1. 文件描述符

先解释输入输出重定向

文件描述符 0 1 2

0标准输入 默认接受来自键盘的输入 stdin in

1标准输出 默认输出到屏幕终端 stdout out

2错误输出 默认输出到屏幕终端 stderr err

其实1其实我理解了一点，将交互时的内容输出到文件

eg: echo “a b” 1>a.txt

这样就实现了标准输出，将内容输出到文本

或者

ls –al > a.txt

是一种将屏幕内容输出到文本的操作

但是0，我几乎没用过

因为这是将文本输出到屏幕。

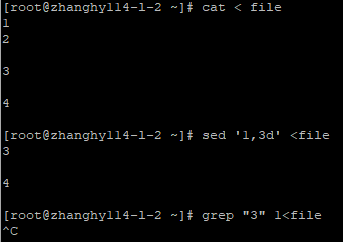
那有时候可以脑子就不好使了，输出屏幕用cat file 不久行了嘛？

cat file 何必写成 cat 1< file呢。

其实举一反三，输出可以将任何输出到屏幕内容哪怕时命令内容都输出到文件

那么输入可不可以将文件内容全部输入到屏幕再给拿出来使用呢。

如果将屏幕内容使用，这个问题就有点难度了。过滤可以吗？

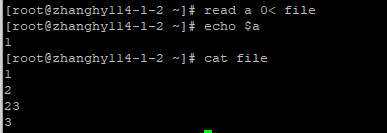


好像sed可以，但也是有点多余了，因为可以直接sed “1,3d” file

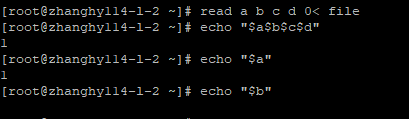
但是有一个可以，交互式的时候需要我们输入内容到屏幕啊。

比如read –p “nihao” a

这时候需要将内容输入到屏幕。但我们可以直接将文本内容赋值给变量a



但是只会赋值文本第一行内容给变量a



如何将文本内容（假如5行）依次赋值给一个变量呢？

read a b c d e < file （这个是错误的，因为read没有这个命令）

肯定用到循环了

while read i

do

done<file

键盘----标准输入stdin----process进程------------stdont标准输出到文本

错误输出到屏幕stdeer当然也可以输出到文本

文件描述符就是每打开一个文件都会有一个与之对应的描述符。Linux一切都是文件

其中0就是输入打开啊

1就是输出文件

2就是错误输出

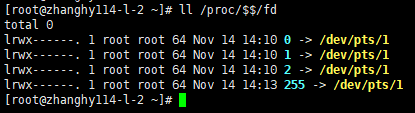
3就是其他文件了

如何查看一个进程打开那些描述符？

ll /proc/进程id/fd

$$当前shell的进程ID，如果在子shell中打印这个结果就是子shell的进程号。

但是在shell中这就是shell的进程号。因为我是用xshell打开的，那么就是



这是什么意思呢？



其实就是四个文件对应了我们打开的文件

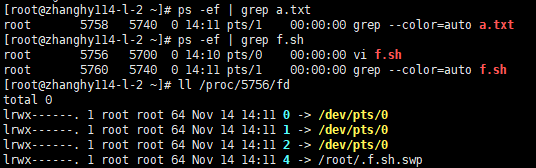
这表示我打开的终端号文件描述符



测试其他文件描述符！

打开一个终端，并且打开一个文件f.sh

再打开一个终端，查看你文件进程，并查看文件标识符

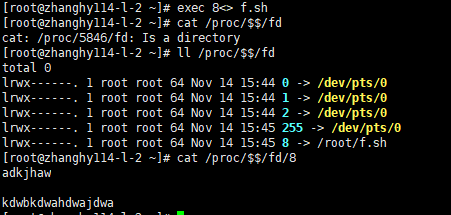


文件标识符内容和文件内容是一样的。但是不知道为什么我的文件标识符打开时错码？

会不会是用不同终端导致的结果。一个终端修改了文件描述符，打开这个文件的确实另一个终端的缘故？

所以接下里我在一个终端打开文件。

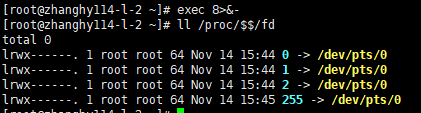
用上exec 8<> file 这个命令是将file打开而且将8作为他的文件描述符



我们想删除这个8文件软连接可以吗？

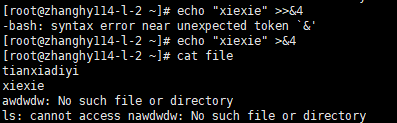


不可以，怎么办？



好了，学会了文件描述符。我发现在一个终端下面，操作文件描述符（肯定必须在一个终端下，因为你这个文件进程在$$下面）

那接下来就简单了，操作软连接文件，也就是描述符，看文件内容是否修改



果然修改了

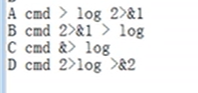
输出

ls 文件 >/root.log 2>&1

将2的输出到1，这样不管错误还是正确都输入到log文件下面

那么我想输出到描述符4上面呢

ls 文件 > &4



上面的那个是与其他三个不同。

B其中第四个是将标准输出到错误输出，然后错误输出到log文件

那么文件描述符是否可以无限打开呢，就是说在/dev/pts这个终端进程下，可以有多少个文件被打开？

ulimit –n

1024

这也就是说，可以有1024个被打开

当然可以设置 ulimit –n 2048

1. exec命令

xargs 和 –exec有点像，将之前命令的输出作为后一个命令的输入

ls –al | xargs rm –rf

ls –al –exec rm -rf

用上exec 8<> file 这个命令是将file打开而且将8作为他的文件描述符

ls 文件 > &4

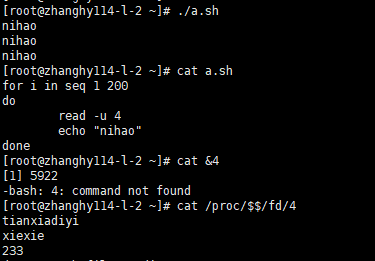
就是将内容输出到4被描述符，当然，内容也会随着修改

删除描述符

exec >&4-

1. read –u 文件描述符

这是读取文件描述符，当文件描述符内容有几行，就会读取几次，读取完毕就会停止



1. 管道

特征1.只能用一次，也就是说内容放进去，拿出来，管道里面就空了

2.先进先出，先放进去内容，出来的总是先进去的

匿名管道：

类似| 管道命令符 ，其实这也是文件

ls | grep “a.sh”

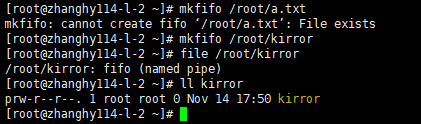
将ls的内容放进管道文件中 其实可以看做 grep “a.sh” | 这个|里面有了ls的内容，用完给grep后，就失去作用了，所以|可以一直使用

