GNU/Linux Shell

GNU/Linux Shell

Shell

是 Linux 的一个外壳，它包在 Linux 内核的外面，为用户和内核之间的交互提供了一个接口。当用户下达指令给操作系统的时候，实际上是把指令告诉 shell ，经过 shell 解释，处理后让内核作出相应的动作。而系统的回应和输出的信息也由 shell 处理，然后显示在用户的屏幕上 .

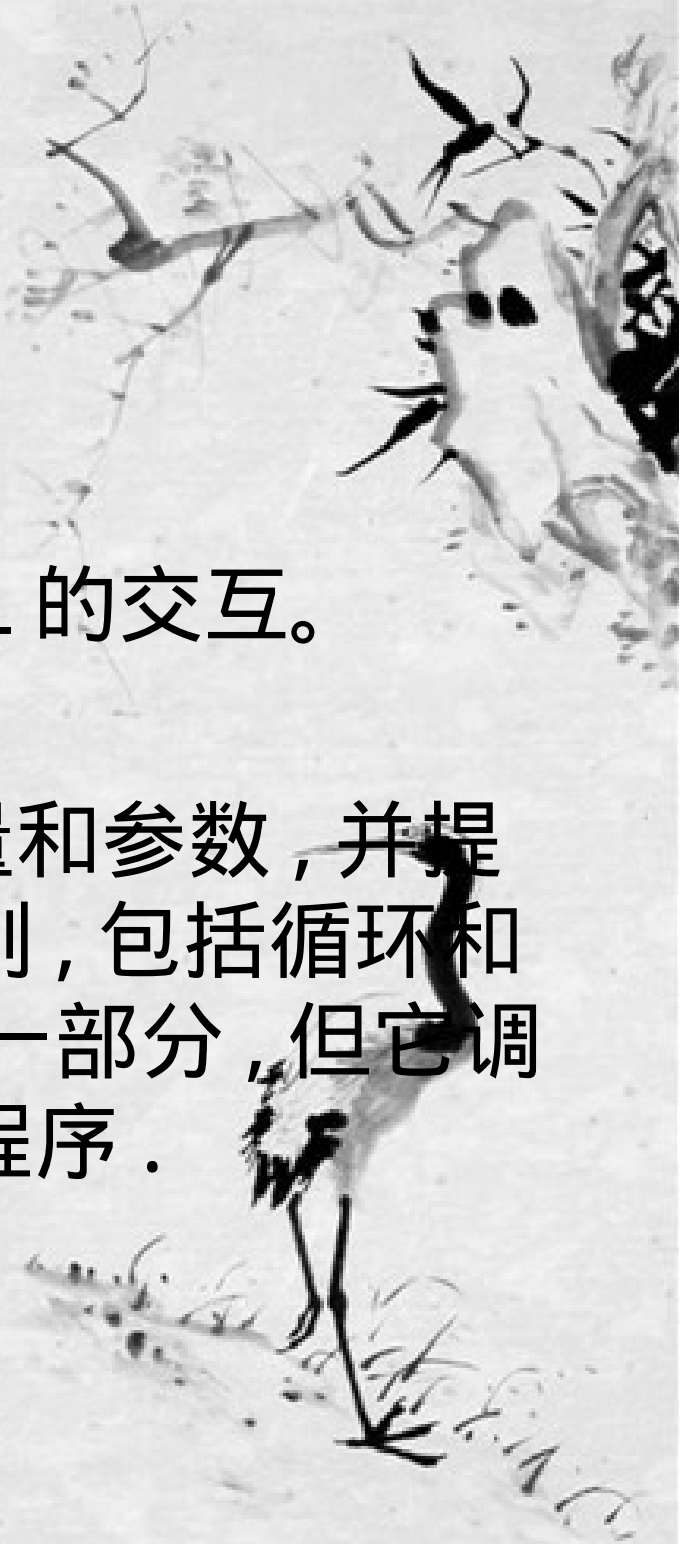
是一个强大的界面和伟大的脚本环境，称为命令行的解释器。

GNU/Linux Shell

Shell 遵循一定的语法，将输入的命令加以解释后传给系统， shell 为用户提供一个向系统发送请求以便运行程序界面的接口，用户可以用 shell 来编写一些程序或者脚本。

Shell 是用 C 语言编写的程序，它是用户和 linux 沟通的桥梁 .

