# 从windows传文件

windows：

下载并安装 FileZilla

https://filezilla-project.org/download.php?type=client

Linux:

$ sudo apt-get update

$ sudo apt-get install openssh-server

windows中就可以使用FileZilla上传文件到虚拟机中的Linux中了

# ubuntu添加root用户登录

1. $ sudo gedit /usr/share/lightdm/lightdm.conf.d/50-ubuntu.conf

在编辑区添加：

greeter-show-manual-login=true

all-guest=false

1. $ sudo passwd root

输入设置root密码

1. $ cd /etc/pam.d

$ vim gdm-autologin

注释掉auth required pam\_succeed\_if.so user != root quiet\_success这一行，保存

$ vim gdm-password   
 注释掉 auth required pam\_succeed\_if.so user != root quiet\_success这一行，保存

1. $ vim /root/.profile

将文件末尾的mesg n || true这一行修改成tty -s&&mesg n || true， 保存

1. 重启系统，输入root用户名和密码，登录系统

# shell脚本

## 第一个shell脚本

#!/bin/bash  
echo "Hello World !"

#! 是一个约定的标记，它告诉系统这个脚本需要什么解释器来执行，即使用哪一种 Shell。

echo 命令用于向窗口输出文本。

## shell脚本的执行方式

* 作为可执行程序

将上面的代码保存为 test.sh，并 cd 到相应目录：

chmod +x ./test.sh #使脚本具有执行权限

./test.sh #执行脚本

***注意：***一定要写成 ./test.sh，而不是 test.sh，运行其它二进制的程序也一样，直接写 test.sh，linux 系统会去 PATH 里寻找有没有叫 test.sh 的，而只有 /bin, /sbin, /usr/bin，/usr/sbin 等在 PATH 里，你的当前目录通常不在 PATH 里，所以写成 test.sh 是会找不到命令的，要用 ./test.sh 告诉系统说，就在当前目录找。

* 作为解释器参数

这种运行方式是，直接运行解释器，其参数就是 shell 脚本的文件名，如：

/bin/sh test.sh

/bin/php test.php

这种方式运行的脚本，不需要在第一行指定解释器信息，写了也没用。

## shell中的变量

变量名和等号之间不能有空格；

可以不加$符号

your\_name="runoob.com"

可以用语句给变量赋值，如：

for file in $(ls /etc)

do

echo $file

done

以上语句将 /etc 下目录的文件名循环出来。

使用一个定义过的变量，只要在变量名前面加$符号即可

your\_name="qinjx"

echo $your\_name

echo ${your\_name}

花括号是可选的，是为了帮助解释器识别变量的边界

## SHELL常见的系统变量：

$0 当前程序的名称

$n 当前程序的第n个参数，n=1,2,...9

$\* 当前程序的所有参数（不包括程序本身）

$# 当前程序的参数个数（不包括程序本身）

$？ 命令或程序执行完后的状态，一般返回0表示执行成功

echo "\033[32m====================\033[0m"

#echo "\033[32m====================\033[1m" #1->down all is green

echo "The \$0 is $0"

echo "The \$1 is $1"

echo "The \$2 is $2"

echo "The \$? is $?"

echo "The \$\* is $\*"

echo "The \$# is $#"

## If条件语句

### 语法：

if condition1

then

command1

elif condition2

command2

else

commandN

fi

***注意事项：***

if (($NUM> 60)) //((计算)) / [[比较]] :可以是比较，也可以是计算

### 逻辑运算符解析：

-f 判断文件是否存在 eg：if [ -f filename ]

-d 判断目录是否存在 eg：if [ -d dir ]

-eq 等于 应用于：整型比较

-ne 不等于 应用于：整型比较

-lt 小于 应用于：整型比较

-gt 大于 应用于：整型比较

-le 小于等于 应用于：整型比较

-ge 大于等于 应用于：整型比较

-a and 都成立 逻辑表达式 -a 逻辑表达式

-o or 单方成立 逻辑表达式 -o 逻辑表达式

-z 空字符串

### 示例1：判断是否存在目录

#!/bin/bash

if [ -d ./test ];then

echo "test dir is exist"

else

echo "create test dir in myfile"

mkdir ./test

fi

### 示例2：判断成绩

#!/bin/bash

score=$1

if [ -z $score ];then

echo "Usage:{$0 60|80};"

exit

fi

if [[ $score -gt 85 ]];then

echo "very good"

elif [[ $score -gt 75 ]];then

echo "good"

elif [[ $score -ge 60 ]];then

echo "pass"

else

echo "no pass"

fi

### 示例3：备份mysql数据库

#!/bin/bash

#auto backup mysql db

#by authors nyf

#define var

BAK\_DIR=/data/backup/`date +%Y%m%d`

MYSQLDB=test1

MYSQLUSER=backup

MYSQLPW=123456

MYSQLCMD=/usr/bin/mysqldump

#check is root user

if [ $UID -ne 0 ];then

echo "Must to be use root for exec shell."

