRL-D is typed.
`cat >> file`	Any subsequent lines are appended to the file until CTRL-D is typed.

old	new	`mv old new`
目录	目录	移动目录及其中的内容
文件	新的文件名	重命名文件
文件	新的目录	移动文件，文件名保持不变
文件	新的目录/新的文件名	移动文件并进行重命名

选项	助记	说明
-f	Force	使用强制模式，忽略所有警告
-i	Interactive	使用交互模式，提示用户确认每个删除
-r	Recursive	使用递归模式，删除所有子目录以及它们包含的文件

选项	助记	说明
-c	byte Counts	显示文件中字符的总数
-l	Line	显示文件的总行数
-L	Longest/Length	显示文件中最长一行的长度
-w	Word	显示文件中单词的总数

选项	助记	说明
-d	Dictionary	以字典的顺序进行排序，忽略非字符、数字和空格
-f	—	排序时忽略大小写
-g	General-number	按数值排序
-k	Key	以第几个字段进行排序
-M	Month	按月份排序
-m	Merge	合并经过排序的文件
-n	Numeric	以数值大小进行排序
-R	Random	随机排序
-r	Reverse	倒序排序
-t	—	指定分隔符
-u	Unique	排序，只考虑唯一的值
-V	Version	版本号排序（能正确排序字符数字混杂的情况，如：chr1,chr2,chr10,...）

选项	助记	说明
-c	Count	统计每行重复的次数
-d	Dupicate	只显示重复行
-u	Unique	只显示没有重复的行

选项	助记	说明	实例
-d	Delimiter	指定分隔符	-d:
-f	Fields	表示取第几个字段	-f1; -f1,5; -f1-5

教学提纲

命令	助记	说明
chown	CHange OWNer	改变文件或目录的所有者
chgrp	CHange GRouP	改变文件或目录的所属组
chmod	CHange MODe	修改权限
useradd	—	添加用户帐号
usermod	—	修改用户账户的属性
userdel	—	删除用户帐号
groupadd	—	添加组账户
groupmod	—	修改组账户的属性
groupdel	—	删除组账户
passwd	PASSWorD	修改密码
umask	—	显示、设置默认的权限方案

命令	助记	说明
w	Who	显示登录的用户
who	—	显示当前所有登录用户的信息
whoami	—	显示当前用户名
who am i	—	最初作为什么用户登录
last	—	显示用户最近登录信息
finger	—	显示用户的相关信息
id	—	显示登录的用户以及用户的组的相关信息
groups	—	显示用户所属的组

教学提纲

命令	助记	说明
which	—	如果文件位于用户的 PATH 内，则显示文件位置
whereis	—	显示文件的位置
find	—	在目录结构中查找文件/目录
locate	—	通过名字在数据库中查找文件
updatedb	UPDATE DataBase	建立整个系统目录文件的数据库
alias	—	查看别名信息
df	Disk Free	显示磁盘空间的使用情况
mount	—	挂载文件系统
date	—	显示当前日期和时间
cal	CALendar	显示当月的日历
set	—	显示/设置 shell 变量
env	ENVironment	显示/设置用户变量
export	—	显示/设置导出成用户变量的 shell 变量
unset	—	清除环境变量

教学提纲

命令	助记	说明
w	—	用户进程信息
top	—	显示所有正在运行的进程（任务管理器）
ps	Process Status	显示当前的活动进程
pstree	—	查看进程树
kill	—	结束进程
killall	—	根据进程名结束进程
uptime	—	显示系统从开机到现在所运行的时间
uname -a	—	显示内核等系统信息
free	—	显示内存及交换区占用情况
jobs	—	显示正在运行的 shell 作业
bg	BackGround	列出已停止或后台运行的作业
fg	ForeGround	将作业从后台转到前台
cron	—	周期性执行计划任务
at	—	一次性执行特定任务
&	CMD &	在后台运行 CMD 命令

信息解读

- load average：分别显示系统在过去 1、5、15 分钟内的平均负载程度
- FROM：显示用户从何处登录系统，“:0”代表该用户是从 X Window 下打开文本模式窗口登录的
- IDLE：用户闲置的时间。这是一个计时器，一旦用户执行任何操作，该计时器便会被重置
- JCPU：以终端代号来区分，该终端所有相关的进程执行时，所消耗的 CPU 时间会显示在这里
- PCPU：CPU 执行程序耗费的时间
- WHAT：用户正在执行的操作

Load average is the average of the **load number** for a given period of time. It takes into account processes that are:

- Actively running on a CPU.
- Considered runnable, but waiting for a CPU to become available.
- Sleeping: i.e., waiting for some kind of resource (typically, I/O) to become available.

The load average can be obtained by running **w**, **top** or **uptime**.