GNU/Linux Shell

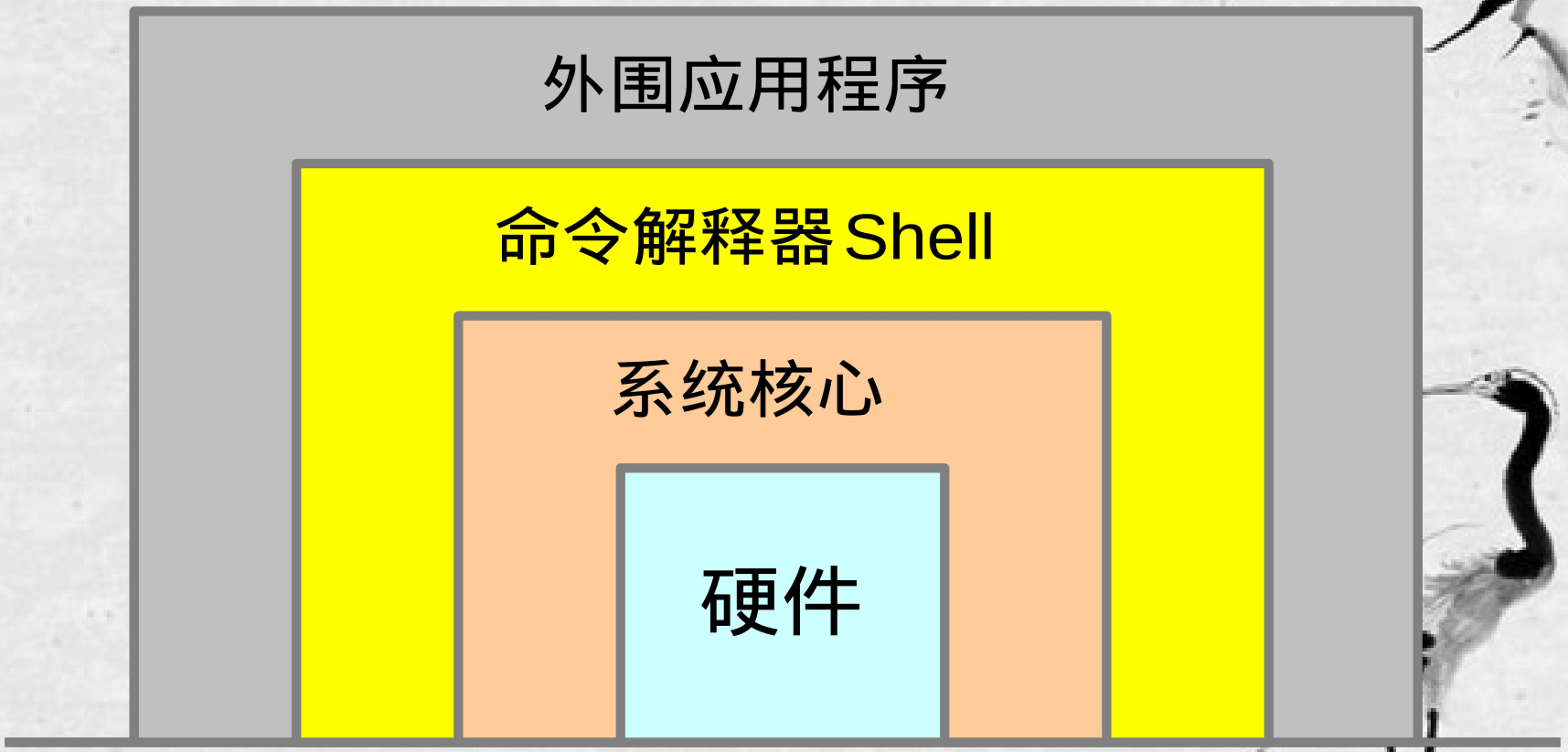


Shell 特点

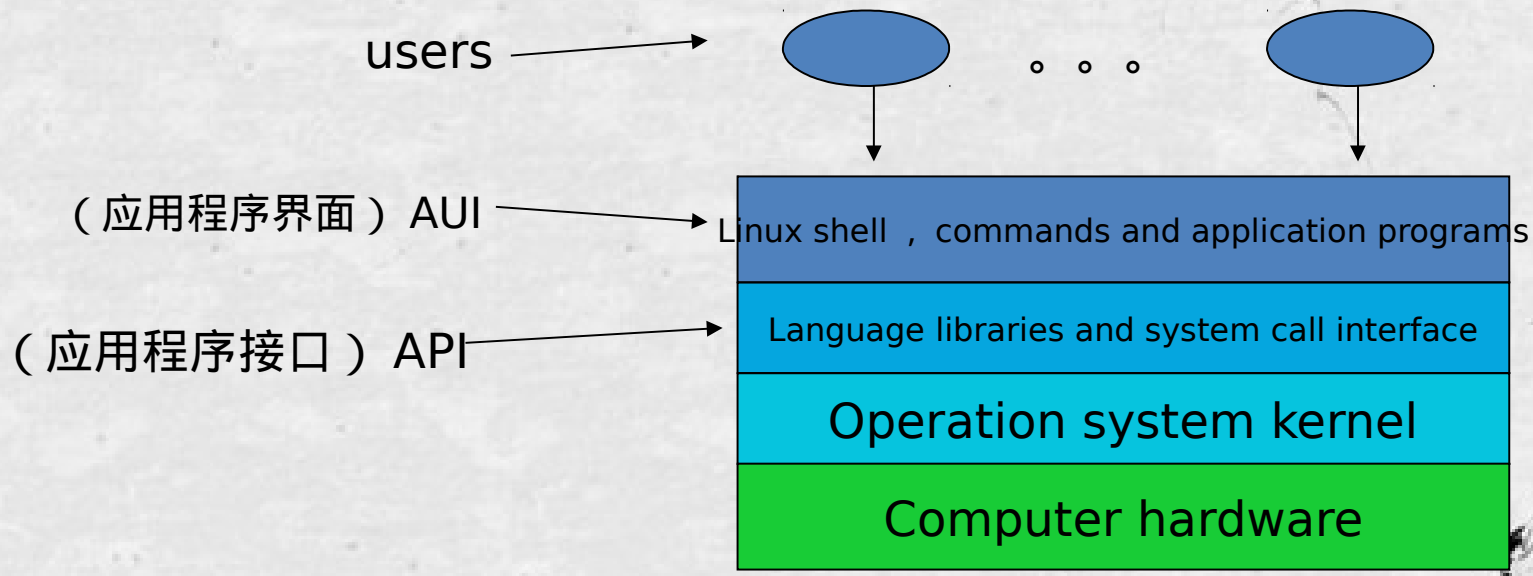
1. 作为命令语言，提供用户和 SHELL 的交互。

2. 作为程序设计语言，提供各种变量和参数，并提供了在高级语言中才具有的结构控制，包括循环和分支。 shell 虽然不是 linux 系统的一部分，但它调用了系统核心的大部分功能来执行程序 .

