**第二十三章 expect-正则表达式-sed-cut的使用**

**本节所讲内容：**

**23.1 expect实现无交互登录**

**23.2 正则表达式**

**23.3 sed流编辑器**

**23.4 cut命令**

**23.5 实战-bash脚本语法检查和查看详细的执行过程**

# 23.1 expect实现无交互登录

**expect ([ɪkˈspekt] 期待 )是从它发展出来的。如果你想要写一个能够自动处理输入输出的脚本（如向用户提问并且验证密码）又不想面对C或者Perl，那么expect是你的最好的选择。它可以用来做一些linux下无法做到交互的一些命令操作**

## 23.1.1 安装和使用expect

**[root@xuegod63 ~]# yum -y install expect**

**使用expect创建脚本的方法**

**1）定义脚本执行的shell**

**#!/usr/bin/expect**

**这里定义的是expect可执行文件的链接路径（或真实路径），功能类似于bash等shell功能**

**2）set timeout 30**

**设置超时时间，单位是秒，如果设为timeout -1 意为永不超时**

**3）spawn**

**spawn 是进入expect环境后才能执行的内部命令，如果没有装expect或者直接在默认的SHELL下执行是找不到spawn命令的。不能直接在默认的shell环境中进行执行主要功能，它主要的功能是给ssh运行进程加个壳，用来传递交互指令。**

**4）expect**

**这里的expect同样是expect的内部命令**

**主要功能：判断输出结果是否包含某项字符串，没有则立即返回，否则就等待一段时间后返回，等待时间通过timeout进行设置**

**5）send**

**执行交互动作，将交互要执行的动作进行输入给交互指令**

**命令字符串结尾要加上"\r"，如果出现异常等待的状态可以进行核查**

**6）exp\_continue**

**继续执行接下来的交互操作**

**7）interact**

**执行完后保持交互状态，把控制权交给控制台；如果不加这一项，交互完成会自动退出**

**8）$argv**

**expect 脚本可以接受从bash传递过来的参数，可以使用 [lindex $argv n]获得，n从0开始，分别表示第一个，第二个，第三个……参数**

**例1：免密码通过SSH登录服务器（了解） 这里不是用密钥**

**注：运行脚本时，要把#号后面的注释删除，不然无法运行**

**[root@xuegod63 ~]# cat ssh.exp**

**#!/usr/bin/expect**

**set ipaddr "192.168.1.63"**

**set name "root"**

**set passwd "123456"**

**set timeout 30 #设置超时时间，单位是秒；expect超时等待的时间。默认timeout为10s。**

**spawn ssh $name@$ipaddr #** **spawn是进入expect环境后才可以执行的expect内部命令，如果没有装expect或者直接在shell下执行是找不到spawn命令的。这个就好比cd是shell的内建命令，离开shell，就无法执行cd一样。 它主要的功能是给ssh运行进程加个壳，用来传递交互指令。**