常用命令 | 进程管理 | w | Load Averages

The load average is displayed using three different sets of numbers, as shown in the following example:

The last piece of information is the average load of the system. Assuming our system is a single-CPU system, the 0.25 means that for the past minute, on average, the system has been 25% utilized. 0.12 in the next position means that over the past 5 minutes, on average, the system has been 12% utilized; and 0.15 in the final position means that over the past 15 minutes, on average, the system has been 15% utilized. If we saw a value of 1.00 in the second position, that would imply that the single-CPU system was 100% utilized, on average, over the past 5 minutes; this is good if we want to fully use a system. A value over 1.00 for a single-CPU system implies that the system was over-utilized: there were more processes needing CPU than CPU was available.

If we had more than one CPU, say a quad-CPU system, we would divide the load average numbers by the number of CPUs. In this case, for example, seeing a 1 minute load average of 4.00 implies that the system as a whole was 100% (4.00/4) utilized during the last minute.

Short term increases are usually not a problem. A high peak you see is likely a burst of activity, not a new level. For example, at start up, many processes start and then activity settles down. If a high peak is seen in the 5 and 15 minute load averages, it would may be cause for concern.

While a static view of what the system is doing is useful, monitoring the system performance live over time is also valuable. One option would be to run **ps** at regular intervals, say, every two minutes. A better alternative is to use **top** to get constant real-time updates (every two seconds by default) until you exit by typing `q`. **top** clearly highlights which processes are consuming the most CPU cycles and memory (using appropriate commands from within **top**.)