实名管道：

可以将一个文件作为管道

mkfifo + 文件路径（但是这个文件不能存在）

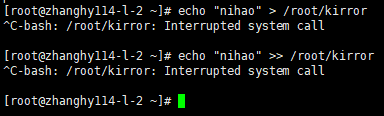
然后file + 文件路径就可以查看状态了



named pipe命令管道

可以看见我创建成功了

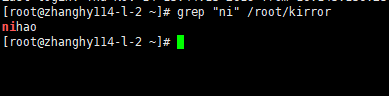
那么接下来接下来就是运用



我尝试将内容写入管道，但是失败了，所以我猜想，应该是写入无法退出状态

，应该就是在管道里面





一个终端输入管道内容，一个终端运行

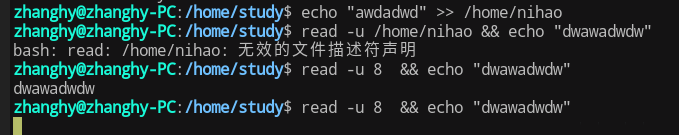
这样就成功了。

### 管道用法：

1.写入内容一次，就可以使用管道内容，使用完内容删除。

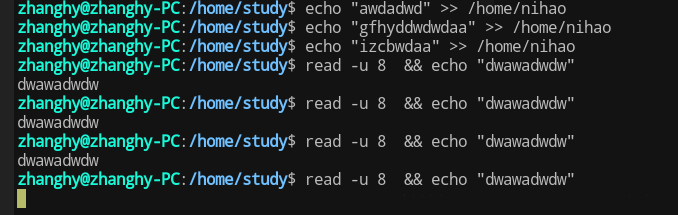
2.一个终端打开管道写，一个终端打开读，二者就会同步

3.不管写入多少内容，读一次管道就清空（但read -u+管道描述符是读取内容，会清理一行）。



我将管道符文件之前打开，描述符为8，接下来写入内容，使用一次管道符，下一次再使用就失效了。

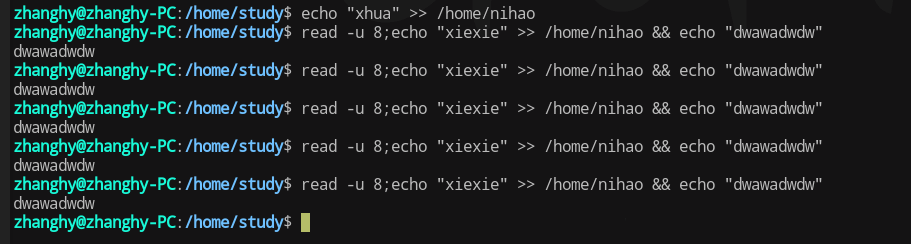
那么接下来写入多个内容呢？



这就是最有意思的地方了。写入三行内容，read -u +描述符会读取描述符文件内容，其实就是管道内容，读三次，三次用完，内容删除。而且内容是先写入的写出来。

那么如果我read -u 8之后再写入管道符内容如何呢。

read -u 8；echo “zai lai yici ” >> /home/nihao && echo “dayin jieguo”



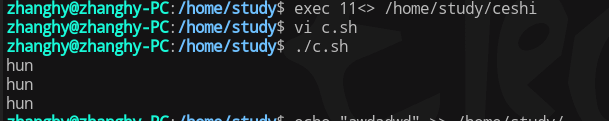
可以无限使用了。

那么如何是文件可以吗？

比如文件a.txt，我也给他写入内容，然后exec 9<> a.txt

### 可以read -u 9（其他非管道文件）吗？

不可以，我试过了，不管我写入内容到文件多少行，read -u 这个文件描述符只会读取3次，不知道啥原因。反正读取三次，只要你是文件，那样通过读取控制执行次数无法实现。



我查阅了read -u，但是只是说是读取fd描述符，也没更多讲述，我猜测是读取了

终端三个描述符 、dev/pts/0 的三个描述符0 1 2

但是也没证据。

实战：多并发ping IP地址，但是控制每次执行次数，因为一次太多可能顶不住

# expect交互

如何我们将在其他用户干一些事情，这个是批量的怎么办？用脚本写？

就是说一些交互型的操作，我们必须要批量执行怎么办？

还是不懂？

比如我们要登录alice1到alice100这一百个用户分别创建一个文件怎么办？

要登录用户，提示输入密码，输入密码，然后按exit退出，这些都是必须要通过交互性操作完成。

那么就需要用得上expect协议系列交互操作可以免除我们做一些交互操作.

首先安装expect： yum install expect

编写expect脚本。which expect可以看见这是在/usr/bin/expect

每次写脚本都要在首行加入#!/usr/bin/expect

内容：

spawn 这个是操作内容，后面接上需要在shell中操作的命令，比如su alice

expect 这个是只有apawn执行完后，才能捕捉到的信息。最最重要的是标准的结束信息，比如expect eof，这个一会介绍。比如上面su alice,那么系统就是提示Password： ,这个Password：就是我们要捕捉的信息，我们需要将其添加到expect后面，eg：expect “Password:”

send 这个是我们要在expect后面输入的信息，比如正常提示Password：后。我要输入密码123456，那么我们在send后面输入123456就是间接在expect后面输入信息了，

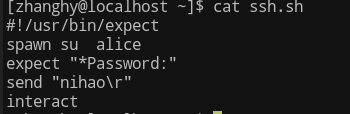
注意注意：\n是换行，\r是回车，我们正常情况输入密码后都要按回车键，所以这里脚本send后面也要加：

interact 会继续停留在交互状态，比如干完一些活后还需要我们手动交互的话，就用这个

expect eof 这个是终止交互，就意味着退出交互

### 1.下面演示登录用户：

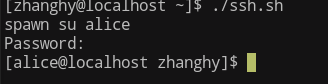
1. 执行expect脚本登录从zhanghy用户登录alice用户



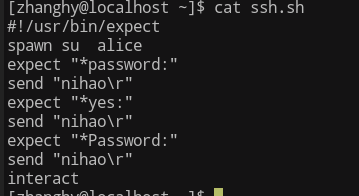
内容说明：

第一行不说了，第二行会执行命令切换alice，然后如果是系统会提示Password :

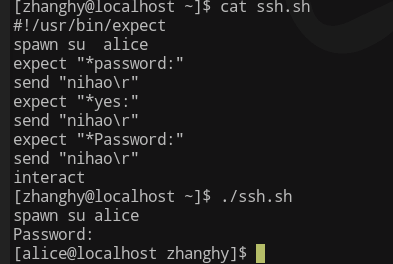
我这里用通配符\*加Password:来代替免提示了，然后将send的内容给这个expect，但是这个/r回车一定要加，不要不会继续往下走，会失败的。最后interact结束，下面演示（执行脚本无需交互操作直接进入alice用户）



但是我建议多次提示交互：失败了交互也没关系，比如我们加三个提示，只要有一个符合就登录alice:



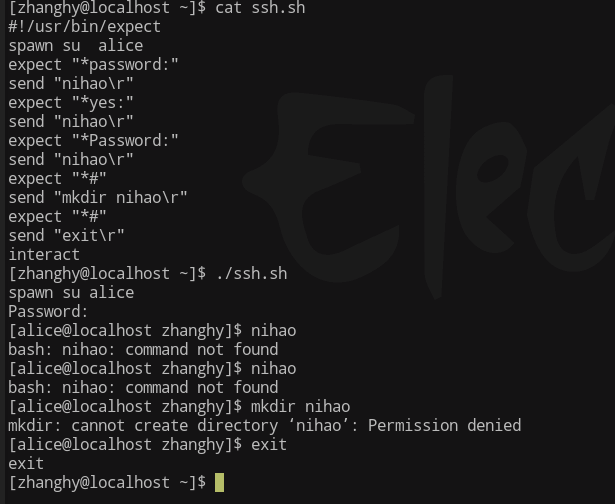
我们无法确认会提示什么，那就把每种可能都写上！



这个脚本由于是一行行由上到下执行的，所以加的提示越多越慢。

还是执行成功了。如果错误的放后面大不了就会提示错误，反正影响不大。不过最好删除其他提示，因为会影响执行速度。

### 登录用户并创建文件退出



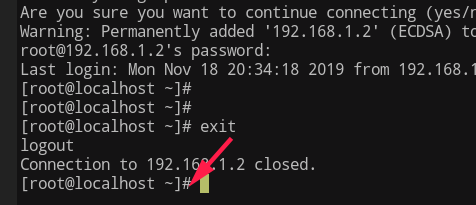
脚本解释：

expect \*#

send 内容

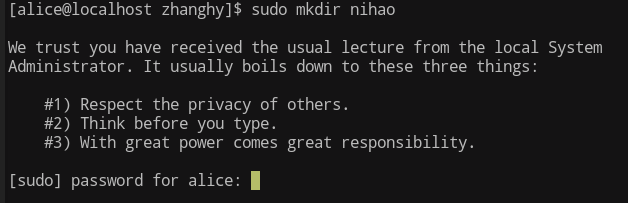
当我们登录用户或者其他用户，是不是会有提示符#

没错，这个expect #就是我们登录后的交互状态，这样我们就可以在其他用户做任何事

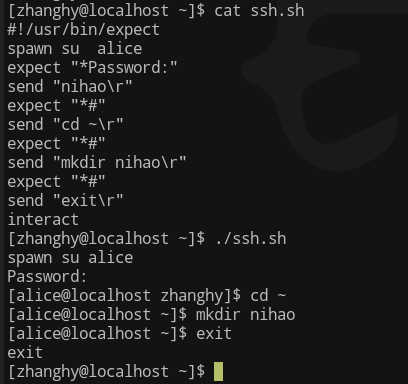


1.执行过程有点慢，所以执行的错误其他提示我建议删除。

2.创建目录权限不足，因为aice用户在zhanghy目录下创建确实禁止

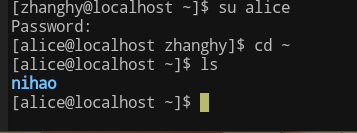


所以我加上提权，但是，哎，提示输入alice密码，所以估计不行，我们尝试切换到alice 目录



这次终于成功了。

我们查看

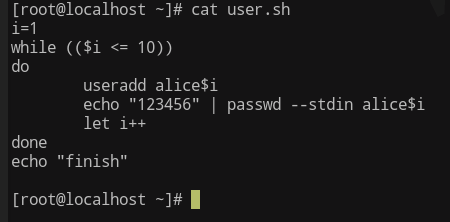


由此可以看见，这种自动免交互式操作真的挺方便的。

### 批量给每个用户根目录下面创建一个文件（结合大并发）

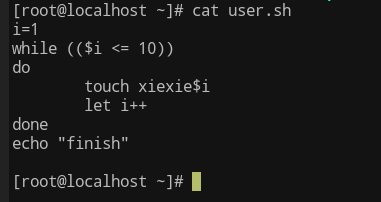
（本来打算创建用户加一起，但是创建用户必须是root，而root进入其他用户不需要密码）

批量创建用户



这个脚本真的很简单，如果要是文件呢？

touch nihao$i



记住，如果是批量创建nihao1xiexie 到nihao10xiexie 10个文件怎么办？

用{}，对，touch nihao${i}xiexie即可。

脚本开始（混合bash和expect）：

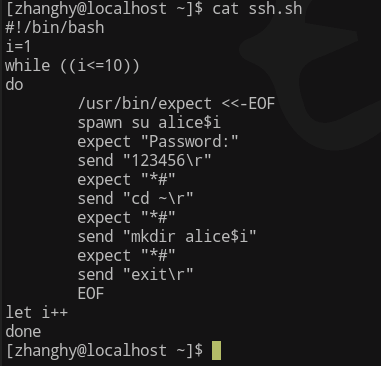
一个脚本如何添加另一个shell命令

加入 命令解释器 << -EOF

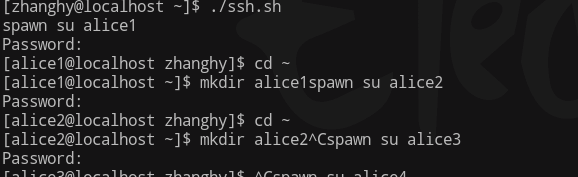
内容

EOF

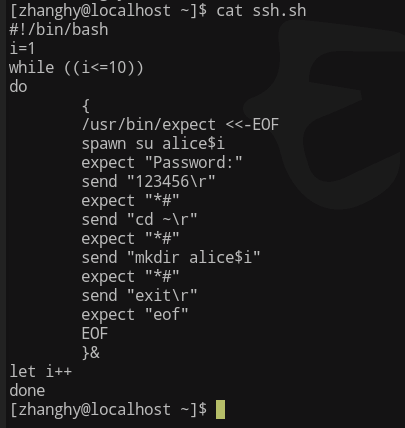
即可，记住，内容最后一条内容一定要是EOF，这是结束符



**注意，我没写结束符，这是失误，而且执行非常的慢**



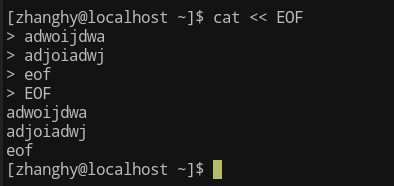
所以我这里尝试多并发，并且加上结束符



很快就完成了。哈哈。

但是还有一个疑问

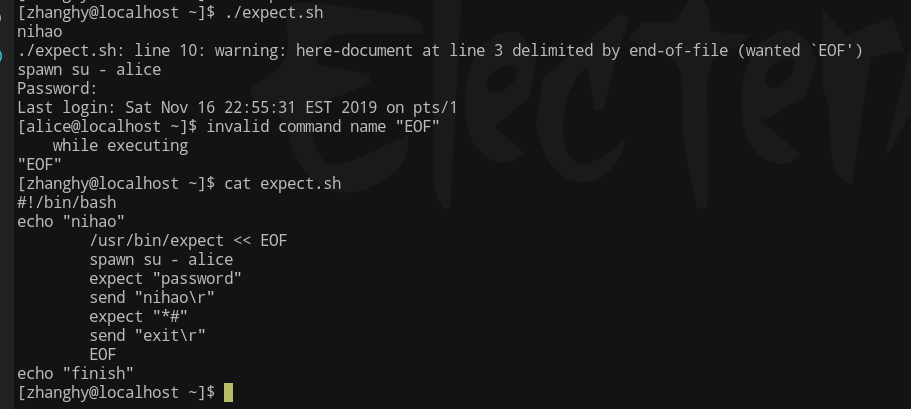
### 4.expect EOF与expect eof区别，以及EOF和-EOF区别：



EOF提示符号不是应该是EOF吗，为什么是expect eof也可以，我在shell中尝试发现

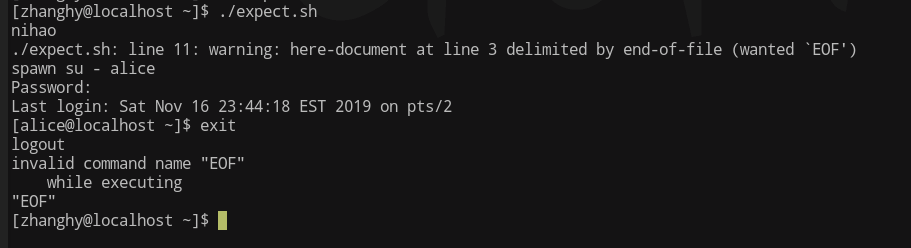
必须是EOF才可以。

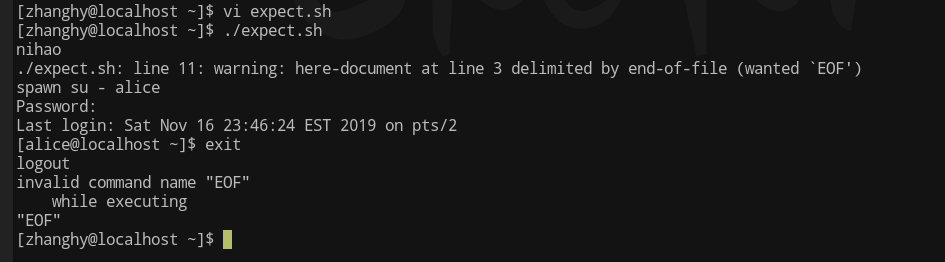
我发现我是真的蠢，这个-EOF和 EOF明明就是打印内容，和expect eof根本没关系，expect eof根本就是结束交互，这个expect 的一种用法。



果然这是必须要加的，不然会提示没有结束符

那么我试试加eof和EOF的区别？



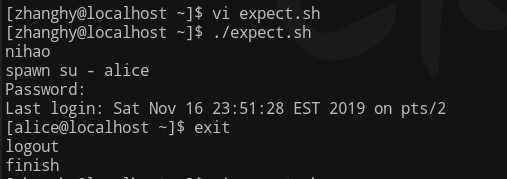


都一样，都提示没有结束符～

我猜想是-EOF的区别，因为我记得之前老师说如果缩进就必须用-EOF

那么加了-EOF再试试区别

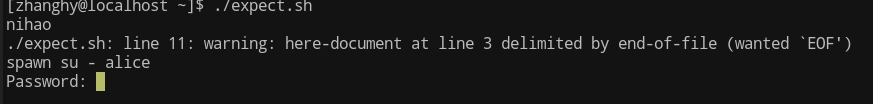
我擦，原来这二者没区别，只是我傻逼<<-EOF 写成<< -EOF，这个不需要加空格的。前后都不需要加。