**expect {**

**"yes/no" { send "yes\r";exp\_continue }**

**"password" { send "$passwd\r" } #执行交互动作，与手工输入密码的动作等效。**

**}**

**expect "#" #判断上次输出结果里是否包含“password:”的字符串，如果有则立即返回，向下执行；否则就一直等待，直到超时时间到**

**send "touch /root/xuegod1011.txt\r"**

**send "ls /etc > /root/xuegod1011.txt\r"**

**send "mkdir /tmp/xuegod1011\r"**

**send "exit\r"**

**expect eof #执行完成上述命令后，退出Expect，把控制权交给控制台，变回手工操作**

**[root@xuegod63 ~]# expect ssh.exp #开始执行**

**例2：对服务器批量管理（了解一下）**

**[root@xuegod63 ~]# cat ip\_pass.txt #这里写上要执行的IP地址和root用户密码**

**192.168.1.63 123456**

**192.168.1.63 123456**

**192.168.1.63 123456**

**[root@xuegod63 ~]# cat ssh2.exp #编写要执行的操作**

**#!/usr/bin/expect**

**set ipaddr [lindex $argv 0]**

**set passwd [lindex $argv 1]**

**set timeout 30**

**spawn ssh root@$ipaddr**

**expect {**

**"yes/no" { send "yes\r";exp\_continue }**

**"password" { send "$passwd\r" }**

**}**

**expect "#"**

**send "touch /root/xuegod1011.txt\r"**

**send "ls /etc > /root/xuegod1011.txt\r"**

**send "mkdir /tmp/xuegod1011\r"**

**send "exit\r"**

**expect eof**

**[root@xuegod63 ~]# cat login.sh #开始执行**

**#!/bin/bash**

**echo**

**for ip in `awk '{print $1}' /root/ip\_pass.txt`**

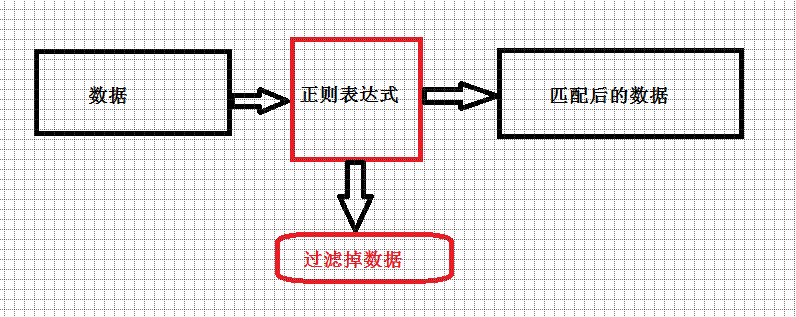
**do**

**pass=`grep $ip /root/ip\_pass.txt|awk '{print $2}'`**

**expect /root/ssh.exp $ip $pass**

**done**

# 23.2 正则表达式的使用

****

**正则表达式，又称规则表达式。（英语：Regular Expression [ˈreɡjulə] 规则的 [ iksˈpreʃən] 表达 ），在代码中常简写为regex、regexp或RE），计算机科学的一个概念。正则表达式通常被用来检索、替换那些符合某个模式(规则)的文本。**

**正则表达式不只有一种，而且LINUX中不同的程序可能会使用不同的正则表达式，如：**

**工具：grep sed awk**

**LINUX中常用的有两种正则表达式引擎**

* **基础正则表达式：BRE**
* **扩展正则表达式: ERE**

## 23.2.1 Shell正则表达式的组成

**基础正则表达式**

|  |  |
| --- | --- |
| **特别字符** | **描述** |
| **$** | **匹配输入字符串的结尾位置。要匹配 $ 字符本身，请使用 \$** |
| **( )** | **标记一个子表达式的开始和结束位置。要匹配这些字符，请使用 \( 和 \)** |
| **\*** | **匹配前面的子表达式零次或多次。要匹配 \* 字符，请使用 \\*** |
| **+** | **匹配前面的子表达式一次或多次。要匹配 + 字符，请使用 \+** |
| **.** | **匹配除换行符 \n 之外的任何单字符。要匹配 . ，请使用 \.** |
| **[** | **标记一个中括号表达式的开始。要匹配 [，请使用 \[** |
| **?** | **匹配前面的子表达式零次或一次，或指明一个非贪婪限定符。要匹配 ? 字符，请使用 \?** |
| **\** | **将下一个字符标记为或特殊字符、或原义字符、或向后引用、或八进制转义符。例如， 'n' 匹配字符 'n'。'\n' 匹配换行符。序列 '\\' 匹配 "\"，而 '\(' 则匹配 "("** |
| **^** | **匹配输入字符串的开始位置，除非在方括号表达式中使用，此时它表示不接受该字符集合。要匹配 ^ 字符本身，请使用 \^** |
| **{** | **标记限定符表达式的开始。要匹配 {，请使用 \{** |
| **|** | **指明两项之间的一个选择。要匹配 |，请使用 \| 如： Y | y** |
| **定位符** | |
| **^** | **匹配输入字符串开始的位置** |
| **$** | **匹配输入字符串结尾的位置** |
| **非打印字符** | |
| **\n** | **匹配一个换行符** |
| **\r** | **匹配一个回车符** |
| **\t** | **匹配一个制表符** |

**例：统计/etc/ssh/sshd\_config文件中除去空行和#号开头的行的行数**

**[root@xuegod63 ~]# grep -v "^$\|^#" /etc/ssh/sshd\_config #使用基础正则表达式**

**[root@xuegod63 ~]# grep -E -v "^$|^#" /etc/ssh/sshd\_config #扩展正则表达式**

**[root@xuegod63 ~]# egrep -v "^$|^#" /etc/ssh/sshd\_config #扩展正则表达式**

**例2：点字符**

**[root@xuegod63 ~]# grep .ot /etc/passwd #查找passwd文件包括.ot 的字符**

**root:x:0:0:root:/root:/bin/bash**

**operator:x:11:0:operator:/root:/sbin/nologin**

**setroubleshoot:x:993:990::/var/lib/setroubleshoot:/sbin/nologin**

# 23.3 sed流编辑器--用分组-正则

## 23.3.1 sed strem editor 流编辑器

**sed编辑器是一行一行的处理文件内容的。正在处理的内容存放在模式空间(缓冲区)内，处理完成后按照选项的规定进行输出或文件的修改。**

**接着处理下一行，这样不断重复，直到文件末尾。文件内容并没有改变，除非你使用重定向存储输出。Sed主要用来自动编辑一个或多个文件；简化对文件的反复操作；**

**sed也是支持正则表达式的，如果要使用扩展正则加参数-r**

**sed的执行过程：**

1. **一次读取一行数据**
2. **根据我们提供的规则来匹配相关的数据，比如查找root。**
3. **按照命令修改数据流中的数据，比如替换**
4. **将结果进行输出**
5. **重复上面四步**

