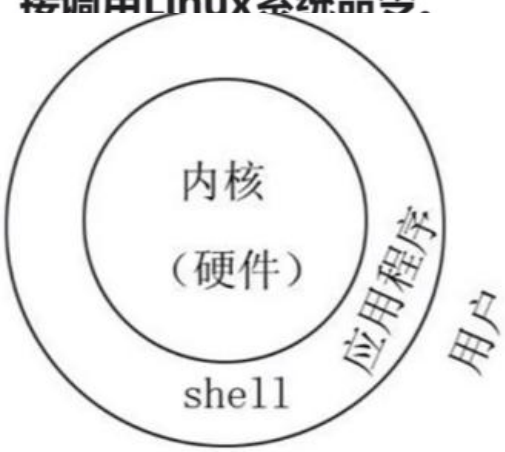
shell编程

1、Shell是什么

- Shell是一个命令行解释器，它为用户提供了一个向Linux内核发送请求以便运行程序的界面系统级程序，用户可以用Shell来启动、挂起、停止甚至是编写一些程序。
- Shell还是一个功能相当强大的编程语言，易编写，易调试，灵活性较强。Shell是解释执行的脚本语言，在Shell中可以直接调用Linux系统命令。



2、Shell的分类

- Bourne Shell：从1979起Unix就开始使用Bourne Shell，Bourne Shell的主文件名为sh。
- C Shell： C Shell主要在BSD版的Unix系统中使用，其语法和C语言相类似而得名

Shell的两种主要语法类型有Bourne和C，这两种语法彼此不兼容。Bourne家族主要包括sh、ksh、Bash、psh、zsh；C家族主要包括：csh、tcsh

- Bash： Bash与sh兼容，现在使用的Linux就是使用Bash作为用户的基本Shell。

1.echo输出命令
echo [选项] [输出内容]
-e 支持控制字符
\e[1;31m 开启颜色显示
\e[0m 取消颜色
例：echo -e "\e[1;31m 你好 \e[0m"
#Linux不支持中文，远程工具可以支持

控制字符	作　用
\a	输出警告音
\b	退格键，也就是向左删除键
\n	换行符
\r	回车键
\t	制表符，也就是Tab键
\v	垂直制表符
\0nnn	按照八进制ASCII码表输出字符。其中0为数字零，nnn是三位八进制数
\xhh	按照十六进制ASCII码表输出字符。其中hh是两位十六进制数

2.第一个脚本
vim hello.sh
##############
#!/bin/bash
echo -e "\e[1;31mHello World\e[0m"
##############

#警号在Shell中是注释

3.脚本执行
1)赋予指向权限，直接运行
chmod 755 hello.sh
./hello.sh #相对路径方式执行
2)通过Bash调用执行脚本
bash hello.sh

都是用shell来进行系统管理的

Shell基础
1 命令别名与快捷键

别名的快捷键 是alias
取消别名的快捷键 是unalias
这是别名永久生效与删除别名

- vi ~/.bashrc
  #写入环境变量配置文件

alias
#查看系统中所有的命令别名

- unalias 别名
  #删除别名

alias 别名= '原命令'
#设定命令别名

命令生效顺序

第一顺位执行用绝对路径或相对路径执行的命令。
第二顺位执行别名。
第三顺位执行Bash的内部命令。
第四顺位执行按照$PATH环境变量定义的目录查找顺序找到的第一个命令

- ctrl+c 强制终止当前命令
- ctrl+l 清屏
- ctrl+a 光标移动到命令行首
- ctrl+e 光标移动到命令行尾
- ctrl+u 从光标所在位置删除到行首
- ctrl+z 把命令放入后台
- ctrl+r 在历史命令中搜索