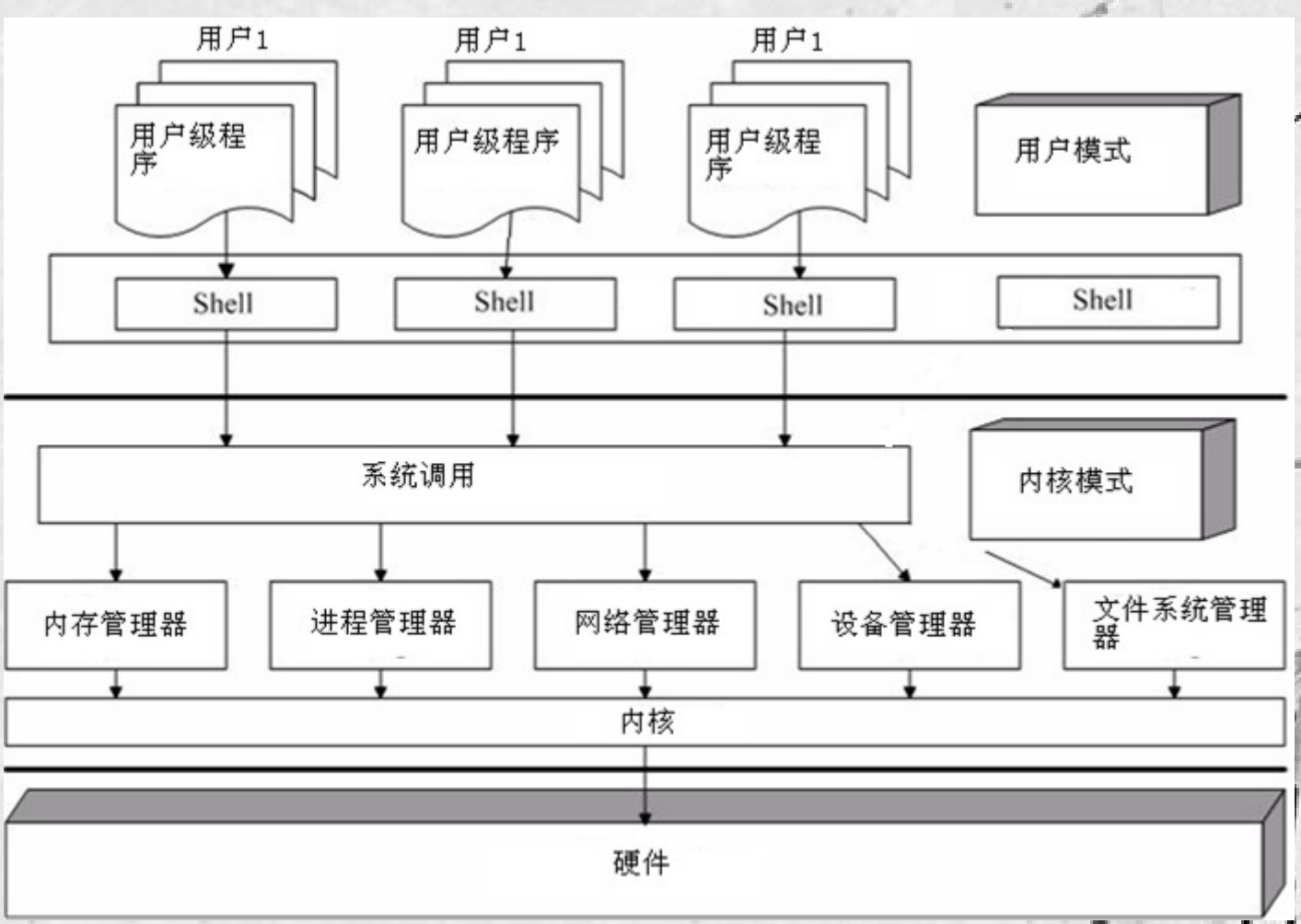
GNU/Linux Shell



GNU/Linux Shell



GNU/Linux Shell



GNU/Linux Shell

Shell 执行命令动作：

在每个程序被打开时，每个进程都和三个文件相关联，并使用文件描述符来引用这些文件 .

文件描述符：

输入文件：标准输入 (stdin, 代码 0).

输出文件：标准输出 (stdout, 代码 1).

错误输出文件：标准错误 (stderr, 代码 2).

GNU/Linux Shell

stdin 所使用的符号

<

stdout 所使用的符号

> >>

stderr

2

代表 stdout 及 stderr 符号

&

GNU/Linux Shell

示例：

1. 将文件 a.txt 文件内容的输出至 b.txt

```
#cat  a.txt  >  b.txt
```

2. 将文件 a.txt 文件内容的输出追加至 b.txt

```
#cat  a.txt  >>  b.txt
```

3. 如产生错误则记录至 err.txt

```
#cat  a.txt  2>err.txt
```

GNU/Linux Shell

示例：

4. 忽略错误信息

```
#cat  a.txt  2>/dev/null
```

5. 将标准输出结果或 / 和错误信息输出到 info.txt

```
#cat  a.txt  &>info.txt
```

6. 将标准输出结果或 / 和错误信息追加至 info.txt

```
#cat  a.txt  >>  info.txt  2>&1
```

或

```
#cat  a.txt  &>>info.txt
```

GNU/Linux Shell

示例：

7. 将标准输入存放至 info.txt, 将错误存放至 err.txt

#find /etc -name passwd > /tmp/output 2> /tmp/err.txt

8. 将标准输入存放至 info.txt, 忽略错误

#find /etc -name passwd > /tmp/output 2> /dev/null

GNU/Linux Shell

Shell 特点

即使一种命令语言又是一种程序设计语言

1. 作为命令语言，提供用户和 SHELL 的交互。

2. 作为程序设计语言，提供各种变量和参数，并提供了在高级语言中才具有的结构控制，包括循环和分支。 shell 虽然不是 linux 系统的一部分，但它调用了系统核心的大部分功能来执行程序。

GNU/Linux Shell

Shell 特点

3. Shell 本身有些命令就包含在自身当中，一般称之为内部命令。

GNU/Linux Shell

Shell 的种类

ash 是由 Kenneth Almquist 编写的，是 Linux 中占用系统资源最少的一个小 Shell，它只包含 24 个内部命令，因而使用起来很不方便

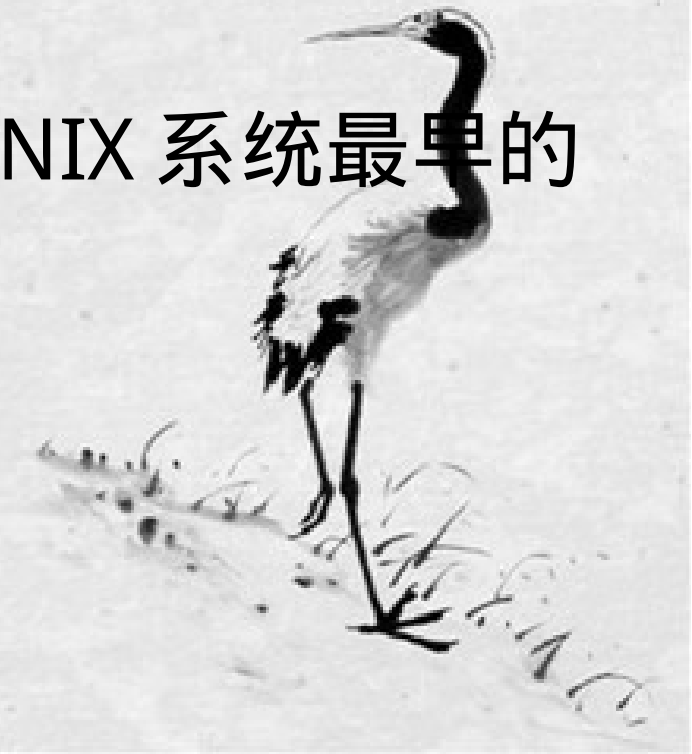
ksh 是 Korn shell 的缩写，由 Eric Gisin 编写，共有 42 条内部命令

csh 是 Linux 比较大的内核，它由以 William Joy 为代表的共计 47 位作者编成，共有 52 个内部命令。该 Shell 其实是指向 /bin/tcsh 的，也就是说，csh 其实就是 tcsh