## 23.3.2 如何使用

**语法格式：sed [options] ‘[commands]’ filename**

**例1：**

**[root@xuegod63 ~]# echo "this is aplle" | sed 's/aplle/dog/'**

**this is dog**

**[root@xuegod63 ~]# echo "this is aplle" > a.txt**

**[root@xuegod63 ~]# sed 's/apple/dog/' a.txt**

**this is aplle**

**[root@xuegod63 ~]# cat a.txt #发现并没有修改文件**

**this is aplle**

## 23.3.3 sed选项|参数

**options:**

**-a 在当前行下面插入文件**

**-n 读取下一个输入行，用下一个命令处理新的行而不是用第一个命令**

**-e 执行多个sed指令**

**-f 运行脚本**

**-i 编辑文件内容 \*\*\***

**-i.bak 编辑的同时创造.bak的备份**

**-r 使用扩展的正则表达式**

**命令：**

**i 在当前行上面插入文件**

**c 把选定的行改为新的指定的文本**

**p 打印 \*\*\***

**d 删除 \*\*\***

**r/R 读取文件/一行**

**w 另存**

**s 查找**

**y 替换**

**h 拷贝模板块的内容到内存中的缓冲区。**

**H 追加模板块的内容到内存中的缓冲区。**

**g 获得内存缓冲区的内容，并替代当前模板块中的文本。**

**G 获得内存缓冲区的内容，并追加到当前模板块文本的后面**

**D 删除\n之前的内容**

**P 打印\n之前的内容**

**替换标记：**

* **数字：表明新文本将替换第几处模式匹配的地方**
* **g：表示新文本将会替换所有匹配的文本**
* **\1：子串匹配标记，前面搜索可以用元字符集\(..\),**
* **&：保留搜索到的字符用来替换其它字符**

**sed匹配字符集**

**^ 匹配行开始，如：/^sed/匹配所有以sed开头的行。**

**$ 匹配行结束，如：/sed$/匹配所有以sed结尾的行。**

**. 匹配一个非换行符的任意字符，如：/s.d/匹配s后接一个任意字符，最后是d。**

**\* 匹配0个或多个字符，如：/\*sed/匹配所有模板是一个或多个空格后紧跟sed的行。**

**例1：s 只替换第一个匹配到的字符，将passwd中的root用户替换成xuegod**

**[root@xuegod63 ~]# sed 's/root/xuegod/' /etc/passwd**

**xuegod:x:0:0:root:/root:/bin/bash #发现只替换了第一个匹配的root，后面的没有替换**

**bin:x:1:1:bin:/bin:/sbin/nologin**

**例2：全面替换标记g**

**[root@xuegod63 ~]# sed 's/root/xuegod/g' /etc/passwd |more**

**xuegod:x:0:0:xuegod:/xuegod:/bin/bash #全部替换了**

**例2： 将sed中默认的/ 定界符改成#号**

**[root@xuegod63 ~]# sed 's#/bin/bash#/sbin/nologin#' /etc/passwd | more**

**root:x:0:0:root:/root:/sbin/nologin**

**bin:x:1:1:bin:/bin:/sbin/nologin**

**以/来做定界符：**

**[root@xuegod63 ~]# sed 's/\/bin\/bash/\/sbin\/nologin/' /etc/passwd**

**后面的文件可以使用通配符去操作**

grep '^(barlow).\*\1' /etc/passwd 搜索/etc/passwd中以barlow开头，而后面还存在barlow的行.