exit

fi

#check dir exist

if [ ! -d $BAK\_DIR ];then

mkdir -p $BAK\_DIR

echo "\033[32mCreate $BAK\_DIR successfully!\033[0m"

else

echo "\033[32mThis $BAK\_DIR is exist...\033[0m"

fi

#backup database commend

#mysqldump -ubackup -p123456 test1 > $BAK\_DIR/test1.sql

$MYSQLCMD -u$MYSQLUSER -p$MYSQLPW -d $MYSQLDB > $BAK\_DIR/$MYSQLDB.sql

#check pro is true

if [ $? -eq 0 ];then

echo "\033[32mThe Mysql Backup $MYSQLDB Successfully !\033[0m"

else

echo "\033[32mThe Mysql Backup $MYSQLDB Failed,Please check.\033[0m"

//创建备份用用户，并赋予权限

mysql> grant all on test1.\* to backup@'localhost' identified by "123456";

### 示例4 一键安装LAMP

一键源码安装LAMP脚本：

打印菜单：

1.安装apache web服务器

2.安装Mysql DB服务器

3.安装PHP服务器

4.整合LAMP架构并启动服务

1.Apache服务器安装部署

下载httpd-2.2.27.tar.gz版本，下载URL，解压，进入安装目录

configure;make;make install

2.Mysql服务器安装

下载mysql-5.5.20.tar.bz2版本，下载URL，解压，进入安装目录

configure;make;make install

3.PHP服务器安装

下载php-5.3.8.tar.bz2版本，下载URL，解压，进入安装目录

configure;make;make install

4.LAMP架构的整合和服务启动

/usr/local/apache2/bin/apachectl start

vi htdocs/index.php

<?php

phpinfo();

?>

#!/bin/bash

#Auto install LAMP

#print list

if [ -z $1 ];then

echo -e "\033[32mPlease select install Menu:\033[0m"

echo "1.install apache;"

echo "2.install mysql;"

echo "3.install php;"

echo "4.config index.php and start LAMP;"

echo -e "\033[34mUsage:{$0 1|2|...|help};\033[0m"

fi

if [ "$1" -eq "help" ];then

echo -e "\033[32mPlease select install Menu:\033[0m"

echo "1.install apache;"

echo "2.install mysql;"

echo "3.install php;"

echo "4.config index.php and start LAMP;"

echo -e "\033[34mUsage:{$0 1|2|...|help};\033[0m"

fi

#Http define path variable

H\_FILES=httpd-2.2.27.tar.bz2

H\_FILES\_DIR=HTTPD-2.2.27

H\_URL=http://mirrors.cnnic.cn/apache/httpd/

#Mysql define path variable

M\_FILES=mysql-5.5.20.tar.gz

M\_FILES\_DIR=mysql-5.5.20

M\_URL=http://down1.chinaunix.net/distfiles/

M\_PREFIX=/usr/local/mysql/

#PHP define path variable

p\_FILES=php-5.3.28.tar.bz2

p\_FILES\_DIR=php-5.3.28

p\_URL=http://mirrors.sohu.com/php/

p\_PREFIX=/usr/local/php5/

echo -e "\033[32m------------------------------------------\033[0m"

#install httpd web server

if [[ "$1" -eq "1" ]];then

wget -c $H\_URL/$H\_FILES && tar -jxvf $H\_FILES && cd $H\_FILES\_DIR &&./configure --prefix=$H\_PREFIX

if [ $? -eq 0 ];then

make && make install

echo -e "\n\033[32m-------------------------------\033[0m"

echo -e "\033[35mThe $H\_FILES\_DIR Server Install Success\033[0m"

else

echo -e "\033[35mThe $H\_FILES\_DIR Server Install Failed\033[0m"

exit 0

fi

fi

#auto install Mysql

if [[ "$1" -eq "2" ]];then

wget -c $M\_URL/$M\_FILES && tar -jxvf $M\_FILES && cd $M\_FILES\_DIR &&./configure --prefix=$M\_PREFIX

if [ $? -eq 0 ];then

make && make install

echo -e "\n\033[32m-------------------------------\033[0m"

echo -e "\033[35mThe $M\_FILES\_DIR Server Install Success\033[0m"

else

echo -e "\033[35mThe $M\_FILES\_DIR Server Install Failed\033[0m"

