Linux常用命令：

pwd显示当前所在目录，

查看文件的命令后面加-h可以显示文件大小，后面加-a显示所有文件包括隐藏文件

mkdir -p 是递归创建；

cd - 返回上次目录

rmdir 删除空白目录 ，rm –r 删除目录，-f 强制删除

cp 拷贝；-r 拷贝目录；-p连带文件属性复制；-d若文件是连接文件则复制连接属性，-a 相当于-pdr，所有内容属性都一样。

touch 将已存在文件的时间标签更新为当前时间， 创建新的空文件夹

locate 文件名，在后台数据库（每天更新，updatedb强制更新数据库）中按文件名搜索，速度快，但只能按照文件名搜索。在/etc/updated.com中配置搜索限制条件。

Whereis 命令名：搜索系统命令所在路径及帮助文档所在位置，-b只查找可执行文件，-m只查找帮助文件，which同样的作用，但which会显示别名。

find：find 范围 条件；eg：find / -name game.log 在根目录下按文件名搜索，很耗费资源。通配符：\*匹配任意内容；？匹配任意一个字符；[]屁匹配任意一个中括号内的字符；

-iname不区分大小写。-user按所有者搜索。atime文件访问时间，ctime改变文件属性，mtime改变文件内容，-10 十天内修改的，10十天当天修改的，+10十天前修改的eg：find /game/game.log -mtime +10。Find . –size 25k:查找当前文件夹下文件大小是25kb的文件

grep：grep 选项 字符串 文件名：在文件中匹配符合条件得字符串；选项：-i 忽略大小写，-v 排除指定字符串。

mount：查询系统中已经挂载的设备

w：查看本机用户；last：查看过去和正在登陆的所有用户信息已经服务器重启时间。

程序的第一行：#!/bin/sh（不可省略）

给文件赋予可执行权限：编译chmod +x filename这样才能用./filename来运行

注释：#

打印num变量：num=2

echo “this is the ${num}nd”

将打印：this is the 2nd

echo"sometext":将文字内容打印在屏幕上

-eq           //等于

-ne           //不等于

-gt            //大于

-lt            //小于

ge            //大于等于

le            //小于等于

ls:文件列表

wc –l file ;wc -w file; wc c file:计算文件行数 计算文件中的单词数 计算文件中的字符数

cp sourcefile destfile:文件拷贝

mv oldname newname:重命名文件或移动文件

rm file:删除文件

grep 'pattern' file:在文件内搜索字符串比如：grep 'searchstring' file.txt

cut -b colnumfile:指定欲显示的文件内容范围，并将它们输出到标准输出设备比如：输出每行第5个到第9个字符cut-b5-9 file.txt千万不要和cat命令混淆，这是两个完全不同的命令

cat file.txt:输出文件内容到标准输出设备（屏幕）上

file somefile:得到文件类型

readvar:提示用户输入，并将输入赋值给变量

sort file.txt:对file.txt文件中的行进行排序

uniq:删除文本文件中出现的行列比如：sort file.txt|uniq

expr:进行数学运算Example:add 2and3 expr 2"+"3

find:搜索文件比如：根据文件名搜索find filename -print

tee:将数据输出到标准输出设备(屏幕)和文件比如：somecommand|teeoutfile

basenamefile:返回不包含路径的文件名比如：basename/bin/tux将返回tux

dirnamefile:返回文件所在路径比如：dirname/bin/tux将返回/bin

head file:打印文本文件开头几行

tail file:打印文本文件末尾几行

sed:Sed是一个基本的查找替换程序。可以从标准输入（比如命令管道）读入文本，并将结果输出到标准输出（屏幕）。该命令采用正则表达式（见参考）进行搜索。不要和shell中的通配符相混淆。比如：将linuxfocus替换为LinuxFocus：cat text.file|sed's/linuxfocus/LinuxFocus/>newtext.fileawk:awk用来从文本文件中提取字段。缺省地，字段分割符是空格，可以使用-F指定其他分割符。catfile.txt|awk-F,'{print$1","$3}'这里我们使用，作为字段分割符，同时打印第一个和第三个字段。如果该文件内容如下：AdamBor,34,IndiaKerryMiller,22,USA命令输出结果为：AdamBor,IndiaKerryMiller,USA

重定向：将命令的结果输出到文件，而不是标准输出（屏幕）。>写入文件并覆盖旧文件 >>加到文件的尾部，保留旧文件内容

用来查找过去24小时（-mtime–2则表示过去48小时）内修改过的文件。如果您想将所有查找到的文件打一个包，则可以使用以下脚本：

#!/bin/sh

#The ticks are backticks(‘)not normalquotes('):

tar-zcvflastmod.tar.gz’find.-mtime-1-typef-print’

