



Linux Shell 介绍

1

主要内容和学习要求

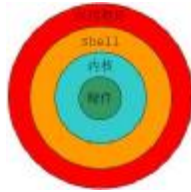
- q 知道什么是 shell 和一些常见的 shell
- q 掌握 bash 的基本功能（通配符、别名等）
- q 了解 bash 的启动脚本
- q 了解 shell 变量，学会查看和修改变量的值
- q 理解如何定制 bash

2

Shell 简介

q shell 是系统的用户界面，它提供了用户和 Linux（内核）之间进行交互操作的一种接口。用户在命令行中输入的每个命令都由 shell 先解释，然后传给 Linux 内核去执行。

q 如果把 Linux 内核想象成一个球体的中心，shell 就是围绕内核的外层，从 shell 向 Linux 操作系统传递命令时，内核就会做出相应的反应。



3

Shell 简介

q shell 是一个命令语言解释器，拥有自己内建的 shell 命令集。此外，shell 也能被系统中其他应用程序所调用。

q shell 的另一个重要特性是它自身就是一个解释型的程序设计语言，shell 程序设计语言支持在高级语言里所能见到的绝大多数程序控制结构，比如循环，函数，变量和数组等。shell 编程语言简单易学，一旦掌握后它将成为你的得力工具。任何在命令行中能键入的命令也能放到一个可执行的 shell 程序里。

4

常用的 Shell

q 常用的 shell 有 Bourne shell, C shell, 和 Korn shell。

q 三种 shell 都有它们的优点和缺点。不同 shell 之间的转换非常方便。

q Bourne shell (sh)

作者是 Steven Bourne，它是 UNIX 最初使用的 shell 并且在每种 UNIX 上都可以使用。Bourne shell 在 shell 编程方面相当优秀，但在处理与用户的交互方面不如其他几种 shell。

5

常用的 Shell (续)

q C shell (csh)

C shell 由 Bill Joy 所写，它更多的考虑了用户界面的友好性。它支持象命令补齐等一些 Bourne shell 所不支持的特性。因为 C shell 的语法和 C 语言的很相似，C shell 被很多 C 程序员使用，这也是 C shell 名称的由来。

q Korn shell (ksh)

由 Dave Korn 所写。它集合了 C shell 和 Bourne shell 的优点并且和 Bourne shell 完全兼容。

6

常用的 Shell (续)

q 其它 shell

许多其它的 shell 基本上都是吸收了这些 shell 的优点扩展而成的 shell。常见的有 tcsh (csh 的扩展)，Bourne Again shell (bash, sh 的扩展)，和 Public Domain Korn shell (pdksh, ksh 的扩展)。

q bash 是现在大多数 Linux 系统的缺省 shell

bash 与 Bourne shell 完全向后兼容，并且在 Bourne shell 的基础上增加和增强了很多特性。bash 也包含了很多 csh 和 ksh 里的优点。bash 有很灵活和强大的编程接口，同时又有很友好的用户界面。

7

Bash 的功能

q 命令行

当用户打开一个（虚拟）终端时，可以看到一个 shell 提示符，标识了命令行的开始。用户可以在提示符后面输入任何命令

```
command [选项] [参数]
```

```
例: ls -l /home/jypan/linux/
```

注意：命令行中选项先于参数输入

8

命令行特征

q 在一个命令行中可以输入多个命令，用分号将各个命令隔开。例如：

```
ls -F; cp -i mydata newdata
```

q 如果一个命令太长，无法在一行中显示，可以使用反斜杠 \ 来续行，在多个命令行上输入一个命令或多个命令。例如：

```
ls -F; \  
cp -i mydata newdata
```

大多数 shell 在达到命令行尾时都会自动断开长命令

9

命令行特征 (续)

q 命令行编辑

命令行实际上是可以编辑的一个文本缓冲区，在按回车之前，可以对输入的命令进行编辑。如用 BACKSPACE 键可以删除刚键入的字符，也可以进行整行删除，还可以插入字符等。