exit 0

fi

fi

#auto install php server

if [[ "$1" -eq "3" ]];then

wget -c $P\_URL/$M\_FILES && tar -jxvf $P\_FILES && cd $P\_FILES\_DIR &&./configure --prefix=$P\_PREFIX

if [ $? -eq 0 ];then

make && make install

echo -e "\n\033[32m-------------------------------\033[0m"

echo -e "\033[35mThe $P\_FILES\_DIR Server Install Success\033[0m"

else

echo -e "\033[35mThe $P\_FILES\_DIR Server Install Failed\033[0m"

exit 0

fi

fi

if [[ "$1" -eq "4" ]];then

sed -i '/DirectoryIndex/s/index.html/index.php index.html/g' $H\_PREFIX/conf/httpd.conf

$H\_PREFIX/bin/apachectl restart

echo "addType application/x-httpd-php .php" >> $H\_PREFIX/conf/httpd.conf

IP=`ifconfig eth1|grep "Bcast"|awk '{print $2}'|cut -d: -f2`

echo "You can access http://$IP/"

cat>$H\_PREFIX/htdocs/index.php <<EOF

<?php

phpinfo();

?>

EOF

fi

## for循环语句

### 示例1 循环seq中的数

#!/bin/bash

for i in `seq 1 15`

do

echo -e "\033[31mThe number is $i\033[0m"

done

### 示例2 0-100求和

#!/bin/bash

j=0

#for i in `seq 1 100` #true

for ((i=1;i<=100;i++)) #true

#for [[ i=1;i<=100;i++ ]] #false

do

j=`expr $i + $j`

done

echo $j

### 示例3 找到相关文件，批量打包

#!/bin/bash

for i in `find . -name "\*.sh"`

do

tar czf sh.tgz $i

done

### 示例4 远程主机批量传输文件

#!/bin/bash

for i in `seq 100 110`

do

echo -e "\033[33mscp -r /home/glzj/myFile/ glzj@127.0.0.1:/home/glzj/Download\033[0m"

done

#!/bin/bash

FILE=$\*

if [ -z $FILE ];then

echo -e "\033[34mUsage:{$0 /home|/home/glzj/test.sh|help}\033[0m"

exit

fi

for i in "127.0.0.1 192.168.80.129"

do

scp -r $FILE glzj@$i:/home/glzj/test2

done

### 示例5 远程主机批量执行命令

#!/bin/bash

for i in "127.0.0.1 192.168.80.129"

do

ssh -l glzj $i "ls /home" >> log.txt

done

## while循环语句

### 示例1 循环打印数字

#!/bin/bash

i=0

#while [[ $i -lt 10 ]]

while (($i<10))

do

echo -e "\033[35mThe number is $i\033[0m"

((i++))

done

### 示例2 while逐行读取某个文件

#!/bin/bash

while read line

do

echo -e "\033[32m$line\033[0m"

done < ifDir.sh

#!/bin/bash

while read addr

do

echo -e "\033[32mscp -r test glzj@$addr:/home/glzj\033[0m"

done < addr.txt

#!/bin/bash

for i in `cat addr.txt`

do

echo -e "\033[36m$i\033[0m"

done

#!/bin/bash

while read i

do

IP=`echo $i | awk '{print $2}'`

echo -e "\033[36m$IP\033[0m"

done < addr.txt

## Until循环

until: 为条件满足的时候，退出循环；

#!/bin/bash

i=10

until [[ $i -lt 0 ]]

do

echo "$i"

((i--))

done

## case选择

#!/bin/bash

case $1 in

apache)

echo -e "\033[32mYou select apache\033[0m"

;;

php)

echo -e "\033[32mYou select php\033[0m"

;;

\*)

echo -e "\033[32mUsage:{$0 apache|php|help}\033[0m"

;;

esac

## Select选择语句

Select一般用于选择菜单的创建，可以配合PS3来做菜单的打印输出信息

#!/bin/bash

PS3="Please select you excu mune:"

select i in "php" "java" "golong"

do

echo "You select $i"

done

#!/bin/bash

PS3="Please select your exec mume"

select i in "apache" "php"

do

case $i in

apache)

echo -e "\033[32mYou select apache\033[0m"

;;

php)

echo -e "\033[32mYou select php\033[0m"

;;

\*)

echo -e "\033[32mUsage:{$0 apache|php|help}\033[0m"

;;

esac

done

## 函数

[ function ] funname [()]{

action;

[return int;]

}

***说明：***

1、可以带function fun() 定义，也可以直接fun() 定义,不带任何参数。

2、参数返回，可以显示加：return 返回，如果不加，将以最后一条命令运行结果，作为返回值。return后跟数值n(0-255)，只能是int类型的数，如命令运行没错，即为0。

### 第一个函数

#!/bin/bash

myfunction(){

echo "This my first function"

}

echo "function start"

myfunction

echo "function end"

### 带return的函数

#!/bin/bash

function myfun(){

echo "This function is sum tow numbers"

read -p "first var a :" a

read -p "secound var b :" b

return $(($a+$b))

}

myfun

echo "a+b=$?"