GNU/Linux Shell

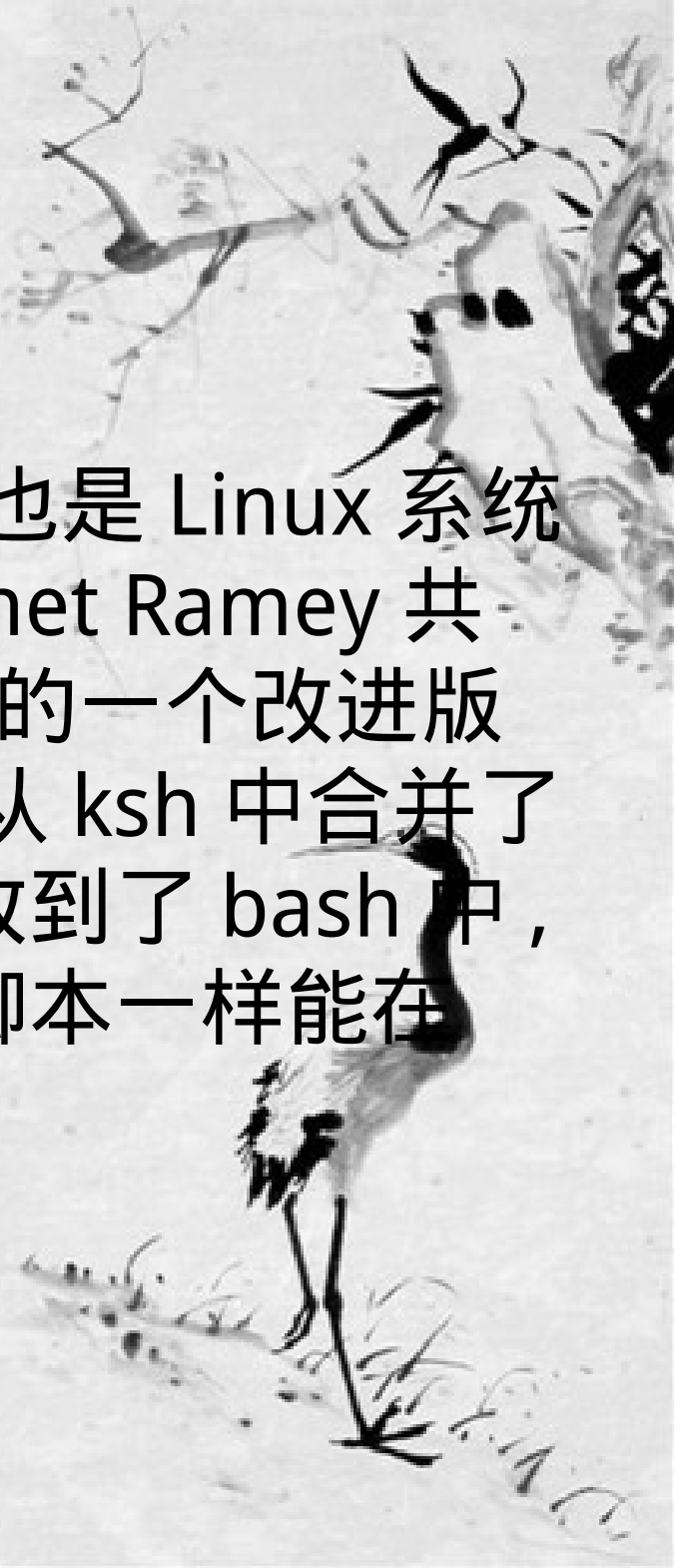
Shell 的种类

zsh 是 Linux 最大的 Shell 之一，由 Paul Falstad 完成，共有 84 个内部命令。如果只是一般的用途，是没有必要安装这样的 Shell 的。



sh 就是 Bourne Shell 的缩写，是 UNIX 系统最早的 shell 。

GNU/Linux Shell



Shell 的种类

bash 是 Bourne Again Shell 的缩写也是 Linux 系统默认使用的 shell, 由 Brian Fox 和 Chet Ramey 共同完成 . 提供 40 个内部命令 . 是 sh 的一个改进版本 , 兼容所有 sh 下运行的脚本，还从 ksh 中合并了许多特性也将 csh 中的一些特性吸收到了 bash 中 , 这也意味着在 csh 和 ksh 上编写的脚本一样能在 bash 环境下很好的运行 .

GNU/Linux Shell

bash 特点：

1. 可以使用类似 DOS 下面 doskey 的功能，用上下方向键查阅和快速输入并修改命令。

2. 自动通过查找匹配的方式，给出以某字串开头的命令。

3. 包含了自身的帮助功能，你只要在提示符下面键入 help 就可以得到相关的帮助。

GNU/Linux Shell

bash 种类

1. login shell

2. normal shell

3. interactive shell

GNU/Linux Shell

bash 的控制文件

1. /etc/profile

　　此文件为系统的每个用户设置环境信息，当用户第一次登录时，该文件被执行．并从 /etc/profile.d 目录的配置文件中搜集 shell 的设置．

　　即：用户工作环境设定在此文件中进行设置

GNU/Linux Shell

bash 的控制文件

2. /etc/bashrc

为每一个运行 bash shell 的用户执行此文件 . 当 bash shell 被打开时 , 该文件被读取 . 设定 bash shell 的环境信息

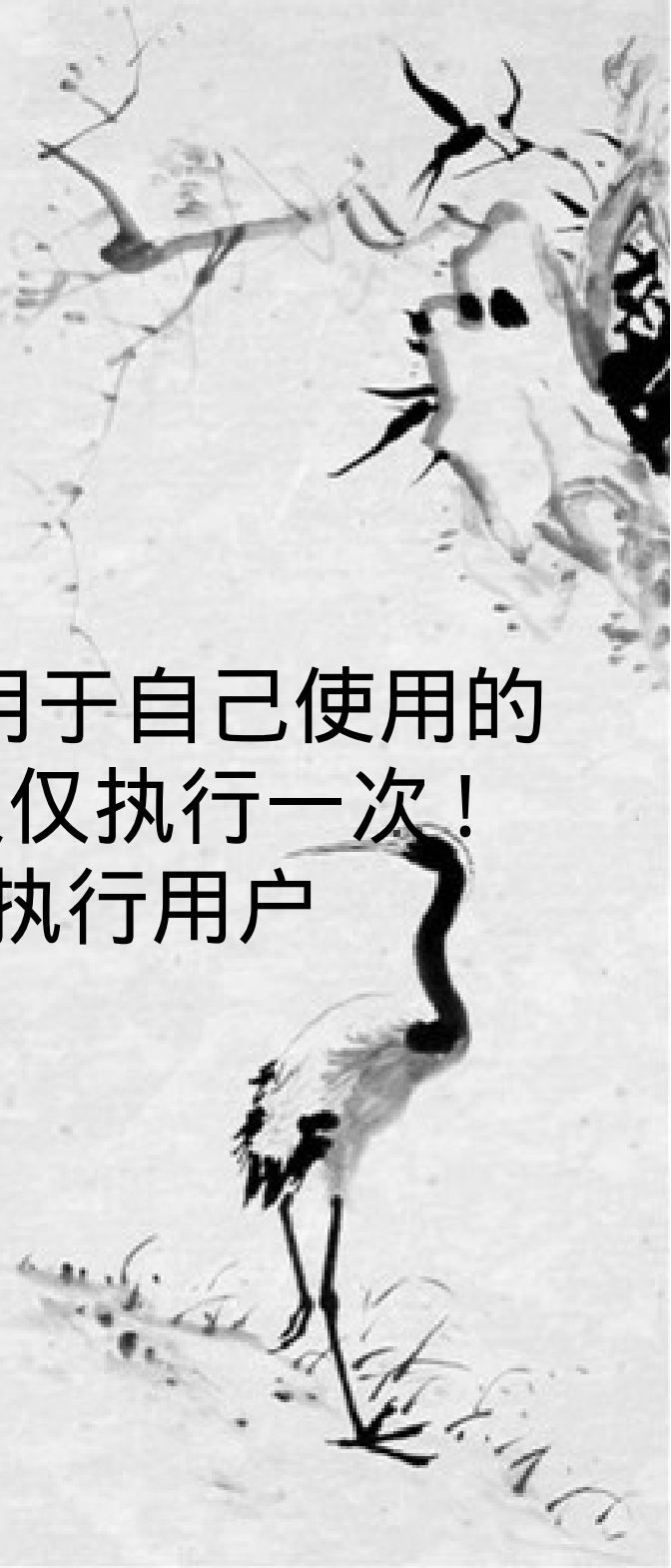
换句话说即 : 设定系统方面的功能 , 函数和别名

GNU/Linux Shell

bash 的控制文件

3. ~/.bash_profile

每个用户都可使用该文件输入专用于自己使用的 shell 信息，当用户登录时，该文件仅仅执行一次！默认情况下，他设置一些环境变量，执行用户的 .bashrc 文件 .