10

常用的快捷键和组合键

左/右箭头键	向左/向右移动一个字符
Ctrl+a	移动到当前行的行首
Ctrl+e	移动到当前行的行尾
Ctrl+f	向前移动一个字符
Ctrl+b	向后移动一个字符
Ctrl+k	从光标处删除到本行的行尾
Ctrl+u	从光标处删除到本行的行首
Ctrl+l	清屏
Alt+f	向前移动一个单词
Alt+b	向后移动一个单词

stty -a 可以看到更多的快捷键。

11

通配符

q 通配符

u bash 提供许多功能用来帮助用户节省输入命令的时间，其中最常用的一种方法就是使用通配符。

u 通配符就是一些特殊的字符，可以用来在引用文件名时简化命令的书写。用户在使用时可以用通配符来指定一种模式，即所谓的“模式串”(pattern)，然后 shell 会把那些与这种模式能够匹配的文件作为输入文件。

u 在 bash 中可以使用三种通配符：*、?、[]。

12

通配符的含义

*	匹配任意长度的字符串（包括零个字符）
?	匹配任何单个字符
[]	创建一个字符表列，方括号中的字符用来匹配或不匹配单个字符。如： [xyz] 匹配 x、y 或 z，但不能匹配 xx、xy 或者其它任意组合。 无论列表中有多少个字符，它只匹配一个字符。 [abcde] 可以简写为 [a-e]。 另外，用感叹号作为列表的第一个字符可以起到反意作用，如： [!xyz] 表示匹配 x、y、z 以外的任意一个字符。

13

通配符举例

u 通配符“*”的常用方法就是查找具有相同扩展名的文件

```
ls *.tar.gz
```

通配符“*”有时可以将几百的命令缩短成一个命令。假设当前目录下有许多文件，现在要删除扩展名为“.old”的文件，如果有几百个这样的文件，逐个删除显然很麻烦，这时可以使用通配符：

```
rm *.old
```

u 问号通配符“?”必须匹配一个且只能匹配一个字符，通常用来查找比 * 更为精确的匹配。

```
ls *.???
```

14

方括号通配符举例

u 方括号通配符使用括号内的字符作为被匹配的字符，且只能匹配其中的一个字符。如列出以 a、b、c 开头，且以 .dat 为扩展名的所有文件：

```
ls [abc]*.dat
```

可以在方括号中使用连字符 - 来指定一个范围，如列出以字母开头，数字结尾的所有文件：

```
ls [a-zA-Z]*[0-9]
```

15

通配符使用注意事项

u 文件名最前面的圆点“.”和路径名中的斜杠“/”必须显式匹配。例如“*”不能匹配“.bashrc”，而“.*”才可以匹配“.bashrc”。

u 连字符 - 仅在方括号内有效，表示字符范围。如果在方括号外面就成为普通字符了。而 * 和 ? 在方括号外面是通配符，若出现在方括号之内，它们也失去通配符的能力，成为普通字符了。

```
ls *  
ls mem*  
ls *x
```

```
ls *lax*  
ls .*  
ls mem?
```

```
ls mem?t  
ls mem[1-9]  
ls mem[*1-9]
```

16

别名

q 别名是 `bash` 中用来节省时间的另一项重要功能，它允许用户按照自己喜欢的方式对命令进行自定义。

q 别名的创建：创建别名的命令是 `alias`，例：

```
alias lf='ls -F'
```

注：等号两边不能有空格！

q 别名的取消：所设置的别名在当前 `shell` 中一直有效，直到退出当前 `shell` 或用 `unalias` 取消别名，例：

```
unalias lf
```

17

别名 (续)

u 查看已创建的别名：输入 `alias` 直接回车即可。

```
alias
```

u 如果需要一直使用某些别名，可以在 `bash` 启动脚本中添加设置别名的命令，这样每次打开一个（虚拟）终端时，系统就会自动设置别名。有关 `bash` 的启动脚本，我们将在后面讨论。

18

命令行自动补齐功能

q 命令行自动补齐功能

u 通常用户在 `bash` 下输入命令时不必把命令输全，`shell` 就能判断出你所要输入的命令。

u 该功能的核心思想是：`bash` 根据用户已输入的信息来查找以这些信息开头的命令，从而试图完成当前命令的输入工作。用来执行这项功能的键是 `Tab` 键，按下一次 `Tab` 键后，`bash` 就试图完成整个命令的输入，如果不成，可以再按一次 `Tab` 键，这时 `bash` 将列出所有能够与当前输入字符相匹配的命令列表。

19

命令行自动补齐功能

例：查看用户的命令历史

```
his<Tab>
```

u 这项功能同样适用于文件名的自动补齐

例：要进入目录：

```
/etc/sysconfig/network-scripts/
```

```
cd /e<Tab>sys<Tab>c<Tab>ne<Tab>-<Tab>
```