现在完美执行。二个都可以啦

那么我再试试-EOF和EOF的区别

因为我缩进了，我用-EOF完美，我试试EOF看看



完美报错。看来那个老师没骗我，这个必须用—EOF才可以，总之以后能用-EOF就用他吧

### expect高级用法

1. exp\_continue
2. expect {}内容添加使用

3.expect “内容” 这个内容应该如何填写，可以模糊填写吗，比如一个关键词password 就可以吗

4,.变量及位置变量与延迟

1. 遇到条件判断该怎么办
2. **expect eof 与 interact（很多时候不继续执行都和这个有关）**
3. expect # send “内容”
4. 可以只用多条expect或者spawn吗？

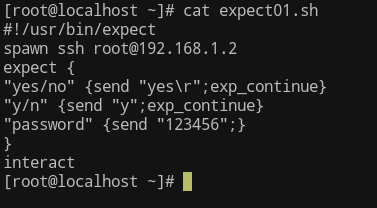
掌握上面8条内容，expect就算入门

注意：这个expect左边千万不要有空格，脚本符号查看可以用 **:set list查看**

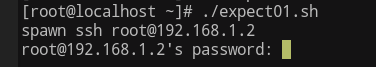


^是开始，$结束，^I是缩进

1.研究exp\_continue以及expect {}内容添加

1

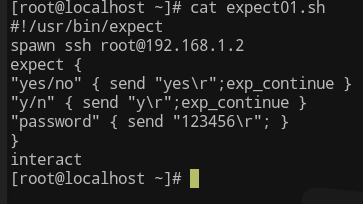
以上是我编写的脚本，要求是登录2服务器,但是2服务器有时候会提示 yes/no，这里我就用了exp\_continue



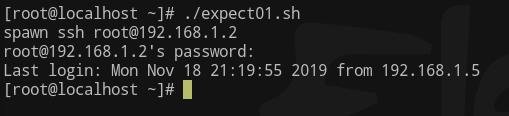
停留在这个界面无法使用了。

我改进了 { send 内容 } 将括号二边都加了空格，但还是不行

仔细一看，我去，原来是send 内容后面没加\r



修改完毕后：



完美执行。所以exp\_continue的效果果然很牛逼，如果有这条内容则执行，没有则继续执行下一行

我还作死将{send 内容}这二边的空格全删除了，发现完全没影响

这和do {} 完全一样啊。

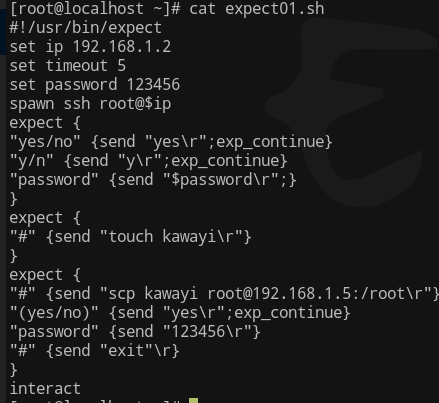
do {内容}

expect {内容}

这个二者是要加空格，或者换行，但是{内容}是不需要的，加了也没关系，毕竟

我查了which {不是关键词也不是命令

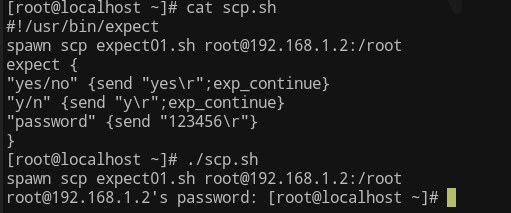
2可以使用多条expect {}吗？



这里我写了多条expect,其实也不用写那么多，就是想试试。

执行结果是成功创建成功，而且执行了scp，但是卡在了yes /no上面，这个很无语，总是卡在关键一步，之前的秘钥生成也是

**原因：没加interact**



发现不执行交互了

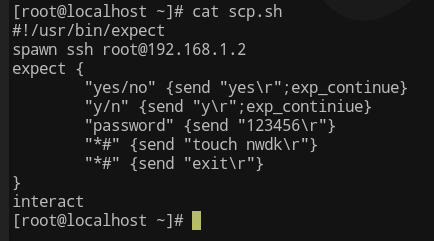
加了interact后：



成功了。

但是我用expect eof和interact互相换之后就不行。

1. 无法继续执行，这到底什么情况啊



我已经试过N遍了，根本不知道原因，只是说

expect: spawn id exp6 not open

while executing

"expect "#""

哎，根本不懂这个，照着视频一步步敲，但就是不行。到底哪里错了

实战：生成秘钥，传递一个主机的IP，即可传递给它公钥

1. 生成秘钥有二种交互性可能
2. 不存在秘钥

确认目录

再次确认回车

确认不要密码  
吗码

所以一共有三个enter

1. 存在秘钥

y

n

退出

确认不要密码同上

overwrite覆盖确认

确认目录

所以出现交互性多种选择时候该怎么办？

就需要用expect多种选择了，而不是一行行的敲

expect {

overwrite y/n: send “n\r”

eof

}exp\_continue

expect{

“passphrase” send “\r”

“again” send “\r”

eof

}

# 各种括号符号练习

这个一点都不熟悉啊，我打算举几个例子练习练习。

# 文件传输协议

**linux之间：**

**本机文件传输给其他主机**

**scp /root/a.sh** [**root@192.168.1.2:/root**](mailto:root@192.168.1.2:/root)

**当然如果有端口号的话，就加-p port,一般是22无需加**

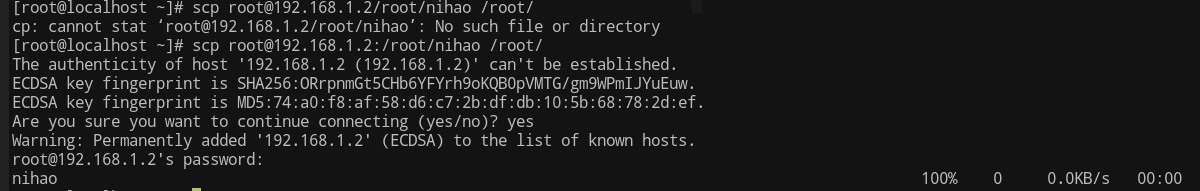
**如果root/a.sh只写成a.sh也可以，但必须在目录下，**

**：/root不加默认传输到root下，因为是root用户**

**获取其他主机文件：**

**很简单，文件放前面，目标放后面**

**scp root@192.168.1.2：/root/nihao /root/**

****

# 正则表达式

**用于代表字符串的内容**

## 基础字符

**^代表字符串以什么开头的（千万别理解成代表前面内容） 如： ^a 就会匹配到 a.sh**

**$ 代表以什么结尾 （千万别理解成代表后面内容） 如: .sh$ 就会匹配a.sh等内容**

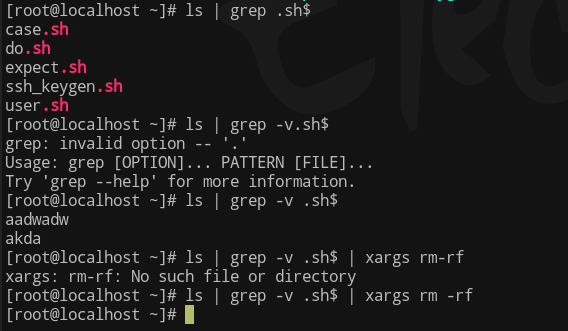
**我之前一直将这二个弄错,用法也弄错，这用法不是和通配符一样哦，通配符是文件才会使用。**