GNU/Linux Shell

bash 的控制文件

4. ~/.bashrc

该文件包含专用于你的 bash shell 的 bash 信息，当登录时以及每次打开新的 shell 时，该文件被读取 .

GNU/Linux Shell

bash 的控制文件

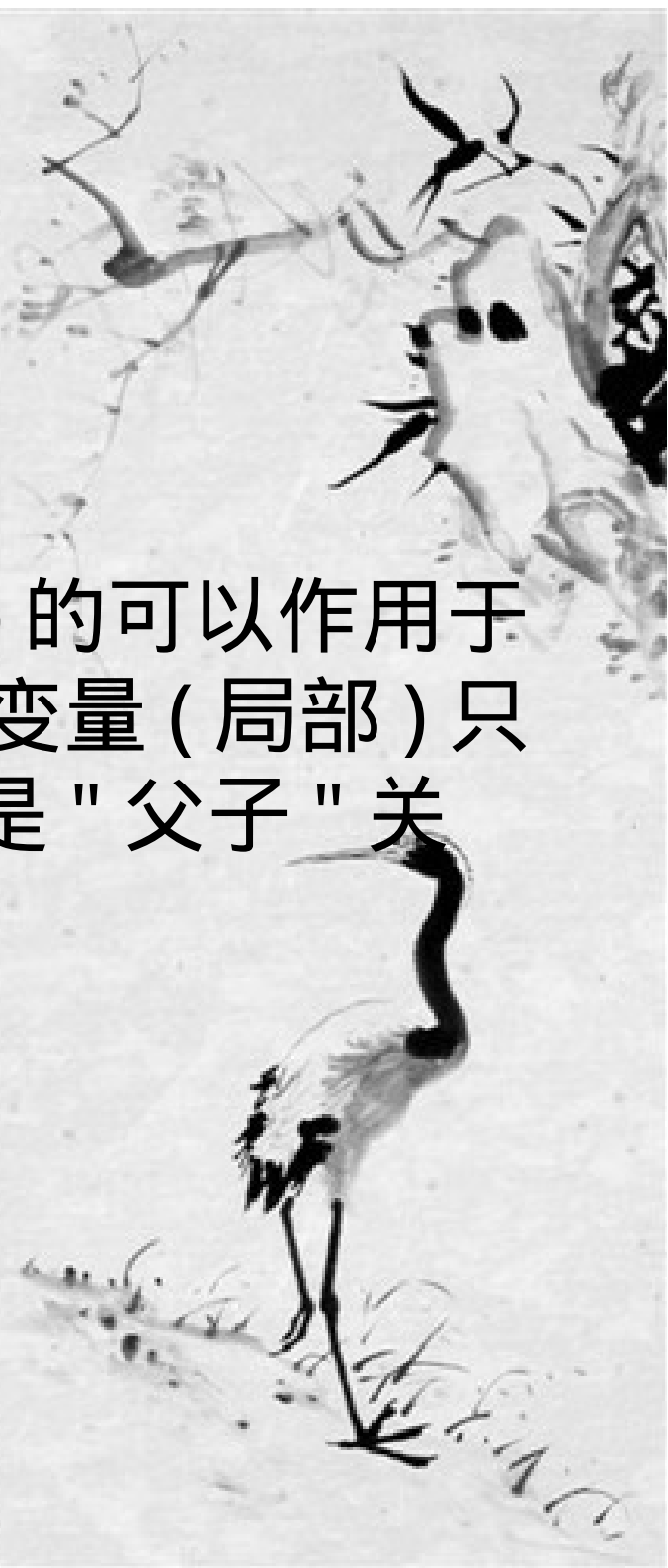
5. ~/.bash_logout

每次退出系统 (退出 bash shell) 时 , 执行该文件 .

GNU/Linux Shell

bash 的控制文件

/etc/profile 中设定的变量 (全局) 的可以作用于任何用户 , 而 ~/.bashrc 等中设定的变量 (局部) 只能继承 /etc/profile 中的变量 , 他们是 " 父子 " 关系 .



GNU/Linux Shell

bash 的控制文件

~/.bash_profile 是交互式、 login 方式进入 bash 运行的

~/.bashrc 是交互式 non-login 方式进入 bash 运行的

通常二者设置大致相同，所以通常前者会调用后者。

GNU/Linux Shell

bash 的环境：

1. 环境变量：

对环境的设置可以通过给各种环境变量赋值来实现．每一种环境变量控制了一项工作环境的设置．

GNU/Linux Shell

bash 的环境 :

2. 查看所有的环境变量 :
   #set

3. 查看各选项的功能是否开启
   #set -o

4. 查看系统环境变量
   #env

GNU/Linux Shell

bash 变量设置

1. 设置变量应注意以下几点

    1) 系统变量一般都是大写

    2) 用户自定义变量一般为小写

    3) 可以用任何数字，任何字母及下划线的组合．但首字符必需是下划线或字母

GNU/Linux Shell

bash 变量设置

2. 变量设定

1) 声明本级 shell 的变量

```
#LANG=zh_CN.UTF-8
#localectl  set-locale LANG=zh_CN.utf8
```

2) 声明全局 shell 变量 (向子 shell 传递此变量)

```
#export  LC_CTYPE=zh_CN.UTF8
```

3) 取消 shell 变量

```
#unset  linux
```

GNU/Linux Shell

bash 变量设置

3. 显示变量设定
   1) 显示变量的值
      #echo $SHELL

GNU/Linux Shell

bash 变量设置

4. 常用 shell 环境说明
   #env

GNU/Linux Shell

bash 变量设置

环境变量	说明
LOGNAME	登陆名，也就是账户名
PATH	命令搜索路径
PS1	命令提示符
PWD	用户的当前目录
SHELL	用户的 shell 类型
TERM	终端类型
HOME	用户主目录的位置，通常是 /home/ 用户名

GNU/Linux Shell

bash 变量设置

5. PS 提示符
   1)PS1/PS2 提示符
   实际是 shell 变量之一

   2) 查看 PS1 提示符
   #set | grep PS1

   3) 查看 PS2 提示符
   #set | grep PS2

GNU/Linux Shell

bash 变量设置

6. PS1 提示符常用变量值说明

\d 今天日期
\h 用户的主机名
\H 用户系统完全符合规范的主机名
\s 用户 shell 名称
\t HH:MM:SS 24 小时制
\T HH:MM:SS 12 小时制

