1. 获取当前目录的路径，赋值给某个变量：

/home/linux/bin/test/test.sh执行该shell文件工作目录为：/home/linux/bin

|  |
| --- |
| #!/bin/bash  currentdir=`pwd`;  echo $currentdir |

执行结果：

|  |
| --- |
| linux:bin>./test/test.sh  /home/linux/bin |

注：`linux系统命令`：表示执行linux系统命令，`pwd`表示执行pwd命令，并且将返回值赋值给currentdir。

在linux中不需要用set来赋值，注意和bat区别,在bat中要用set currentdir=%cd%。

1. 获取正在运行的shell文件所属的目录

/home/linux/bin/test/test.sh执行该shell文件工作目录为：/home/linux/bin

|  |
| --- |
| #!/bin/bash  shellfile\_dir=”$(dirname “$0”)”  echo $ shellfile\_dir |

执行结果：

|  |
| --- |
| linux:bin>./test/test.sh 小明  /home/linux/bin/test |

注：其中$0表示正在运行的shell文件本身./test/test.sh得到的是一个相对路径，$1表示第1个参数：小明,$2代表第二个参数，以此类推

dirname：将文件路径去掉最后一个/，仅展示到最近目录层级，不展示文件即./test/

或者

/home/linux/bin/test/test.sh执行该shell文件工作目录为：/home/linux/bin

|  |
| --- |
| #!/bin/bash  currentdir =”$(dirname $(readlink -f “$0”))”  echo $currentdir |

由此得到的是绝对路径

readlink -f 文件名 递归跟随给出文件名的所有符号链接以标准化，除最后一个外所有组件必须存在。

用bat时可以使用变量扩充：%~dp0：

%~dp0 “d”为Drive的缩写，即为驱动器，磁盘、“p”为Path缩写，即为路径，目录

cd是转到这个目录，cd /d %~dp0 可能会好些

选项语法:

|  |  |
| --- | --- |
| ~0 | 删除任何引号(")，扩充 %0 |
| %~f0 | 将 %0 扩充到一个完全合格的路径名(“f”是file，即文件) |
| %~d0 | 仅将 %0 扩充到一个驱动器号 |
| %~p0 | 仅将 %0 扩充到一个路径 |
| %~n0 | 仅将 %0 扩充到一个文件名(“n”是name 文件名) |
| %~x0 | 仅将 %0 扩充到一个文件扩展名 |
| %~s0 | 扩充的路径只含有短名(“s”为Short，短的) |
| %~a0 | 将 %0 扩充到文件的文件属性(“a”为attribute，即属性) |
| %~t0 | 将 %0 扩充到文件的日期/时间(“t”time) |
| %~z0 | 将 %0 扩充到文件的大小(Size 大小) |
| %~$PATH:0 | 查找列在路径环境变量的目录，并将 %0扩充到找到的第一个完全合格的名称。如果环境变量名未被定义，或者没有找到文件，此组合键会扩充到空字符串 |

可以组合修饰符来得到多重结果:

|  |  |
| --- | --- |
| %~dp0 | 仅将 %0 扩充到一个驱动器号和路径 |
| %~nx0 | 仅将 %0 扩充到一个文件名和扩展名 |
| %~fs0 | 仅将 %0 扩充到一个带有短名的完整路径名 |
| %~dp$PATH:0 | 查找列在路径环境变量的目录，并将 %I 扩充到找到的第一个驱动器号和路径 |
| %~ftza0 | 将 %0 扩充到类似输出线路的 DIR |
| %0 | 为当前批处理文件 |

如果0换成1为第一个文件，2为第2个

1. cp

|  |  |
| --- | --- |
| option/语法： | cp [options] source dest 或 cp [options] source... Directory |
| -a | 此选项通常在复制目录时使用，它保留链接、文件属性，并复制目录下的所有内容。其作用等于dpR参数组合 |
| -d | 复制时保留链接。这里所说的链接相当于Windows系统中的快捷方式 |
| -i | 在覆盖文件之前提示用户，由用户确认 |
| -p | 除复制文件的内容外，还把修改时间和访问权限也复制到新文件中 |
| -r | 若给出的源文件是一个目录文件，此时将复制该目录下所有的子目录和文件 |
| -f | 覆盖已经存在的目标文件而不给出提示 |
| -l | 不复制文件，只是生成链接文件 |