**^$ 代表空行,不是空格**

**如果是字符串，能加“”就加，这样就代表是字符串**

**用法演示。删除root下面除了脚本文件以外的文件。grep 无法选择文件，他只会筛选字符所以最好加“”啦，那这时候就需要正则表达式了**

**ls | grep -v .sh$ | xargs rm -rf**

****

**当然用通配符也可以**

**ls | grep -v “`ls \*.sh` “ |xargs rm- rf**

## 延伸符号：

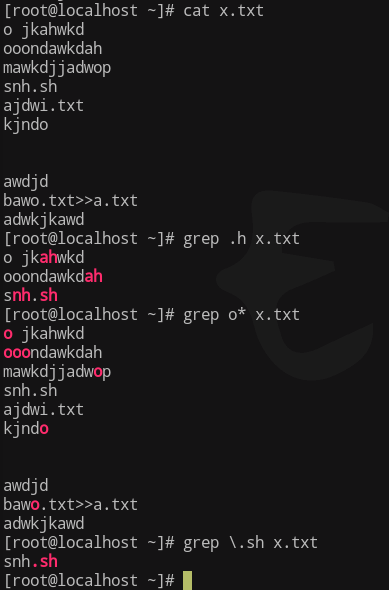
**. 匹配任意一个字符**

**\. 将.转义成单纯的.字符**

**\* 匹配一个或多个前面的字符。这个啥意思呢？**

**（比如我们匹配一个字符串有o 或者ooooo，或者没有o的，那么我们就用o\*）**

**如果用.\*就代表所有字符**

****

**如上图：**

**grep o\* x.txt 就会匹配所有行，因为有o的行和没o的行都匹配到**

**grep .h 就会匹配和h一起的字符**

**grep \.sh 就会匹配 .sh的内容字符**

**话说,ls 筛选出来的是文件名哦，不是字符串哦**

### 说一下.的用法

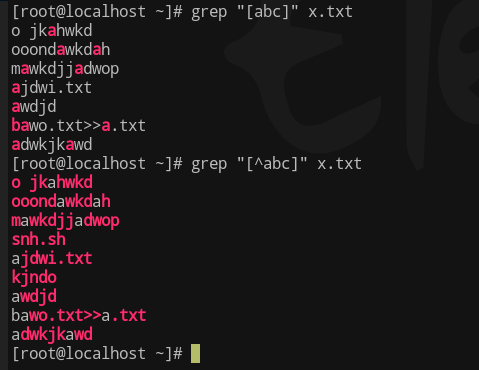
1. 用于当前目录 ，一般./就是当前目录下
2. 正则表达式
3. 隐藏文件
4. 执行文件或者让配置文件生效 . a.txt . /etc/profile

## 字符再延伸

[abc] 这个我好像学过，不是通配符吗

[abc] 匹配abc三个中任意一个字符

[^abc]匹配除了abc三个中的以外字符 这个^是取反



字符\{数字\} 匹配数字次字符

如 o\{3\} 匹配3次o



那么3次以上呢？

o\{3,\}就是了

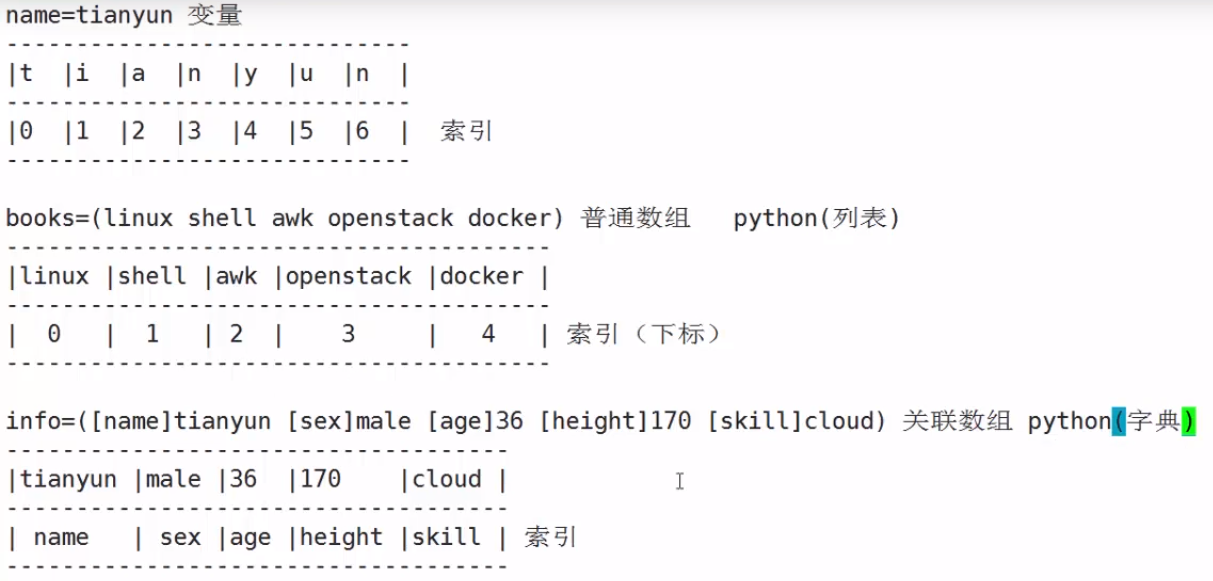
grep -v 除外选择

grep -n 打印行

grep -i 不区分大小写

grep -w 严格区分大小写

# 数组



数组定义，如上，用括号表示

数组引用 echo ${book[4]}即可打印docker

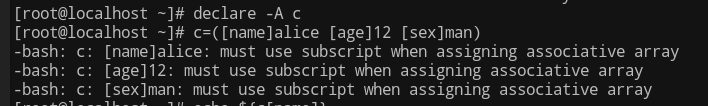
**但是关联数组呢**

echo {info[name]}

你以为这样就能打印tianyun了，不可能

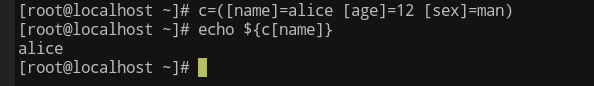
因为关联数组压根就不是这么定义的。

要先加declare -A 数组名



还是不行

正确应该是



下面肯定又是日常增删减改查的环节了：

## 查数组：

echo ${a[4]}

关联数组

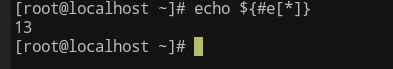
echo ${a[name]}

查看所有数组信息

echo ${a{\*}}

查看数组长度（这个跟查看变量长度一样）

echo ${#a{\*}



查看所有数组信息

declare -a

查看所有关联数组信息

declare -A

declare -a |grep a 查看a数组

## 给数组赋值：

alice=(a b c)

alice=(a b “c d”) 这是三个值

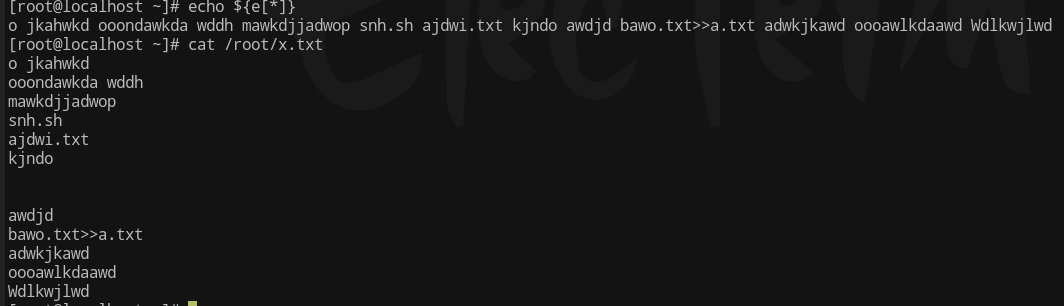
declare -A alice

alice=([a]=wdw [b]=awdaw [c]=awdd)

将文件内容赋值给列表

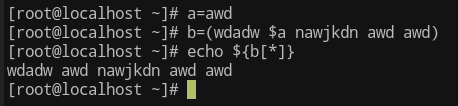
alice=(`cat /root/a.txt`) 将文本行赋值给alice每一个值

但是空格也会作为分割符号



将变量值赋值给数组

a=($z $x $c)



## 切片操作：

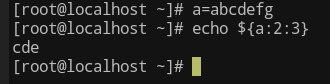
切取已备份数据来使用

比如变量内容

a=adcdefg

要切取cde的值怎么办：

echo ${a:2:3}

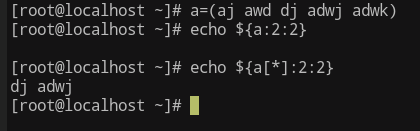


因为变量值也是从0开始

举一反三：

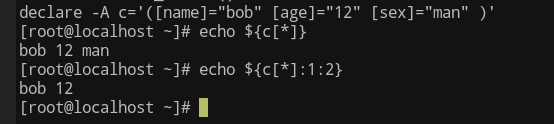
切取数组的值：

a=(aj awd dj adwj adwk)