常用命令 | 进程管理 | top

```
top - 16:14:55 up 4 days, 14:40,  3 users,  load average: 0.48, 0.20, 0.11
Tasks: 167 total,   2 running, 164 sleeping,   0 stopped,   1 zombie
%Cpu(s):  2.1 us,  8.0 sy,  0.0 ni, 89.7 id,  0.0 wa,  0.0 hi,  0.2 si,  0.0 st
KiB Mem:   1020440 total,   946556 used,    73884 free,    20380 buffers
KiB Swap:  1589244 total,   632116 used,   957128 free,   246532 cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1987	test2	9	-11	442204	2104	1336	S	12.29	0.206	140:12.33	pulseaudio
3185	test2	20	0	444216	2924	1176	S	9.634	0.287	7:28.85	sd_espeak
2229	test2	20	0	2126148	189700	12072	S	0.997	18.59	26:32.41	gnome-shell
2749	test2	20	0	748692	169028	3252	S	0.997	16.56	6:35.50	orca
1270	test2	20	0	103428	168	168	S	0.664	0.016	25:28.50	VBoxClient
19783	test2	20	0	1157756	187008	17036	S	0.664	18.33	6:55.73	firefox
18464	test2	20	0	2341924	8284	4144	S	0.332	0.812	5:00.30	soffice.bin
19131	test2	20	0	660772	17676	10336	S	0.332	1.732	0:07.61	gnome-terminal-
1	root	20	0	47564	2444	1224	S	0.000	0.240	0:02.63	systemd
2	root	20	0	0	0	0	S	0.000	0.000	0:00.36	kthreadd
3	root	20	0	0	0	0	S	0.000	0.000	0:20.02	ksoftirqd/0
5	root	0	-20	0	0	0	S	0.000	0.000	0:00.00	kworker/0:0H
7	root	rt	0	0	0	0	S	0.000	0.000	0:00.35	migration/0
8	root	-2	0	0	0	0	S	0.000	0.000	0:00.00	rcuc/0
9	root	-2	0	0	0	0	S	0.000	0.000	0:00.00	rcub/0
10	root	20	0	0	0	0	S	0.000	0.000	0:32.17	rcu_preempt
11	root	20	0	0	0	0	S	0.000	0.000	0:21.53	rcuop/0
12	**root**	**20**	**0**	**0**	**0**	**0**	**R**	**0.000**	**0.000**	**0:20.93**	**rcuop/1**
13	root	20	0	0	0	0	S	0.000	0.000	0:00.00	rcu_bh
14	root	20	0	0	0	0	S	0.000	0.000	0:00.00	rcuob/0
15	root	20	0	0	0	0	S	0.000	0.000	0:00.00	rcuob/1
16	root	20	0	0	0	0	S	0.000	0.000	0:00.00	rcu_sched
17	root	20	0	0	0	0	S	0.000	0.000	0:00.00	rcuos/0

The first line of the **top** output displays a quick summary of what is happening in the system including:

- How long the system has been up
- How many users are logged on
- What is the load average

The **load average** determines how busy the system is. A load average of 1.00 per CPU indicates a fully subscribed, but not overloaded, system. If the load average goes above this value, it indicates that processes are competing for CPU time. If the load average is very high, it might indicate that the system is having a problem, such as a runaway process (a process in a non-responding state).

The second line of the **top** output displays the total number of processes, the number of running, sleeping, stopped and zombie processes. Comparing the number of running processes with the load average helps determine if the system has reached its capacity or perhaps a particular user is running too many processes. The stopped processes should be examined to see if everything is running correctly.

The third line of the **top** output indicates how the CPU time is being divided between the users (**us**) and the kernel (**sy**) by displaying the percentage of CPU time used for each.

The percentage of user jobs running at a lower priority (niceness - **ni**) is then listed. Idle mode (**id**) should be low if the load average is high, and vice versa. The percentage of jobs waiting (**wa**) for I/O is listed. Interrupts include the percentage of hardware (**hi**) vs. software interrupts (**si**). Steal time (**st**) is generally used with virtual machines, which has some of its idle CPU time taken for other uses.

The fourth and fifth lines of the **top** output indicate memory usage, which is divided in two categories:

- Physical memory (RAM) −displayed on line 4.
- Swap space −displayed on line 5.

Both categories display total memory, used memory, and free space.

You need to monitor memory usage very carefully to ensure good system performance. Once the physical memory is exhausted, the system starts using **swap** space (temporary storage space on the hard drive) as an extended memory pool, and since accessing disk is much slower than accessing memory, this will negatively affect system performance.

If the system starts using swap often, you can add more swap space. However, adding more physical memory should also be considered.