**函数返回值在调用该函数后通过 $? 来获得**

### 带参数的函数

#!/bin/bash

function myfunc(){

echo "This is 1 para: $1"

echo "This is 2 para: $2"

echo "This is 10 para: $10"

echo "This is 10 para: ${10}"

echo "This is 11 para: ${11}"

echo "All paras numbers: $#"

echo "This is all para: $\*"

}

myfunc 1 3 4 5 6 7 8 9 12 13 14 15 16

注意，$10 不能获取第十个参数，获取第十个参数需要${10}。当n>=10时，需要使用${n}来获取参数；

### 函数中的一些特殊符号

|  |  |
| --- | --- |
| 参数处理 | 说明 |
| $# | 传递到脚本的参数个数 |
| $\* | 以一个单字符串显示所有向脚本传递的参数 |
| $$ | 脚本运行的当前进程ID号 |
| $! | 后台运行的最后一个进程的ID号 |
| $@ | 与$\*相同，但是使用时加引号，并在引号中返回每个参数。 |
| $- | 显示Shell使用的当前选项，与set命令功能相同。 |
| $? | 显示最后命令的退出状态。0表示没有错误，其他任何值表明有错误。 |

## 数组

Shell 数组用括号来表示，元素用"空格"符号分割开，语法格式如下：

array\_name=(value0 value1 ... valuen)

#!/bin/bash

soft=(

jdk.tar.gz

tomcat.tar.gz

nginx.tar.gz

)

echo "This soft total is ${#soft[@]}" //元素总数

echo "This soft total is ${#soft[\*]}"

echo "This soft all is ${soft[@]}" //所有元素列表

echo "This soft all is ${soft[\*]}"

echo "This soft first is ${soft[0]}"

tar -xzf ${soft[2]} ; cd nginx; ./configure ; make install //一般的用法

echo "Update soft two is ${soft[@]/tomcat.tar.gz/tomcat.tgz}"

echo "Delete soft two is"

unset soft[1]

echo ${soft[\*]}

## shell编程四剑客

### sed：替换

替换文件中的字符

使用vi打开文件替换：

:%s/oldstring/newstring/g

:%s/oldstring/newstring/

不打开文件，使用sed修改：

$ sed ‘s/oldstring/newstring/g’ filename //预修改，不会真的去修改

$ sed -i ‘s/oldstring/newstring/g’ filename //加参数-i，修改文件

$ sed ‘s/^/& /g’ test.txt //给test.txt文件中的每行的开头加上空格

$ sed ‘s/$/& id/g’ test.txt //给每行的结尾加上 空格id

$ sed ‘/wgk/a ###’ test.txt //给wgk行之后加上一行，输入###

$ sed ‘/wgk/i ###’ test.txt //给wgk行之前加上一行，输入###

$ sed ‘/wgk/p’ test.txt //print包含wgk的行

$ sed -n ‘1p’ test.txt //print第1行

$ sed -n ‘1,5p’ test.txt //print第1-5行

//查看文件，把空格改成换行，去掉空行，排序由大到小，取第一行和最后一行

$ cat number.txt |sed ‘s/ /\n/g’ |grep -v “^$” |sort -nr | sed -n ‘1p;$p’

### grep：行筛选

$ cat number.txt |grep “45” //打印包含45的行

$ cat number.txt |grep -v “45” //打印不包含45的行

$ grep “11$” //匹配以11结尾的行

$ grep “^10” //匹配以10开头的行

$ cat test.txt |grep -E “([0-9]{1,3}\.){3}[0-9]{1,3}” //匹配ip地址

$ egrep “11|10” test.txt //匹配11或者10的行，egrep即grep -E

### awk：列筛选

this is my file //test.txt

$ cat test.txt |awk ‘{print $1}’ //this

$ cat test.txt |awk ‘{print $NF}’ //取最后一列

//etc/passwd文件，把 : 变成空格，取第一列的元素

$ cat /etc/passwd |sed ‘s/:/ /g’ |awk ‘{print $1}’

$ cat /etc/passwd |awk -F: '{print $1}' //以 : 分割，第1列

$ ifconfig ens33 |grep “netmask” |awk '{print $2}' //获取ip

$ df -h |grep "/$" |awk '{print $5}' |sed 's/%/ /g' //获取根路径文件利用率

$ cat test.txt|awk '{print "id: "$2}' //给筛选出的内容前添加”id: ”