要先打印所有制，然后开始切取关键值

切取关联数组的值呢：

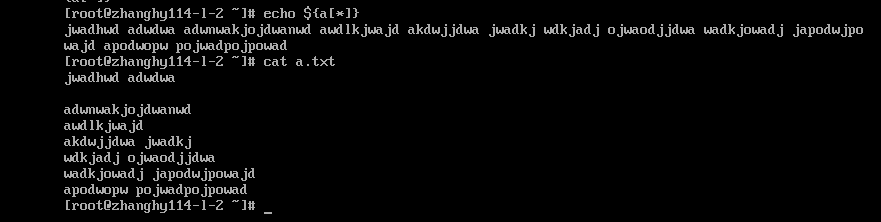


一样的，虽然他们是有键，但是值可以用顺序代替

遍历整个数组

a=(`cat a.txt`)

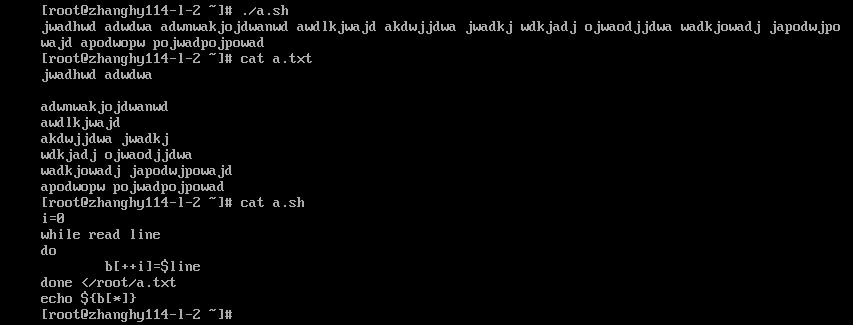
的确能够将文件每一行赋值给数组每一个值。但是如果文件有空格，他会将空格也当作分隔符

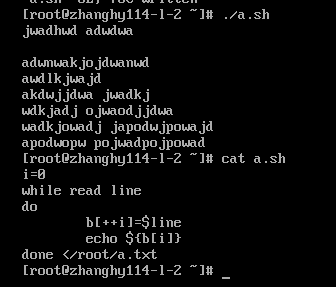


手动编写脚本将值赋值给数组，而且是一行一个值？这就需要循环了。

但是有一个问题，我运行这个脚本还必须是. a.sh

因为在子脚本中赋值变量是无法在shell中生效的。



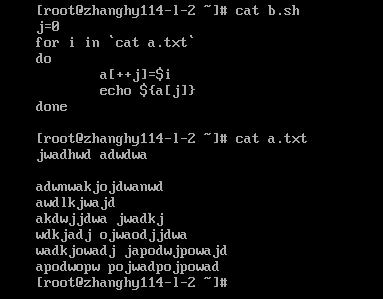


可以清楚得看见，这是以行为标准。

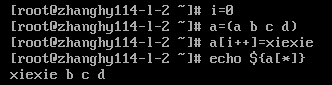
将一行为标准赋值给数组

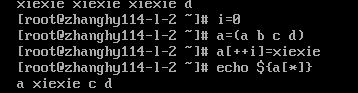
接下里我们已for为标准。

其实for是以空格作为分隔符号的。



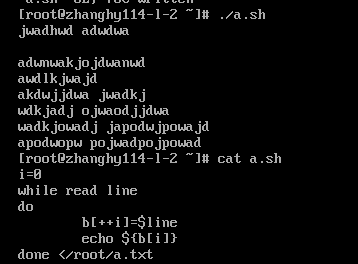
注意：数组遍历，注意的是a[++j]和a[j++].

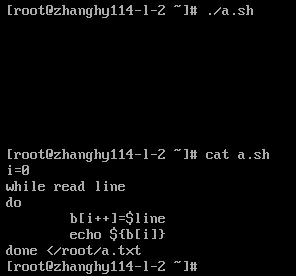




我当然知道二者区别。

但是问题是：为什么遍历必须用++j,而用j++就无效





我来教大家为啥只能用++i不能用i++

就以上面图片为例：

b[i++]=$line

是将b[0]=$line

然后我们打印 ${b[i]}这是打印b[1]的值。很遗憾，我还没有给b[1]赋值。

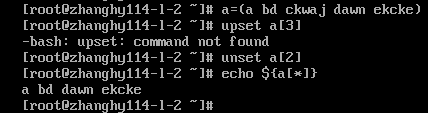
然后接下里我给b[1]赋值了，但是我打印的是b[2]

所以要打印全部，只有从do done 后面放置信息

## 数组的删

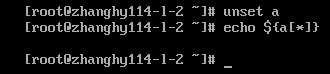
删除单个值

unset array[2]



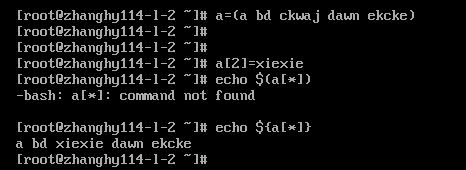
删除整个数组：

unset array



## 数组的改：

直接array[num]=值

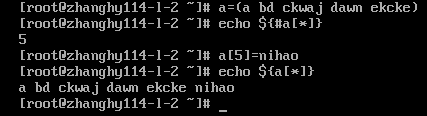


数组的增：

a=${#array[\*]}

先打印长度

array[a]=新增的值（因为数组的值是从0开始）



针对关联数组，这个还要定义

declare –A a (前提这个数组不存在)

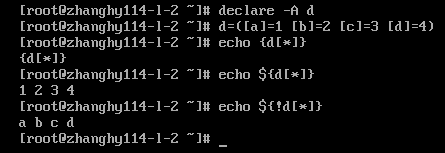
a=([a]=1 [b=2 [c]=3 [d]=4])

查看数组索引

echo ${!a[\*]}

查看数组值

echo ${a[\*]}



## 数组：功能：统计列表的值

eg:

|  |  |  |
| --- | --- | --- |
| Name | Age | Sex |
| Zhang | 12 | M |
| Ua | 12 | W |
| Dawd | 32 | W |
| Adwdw | 22 | M |
| Wda | 33 | W |
| dwa | 11 | x |

## 其实还要讲到一个功能let：

let 变量++

1. 变量不需要加$
2. 变量不需要赋值0，默认就是0

啥意思

比如

i=0

for I in seq 1 12

do

let m++

done

echo $m

这个是成立的

虽然m没定义，虽然没写$m,但是就是这样

结果 12

let 可以做很多运算

let i++

let a=a+2

let a\*=2

但是let进行运算很完美，但是没有输出

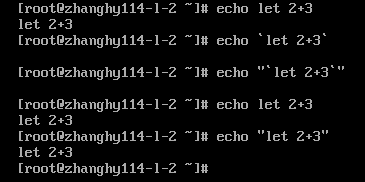
我们可以将输出给一个变量，然后打印 eg let a\*=10 echo $a

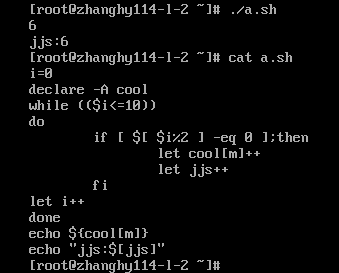
绝对绝对不要用echo let 运算

就算是echo `let 2+3`

echo “let 2+3”

都不行

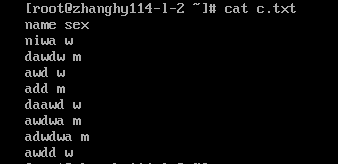




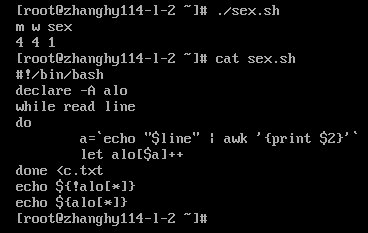
那么开始数组，统计上面表格男人数量和女人数量

eg:sex=([man]=12 [woman]=23)