Each line in the process list of the **top** output displays information about a process. By default, processes are ordered by highest CPU usage. The following information about each process is displayed:

- Process Identification Number (PID)
- Process owner (USER)
- Priority (PR) and nice values (NI)
- Virtual (VIRT), physical (RES), and shared memory (SHR)
- Status (S)
- Percentage of CPU (%CPU) and memory (%MEM) used
- Execution time (TIME+)
- Command (COMMAND)

Besides reporting information, **top** can be utilized interactively for monitoring and controlling processes. While **top** is running in a terminal window you can enter single-letter commands to change its behaviour. For example, you can view the top-ranked processes based on CPU or memory usage. If needed, you can alter the priorities of running processes or you can stop/kill a process.

Command	Output
t	Display or hide summary information (rows 2 and 3)
m	Display or hide memory information (rows 4 and 5)
A	Sort the process list by top resource consumers
r	Renice (change the priority of) a specific processes
k	Kill a specific process
f	Enter the top configuration screen
o	Interactively select a new sort order in the process list

作用

进程状态显示和进程控制；动态显示（每 5 秒钟自动刷新一次）。

选项

- c：显示整个命令行而不仅仅显示命令名
- d：指定刷新的时间间隔

命令

- h，？：获得帮助
- k：终止执行中的进程
- r：重新设置进程优先级
- s：改变刷新的时间间隔
- u：查看指定用户的进程
- W：将当前设置写入 /.toprc 文件中

作用

进程状态显示和进程控制；动态显示（每 5 秒钟自动刷新一次）。

选项

- c：显示整个命令行而不仅仅显示命令名
- d：指定刷新的时间间隔

命令

- h, ？：获得帮助
- k：终止执行中的进程
- r：重新设置进程优先级
- s：改变刷新的时间间隔
- u：查看指定用户的进程
- W：将当前设置写入 /.toprc 文件中

常用命令 | 进程管理 | top

作用

进程状态显示和进程控制；动态显示（每 5 秒钟自动刷新一次）。

选项

- c：显示整个命令行而不仅仅显示命令名
- d：指定刷新的时间间隔

命令

- h，？：获得帮助
- k：终止执行中的进程
- r：重新设置进程优先级
- s：改变刷新的时间间隔
- u：查看指定用户的进程
- W：将当前设置写入 /.toprc 文件中

选项

- a：显示所有用户的进程
- e：显示所有进程，包括没有控制终端的进程
- l：（小写 L）长格式显示
- u：显示用户名和启动时间
- w：宽行显示，可以使用多个 w 进行加宽显示
- x：显示没有控制终端的进程

实例

- ps：查看隶属于自己的进程
- ps -u or -l：查看隶属于自己的进程的详细信息
- ps -le or -aux：查看所有用户执行的进程的详细信息
- ps -aux --sort pid：可按进程执行的时间、PID、UID 等对进程进行排序
- ps -aux | grep USER，ps -uU USER：查看系统中指定用户执行的进程
- ps -le | grep PROCESS：查看指定进程信息

信息解读

- PID：进程号
- PPID：父进程的进程号
- TTY：进程启动的终端
- STAT：进程当前状态（S 休眠状态，D 不可中断的休眠状态，R 运行状态，Z 僵死状态，T 停止）
- NI：进程优先级
- TIME：进程自从启动以来启用 CPU 的总时间
- COMMAND/CMD：进程的命令名
- USER：用户名
- %CPU：占用 CPU 时间和总时间的百分比
- %MEM：占用内存与系统内存总量的百分比

ps provides information about currently running processes, keyed by **PID**. If you want a repetitive update of this status, you can use **top** or commonly installed variants such as **htop** or **atop** from the command line, or invoke your distribution's graphical system monitor application.

ps has many options for specifying exactly which tasks to examine, what information to display about them, and precisely what output format should be used.

Without options **ps** will display all processes running under the current shell. You can use the `-u` option to display information of processes for a specified username. The command `ps -ef` displays all the processes in the system in full detail. The command `ps -eLf` goes one step further and displays one line of information for every **thread** (remember, a process can contain multiple threads).

ps has another style of option specification which stems from the **BSD** variety of UNIX, where options are specified without preceding dashes. For example, the command `ps aux` displays all processes of all users. The command `ps axo` allows you to specify which attributes you want to view.

Command	Output

`ps aux`	`USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND`
	`root 1 0.0 0.0 19356 1292 ? Ss Feb27 0:08 /sbin/init`
	`root 2 0.0 0.0 0 0 ? S Feb27 0:00 [kthreadd]`
	`root 3 0.0 0.0 0 0 ? S Feb27 0:27 [migration/0]`

Command	Output

`ps axo`
`stat,priority,pid,pcpu,comm`
	`STAT PRI PID %CPU COMMAND`
	`Ss 20 1 0.0 init`
	`S 20 2 0.0 kthreadd`
	`S -100 3 0.0 migration/0`

pstree displays the processes running on the system in the form of a **tree diagram** showing the relationship between a process and its parent process and any other processes that it created. Repeated entries of a process are not displayed, and threads are displayed in curly braces.