1. scp

从本地复制到远程：

scp [option] local\_filename remote\_username@remote\_ip:remote\_folder

scp [option] local\_filename remote\_ip:remote\_folder

从远程复制到本地：

scp [option] remote\_username@remote\_ip:remote\_folder local\_filename

scp [option] remote\_ip:remote\_folder local\_filename

|  |  |
| --- | --- |
| -1 | 强制scp命令使用协议ssh1 |
| -2 | 强制scp命令使用协议ssh2 |
| -4 | 强制scp命令只使用IPv4寻址 |
| -6 | 强制scp命令只使用IPv6寻址 |
| -B | 使用批处理模式（传输过程中不询问传输口令或短语） |
| -C | 允许压缩。（将-C标志传递给ssh，从而打开压缩功能） |
| -p | 保留原文件的修改时间，访问时间和访问权限。 |
| -q | 不显示传输进度条。 |
| -r | 递归复制整个目录。 |
| -v | 详细方式显示输出。scp和ssh(1)会显示出整个过程的调试信息。这些信息用于调试连接，验证和配置问题。 |
| -c cipher | 以cipher将数据传输进行加密，这个选项将直接传递给ssh。 |
| -F ssh\_config | 指定一个替代的ssh配置文件，此参数直接传递给ssh。 |
| -I identity\_file | 从指定文件中读取传输时使用的密钥文件，此参数直接传递给ssh。 |
| -l limit | 限定用户所能使用的带宽，以Kbit/s为单位。 |
| -o ssh\_option | 如果习惯于使用ssh\_config(5)中的参数传递方式， |
| -P port | 注意是大写的P,port是指定数据传输用到的端口号 |
| -S program | 指定加密传输时所使用的程序。此程序必须能够理解ssh(1)的选项。 |

1. If语句

|  |
| --- |
| if [….] then  …..  fi  或者  if [….] then  …..  elif […] then  …  else  ….  fi |

|  |  |
| --- | --- |
| -a |  |
| -b | 当file存在并且文件是块文件时，返回真 |
| -c | 当file存在并且是字符文件时，返回真 |
| -d | 当pathname存在并且是一个目录时，返回真 |
| -e | 当pathname指定的文件或者目录存在时，返回真 |
| -f | 当file文件存在，并且是正规文件上时，返回真 |
| -g | 当pathname指定的文件或目录存在并指定了SGID位时，返回真 |
| -h | 当file存在并且是符号连接文件时，返回真，但该选项在一些老系统上无效 |
| -k | 当由pathname指定的文件或目录存在并且设置了“粘滞”位时返回真 |
| -p | 当file存在并且是命令管道时返回为真 |
| -r | 当由pathname指定的文件或目录存在并且可读时返回为真 |
| -s | 当file存在文件大小大于0时返回真 |
| -u | 当由pathname指定的文件或目录存在并且设置了SUID位时返回真 |
| -w | 当由pathname指定的文件或目录存在并且可写时返回真。一个目录为了它的内容被访问必然是可执行的。 |
| -o | 当由pathname指定的文件或目录存在并且被子当前进程的有效用户ID所指定的用户拥有时返回真。 |

UNIX Shell 里面比较字符写法：注意使用引号可以排除空格对代码的干扰

|  |  |
| --- | --- |
| -eq | 等于 |
| -ne | 不等于 |
| -gt | 大于 |
| -lt | 小于 |
| -le | 小于等于 |
| -ge | 大于等于 |
| -nt | filename1 -nt filename2如果filename1比filename2新，则为真 |
| -ot | filename1 -ot filename2   如果 filename1 比 filename2 旧，则为真 |
| -z | 空串，如[-z $var]如果$var的值长度为0，则返回真 |
| = | 两个字符相等,如：[string1 = string2]，两个字符串相等则返回真 |
| != | 两个字符不等如：[string1!=string2]，两个字符串不相等则返回真 |
| -n | 非空串，如[-z$var]如果$var的值长度为不为0，则返回真 |
| -x | 如果 filename 可执行，则为真 |

1. date
2. date命令的一些参数

|  |  |
| --- | --- |
| date -d datestr | 显示 datestr 中所设定的时间 (非系统时间) |
| date –help | 显示辅助讯息 |
| date -s datestr | 将系统时间设为 datestr 中所设定的时间 |
| date -u | 显示目前的格林威治时间 |
| date –version | 显示版本编号 |

1. 如果以+开头，则是显示获取时间

示例：

$(date “+%Y-%m-%d %H:%M:%S”) ：格式如2019-4-22 23:14:20

$(date -d "-2 day" "+%Y%m%d %A") 或 $(date --date="2 days ago" "+%Y%m%d %A")：获取两天前的时间

日期格式：

|  |  |
| --- | --- |
| %a | 星期几 (Sun..Sat) |
| %A | 星期几 (Sunday..Saturday) |
| %b | 月份 (Jan..Dec) |
| %B | 月份 (January..December) |
| %c | 直接显示日期与时间 |
| %d | 日 (01..31) |
| %D | 直接显示日期 (mm/dd/yy) |
| %h | 同 %b |
| %j | 一年中的第几天 (001..366) |
| %m | 月份 (01..12) |
| %U | 一年中的第几周 (00..53) (以 Sunday 为一周的第一天的情形) |
| %w | 一周中的第几天 (0..6) |
| %W | 一年中的第几周 (00..53) (以 Monday 为一周的第一天的情形) |
| %x | 直接显示日期 (mm/dd/yy) |
| %y | 年份的最后两位数字 (00.99) |
| %Y | 完整年份 (0000..9999) |