GNU/Linux Shell

bash 变量设置

6. PS1 提示符常用变量值说明

\u 用户的用户名

\v bash 版本

\w 用户当前工作目录的绝对路径

\W 用户当前工作目录的相对路径

\! bash 命令历史记录中的编号

\@ AM 或 PM

\$ $ 提示符 , 如果是 root 用户则显示 # 提示符

GNU/Linux Shell

bash 变量设置

7. 开启子 shell
   #bash

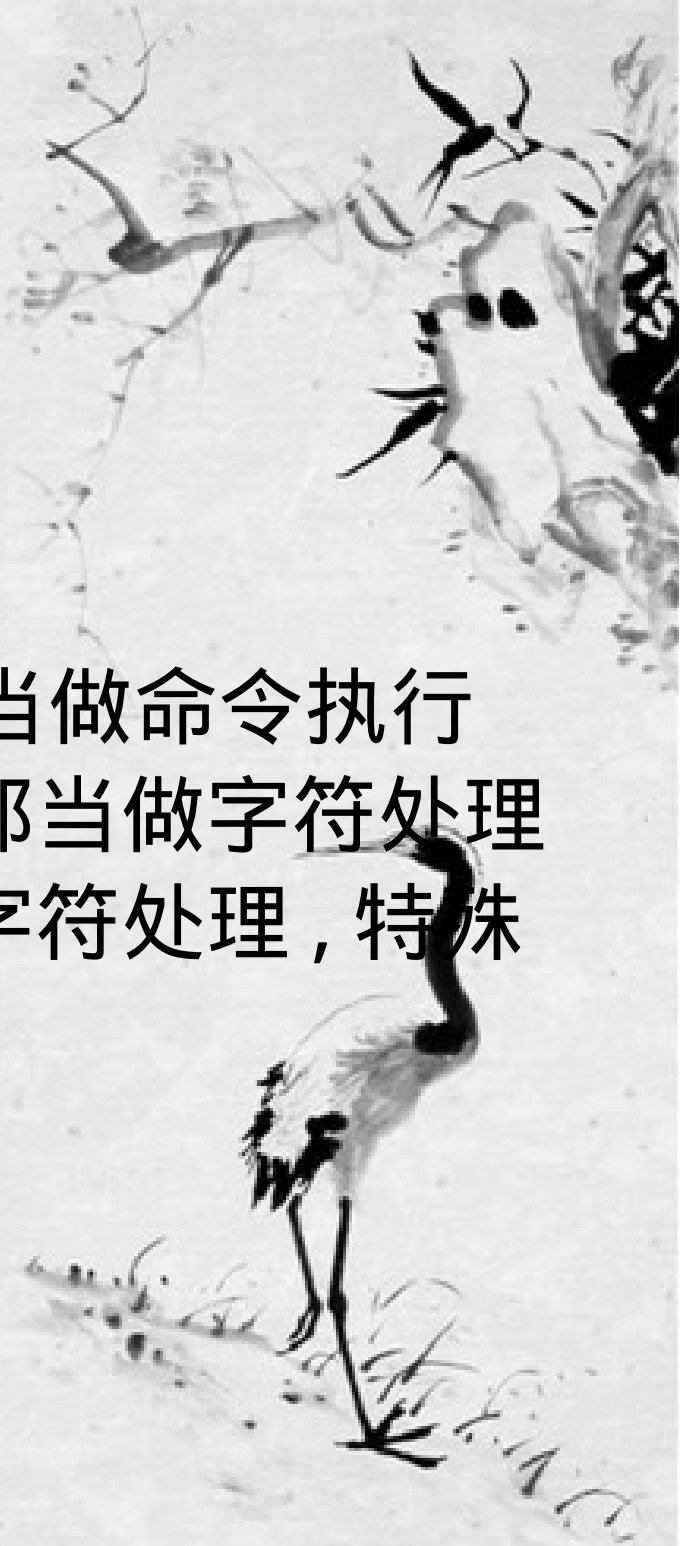
8. 退出子 shell
   #exit

GNU/Linux Shell

bash 特殊变量

$*	所有位置的参数内容
$#	位置参数的数量
$$	当前程序的 PID
$!	后台运行的最后一个 PID 号
$?	命令执行后返回的状态
$0	当前执行的进程名

