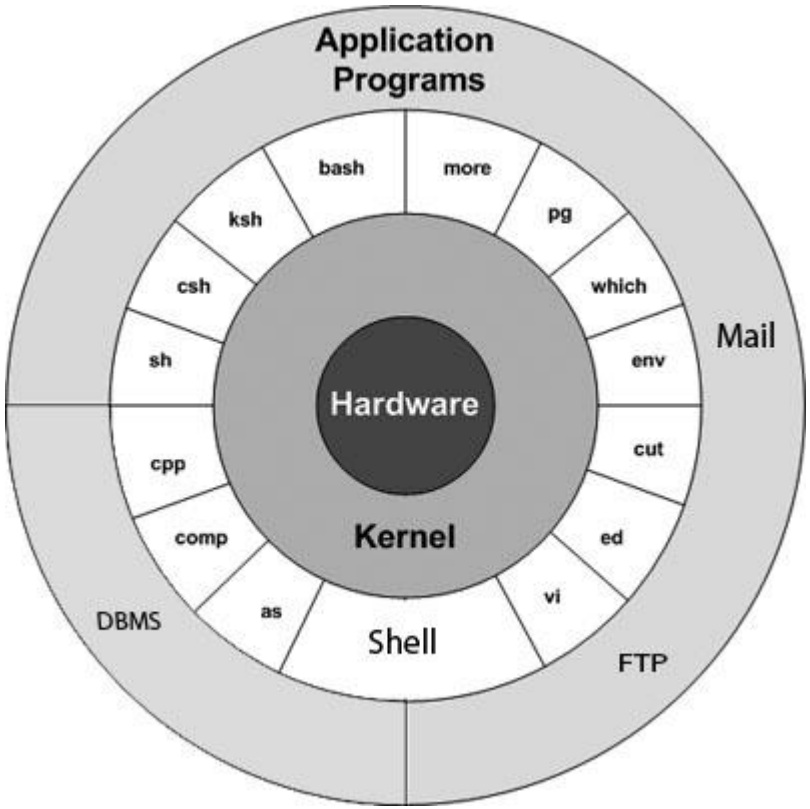
Shell脚本编程基础



内容

①

- Shell概述
  - Shell的概念
  - Shell的特点
  - Shell版本
  - Shell程序示例
  - Shell程序执行方法
- Linux命令帮手
  - 命令行编辑
  - screen工具

②

- Shell特殊字符
  - 通配符
  - 正则表达式
  - 引号
  - 转义符与路径符
  - 输入输出重定向
  - 注释和后台命令
  - 命令组合符
  - 成组命令

③

- Shell变量
  - 用户自定义变量
  - 位置变量
  - Shell预定义变量
  - 环境变量

④

- 算术运算
- 控制结构
- 函数

⑤

- 作业控制
- Shell内置命令
- Shell脚本调试
- Shell脚本示例

算术运算

- 整数运算

  (( ))

  - ((i=i+1))
  - j=$((i*3))

  let

  - i=2; let i=i+1

  expr

  - expr 1 + 1　#运算符两侧需有空格
  - expr 3 * 5　#*需转义

  $[ ]

  - j=$[i*2]

- 非整数运算（**bc**）
  - echo “scale=4;10/3” | bc

控制结构

- 顺序执行：换行或分号
- 判断：if、case
- 循环：for、while、until、select

条件测试

- **test（shell内部命令）**

  $ test -d ~/genome

- **[ ]（shell内部命令）**

  $ [$num -gt 2 -a $num -le 5]

  其中使用-a/-o表示逻辑与和逻辑或

- **[[ ]]（Shell关键字）**

  $ [[$color == “Green” || color = “Red”]]

  其中逻辑与/或用&&/||表示，通配符不需加引号，变量前可以不加$

- **命令（执行成功则条件为真）**

  $ echo $1 | grep “-h” >/dev/null　　#$1中包含-h则条件为真

条件测试

- 数值测试运算符
- 字符串测试运算符
- 逻辑运算符
- 文件测试运算符
- 特殊条件测试

数值测试运算符

n1 –eq n2	n1等于n2时为真（equal）
n1 –ne n2	n1不等于n2时为真（not equal）
n1 –lt n2	n1小于n2时为真（less than)
n1 –le n2	n1小于或等于n2时为真（less than or equal）
n1 –gt n2	n1大于n2时为真（greater than）
n1 –ge n2	n1大于或等于n2时为真（greater than or equal）