1 历史命令
history【选项】【历史命令保存文件】

- 选项：
  - -c：清空历史命令
  - -w：把缓存中的历史命令写入历史命令保存文件~/.bash_history

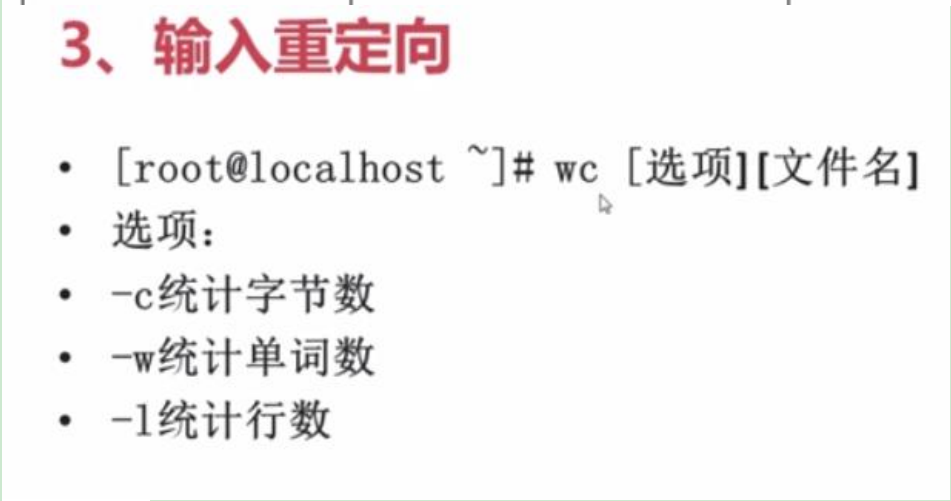
1.历史命令
history [选项] [历史命令保存文件]
选项
-c 清空历史命令
-w 把缓存中的历史命令写入历史命令保存文件～/.bash_history
#历史命令默认保存1000条，如果不够可以修改环境变量进行配置/etc/profile HISTSIZE
#使用上下箭头调用以前的历史命令
#使用"!n"重复执行第n条历史命令
#使用"!!"重复执行上一条命令
#使用"!字符串"重复执行以该字符串开头的命令

2.命令和文件补全 Tab键
命令补全依赖PATH环境变量

设备	设备文件名	文件描述符	类型
键盘	/dev/stdin	0	标准输入
显示器	/dev/sdtout	1	标准输出
显示器	/dev/sdterr	2	标准错误输出

在linux中有定时任务

正确输出和错误输出同时保存	命令 > 文件 2>&1	以覆盖的方式，把正确输出和错误输出都保存到同一个文件当中。
	命令 >> 文件 2>&1	以追加的方式，把正确输出和错误输出都保存到同一个文件当中。
	命令 &>文件	以覆盖的方式，把正确输出和错误输出都保存到同一个文件当中。
	命令 &>>文件	以追加的方式，把正确输出和错误输出都保存到同一个文件当中。
	命令>>文件1 2>>文件2	把正确的输出追加到文件1中，把错误的输出追加到文件2中。



1.标准输入输出
设备 设备文件名 文件描述 类型
键盘 /dev/stdin 0 标准输入
显示器 /dev/sdtout 1 标准输出
显示器 /dev/sdterr 2 标准错误输出

2.输出重定向
标准输出重定向：
命令 > 文件 以覆盖的方式，把命令的正确输出输出到指定的文件或设备当中
命令 >> 文件 以追加的方式，把命令的正确输出输出到指定的文件或设备当中
标准错误输出重定向：
错误命令 2>文件 以覆盖的方式，把命令的错误输出输出到指定的文件或设备当中
错误命令 2>>文件 以追加的方式，把命令的错误输出输出到指定的文件或设备当中
正确输出和错误输出同时保存
命令 > 文件 2>&1 以覆盖的方式，把正确输出和错误输出都保存到同一个文件中
命令 >> 文件 2>&1 以追加的方式，把正确输出和错误输出都保存到同一个文件中
命令 &>文件 以覆盖的方式，把正确输出和错误输出都保存到同一个文件中
命令 &>>文件 以追加的方式，把正确输出和错误输出都保存到同一个文件中
命令>>文件1 2>>文件2 把正确输出追加到文件1中，错误输出追加到文件2中
#/dev/null 文件黑洞

3.输入重定向
wc [选项] [文件名] ctrl+d结束输入
-c:统计字节数

-w:统计单词数
-l:统计行数

命令 < 文件把文件作为命令的输入
命令 << 标识符 [内容...] 标识符

正确执行

多命令执行符	格式	作　用
；	命令1；命令2	多个命令顺序执行，命令之间没有任何逻辑联系
&&	命令1&& 命令2	逻辑与 当命令1正确执行，则命令2才会执行 当命令1执行不正确，则命令2不会执行
||	命令1 || 命令2	逻辑或 当命令1执行不正确，则命令2才会执行 当命令1正确执行，则命令2不会执行

2、管道符

- **命令格式：**
- **[root@localhost ~]# 命令1 | 命令2**
  #命令1的正确输出作为命令2的操作对象

netstat -an这个命令是查看网络下所有的网络连接

1、通配符

通配符	作　用
?	匹配一个任意字符
*	匹配0个或任意多个任意字符，也就是可以匹配任何内容
[]	匹配中括号中任意一个字符。例如：[abc]代表一定匹配一个字符，或者是a，或者是b，或者是c。
[-]	匹配中括号中任意一个字符，-代表一个范围。例如：[a-z]代表匹配一个小写字母。
[^]	逻辑非，表示匹配不是中括号内的一个字符。例如：[^0-9]代表匹配一个不是数字的字符。