**拿到xxx yyyy**

### echo "aaaaaaa balabala name=xxx,yyy&zzzzz\_balabala" | sed -r 's/.\*name=(.\*?),(.+)&(.\*?)\_\*\*/\1 \2/'

### （2）按行查找替换

**写法如下：**

* **用数字表示行范围；$表示行尾**
* **用文本模式配置来过滤**

**例1：单行替换，将第2行中bin替换成xuegod**

**[root@xuegod63 ~]# sed '2s/bin/xuegod/' /etc/passwd | more**

**root:x:0:0:root:/root:/bin/bash**

**xuegod:x:1:1:bin:/bin:/sbin/nologin**

**例2：多行替换，如果涉及到多行处理，用逗号表示行间隔。 将第3行到最行尾中bin替换成xuegod**

**[root@xuegod63 ~]# sed '2,$s/bin/xuegod/' /etc/passwd | more**

**root:x:0:0:root:/root:/bin/bash**

**xuegod:x:1:1:bin:/bin:/sbin/nologin**

**daemon:x:2:2:daemon:/sxuegod:/sbin/nologin**

**adm:x:3:4:adm:/var/adm:/sxuegod/nologin**

### （3）d 删除第2行到第4行的内容

**[root@xuegod63 ~]# cat /etc/hosts**

**127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4**

**::1 localhost localhost.localdomain localhost6 localhost6.localdomain6**

**192.168.1.63 xuegod63.cn**

**192.168.1.64 xuegod64.cn**

**192.168.1.62 xuegod62.cn**

**[root@xuegod63 ~]# sed '2,4d' /etc/hosts**

**127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4**

**192.168.1.62 xuegod62.cn**

**[root@xuegod63 ~]# sed '/192.168/d' /etc/hosts #将包括192.168的行删除**

**127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4**

**::1 localhost localhost.localdomain localhost6 localhost6.localdomain6**

**（4）添加行**

* **命令i(insert插入)，在当前行前面插入一行 i\**
* **命令a(append附加)，在当前行后面添加一行 a\**

**例1：插入**

**[root@xuegod63 ~]# echo "hello world" | sed 'i\ xuegod '**

**xuegod**

**hello world**

**例2：追加**

**[root@xuegod63 ~]# echo "hello world"|sed 'a\xuegod'**

**hello world**

**xuegod**

**例3：在文件最后追加内容**

**[root@xuegod63 ~]# sed '$a\192.168.1.65 xuegod65.cn' /etc/hosts**

**127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4**

**::1 localhost localhost.localdomain localhost6 localhost6.localdomain6**

**192.168.1.63 xuegod63.cn**

**192.168.1.64 xuegod64.cn**

**192.168.1.62 xuegod62.cn**

**192.168.1.65 xuegod65.cn**

**例4：在文件中第2行之后，开始追加内容**

**[root@xuegod63 ~]# sed '2a\192.168.1.65 xuegod65.cn' /etc/hosts**

**127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4**

**::1 localhost localhost.localdomain localhost6 localhost6.localdomain6**

**192.168.1.65 xuegod65.cn**

**192.168.1.63 xuegod63.cn**

**192.168.1.64 xuegod64.cn**

**192.168.1.62 xuegod62.cn**

**例5：在文件中第2行到第4行之后分别追加内容**

**[root@xuegod63 ~]# sed '2,4a\hello world' word1.txt**

**[root@xuegod63 ~]# sed '2,4a\192.168.1.65 xuegod65.cn' /etc/hosts**

**127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4**

**::1 localhost localhost.localdomain localhost6 localhost6.localdomain6**

**192.168.1.65 xuegod65.cn**

**192.168.1.63 xuegod63.cn**

**192.168.1.65 xuegod65.cn**

**192.168.1.64 xuegod64.cn**

**192.168.1.65 xuegod65.cn**

**192.168.1.62 xuegod62.cn**

### （5）修改行命令c (change) c\

**例1：将第4行内容改成192.168.1.65 xuegod65.cn**

**[root@xuegod63 ~]# cat /etc/hosts**

**127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4**

**::1 localhost localhost.localdomain localhost6 localhost6.localdomain6**

**192.168.1.63 xuegod63.cn**

**192.168.1.64 xuegod64.cn**

**192.168.1.62 xuegod62.cn**

**[root@xuegod63 ~]# sed '4c\192.168.1.65 xuegod65.cn' /etc/hosts**