字符串测试运算符

s1 = s2	字符串s1和s2一样时为真
s1 == s2	同上
s1 != s2	字符串s1和s2不一样时为真
s1 < s2	按字典顺序s1在s2之前时为真
s1 > s2	按字典顺序s1在s2之前时为真
-z s1	字符串s1长度为0时为真
-n s1	字符串s1长度大于0时为真
s1	字符串s1不为空时为真

逻辑运算符

- **!**：逻辑非，如：[! –f hello.pl]
- **-a**：逻辑与，用在[ ]中，如：[-f hello.pl -a -f hello.c]
- **-o**：逻辑或，用在[ ]中，如：[-f hello.pl -o -f hello.sh]
- **&&**：逻辑与，用在[[ ]]中，如：[[-f hello.pl && -f hello.c]]
- **||**：逻辑或，用在[[ ]]中，如：[[-f hello.pl || -f hello.c]]

- **()**：优先运算：如：

  –[$1 -gt 10 -a $2 -gt 20 -o $3 -lt 10]

  –[$1 -gt 10 -a \($2 -gt 20 -o $3 -lt 10 \)]

文件测试运算符：文件类型

-f filename 文件存在且是普通文件则为真

-d filename 文件存在且是目录文件则为真

-L filename 文件存在且是符号链接则为真

-p filename 文件存在且是管道文件则为真

-b filename 文件存在且是块设备文件则为真

-c filename 文件存在且是字符设备文件则为真

-S filename 文件存在且是socket文件则为真

文件测试运算符：文件权限

-r filename	文件存在且用户可读则为真
-w filename	文件存在且用户可写则为真
-x filename	文件存在且用户可执行则为真

文件测试运算符：文件大小

-e filename	文件（包括目录）存在则为真
-s filename	文件存在且大小不为0则为真

文件测试运算符：文件比较

file1 **-nt** file2　　判断文件file1是否比file2更新（根据mtime），或者判断file1存在但file2不存在

file1 **-ot** file2　　判断文件file1是否比file2更旧，或者判断file2存在但file1不存在

file1 **-ef** file2　　判断文件file1和file2是否互为硬链接（？）

特殊条件测试

- **：**，不作任何事情，退出值为0
- **true**，总为真，退出值为0
- **false**，总为假，退出值为1

[ ]与[[ ]]总结

- 表示逻辑与/或：在[ ]内用-a/-o，在[[ ]]内用&&/||
- 整数比较用-gt、-lt…，字符串比较用>、<…
- [ ]内用>和<时需转义（否则是重定向符）
- [ ]是Shell内置命令，[[ ]]是Shell关键字

- **思考：下面的条件表达式的值。**

  [[5 -gt 10]]

  [[5 > 10]]

5 < $a <9 可写成：

[$a -gt 5 -a $a -lt 9] 或 test $a -gt 5 -a $a -lt 9

[[$a -gt 5 && $a -lt 9]]

[[a -gt 5 && a -lt 9]]　　#[[ ]]内的变量可以没有$

下面的语法不会出错，但结果是按照字符串比较

[[$a > 5]]

[$a \> 5]

if语句

if 条件

then

命令

elif 条件

then

命令

else

命令

fi

if语句示例

```
if test -f /home/pub/seq/at.fa
then
        echo “File exists!”
else
        echo “File does not exist!”
fi
```

case语句

```
case 变量名 in
  模式字符串1) 命令;;
  模式字符串2) 命令;;
  ……
  *) 命令;;
esac
```

case示例

```
#!/bin/bash
week=$1
case $week in
    Saturday|Sunday) echo “Weekend”;;
    Monday|Tuesday|Wednesday|Thursday|Friday)
        echo “Work day”;;
    *) echo “Unknown date”;;
esac
```

for语句：值表形式

- 一般格式：for 变量 in 值表;do 命令表;done
- 如：

```
for i in "${person[@]}"                    #数组
do
        echo $i
done
```