怎么实现



就像这份表



我成功的统计了信息

# 10.函数

功能：方便代码复用化

函数必须先定义才可以使用

fangfu() {

内容

}

fangfu

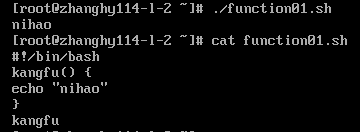
例如：

fangfu() {

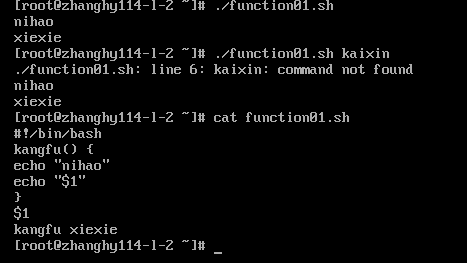
echo “nihao”

}

fangfu



函数的位置参数，如果函数里面有$!，注意这不是脚本的位置参数，这就是函数的。



传递方法，函数名 参数

## shell的C语言循环写法

shell写法：

#!/bin/bash

i=0

**while (($i<**=10))

do

let i++

done

其实也可以用c语言风格

for ((i=1;$i<=10;i++))

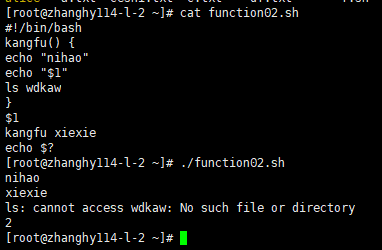
do

done

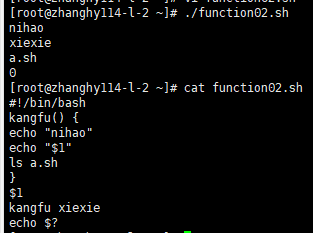
## 函数返回值

返回的是函数最后一条命令的值。

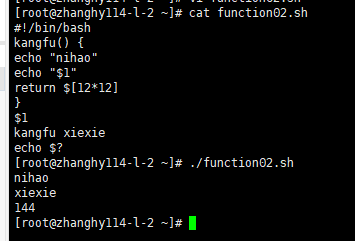
我故意将函数最后一条命令写错



我将函数最后一条命令写对

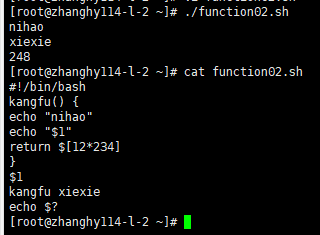


当然，手动定义返回值是最完美的



我return $[12\*12]，返回了144

但是要是$[12\*234]

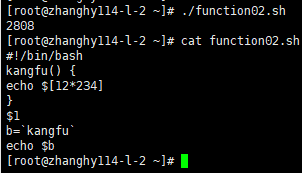


为什么返回值是248，因为返回值最大就是255

返回字符串可以吗？不可以

那怎么办？

将函数的结果打印出来：



传递位置参数：

函数传递位置参数和脚本传递位置参数用法一样，但是传递一定要注意，传递给脚本的实参无法传递给函数

if [ $# -ne 3 ];then

fi

传递参数，其实可以将数组值传递给函数。

num=(1 2 3 4 5 6)

cool() {

for I in $\*

do

done

}

cool ${num[\*]}

接下来演示结果：

其实就是

函数 ${a[\*]}

将数组的所有值当作函数的位置参数，而函数则使用 $i 来即可 ，这是不对的

因为$i代表数组的值 ,而$1代表的是数组位置，二者不可混淆

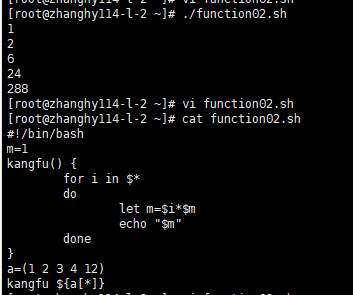
应该是 $\* 代表所有位置参数，

for I in $\*

do

echo $i 就可以遍历所有的位置参数了

done



# 11．循环扩展

break

这个我之前说过了，只要使用，就会跳出循环

但是在嵌套循环中，内循环跳出 就用break，但若是要一起跳出呢

break 2就是跳出2层循环

shift

位置参数每次向左移动一个，直到没有

啥意思呢？

比如我想传递给一个脚本位置参数，但是我自己都不知道要传递多少个，我就写一个函数，用shift将位置参数每次移动一个，第一次是$1,移动之后就是$2

那么演示：

脚本：

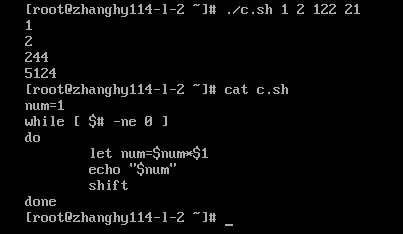
while [ $# -ne 0 ] $#表示位置参数个数

do

let num=$num\*$1

echo $num

done



如下图，可以传递任意参数，每次执行shift 就会删除最左边一个位置参数。

如果不加shift，这就是一个死循环。因为 $#就是代表参数，参数值不等于0就会一直循环

# 12.正则表达式

（建议网上找大量试题练习，匹配grep sed awk egrep）

用于匹配特定字符

元字符是最重要的概念，

正则表达式用于过滤日志工作，化繁为易，简单高效

文本处理工具：vim grep sed awk

基本正则表达式 BRE

扩展正则表达式 ERE

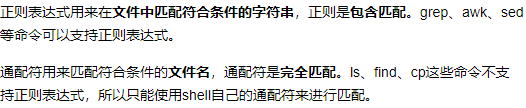
BRE只承认一些基础的元字符，如^ $ [] \* . 这些，要想用扩展的还要用\转义，比如 {} () 还要\{ \}