### find：查找文件/目录

$ find path -name "\*.txt" //寻找path路径下，.txt文件

$ find . -maxdepth 1 -name “test.txt” //寻找当前目录下，第一级中的，名为test.txt文件

$ find . -maxdepth 1 -type f -name “\*.txt” //寻找类型为file，名字为 .txt 的文件

$ find . -maxdepth 1 -type f -name “\*.txt” -mtime +30 //寻找30天前的文件

$ find . -maxdepth 1 -type f -name “\*.txt” -mtime -1 //寻找1天内的文件

$ find . -maxdepth 1 -type f -name “\*.txt” -mtime -1 -exec rm -rf {} \; //delete找到的文件

$ find . -maxdepth 1 -name “\*.txt” -mtime -1 -exec cp {} /tmp/ \; //copy找到的文件到/tmp/

$ find . -maxdepth 1 -name “\*.txt” |xargs rm -rf {} \; //delete找到的文件，xargs不能copy

$ find . -maxdepth 1 -size +50M -type f //查找大于50M的文件

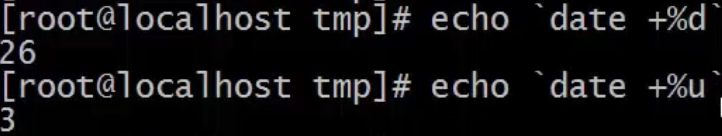
$ find . -maxdepth 1 -size +50M -type f -exec cp {} /download/ \;