通常用"[]"来表示条件测试。注意这里的空格很重要。要确保方括号的空格。

[ -f "somefile"]：判断是否是一个文件

[ -x "/bin/ls"]：判断/bin/ls是否存在并有可执行权限

[ -n "$var"]：判断$var变量是否有值

[ "$a" = "$b"]：判断$a和$b是否相等

&&就是一个快捷操作符，如果左边的表达式为真则执行右边的语句,||则左边为假时执行右边的语句。不用与和或操作符，我们也可以用if表达式作任何事情，但是使用与或操作符会更便利很多

Shell命令和流程控制，if。。。。。fi ；case。。。。。esac ； select

if用法常规；

case 匹配母板 in  
模板1 [ | 模板2 ] … ) 语句组 ;;  
模板3 [ | 模板4 ] … ) 语句组 ;;  
esac

代码如下：

#!/bin/sh  
echo "Please input \"yes\" or \"no\""  
read var  
case "$var" in  
[yY][eE][sS] ) echo "Your input is YES" ;;  
[nN][oO] ) echo "Your input is no" ;;  
\* ) echo "Input Error!" ;;  
esac  
exit 0

用法：

select var in ... ; do

　break;

done

.... now $var can be used ....

示例代码：

#!/bin/bash

echo "What is your favourite OS?"

select var in "Linux" "Gnu Hurd" "Free BSD" "Other"; do

break;

done

echo "You have selected $var"

该脚本的运行结果如下：

What is your favourite OS?

1) Linux

2) Gnu Hurd

3) Free BSD

4) Other

#? 1

You have selected Linux

Loop用法：

While循环：例：从1加到100

#!/bin/bash

# desc : while loop

sum=0

i=0

while [ $i -lt 100 ]

do

i=$(($i+1));

sum=$(($sum+$i))

done

echo "From 1 to 100, sum is : " $sum””

until循环与while相反，条件成立时终止循环。例：

#!/bin/bash

# desc : while loop

until [ "$yn" == "yes" -o "$yn" == "YES" ]

do

read -p "Please input yes/YES to stop : " yn

done

echo "OK"

执行：

[work@www sh]$ sh while.sh

Please input yes/YES to stop : hi

Please input yes/YES to stop : yes

OK

[work@www sh]$

for为已知次数的循环。例：

#!/bin/bash

# desc : for loop

for animal in dog cat pig

do

echo "HI, ${animal}"

done

执行：

[work@www sh]$ sh for.sh

HI, dog

HI, cat

HI, pig

[work@www sh]$

for数值循环，适用于循环数值计算 例：

#!/bin/bash

# desc : for loop

sum=0

for (( i=0; i<=100; i++ ))

do

sum=$(($sum+$i))

done

echo "sum is : " $sum

引号：单引号更严格一些。它可以防止任何变量扩展。双引号可以防止通配符扩展但允许变量扩展。

#!/bin/sh

echo$SHELL

echo"$SHELL"

echo'$SHELL'

运行结果为：/bin/bash

/bin/bash

$SHELL

Here documents：类似于批量执行； 基本型式

cmd << delimiter

Here Document Content

delimiter

终端例：

cat << EOF

> First Line

> Second Line

> Third Line EOF

> EOF

First Line

Second Line

Third Line EOF

Shell文件例：

cat << EOF > output.sh

echo "hello"

echo "world"

EOF

使用 sh here.sh 运行这个脚本文件，会得到output.sh 这个新文件，里面的内容如下

echo "hello"

echo "world"

函数：形式：

function function\_name () {

list of commands

[ return value ]

}

Shell 函数返回值只能是整数，一般用来表示函数执行成功与否，0表示成功，其他值表示失败；如果一定要让函数返回字符串，那么可以先定义一个变量，用来接收函数的计算结果，脚本在需要的时候访问这个变量来获得函数返回值。例：

#!/bin/bash

# Calling one function from another

number\_one () {

echo "Url\_1 is http://see.xidian.edu.cn/cpp/shell/"

number\_two

}

number\_two () {

echo "Url\_2 is http://see.xidian.edu.cn/cpp/u/xitong/"

}

number\_one

结果：Url\_1 is http://see.xidian.edu.cn/cpp/shell/

Url\_2 is http://see.xidian.edu.cn/cpp/u/xitong/

删除函数和删除变量一样执行unset f function\_name