1、echo输出命令和脚本执行

格式：echo [选项] [输出内容]
选项： -e：支持反斜线控制的字符转换
echo "bols he cangls ni xihuan nayige"
\a - 输出警告音
\b - 退格符
\n - 换行符
\r - 回车键
\t - 制表符，也就是tab键
\v - 垂直制表符
\0nnn - 按照八进制ASCII码输出字符

\xhh - 按照十六进制ASCII表输出字符
echo -e "bols he cangls ni\b xihuan nayige"
echo -e "hell\bo"
echo -e "h\te\tl\nl\to\t"
echo -e "\x68\t\x65\t\x6c\n\x6c\t\x6f"
echo -e "\e[1;31m嫁人就要嫁凤姐\e[0m"
#输出颜色 \e[1;##m - 开启颜色显示； \e[0m - 关闭颜色显示
#30m=黑色，31m=红色，32m=绿色，33m=黄色
#34m=蓝色，35m=洋红，36m=青色，37m=白色

2、脚本执行：
方法1：赋予执行权限，直接运行
chmod 755 hello.sh
./hello.sh（或者绝对路径执行 /root/hello.sh）
方法2：通过Bash调用执行脚本
bash hello.sh
二、
#！/bin/Bash
不是注释，它标称下面的内容是linux的标准脚本程序
如果该脚本使用纯shell语句完成，不加#！/bin/Bash，运行没问题
但，脚本调用了其他语言，就会报错

2、Bash中其他特殊符号

符 号	作 用
,,	单引号。在单引号中所有的特殊符号，如“$”和“`”(反引号)都没有特殊含义。
""	双引号。在双引号中特殊符号都没有特殊含义，但是“$”、“`”和“\”是例外，拥有“调用变量的值”、“引用命令”和“转义符”的特殊含义。
``	反引号。反引号括起来的内容是系统命令，在Bash中会先执行它。和$()作用一样，不过推荐使用$()，因为反引号非常容易看错。
$()	和反引号作用一样，用来引用系统命令。
#	在Shell脚本中，#开头的行代表注释。
$	用于调用变量的值，如需要调用变量name的值时，需要用$name的方式得到变量的值。
\	转义符，跟在\之后的特殊符号将失去特殊含义，变为普通字符。如\$将输出“$”符号，而不当做是变量引用。

尽量使用$() 而不使用反引号

- VI → VIM
- VIM相对于VI做了哪些提升
  - VIM支持多级撤销
  - VIM可以跨平台运行
  - VIM支持语法高亮
  - VIM支持图形界面

VI编辑器的操作模式

- Command Mode - 命令模式
- Insert Mode - 输入模式
- Last Line Mode - 底行模式（尾行，末行）

光标移动

普通模式（进入Vim后的默认模式，其他模式按Esc切换普通模式）中，使用以下面按键移动光标：

h： 左移
j： 下移
k： 上移
l： 右移
w： 移到下一个单词
b： 移到上一个单词
0： 移到行首

切换插入模式

以下按键从普通模式切换到插入模式，键盘按键输入到文本中：

i： 在当前光标处进行编辑
I： 在行首插入
A： 在行末插入
a： 在光标后插入编辑
o： 在当前行后插入一个新行
O： 在当前行前插入一个新行
cw： 替换从光标所在位置后到一个单词结尾的字符

切换命令模式

从普通模式输入：进入命令行模式。

：w 文件名： 保存文件
：q！： 强制退出Vim，不保存
：x或者：wq： 保存并推出

普通模式下，Shift + zz也可保存并推出Vim

删除文本

普通模式下，以下命令可以实现删除文本目的：

x： 删除光标所在字符
X： 删除光标所在的前一个字符
dd： 删除整行
dw： 删除一个单词
d$ 或 D： 删除至行尾
d^： 删除至行首
dG： 删除至文档结尾处
d1G： 删除至文档开始处

vim + abc 定位到最后一行
vim +/a 定位到a字符串的那行
vim aa bb cc 同时创建3个文件 字母n 字母N 切换

底行模式常用指令
:w 保存
:q 退出
:! 强制退出
:wq 保存退出
:num 将光标快速定位到num行
/xxx 将光标向后搜索xxx字符串，定位到xxx第一次出现的位置
?xxx 将光标向前搜索xxx字符串，定们到xxx第一次出现的位置
dd 删除光标所在行
o 在光标所在行的下方插入一行并切换到输入模式
yy 复制光标所在的行
p 在光标所在行的下方粘贴
P 在光标所在行的上方粘贴
命令模式常用指令
h 光标左移
j 光标下移
k 光标上移
l 光标右移
ctrl+f 向下翻页(front)
ctrl+b 向上翻页(back)
ctrl+d 向下翻半页(down)
ctrl+u 向上翻半页(up)