GNU/Linux Shell



SHELL 特殊符号

特殊字符：

\ 转义符

` 反引号，里面的内容将被当做命令执行

' 单引号，里面的内容将全部当做字符处理

“ 双引号，普通字符按普通字符处理，特殊字符按特殊字符处理

| 管道

GNU/Linux Shell

SHELL 特殊符号

特殊字符：

& 后台执行

$ 变量替代值

< 输入重定向

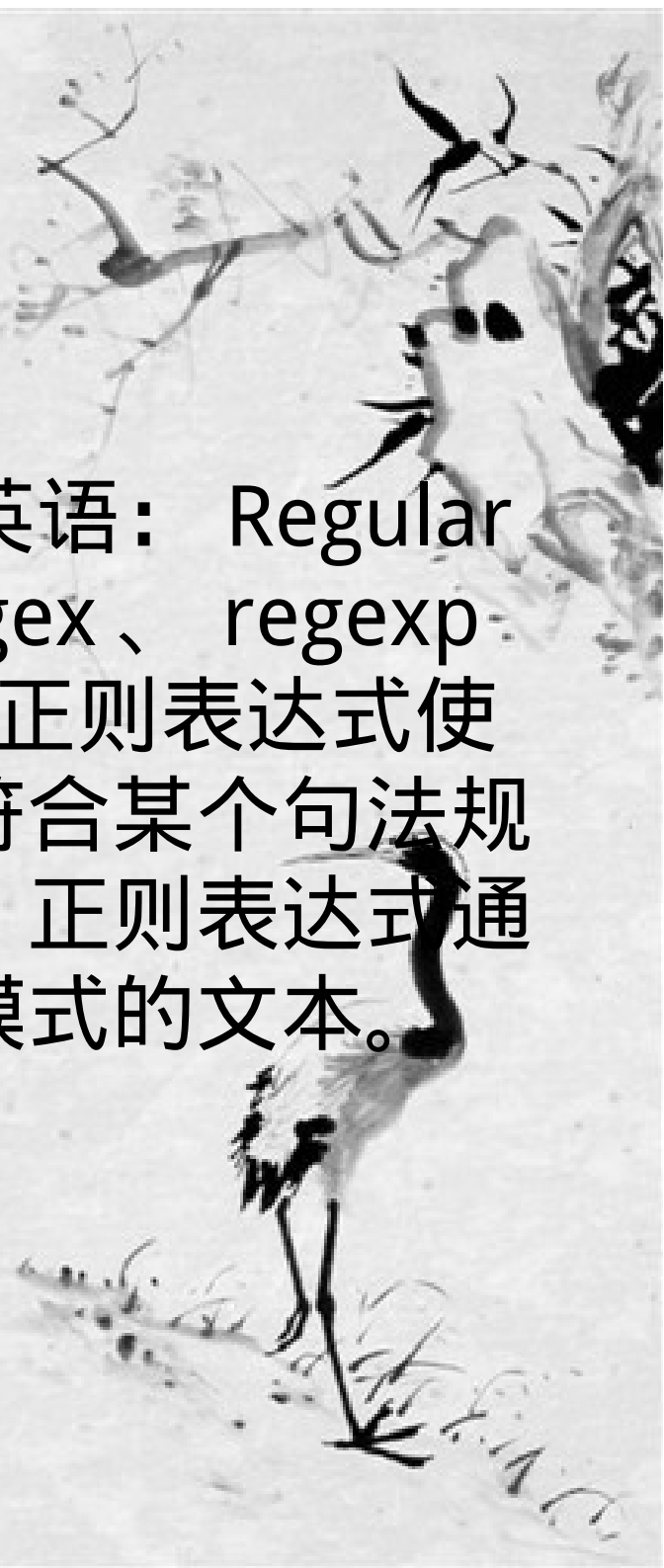
> 输出重定向

>> 输出追加重定向

GNU/Linux Shell

正则表达式

又称正规表示法、常规表示法（英语：Regular Expression，在代码中常简写为 regex、regexp 或 RE），计算机科学的一个概念。正则表达式使用单个字符串来描述、匹配一系列符合某个句法规则的字符串。在很多文本编辑器里，正则表达式通常被用来检索、替换那些符合某个模式的文本。



GNU/Linux Shell

正则表达式

正则表达式是由普通字符（例如字符 a 到 z）以及特殊字符（称为元字符）组成的文字模式。 正则表达式作为一个模板，将某个字符模式与所搜索的字符串进行匹配 .

GNU/Linux Shell

正则表达式

普通字符：

大写和小写字母字符，所有数字，所有标点符号以及一些符号 . 及非打印字符：

\f 匹配一个换页符

\n 匹配一个换行符

\r 匹配一个回车符

GNU/Linux Shell

正则表达式

普通字符：

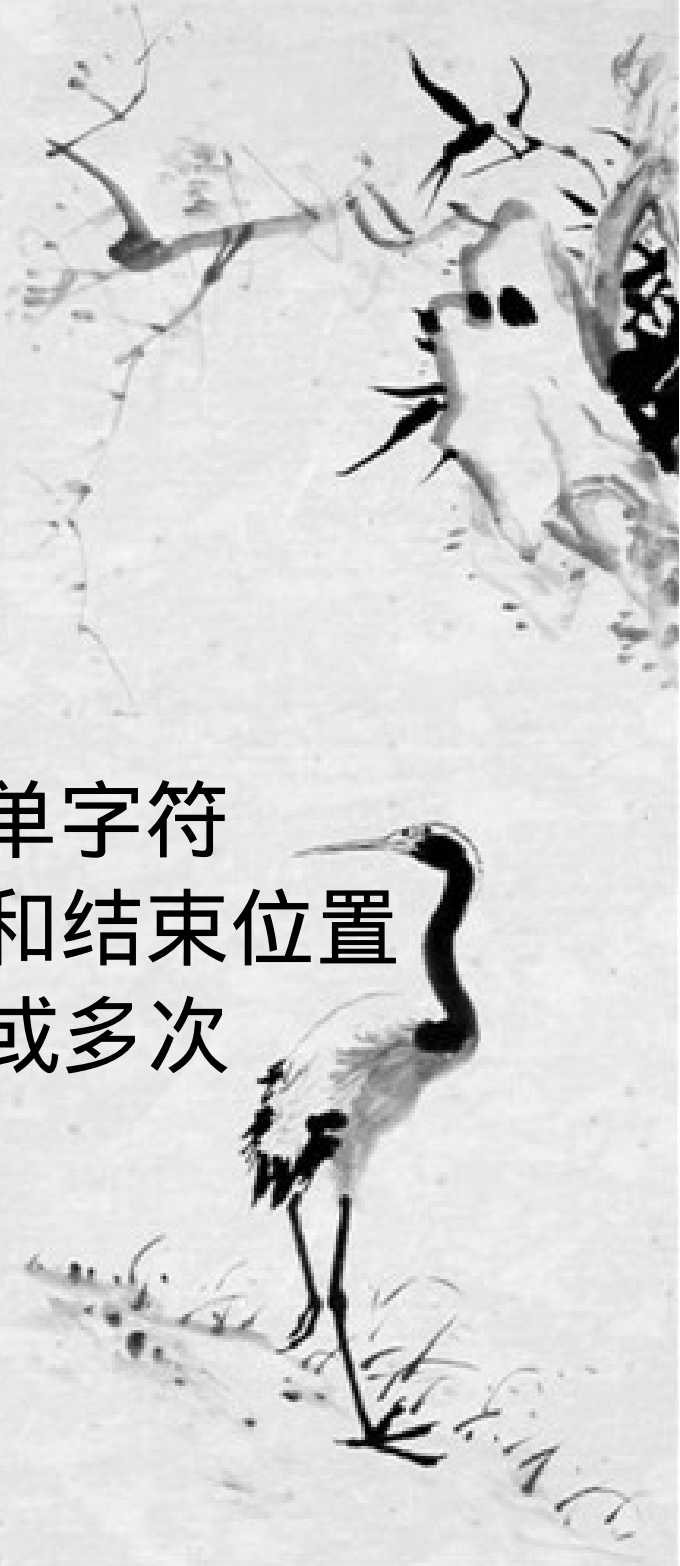
大写和小写字母字符，所有数字，所有标点符号以及一些符号，非打印字符

\s 匹配任何空白字符，包括空格、制表符、换页符等等

\S 匹配任何非空白字符

\t 匹配一个制表符

GNU/Linux Shell



正则表达式

特殊字符：

$	匹配字符串的结尾位置
^	匹配字符串的开始位置
.	匹配除换行符以外的所有单字符
( )	表示一个子表达式的开始和结束位置
*	匹配前面的子表达式 0 次或多次

GNU/Linux Shell

正则表达式

特殊字符：

+ 匹配前面的子表达式 1 次或多次

? 匹配前面的子表达式 0 次或一次

[表达式的开始

] 表达式的结尾

| “或”运算

GNU/Linux Shell

正则表达式

举例

1. dg+

   可匹配 dg 及 dgg

2.dg?