```
test2@OpenSUSE:~> pstree
systemd─┬─ModemManager───2*[{ModemManager}]
        ├─3*[VBoxClient───{VBoxClient}]
        ├─VBoxClient───2*[{VBoxClient}]
        ├─accounts-daemon───2*[{accounts-daemon}]
        ├─agetty
        ├─at-spi-bus-laun─┬─dbus-daemon
        │                 └─3*[{at-spi-bus-laun}]
        ├─at-spi2-registr───{at-spi2-registr}
        ├─avahi-autoipd───avahi-autoipd
        ├─avahi-daemon
        ├─bluetoothd
        ├─colord───2*[{colord}]
        ├─cron
        ├─cupsd
        ├─2*[dbus-daemon]
        ├─dbus-launch
        ├─dconf-service───2*[{dconf-service}]
        ├─dhclient6
```

- 关闭进程：kill 进程号
- 强行关闭：kill -9 进程号
- 重启进程：kill -1 进程号
- 关闭图形程序：xkill
- 结束所有进程：killall
- 查找服务进程号：pgrep 服务名称
- 关闭进程：pkill 进程名称

To terminate a process you can type `kill -SIGKILL <pid>` or `kill -9 <pid>`. Note however, you can only **kill** your own processes: those belonging to another user are off limits unless you are root.

Linux supports **background** and **foreground** job processing. (A job in this context is just a command launched from a terminal window.) **Foreground** jobs run directly from the shell, and when one foreground job is running, other jobs need to wait for shell access (at least in that terminal window if using the GUI) until it is completed. This is fine when jobs complete quickly. But this can have an adverse effect if the current job is going to take a long time (even several hours) to complete.

In such cases, you can run the job in the **background** and free the shell for other tasks. The background job will be executed at lower priority, which, in turn, will allow smooth execution of the interactive tasks, and you can type other commands in the terminal window while the background job is running. By default all jobs are executed in the foreground. You can put a job in the background by suffixing `&` to the command, for example: `updatedb &`.

You can either use **CTRL-Z** to suspend a foreground job or **CTRL-C** to terminate a foreground job and can always use the **bg** and **fg** commands to run a process in the background and foreground, respectively.

The **jobs** utility displays all jobs running in background. The display shows the job ID, state, and command name, as shown here.

`jobs -l` provides a the same information as `jobs` including the PID of the background jobs.

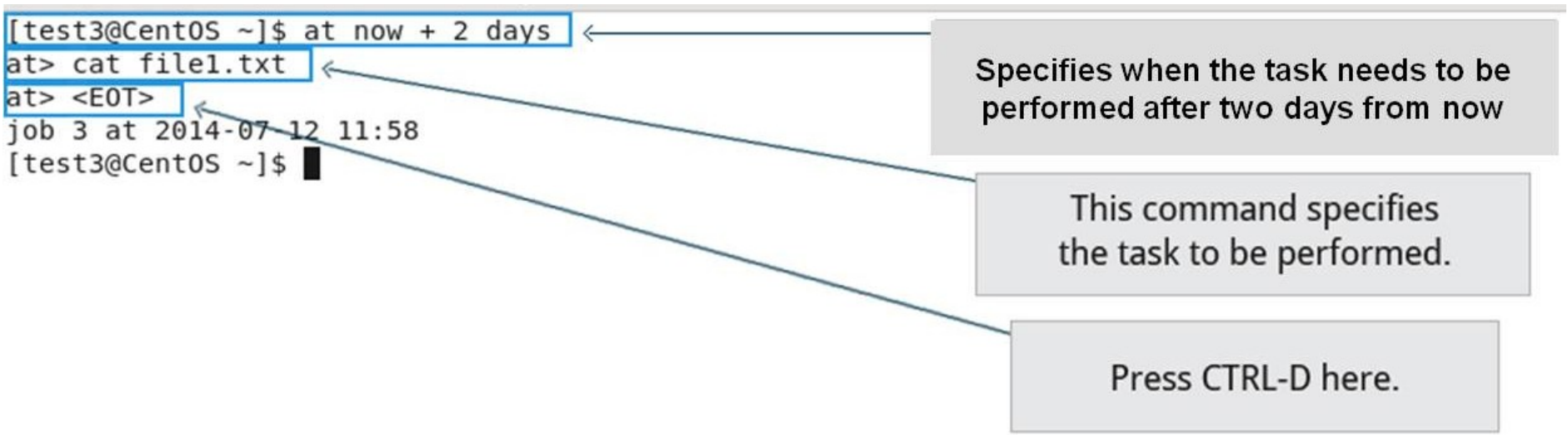
The background **jobs** are connected to the terminal window, so if you log off, the jobs utility will not show the ones started from that window.

- 进程的挂起（中止）：Ctrl+Z
- 进程的终止：Ctrl+C
- 恢复挂起/后台的进程到前台继续运行：fg
- 恢复挂起/后台的进程到后台继续运行：bg
- 查看被挂起的进程：jobs