20

管道

q 管道

u UNIX 系统的一个基本哲学是：一连串的小命令能够解决大问题。其中每个小命令都能够很好地完成一项单一的工作。现在需要有一些东西能够将这简单的命令连接起来，这样管道就应运而生。

u 管道“|”的基本含义是：将前一个命令的输出作为后一个命令的输入。如：

```
ls /local | du -sh *
```

u 利用管道可以实现一些很强的功能。

21

管道举例

一个较复杂的例子：输出系统中用户名的一个排序列表。这里需要用到三个命令：`cat`、`awk`、`sort`，其中 `cat` 用来显示文件 `/etc/passwd` 的内容，`awk` 用来提取用户名，`sort` 用来排序。

```
cat /etc/passwd | \
awk -F: '{print $1}' | \
sort
```

22

重定向

q 数据流

u Linux 中的数据流有三种：标准输入 (STDIN)、标准输出 (STDOUT) 和标准错误 (STDERR)。

u 标准输入通常来自键盘，标准输出是命令的结果，通常定向到显示器，标准错误是错误信息，通常也定向到显示器。

```
[jypan@euler jypan]$ cat y
This is y!
[jypan@euler jypan]$ cat x
cat: x: No such file or directory
```

命令

标准输出

标准输入

标准错误

23

重定向

q 输入输出重定向

u 输入重定向：“<”

可以使用文件中的内容作为命令的输入。

u 输出重定向：“>”

允许将命令的输出结果保存到一个文件中。

```
ls > list
```

```
sort < list > sort_list
```

24

重定向

q 输入输出重定向

u 在使用输出重定向时，如果输出文件已经存在，则原文件中的内容将被删除。

u 如果希望保留原文件的内容，可以使用“>>”代替“>”，这样重定向输出的内容将添加到原文件的后面。

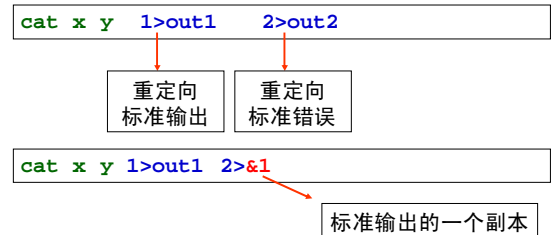
```
ls / > list
```

```
ls /home/ >> list
```

25

文件描述符

q shell 中进程处理文件时会建立一个文件描述符，标准文件描述符有三个：0、1、2，分别对应于标准输入、标准输出和标准错误。



26

命令历史记录

q 在命令行中输入的每个命令都被保存到一个称为 **history**（命令历史记录）的地方，在 **bash** 中，变量 **HISTSIZE** 用来指定存储在命令历史记录中的命令的最多个数。

q 查看命令历史记录：`history`

```
history
```

```
history 30
```

表示查看最近 30 个命令历史记录

27

命令历史记录

q 上下箭头键：除查看命令历史记录外，还可以利用上下箭头键在命令历史记录中移动。此外，还可以对所选的命令进行编辑。

q 感叹号的用法：

!!：执行最近一次使用的命令；

!n：其中 **n** 为一个具体的数字，表示执行在命令历史记录中的第 **n** 个命令；

!s：其中 **s** 为一个字符串，表示执行命令历史记录中以该字符串开头的最近的一个命令。

q `fc` 命令：

```
fc -l 30 50
```

列出命令历史记录中第30到第50之间的命令

28

引用

q 引用

u 在 **bash** 中，有些字符具有特殊含义，如果需要忽略这些字符的特殊含义，就必须使用引用技术。

u 引用可以通过下面三种方式实现：

- ü 使用转义字符：\
- ü 使用单引号：' '
- ü 使用双引号：" "

u 转义字符的引用方法就是直接在字符前加反斜杠

```
例：\$, \', \", \\, \, \!
```

29

引用

u 单引号对中的字符都将作为普通字符，但不允许出现另外的单引号。

u 双引号对中的部分字符仍保留特殊含义，如：`$`、`\`、```、`“`、及换行符等。

u 单引号是强引用，而双引号是弱引用。

30

Shell 中的特殊字符

u 在 **bash** 中，有些字符具有特殊含义，通常称为特殊字符。

字符	含义	字符	含义
\	强引用	*、?、!	通配符
"	弱引用	<、>、>>	重定向
\	转义字符	-	选项标志
\$	变量引用	#	注释符
;	命令分隔符	空格、换行符	命令分隔符
~	命令替换：反引号中的字符串被 shell 解释为命令，在执行时，shell 首先执行该命令，并以它的标准输出取代整个反引号（包括两个反引号）部分		