   可匹配 dga,dgb 及 dgc 等

3. dg*

   可匹配 dg 及 dga,dgaaa 等

GNU/Linux Shell

正则表达式

举例

4. {n} n 为一个非负数，匹配确定的字符 n 次

   例如：dg{2} 不能匹配 dg，能匹配 dgg

5. {n,} n 为一个非负数，能匹配确定的字符至少 n 次

   例如：dg{2,} 能匹配 dgg dggg dgggg 等

GNU/Linux Shell

正则表达式

举例

6. {n,m} n 和 m 都为一个非负数 n<m , 能匹配确定的字符至少 n 次但最多不超过 m 次 , 逗号之间不能有空格

例如 : d{2,4} 能匹配 dd,ddd,dddd

GNU/Linux Shell

正则表达式

示例：

1. 显示包含 Sunday,Monday,Tuesday,Wednesday}.log 的字串

#echo {Sunday,Monday,Tuesday,Wednesday}.log

2. 显示以 file 开头，第 5 个字符由 1 至 5 且后缀为 .txt 的字串

#echo file{1..5}.txt

3. 显示以 file 开头，第 5,6 个字符为 a1,a2,b1,b2 且后缀为 .txt 的字串

#echo file{a,b}{1,2}.txt

GNU/Linux Shell

正则表达式

示例：

4. 显示以 file 开头，第 5,6 个字符为 a1,a2 及第 5 个字符为 b,c 且后缀为 .txt 的字串

#echo file{a{1,2},b,c}.txt

GNU/Linux Shell

正则表达式

操作的优先级

相同的优先级，从左到右进行运算。

\	转义（最高）
(),[]	括号
*,+,?,{n,m}	特殊字符
^,$	定位字符
|	“或”操作（最低）

正则表达式是一个非常有用的工具，很多程序语言里都提供对正则表达式的支持。

GNU/Linux Shell

追寻历史命令

1. #history

操作方法

!n　n 为 history 中第 N 条指令

!!　执行最后一次所执行的命令

!str　执行最后以 str 开头的命令

2. 在 CLI 模式下按 ^R

3. 键盘的”↑” ”↓”箭头翻滚

GNU/Linux Shell

字符操作快捷键

1. 将光标移动到当前命令行头部
   #^a

2. 将光标移动到当前命令的尾部
   #^e

3. 清除光标前至头部的字符
   #^u

GNU/Linux Shell

字符操作快捷键

4. 清除光标前至尾部的字符
   #^k

5. 将光标向左移动一个单词
   #^←

6. 将光标向右移动一个单词
   #^→

GNU/Linux Shell

命令执行分隔符

1. “;”

   无论前面命令执行是否成功，都执行后面的

2.”&&”

   前面命令执行成功，在执行后面的命令

3. “||”

   前面命令执行失败，在执行后面的命令

GNU/Linux Shell

命令执行分隔符优先级

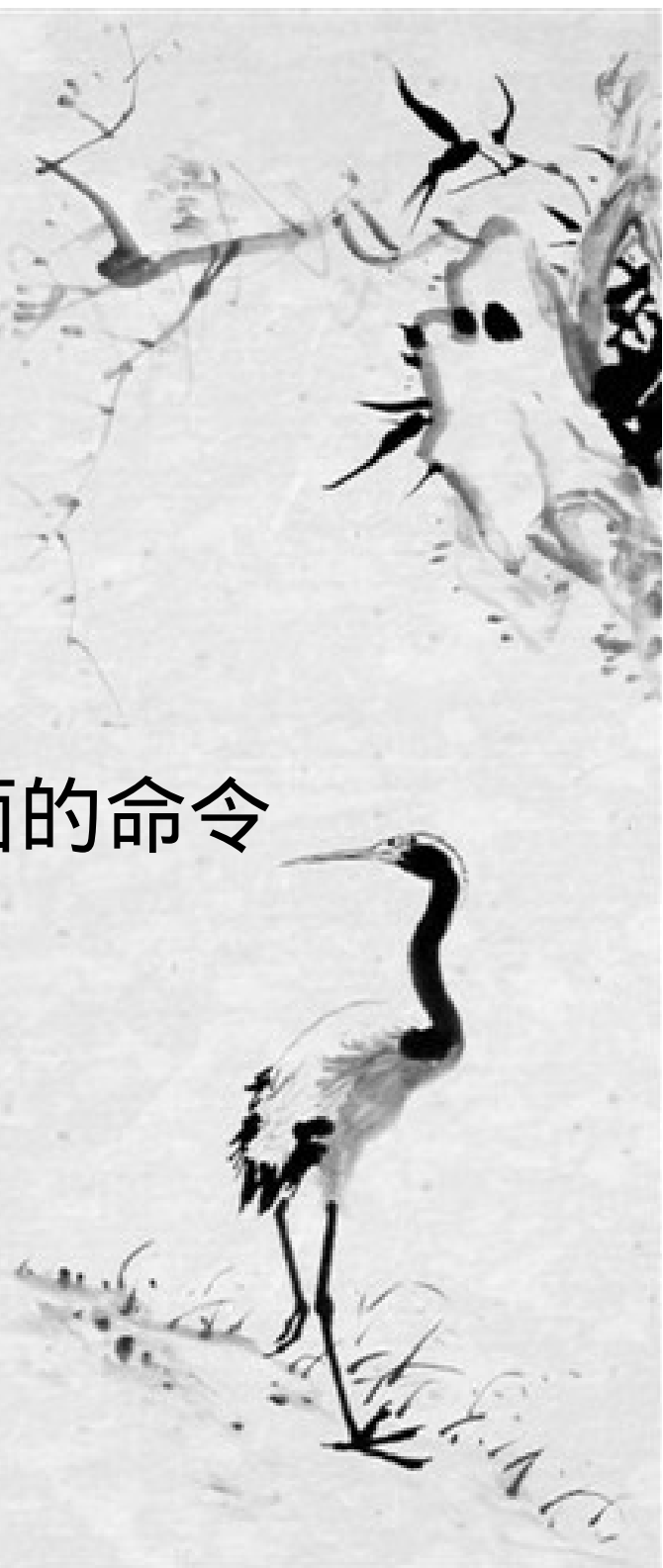
1. “;” 最低

2.”&&” 与” ||” 同级

前面命令执行成功，在执行后面的命令

3. 同级按从左到右分别执行

4. 如同级在在 () 中，则被优先执行



GNU/Linux Shell

shell 别名

别名的作用是让用户自定义新的命令名称来替代原有的命令。

1. 定义别名

```
#alias  xianshi='ls  --color  -F'
```

2. 取消别名

```
#unalias  xianshi
```

GNU/Linux Shell

定义个性化环境、别名方法

将所定义的环境变量、别名按照类别，可加入至：

1. ~/.bash_profile

2. ~/.bashrc

3. ~/.bash_logout

GNU/Linux Shell

TTY 切换与确认

1. 切换 TTY 终端
   GUI->CLI
   Ctrl+Alt+F2—Ctrl+Alt+F6

2. 切换 TTY 终端
   CLI->CLI
   Alt+F2--Alt+F6

GNU/Linux Shell

TTY 切换与确认

3. 切换 TTY 终端
   CLI->GUI
   Alt+F1

4. TTY 说明
   tty1--GUI
   tty2-tty6-- 虚拟终端

GNU/Linux Shell

TTY 切换与确认

5. 确认当前的 tty 终端

   #tty

GNU/Linux Shell

配置虚拟设置

#vim /etc/vconsole.conf

此文件可以设定键盘类型及字体

键盘设定

#localectl set-keymap us