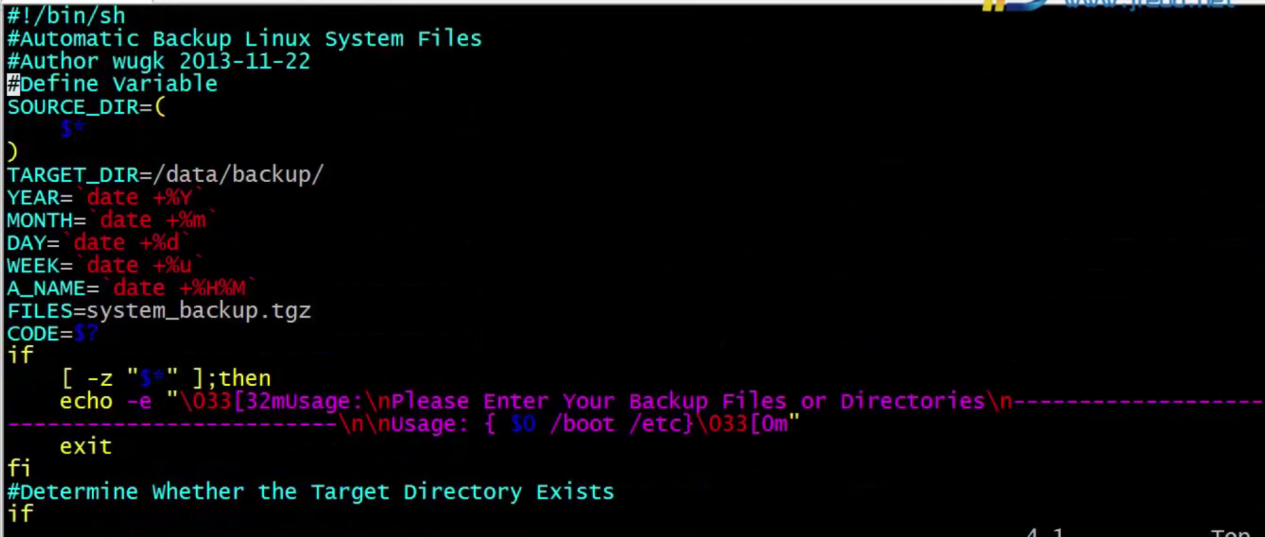
//把大于50M的文件copy到/download/中

## 备份

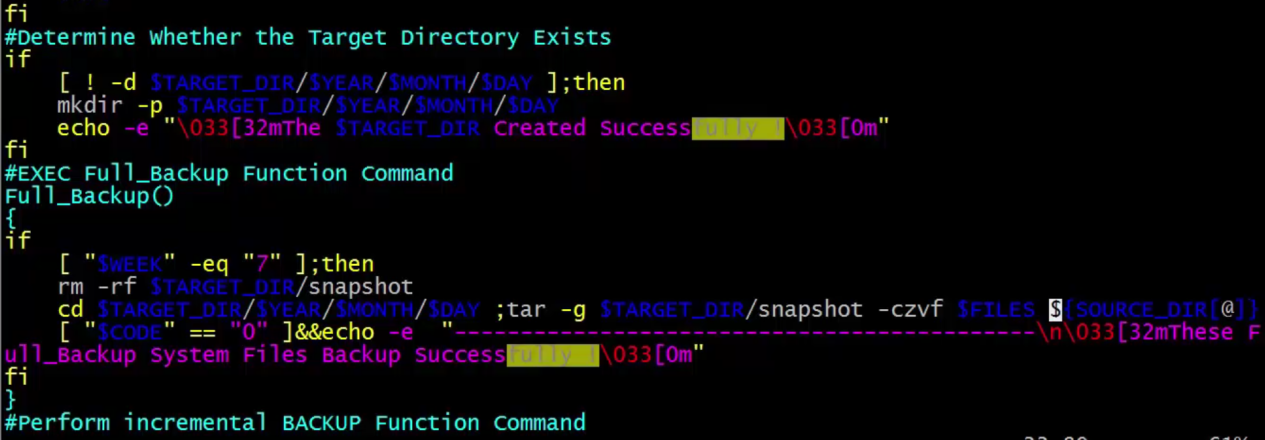
完整备份、增量备份



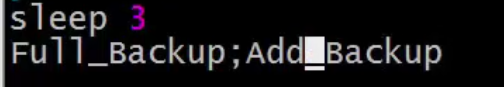






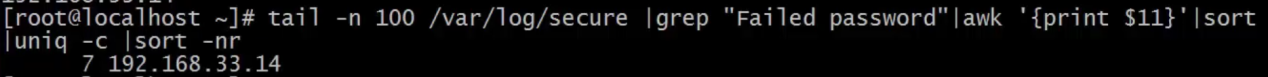


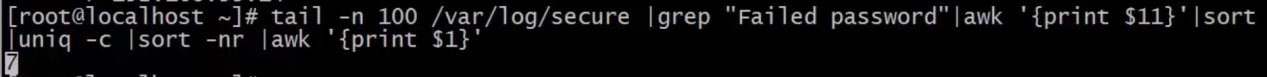




## 防止恶意攻击服务器

$ tail -n 100 /var/log/syslog

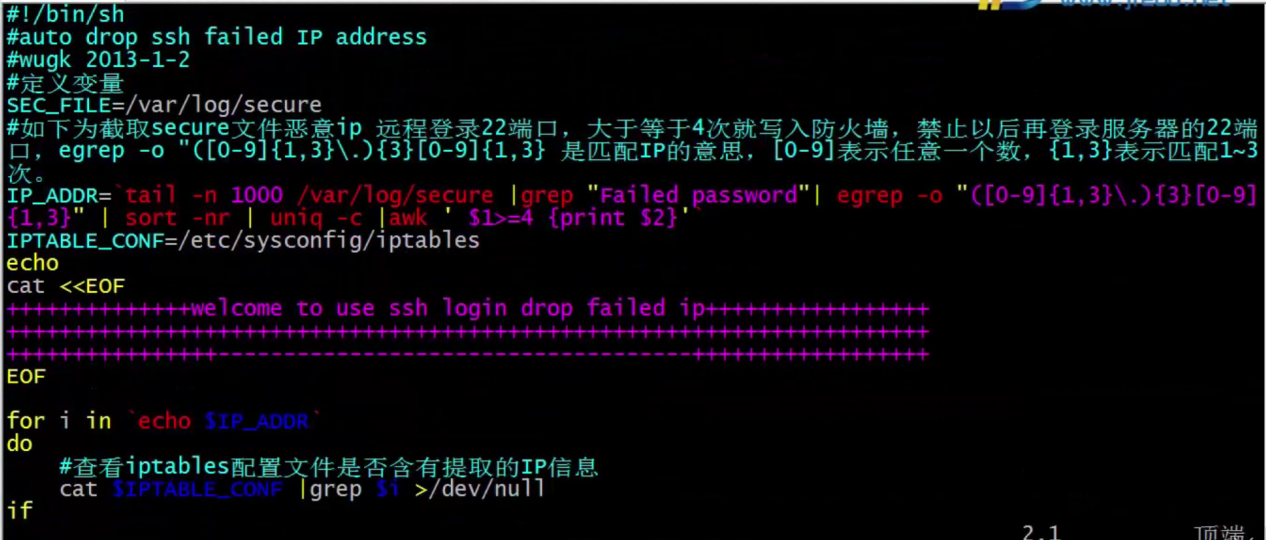








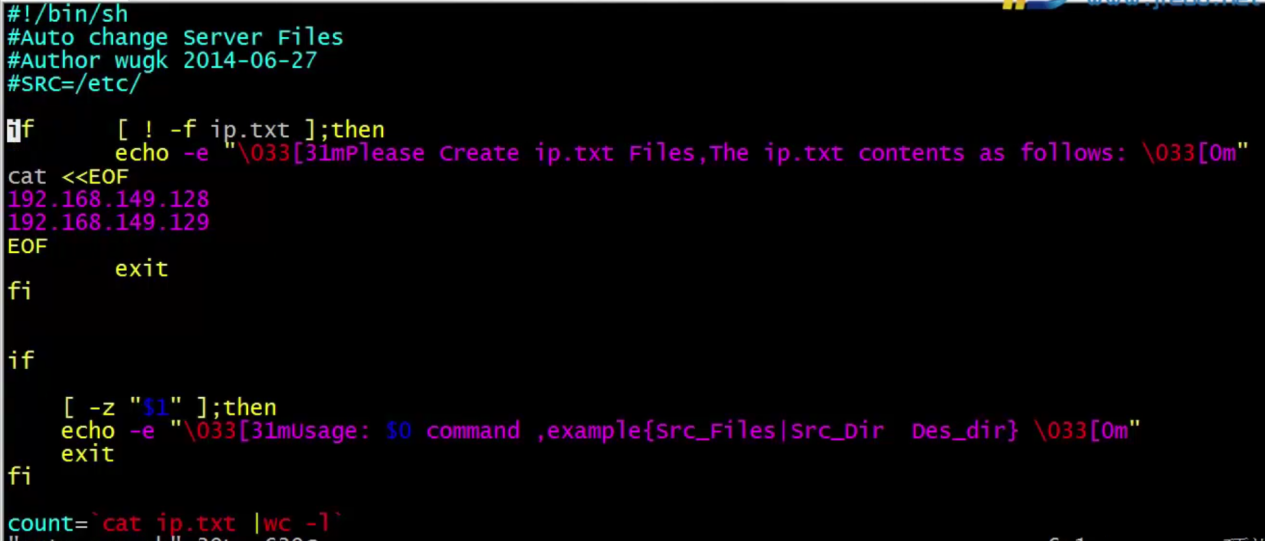


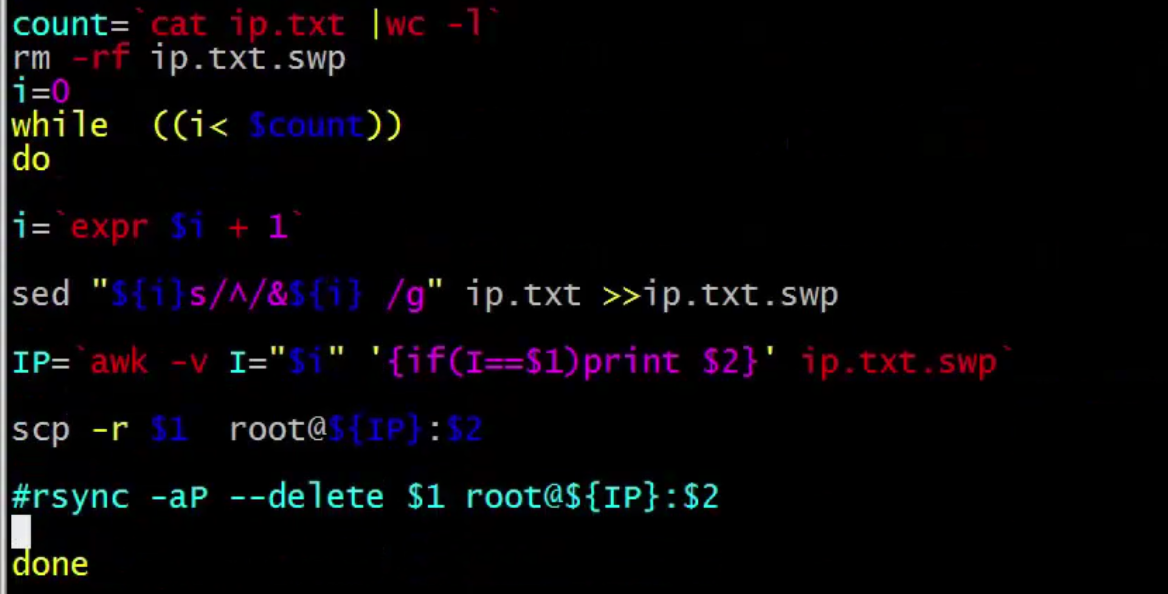