ERE则添加了() {} ? + |等

## 正则表达式和通配符区别：

正则表达式是对于字符，文本内容

而通配符用于文件 ls find cp处理文件





通配符的\*代表多个字符或0个，而正则的\*则表示前面一个字符，必须和字符一起使用

## 基本正则表达式元字符



**^ 放在字符前面，匹配以什么字符开头的字符串**

^ak



$搜索以字符串结尾

dw$



^$ 搜索空行，不是空格



. 代表且只能代表一个字符（不匹配空行）

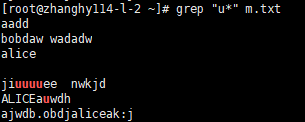


\. 转义字符，让有特殊字符变成单纯的字符。比如\.就是找.字符



\*

重复之前的字符0次或者多次



可以看到多有行都会打印出来，因为0次也会打印，有一次以上会重点标注

结合使用

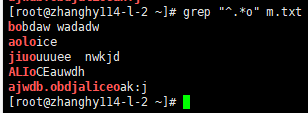
.\* 重复.0次或者多次，而.又代表任意字符，所以意思就是任意多个字符



grep “r.\*t” a.txt

这是搜索以r开头的t的字符段

^.\* 以任意字符开头，\*尽可能多，贪婪性。

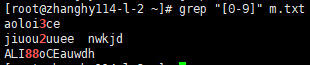


他会匹配O直到这一行没有o为止

括号表达式：

[1-3] 这是匹配1到3之间一个数，1或者2或者3

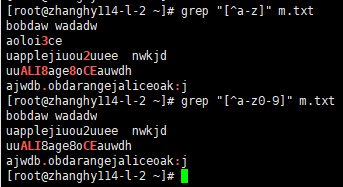
同理[0-9] [a-z] [A-Z] 也一样



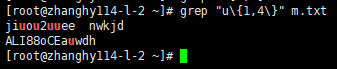
[^1-9a-z] 这是二者都会取反

[^1-9]这是只会对1-9取反

[a-Z]就是所有英文字母



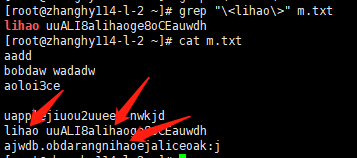
a\{n,m\} 匹配a字符n到m次，因为这是扩展正则，所以要加转义符号



a \{n} 就是n次

< 与 >

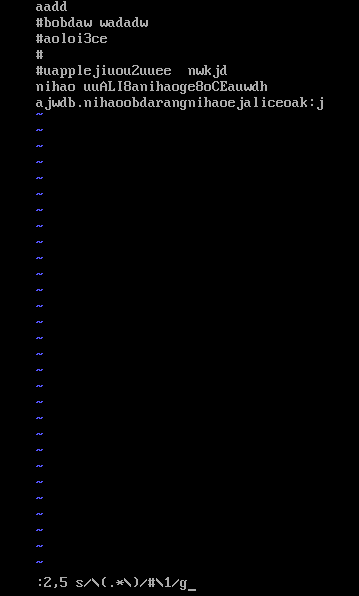
匹配 一个特定字符单词，而不是一个字符串的一部分



可以看到，文本中有三处lihao，但是有了这个限制单词符号，所以只有特定单词才会又结果。

\(内容\) 后面想引用这个内容 \1即可

比如想将5到10每一行前面加注释，怎么办



下面解释一下：

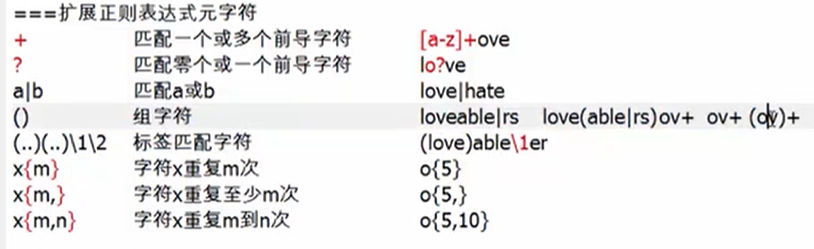
2,5表示2到5行

s g 表示整行替换

替换内容 \(.\*\)由于.\*代表任意字符，所以就是代表2到5行，但是()要转义，所以成了\(\)

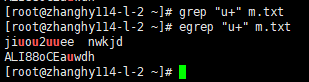
\1就代表\(.\*\)，这个也很简单，前面加#就是将内容替换

## 扩展正则表达式元字符

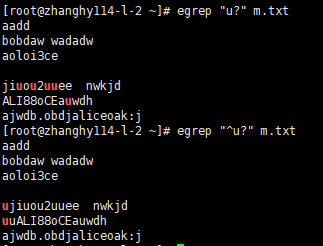


+ 前一个字符至少一次以上 ，但是由于是扩展，所以要加 –E,或者egrep

grep –E “u+” m.txt



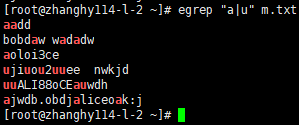
? 重复前面字符一次或0次（有且只有一次，但是多次分开好像也可以）



管道符（同时过滤多个字符）

a| b 同时筛选a和b

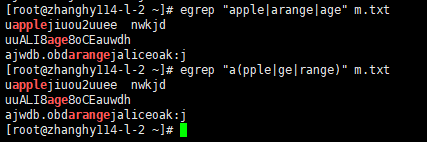
egrep “a|u” m.txt



（） 同时过滤括起来的一个整体 组字符

eg: a(ge|pple|range)

这样就是筛选age 和apple还有arang三个，类似age|apple|arrange



(txt)+ 多次txt字符出现，这个组合使用还是很有用的

空行：

^$ 代表空行

^[\t]\*$ 空行，但可能存在空格\t就是空格

注释行：

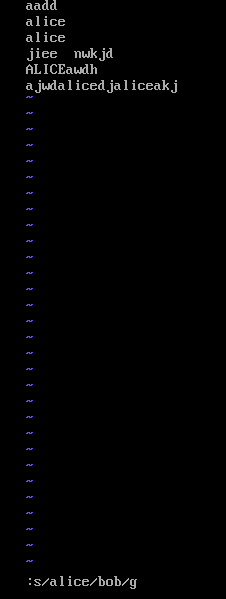
^# 代表注释行

# 13.vim，

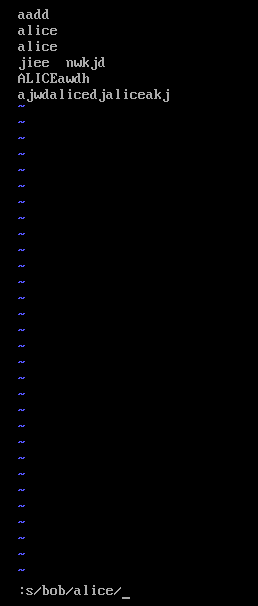
vim 替换文本字符

%：要替换的字符替换后的字符/ 注意：千万不要用\，我记得sed好像可以\也可以/，但是vim还是统一/

vim下面的编辑操作，其实也可以用:上下键来查看之前的输出信息，类似命令history查看一样。但是一旦退出再进去就没有了

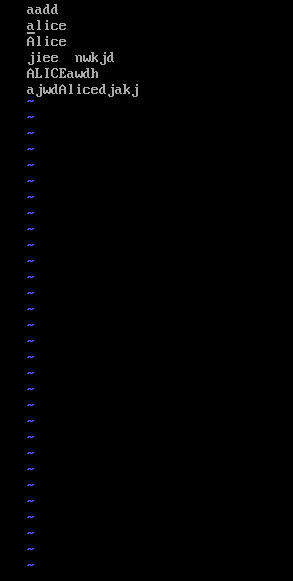


按方向键向上就是上一个输出



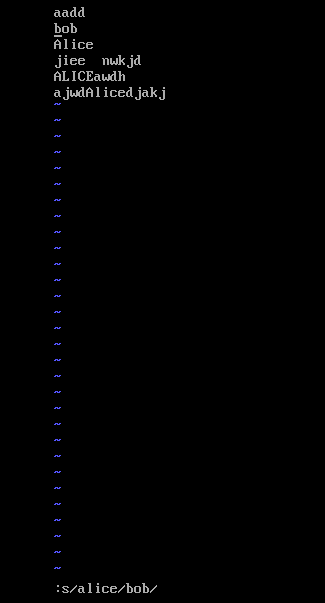
%是全文

那么我们实验一下



:s/alice/bob/

s是替换的意思

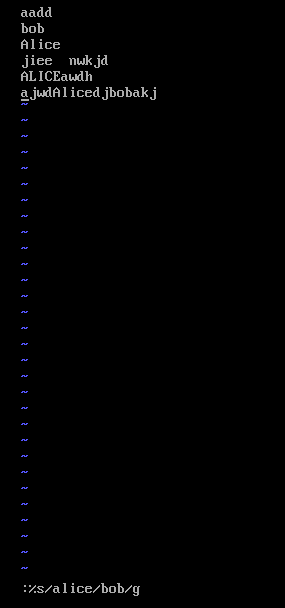


其实这个:s/内容/替换后的内容/只能替换一个，而且必须是光标所在行，如果光标不在所在行，则不会替换。

如果是

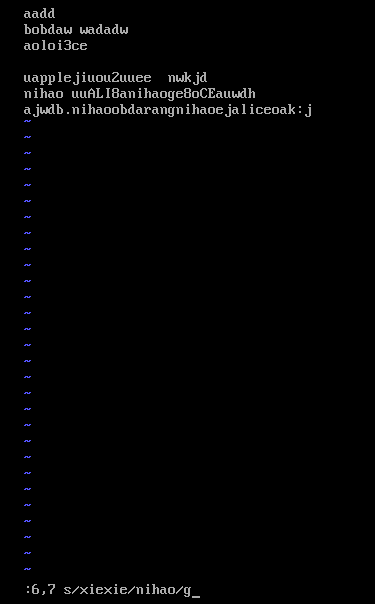
:s/alice/bob/g是什么呢，这个是将鼠标光标所在的这一行的所有的alice替换成bob。

再加全文%试试



成功了，看来全文全局都要用。

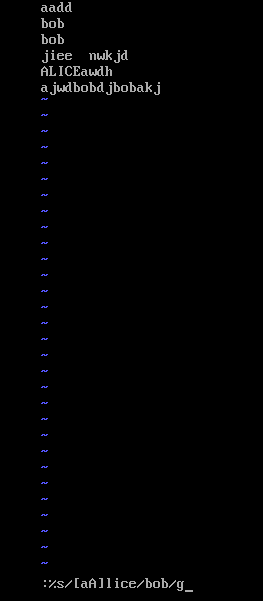
如果想换5-9行而不是全局怎么办：



使用

：5,9 s/内容/替换内容/g

但是如果想替换Alice不管大小写怎么办？



我这里就是使用正则表达式

# linux的一些其他字符作用