或

```
for i in rose lily "Chinese rose"          #列表
for i in {a..z}                            #大括号扩展
for i in `seq 1 10`                        #命令扩展
```

for语句：算术表达式形式

- 一般格式：for ((e1;e2;e3));do 命令表;done
- 如：

```
for ((i=1;i<5;i++))          #注意是两对小括号
do
    echo $i
done
```

while语句

- 一般形式：

  while 测试条件

  do

  　　命令表

  done

until语句

- 一般形式：

  until 测试条件

  do

  　　命令表

  done

select语句

- select语句通常用于菜单设计，其一般形式：

  select identifier [in word…]

  do

  　　命令表

  done

select语句示例

```
$ cat select.sh
#!/bin/bash
echo "What is your favourite fruit?"
select fruit in Apple Banana Orange Pineapple
do
        break
done
echo "Your favourite fruit is $fruit!"
```

```
$ ./select.sh
What is your favourite fruit?
1) Apple
2) Banana
3) Orange
4) Pineapple
#? 2
Your favourite fruit is Banana!
```

跳出循环

- **continue**：跳过本次循环，执行下一次循环
- **break**：退出循环体
- **exit**：退出脚本

shell函数

- **格式：**

  [function] 函数名 ()

  {

  命令表

  }

- **注意事项：**
  - 函数需先定义，后引用
  - 函数返回值可用return n定义，否则为函数最后一个命令的返回值
  - Shell函数定义时不需要指定参数，可直接通过位置变量调用参数

shell函数示例

```
function sum ()
{
    local total=0          #Shell变量默认为全局变量，local定义为局部变量
    while [ $1 ]
    do
        total=$((total+$1))
        shift
    done
    echo $total
}
```

总结：Shell中的括号—小括号

- 单小括号()
  - –命令组（新建一个子Shell）：(ls; pwd) |sort
  - –命令替换：current_path=$(pwd)
  - –数组赋值（列表）：week=(Mon Tue Wed)
- 双小括号(())
  - –整数运算：((j=i+1))
  - –循环：for ((i=1;i<5;i++))

总结：Shell中的括号—中括号

- 单中括号[ ]
  - –条件测试（Bash的内部命令=test）： if [$# -eq 0]
  - –通配符，表示字符组或范围：[aCe]、[0-5]
  - –数组下标：echo ${week[0]}
  - –整数运算：$[2*3]
- 双中括号[[ ]]
  - –Bash程序语言的关键字，用于条件表达式：如

  if [[$1 == “-h” || $1 == “--help”]]

总结：Shell中的括号—大括号

- 变量名

–普通变量或数组：${num}_chr, ${week[*]}

–置换变量：

${var:-string}, ${var:+string}, ${var:=string}, ${var:?string}

–模式匹配替换：

${var%pattern}, ${var%%pattern}, ${var#pattern},

${var##pattern}

–字符串提取和替换： ${var:num}, ${var:num1:num2},

${var/pattern/pattern}, ${var//pattern/pattern}

总结：Shell中的括号—大括号（续）

- 命令组（也叫内部组，里面的命令在当前Shell运行，不新建子Shell）

  $ { cat rna_stat; paste rna_type rna_num; } |sort

- 大括号扩展

  $ touch {a,b}_{1,2,3}.txt

- 函数：

```
function sum()
{
        …
}
```

Shell关键字总结（1）

- if：用作条件判断，与fi一起使用
- elif：用作条件判断
- then：用作条件判断，与if、elif一起使用
- else：用作条件判断
- fi：用作条件判断，与if一起使用
- case：用作条件判断，与esac一起使用
- esac：用作条件判断，与case一起使用
- [[：用在条件表达式中，与]]一起使用
-]]：用在条件表达式中，与[[一起使用

Shell关键字总结（2）

- for：用在循环语句中
- while：用在循环语句中
- until：用在循环语句中
- select：循环的一种，可用来实现选择菜单
- do：用在shell循环中，与done一起使用
- done：用在shell循环中，与do一起使用
- !：表示否定、反义
- function：用于定义函数
- time：用于计算命令的运行时间


未完待续
LOADING