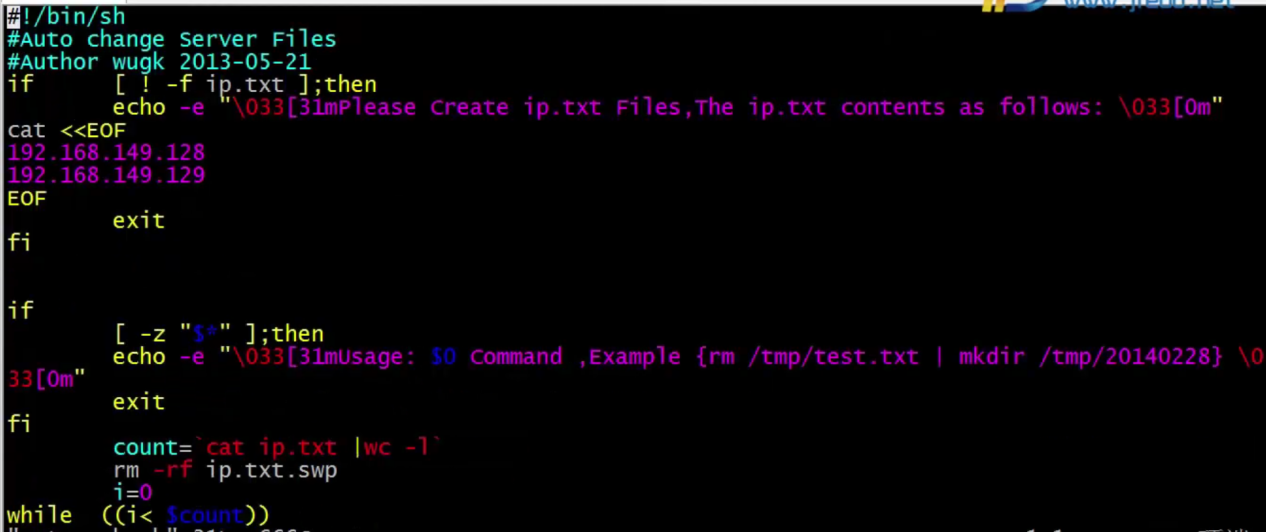
## 网站自动化部署管理

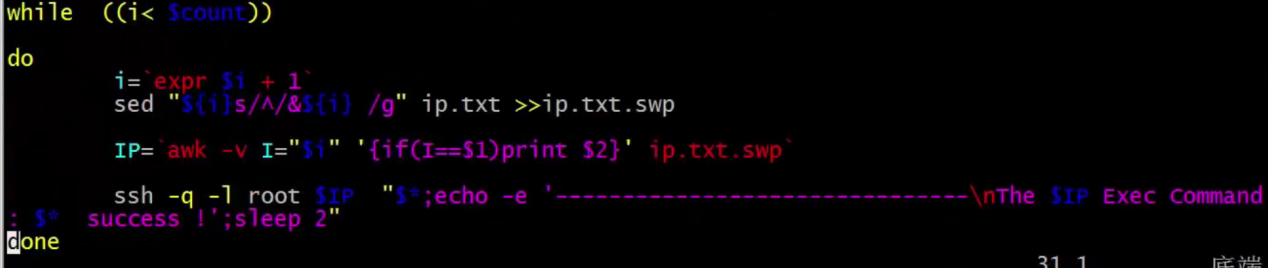
***同步文件到多个服务器：***





***批量远程服务器执行命令：***





***Rsync+ssh自动部署***

# 附录 shell命令参数

1. $ /bin/bash -x for2Sum.sh //查看脚本执行的过程
2. $ /bin/bash -n filename.sh //验证脚本有没有问题
3. tar czf all.tgz \* //打包目录中的所有文件
4. tar xzf all.tgz -C ./test //解压到，./test目录中
5. find . -name "\*.tgz" //查找当前目录中，以 .tgz 结尾的文件
6. find . -name "\*.sh" | tail -2 //查找当前目录中，以 .sh 结尾文件的后两条数据
7. ssh-keygen //两个linux之间使用ssh免密传输时使用
8. scp -r test.sh [glzj@127.0.0.1:/home/glzj](mailto:glzj@127.0.0.1:/home/glzj)

//远程cp文件，-r 表示可以是目录也可以是文件

1. ssh -l glzj 127.0.0.1 'ls /home' //远程主机执行命令
2. read -p "Please input numbers:" input | read input

//读取数据存入变量input；（2.7.2示例2 while逐行读取某个文件）

1. echo a b c | awk '{print $2}' //取第二个参数，用于read读值的时候

# 附录 一些技术资料、博文

1. 两个linux之间使用ssh免密传输时使用：

<https://blog.csdn.net/xinshui151/article/details/79187563>