**例2：将第2行到最后全部修改成192.168.1.65 xuegod65.cn**

**[root@xuegod63 ~]# sed '2,$c\192.168.1.65 xuegod65.cn' /etc/hosts**

**127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4**

**192.168.1.65 xuegod65.cn**

**例3：将包括192.168.1.64行的内容修改成192.168.1.65**

**[root@xuegod63 ~]# sed '/192.168.1.64/c\192.168.1.65' /etc/hosts**

**127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4**

**::1 localhost localhost.localdomain localhost6 localhost6.localdomain6**

**192.168.1.63 xuegod63.cn**

**192.168.1.65**

**192.168.1.62 xuegod62.cn**

### （6）打印，直接输入文件中的内容

**例1：输入第2行内容**

**[root@xuegod63 ~]# sed -n '2p' /etc/hosts**

**::1 localhost localhost.localdomain localhost6 localhost6.localdomain6**

### （7）将修改或过滤出来的内容保存到另一个文件中

**例2：将passwd中的包括root字样的行保存到 c.txt 中**

**[root@xuegod63 ~]# sed -n '/root/w c.txt' /etc/passwd**

**[root@xuegod63 ~]# cat c.txt**

**root:x:0:0:root:/root:/bin/bash**

**operator:x:11:0:operator:/root:/sbin/nologin**

### （8）-i 对原文件修改，保存（ 必会 ） 使用场景： 替换或修改服务器配置文件

**[root@xuegod63 ~]# cp /etc/passwd /opt/**

**[root@xuegod63 ~]# sed -i 's/root/xuegod/' /etc/passwd**

**[root@xuegod63 ~]# head -n 1 /etc/passwd**

**xuegod:x:0:0:root:/root:/bin/bash**

**修改IP地址为192.168.1.65**

**[root@xuegod63 ~]# sed -i 's/IPADDR=192.168.1.63/IPADDR=192.168.1.65/' /etc/sysconfig/network-scripts/ifcfg-ens33**

# 23.4 cut命令

## 23.4.1 cut常用参数

**cut命令用来显示行中的指定部分，删除文件中指定字段。**

**说明：该命令有两项功能，其一是用来显示文件的内容，它依次读取由参数file所指明的文件，将它们的内容输出到标准输出上；其二是连接两个或多个文件，如cut fl f2 > f3将把文件fl和fn的内容合并起来，然后通过输出重定向符“>”的作用，将它们放入文件f3中。**

**语法: cut(选项)(参数)**

**选项**

**-b：仅显示行中指定范围的字节数；**

**-c：仅显示行中指定范围的字符；**

**-d：指定字段的分隔符，默认的字段分隔符为“TAB”；**

**-f：显示指定字段的内容；**

**例1：输出系统中所有用户名**

**使用 -f 选项提取指定字段，使用 -d 选项指定字段分隔符，这里以：冒号做分隔**

**[root@xuegod63 ~]# cut -f1 -d ":" /etc/passwd**

## 23.4.2 cut命令可以将一串字符作为列来显示，字符字段的记法：

**N-：从第N个字节、字符、字段到结尾；**

**N-M：从第N个字节、字符、字段到第M个（包括M在内）字节、字符、字段；**

**-M：从第1个字节、字符、字段到第M个（包括M在内）字节、字符、字段。**

**上面是记法，结合下面选项将摸个范围的字节、字符指定为字段：**

**-b 表示字节；**

**-c 表示字符；**

**-f 表示定义字段。**

**示例**

**例1：打印第1个到第3个字符：**

**[root@xuegod63 ~]# cut -c1-3 /etc/passwd**

**例2：打印前2个字符：**

**[root@xuegod63 ~]# cut -c-2 /etc/passwd**

**例3：打印从第5个字符开始到结尾：**

**[root@xuegod63 ~]# cut -c5- /etc/passwd**

# 23.5 实战-bash脚本语法检查和查看详细的执行过程

**检查语法是否有错：**

**bash -v test.bash #查看bash是否存在语法错误**

**bash -x test.bash #查看bash详细的执行过程**

**[root@xuegod63 ~]# cat a.sh**

**# Script to show debug of shell**

**#**

**tot=`expr $1 + $2`**

**secho $tot #这里故意写错**

**[root@xuegod63 ~]# bash -v a.sh**

**# Script to show debug of shell**

**#**

**tot=`expr $1 + $2`**

**expr: 语法错误 #语法哪错了？ 运行时没有给参数**

**secho $tot #这里故意写错**

**a.sh:行4: secho: 未找到命令**

**[root@xuegod63 ~]# sed -i 's/secho/echo/' a.sh #修改正确后**

**[root@xuegod63 ~]# bash -x a.sh 2 3 #查看详细执行过程。 注：这个脚本是真正执行一遍，不是预执行**

**++ expr 2 + 3**

**+ tot=5**

**+ echo 5**

**例2：查看九九乘法表shell脚本运行过程**

**[root@xuegod63 ~]# cat 99.sh**

**for i in `seq 9`**

**do**

**for j in `seq $i`**

**do**

**echo -n "$i\*$j= `echo $(($i\*$j))` "**

**done**

**echo " "**

**done**

**root@xuegod63 ~]# bash -x 99.sh**

**总结：**

**23.1 expect实现无交互登录**

**23.2 正则表达式**

**23.3 sed流编辑器**

**23.4 cut命令**

**23.5 实战-bash脚本语法检查和查看详细的执行过程**