```
[maple@linux ~]$  ping baidu.com -a >/dev/null &
[1] 12879
[maple@linux ~]$ jobs
[1]+  Running                 ping baidu.com -a > /dev/null &
[maple@linux ~]$
[maple@linux ~]$ ping google.com -a >/dev/null
#Ctrl+Z
[2]+  Stopped                 ping google.com -a > /dev/null
[maple@linux ~]$ jobs
[1]-  Running                 ping baidu.com -a > /dev/null &
[2]+  Stopped                 ping google.com -a > /dev/null
[maple@linux ~]$
[maple@linux ~]$ fg 1
ping baidu.com -a > /dev/null
#Ctrl+Z
[1]+  Stopped                 ping baidu.com -a > /dev/null
[maple@linux ~]$ jobs
[1]+  Stopped                 ping baidu.com -a > /dev/null
[2]-  Stopped                 ping google.com -a > /dev/null
[maple@linux ~]$
[maple@linux ~]$ bg 2
[2]- ping google.com -a > /dev/null &
[maple@linux ~]$ jobs
[1]+  Stopped                 ping baidu.com -a > /dev/null
[2]-  Running                 ping google.com -a > /dev/null &
[maple@linux ~]$
[maple@linux ~]$ %2
ping google.com -a > /dev/null
#Ctrl+Z
[2]+  Stopped                 ping google.com -a > /dev/null
[maple@linux ~]$ jobs
[1]-  Stopped                 ping baidu.com -a > /dev/null
[2]+  Stopped                 ping google.com -a > /dev/null
```

- at：安排（非交互式）作业在指定时间执行一次
- batch：安排作业在系统负载不重（平均负载降到 0.8 以下）时执行一次
- cron：安排周期性运行的作业
- crontab：用于生成 cron 进程所需要的 crontab 文件

Suppose you need to perform a task on a specific day sometime in the future. However, you know you will be away from the machine on that day. How will you perform the task? You can use the **at** utility program to execute any non-interactive command at a specified time, as illustrated in the diagram.



cron is a time-based scheduling utility program. It can launch routine background jobs at specific times and/or days on an on-going basis. **cron** is driven by a configuration file called `/etc/crontab` (**cron** table) which contains the various shell commands that need to be run at the properly scheduled times. There are both system-wide crontab files and individual user-based ones. Each line of a crontab file represents a job, and is composed of a so-called CRON expression, followed by a shell command to execute.

The `crontab -e` command will open the crontab editor to edit existing jobs or to create new jobs. Each line of the crontab file will contain 6 fields.

Field	Description	Values
MIN	Minutes	0 to 59
HOUR	Hour field	0 to 23
DOM	Day of Month	1-31
MON	Month field	1-12
DOW	Day Of Week	0-6 (0 = Sunday)
CMD	Command	Any command to be executed

```
1 * * * * * /usr/local/bin/execute/this/script.
    sh
2 30 08 10 06 * /home/sysadmin/full-backup
```

1. Schedule a job to execute ‘script.sh’ every minute of every hour of every day of the month, and every month and every day in the week.
2. Schedule a full-backup at 8.30am, 10-June irrespective of the day of the week.