31

Shell 变量

q shell 变量大致可以分为三类：内部变量、用户变量和环境变量。

u 内部变量：由系统提供，用户不能修改。

u 用户变量：由用户建立和修改，在 shell 脚本编写中会经常用到。

u 环境变量：这些变量决定了用户工作的环境，它们不需要用户去定义，可以直接在 shell 中使用，其中某些变量用户可以修改。

32

常见的 Shell 变量

变量名	含义
HOME	用户主目录
LOGNAME	登录名
USER	用户名, 与登录名相同
PWD	当前目录/工作目录名
MAIL	用户的邮箱路径名
HOSTNAME	计算机的主机名
INPUTRC	默认的键盘映像
SHELL	用户所使用的 shell 的路径名
LANG	默认语言
HISTSIZE	history 所能记住的命令的最多个数
PATH	shell 查找用户输入命令的路径 (目录列表)
PS1、PS2	shell 一级、二级命令提示符

33

Shell 变量

u **PATH 变量**是最重要的环境变量之一。当用户在命令行中输入命令时, **shell** 就会根据该变量定义的路径 (目录) 和顺序, 查找并执行该命令。如果没有正确设置 **PATH** 变量, 则必须输入完整的路径名来运行某个命令。

u 在 Linux 下输入命令的两种方式:

- ü 直接在命令行中输入命令: 根据 **PATH** 查找该命令
- ü 输入完整的路径名

u 用户可以根据需要修改环境变量

如: **HISTSIZE**, **PATH**, **PS1**, **PS2** 等

34

Shell 变量查询

u 查询当前 **shell** 中的环境变量: **env**

```
env
```

u 查询某个变量的值: **echo**

```
echo ${变量名}
```

35

命令提示符

u 在 **bash** 中, 有两个级别的命令输入提示:

ü 一级提示符是当 **bash** 等待输入命令时所出现的提示符, 由环境变量 **PS1** 控制, 缺省值为 “\$”;

ü 二级提示符是在 **bash** 执行一个命令后, 需要用户进一步输入才能完成次命令时, 所出现的提示符, 由环境变量 **PS2** 控制, 缺省值为 “>”。

u 重设 **PS1** 和 **PS2** 的设置

```
export
```

使变量的值对当前 **shell** 及其所有子进程都可见

```
例: export PS1="\t\w\$"
```

36

命令提示符

u 在创建提示符时, 可以使用下面的特殊字符:

\!	显示命令的历史编号	\h	显示机器的主机名
\#	显示命令的命令编号	\s	显示当前使用的 shell
\\	显示一个反斜杠	\u	显示用户名
\n	显示一个换行符	\W	显示当前目录名
\d	显示当前的日期	\w	显示当前目录完整路径名
\t	显示当前的时间		
\\$	普通用户显示“\$”, 超级用户显示“#”	\nnn	显示与八进制 nnn 相对应的字符

37

bash 配置文件

q bash 配置文件

u 在命令行中设置和修改的变量值, 只在当前的 **shell** 中有效。一旦用户退出 **bash**, 所做的一切改变都会丢失。

u 在启动交互式会话过程中, 在出现提示符前, 系统会读取几个配置文件, 并执行这些文件中的命令。所以这些文件可以用来定制 **bash** 环境。如: 设置 **shell** 变量值或建立别名等。

u **bash** 配置文件:

```
/etc/profile
```

```
~/.bash_profile  
~/.bash_login  
~/.profile
```

```
~/.bashrc
```

38

bash 配置文件

u **/etc/profile**

Linux 系统中的全局 **bash** 启动脚本, 任何用户登录系统时 **/etc/profile** 都会被执行。通常用来设置标准 **bash** 环境, 但修改该文件需 **root** 权限。

u 读取 **/etc/profile** 文件后, **bash** 将在用户主目录中按顺序查找以下文件, 并执行第一个找到的文件:

```
~/.bash_profile  
~/.bash_login  
~/.profile
```

在这些文件中, 用户可以定义自己的环境变量, 而且能够覆盖在 **/etc/profile** 中定义的设置。

39

bash 配置文件

u **bash** 启动后, 将读入配置文件 **~/.bashrc**, 并执行这个文件中的所有内容。

u 另外, 还可以从另一个 **shell** 或者 **bash** 自身启动一个新的 **bash**, 这种过程称为 **非登录交互式**, 启动新 **bash** 的命令为 **bash**, 此时所读入的唯一 **bash** 配置文件是

```
~/.bashrc
```

通常, 个人 **bash** 环境设置都定义在 **~/.bashrc** 文件里

40