```
1 * * * * * /usr/local/bin/execute/this/script.
    sh
2 30 08 10 06 * /home/sysadmin/full-backup
```

1. Schedule a job to execute ‘script.sh’ every minute of every hour of every day of the month, and every month and every day in the week.
2. Schedule a full-backup at 8.30am, 10-June irrespective of the day of the week.

sleep suspends execution for at least the specified period of time, which can be given as the number of seconds (the default), minutes, hours or days. After that time has passed (or an interrupting signal has been received) execution will resume. Syntax:

```
sleep NUMBER[SUFFIX]...
```

where SUFFIX may be:

1. s for seconds (the default)
2. m for minutes
3. h for hours
4. d for days

sleep and **at** are quite different; **sleep** delays execution for a specific period while **at** starts execution at a later time.

用途

通过让提交的命令忽略 HUP（hangup）信号，使进程在用户退出登录后仍旧继续执行。
nohup 命令将执行后的数据信息和错误信息默认储存到文件 nohup.out 中。
远程登录服务器运行大型程序时常用/必用！

使用

nohup COMMADN/PROGRAM &

用途

通过让提交的命令忽略 HUP（hangup）信号，使进程在用户退出登录后仍旧继续执行。
nohup 命令将执行后的数据信息和错误信息默认储存到文件 nohup.out 中。
远程登录服务器运行大型程序时常用/必用！

使用

nohup COMMADN/PROGRAM &

教学提纲

命令	助记	说明	常用后缀
tar	Tape ARchive	创建磁盘归档（打包/归档）	.tar
gzip	GNU Zip	压缩	.gz, .z, .Z, (.tar.gz, .tgz)
gunzip	—	解压	.gz, .z, .Z, (.tar.gz, .tgz)
bzip2	—	压缩	.bz, .bz2, .bzip2, (.tar.bz, .tbz, .tbz2)
bunzip2	—	解压	.bz, .bz2, .bzip2, (.tar.bz, .tbz, .tbz2)
xz	—	压缩	.xz
unxz	—	解压	.xz
zip	—	压缩	.zip
unzip	—	解压	.zip

命令	说明
gzip	Linux 中使用最广泛
bzip2	压缩后的文件比 gzip 节省空间（压缩时间长）
xz	压缩后的文件在 Linux 中是最节省空间的（压缩时间更长）
zip	和其他操作系统（如 Windows）的兼容性好

备注：归档是否是压缩的和采用哪种压缩方式并不取决于其扩展名，扩展名只是为了便于辨识。

选项	助记	说明
-c	Create	创建文件（打包）
-f	File	指定打包文件或设备
-j	—	使用 bzip2 压缩/解压
-r	append	向包中（末尾）追加文件
-t	lisT	显示包中的内容
-u	Update	更新包中的内容或添加新内容
-v	Verbose	显示详细的打包过程
-x	eXtract	提取文件（解包）
-z	gZip	使用 gzip 压缩/解压

常用命令

- tar 归档、gzip 压缩/etc 目录：
  ```
  tar -cvzf /backups/etc_backup.tar.gz /etc
  ```
- gzip 解压缩、tar 解归档：
  ```
  tar -xvzf /backups/etc_backup.tgz
  ```
- tar 归档、bzip2 压缩/etc 目录：
  ```
  tar -cvjf /backups/etc_backup.tar.bz /etc
  ```
- bzip2 解压缩、tar 解归档：
  ```
  tar -xvjf /backups/etc_backup.tbz
  ```

教学提纲

命令	说明
write	向另外一个用户发送信息，以 Ctrl+D 结束
wall	向所有用户广播信息
ping	测试网络连通性
hostname	查看本机的 hostname
ifconfig	查看网络设置信息
route	显示和操作 IP 路由表
ip	整合了 ifconfig 和 route 两个命令

命令	说明
traceroute	追踪数据包在网络上传输时的全部路径
host	返回一个主机的网络地址或返回主机名
nslookup	查询一台机器的 IP 地址和其对应的域名
dig	域名查询工具
ethtool	查询及设置网卡参数
netstat	显示各种网络相关信息
nmap	网络扫描和嗅探工具包
tcpdump	网络数据采集分析工具
iptraf	实时地监视网卡流量

命令	说明
ssh	远程登录客户端工具
scp	基于 SSH 登录进行安全的远程文件拷贝（Secure CoPy）
wget	命令行下载工具
curl	利用 URL 规则在命令行下工作的文件传输工具
ftp, sftp	命令行下的 FTP 客户端
ncftp, yafc	命令行下的 FTP 客户端, 支持 Windows 和 Linux
lftp	功能强大的文件传输客户端程序，支持 FTP、SSH 和 HTTP 等多种文件传输协议
lynx	字符界面下的全功能的 WWW 浏览器
links, elinks	基于 lynx，支持 frame、表格和鼠标
w3m	较新的命令行下的网页浏览器

教学提纲

命令	说明
shutdown	关闭系统
reboot	重新启动系统
halt	立即关闭系统
poweroff	通过切断电源来关闭系统

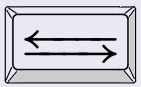
教学提纲

命令	助记	说明
man	MANual	查看命令的联机帮助页
info	Info format	查看命令的信息帮助页
apropos	—	使用关键字来查找文件
whatis	—	显示联机帮助页中对命令的简短描述
makewhatis	—	建立 whatis 和 apropos 搜索使用的数据库
whereis	—	查找软件包
--help/-h	—	命令的 --help/-h 选项

教学提纲

教学提纲

自动补全

自动补全/补齐允许用户输入命令或文件名起始的若干个字母后，按  （Tab）补全命令名或文件名。

命令历史

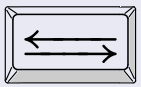
命令历史允许用户浏览先前输入的命令并重新调用它们，用 history 命令可以显示命令列表，按方向键（上、下）可查找以前执行过的命令。

命令别名

- alias 查看、定义别名；unalias 删除别名
- `alias ll="ls -l"; alias la="ls -al"; alias cd3="cd ../../.."`
- `alias ls="ls -l"`; 执行原始的 ls 命令：`\ls`

命令技巧 | 补全、历史和别名

自动补全

自动补全/补齐允许用户输入命令或文件名起始的若干个字母后，按  （Tab）补全命令名或文件名。

命令历史

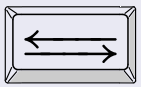
命令历史允许用户浏览先前输入的命令并重新调用它们，用 history 命令可以显示命令列表，按方向键（上、下）可查找以前执行过的命令。

命令别名

- alias 查看、定义别名；unalias 删除别名
- `alias ll="ls -l"; alias la="ls -al"; alias cd3="cd ../../.."`
- `alias ls="ls -l"`; 执行原始的 ls 命令：`\ls`

命令技巧 | 补全、历史和别名

自动补全

自动补全/补齐允许用户输入命令或文件名起始的若干个字母后，按  （Tab）补全命令名或文件名。

命令历史

命令历史允许用户浏览先前输入的命令并重新调用它们，用 history 命令可以显示命令列表，按方向键（上、下）可查找以前执行过的命令。

命令别名

- alias 查看、定义别名；unalias 删除别名
- `alias ll="ls -l"; alias la="ls -al"; alias cd3="cd ../../.."`
- `alias ls="ls -l"`; 执行原始的 ls 命令：`\ls`

教学提纲

命令连接符

; 用 ；间隔的各命令按顺序依次执行

&& 前后命令的执行存在逻辑与关系：只有 && 前面的命令执行成功后，它后面的命令才被执行

|| 前后命令的执行存在逻辑或关系：只有 || 前面的命令执行失败后，它后面的命令才被执行。

后台运行

- COMMAND &：后台运行，关掉客户端/终端会停止运行
- nohup COMMAND &：后台运行，关掉客户端/终端仍会继续运行

在子 shell 中运行命令：(CMD)

- 在其它目录运行一个命令，然后自动返回当前工作目录
- (cd /tmp; ls)

命令连接符

; 用 ; 间隔的各命令按顺序依次执行

&& 前后命令的执行存在逻辑与关系：只有 && 前面的命令执行成功后，它后面的命令才被执行

|| 前后命令的执行存在逻辑或关系：只有 || 前面的命令执行失败后，它后面的命令才被执行。

后台运行

- `COMMAND &`：后台运行，关掉客户端/终端会停止运行
- `nohup COMMAND &`：后台运行，关掉客户端/终端仍会继续运行

在子 shell 中运行命令：(CMD)

- 在其它目录运行一个命令，然后自动返回当前工作目录
- `(cd /tmp; ls)`

命令连接符

; 用 ; 间隔的各命令按顺序依次执行

&& 前后命令的执行存在逻辑与关系：只有 && 前面的命令执行成功后，它后面的命令才被执行

|| 前后命令的执行存在逻辑或关系：只有 || 前面的命令执行失败后，它后面的命令才被执行。

后台运行

- `COMMAND &`：后台运行，关掉客户端/终端会停止运行
- `nohup COMMAND &`：后台运行，关掉客户端/终端仍会继续运行

在子 shell 中运行命令：(CMD)

- 在其它目录运行一个命令，然后自动返回当前工作目录
- `(cd /tmp; ls)`

&&

```
[root@linux ~]# ls /tmp && touch /tmp/testingagin
```

利用 && 来决定，当前面的命令执行结果为正确（如仅有标准输出）时，就可以接着执行后续命令，否则就略过。因此，当 ls /tmp 没有问题时，就会接着执行 touch /tmp/testingagin 了。

```
[root@linux ~]# ls /vbird && touch /vbird/test
```

||

```
[root@linux ~]# ls /tmp/vbirding || touch /tmp/vbirding
```

|| 完全与 && 相反，当前一个命令有错误时，才会执行 || 后面的命令

简单来说，当 ls /tmp/vbirding 发生错误时，才会使用 touch /tmp/vbirding 建立这个文件。

mkdir test; cd test 创建文件夹，并成功切换进去

cd test; mkdir test 切换文件夹报错，创建文件夹

mkdir test && cd test 创建文件夹，并成功切换进去

cd test && mkdir test 切换文件夹报错，不会创建文件夹

mkdir test || cd test 创建文件夹，但不会切换进去

cd test || mkdir test 切换文件夹报错，但会创建文件夹

教学提纲

快捷键	作用
clear	清屏
Ctrl+L	清屏
Ctrl+S	阻止屏幕输出
Ctrl+Q	恢复屏幕输出
Ctrl+C	终止命令
Ctrl+Z	挂起命令
Ctrl+D	注销会话
exit	注销会话

快捷键	作用
Ctrl+A	移动到命令行首
Ctrl+E	移动到命令行尾
Ctrl+F	按字符前/右移
Ctrl+B	按字符后/左移
Ctrl+U	从光标处删除至命令行首
Ctrl+K	从光标处删除至命令行尾
Ctrl+W	从光标处删除至字首
Ctrl+H	删除光标前的字符
Ctrl+D	删除光标处的字符
Ctrl+T	交换光标前的两个字符
Ctrl+Y	粘贴上次的删除
Ctrl+_	撤销操作

命令技巧 | 快捷键（续）

快捷键	作用
Ctrl+R	逆向搜索命令历史
Ctrl+G	退出历史搜索模式
Ctrl+P	历史中的上一条命令
Ctrl+N	历史中的下一条命令
!!	执行上一条命令
!ABC	执行最近的以 ABC 开头的命令
!ABC:p	仅打印输出而不执行
!^	上一条命令的第一个参数
!^:p	打印输出上一条命令的第一个参数
!$	上一条命令的最后一个参数
!$:p	打印输出上一条命令的最后一个参数
!*	上一条命令的所有参数
!*:p	打印输出上一条命令的所有参数
^A^B	将上一条命令中的第一个 A 替换为 B 并执行
^A^B^	将上一条命令中的所有 A 替换为 B 并执行

教学提纲

教学提纲

知识点

- Linux 命令的基本结构：命令、参数（选项、目标）
- 查找命令相关信息的方法：man、info、--help/-h、apropos
- 命令的修改：通配符、转义符、输入输出重定向、管道、命令置换
- 目录和文件操作、权限管理、系统导航、压缩解压等常用命令
- 命令使用技巧：补全、历史、别名、连接符、后台运行、快捷键

技能

- 常用命令及其选项的使用
- 命令的修改
- 命令的使用技巧

教学提纲

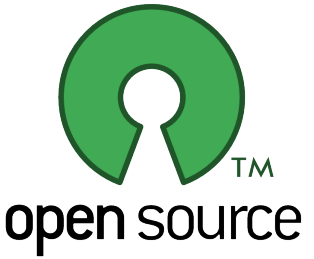
Linux 命令 | 思考题

1. Linux 命令由哪几部分构成？
2. 查找命令相关信息的方法有哪些？
3. 列举三个常用的通配符并举例说明。
4. 如何进行输入、结果输出和错误输出的重定向？
5. 命令置换的方法有哪些，举例说明。
6. 列举几个常用的命令及其常用选项。
7. 列举命令使用技巧并进行说明。

下节预告

你是如何处理文本数据（如生物信息学数据）的，包括查找、替换、修改、简单的统计等。
如果是使用 Word 或 Excel 来处理的，你有没有想吐槽的地方？

Powered by



TEX LATEX XETEX Beamer