时间格式：

|  |  |
| --- | --- |
| % | 印出 |
| % %n | 下一行 |
| %t | 跳格 |
| %H | 小时(00..23) |
| %I | 小时(01..12) |
| %k | 小时(0..23) |
| %l | 小时(1..12) |
| %M | 分钟(00..59) |
| %p | 显示本地AM或PM |
| %r | 直接显示时间(12小时制，格式为hh:mm:ss[AP]M) |
| %s | 从1970年1月1日00:00:00UTC到目前为止的秒数%S:秒(00..61) |
| %T | 直接显示时间(24小时制) |
| %X | 相当于%H:%M:%S |
| %Z | 显示时区 |

1. 如果不以+开头则是设定时间，此时时间格式如下：

|  |  |
| --- | --- |
| MM | 为月份 |
| DD | 为日 |
| hh | 为小时 |
| mm | 为分钟 |
| CC | 为年份前两位数字 |
| YY | 为年份后两位数字 |
| ss | 为秒数 |

1. read从键盘读取变量值

read命令 从键盘读取变量的值，通常用在shell脚本中与用户进行交互的场合。该命令可以一次读取多个变量的值，变量和输入的值都需要使用空格隔开。在read命令后面，如果没有指定变量名，读取的数据将被自动赋值给特定的变量REPLY

|  |  |
| --- | --- |
| 选项/语法 | read (选项) (参数) |
| -p | 指定读取值时的提示符； |
| -t | 指定读取值时等待的时间（秒） |
| -s | 不显示输入内容 |
| -d<"定界符"> | 指定结束输入的定界符，无该参数默认为回车 |
| -n<字符数> | 指定存入变量的字符数量，不需要回车 |
| -r | 允许输入反斜杠 |
| -a | 把单词清单读取到变量中，该变量为数组，可以用${var[0]}获取 |

示例：

read –p ”请输入密码：” –s pwd 在打印提示后输入密码，不显示输入密码，将其存入变量pwd中

read –d “:” var 读取输入数据，以：结束，存在变量var中

read –a friends 输入数据到数组变量friends中，可以用${ friends[0]}、${ friends[0]}等获取

read var1 var2 一次输入多个变量值，用空格隔开

1. shell中对其他shell脚本的调用有三种方式

bash shell的命令分为两类：外部命令和内部命令。外部命令是通过系统调用或独立的程序实现的，如sed、awk等等。内部命令是由特殊的文件格式（.def）所实现，如cd、history、exec等等。

1. fork

即用/path/\*.sh的这种方式调用，这种方式调用是由父进程开启一个新的子进程来进行调用，子进程开启后父进程仍然存在，等待子进程执行结束后返回父进程，然后父进程继续执行。其中子进程由父进程处分配资源，并继承父进程的环境,可以获取父进程定义的环境变量，但是返回时不会带回环境变量，即父进程不能访问子进程中定义的环境变量。

1. exec

即exec /pathname/\*.sh的调用方式，这种方式调用也会由父进程开启一个新的子进程，但是在子进程开启后，父进程被干掉，子进程结束后不会再返回父进程，从子进程开启位置起，父进程的脚本将不会再执行，只执行子脚本中内容后结束。

1. source

即source /pathname/\*.sh的调用方式，这种方式调用不会开启子进程，被调用的脚本在原有的进程中执行，执行完毕后，继续执行父脚本内容，因此在这种执行方式下，当子脚本改变了父脚本中的变量，在父脚本中也是可见的。

1. 比较文件的差异
2. 转换文件字符集iconv

iconv -f utf-8 -t gb2312 /server\_test/reports/software\_.txt > /server\_test/reports/software\_asserts.txt

|  |  |
| --- | --- |
| -f,--from-code=名称 | 原始文本编码 |
| -t,--to-code=名称 | 输出编码 |
| -l,--list | 列举所有已知的字符集 |
| -c | 从输出中忽略无效的字符 |
| -o,--output=FILE | 输出文件 |
| -s,--silent | 关闭警告 |
| --verbose | 打印进度信息 |

1. 文件压缩备份以及解压缩

tar

zip

zip –r text.zip /home/\*.txt

unzip text.zip –d text/

gzip

1. Ubantu系统安装从源中安装和卸载软件

第一种方式，通过apt-get安装：

使用apt-get方法安装的软件，所有下载的deb包都缓存到了/var/cache/apt/archives目录下了，所以可以把常用的deb包备份出来，甚至做成ISO工具包、刻盘，以后安装Ubuntu时就可以在没有网络环境的情况下进行了。可以将archives这个目录到/var/cache/apt/目录下，替换原有的archives，命令为：sudo cp -r archives/ /var/cache/apt/

|  |  |
| --- | --- |
| sudo apt-get install 软件名 | 安装软件 |
| sudo apt-get remove 软件名 | 卸载软件 |
| sudo apt-get update | 更新软件信息数据库，每个系统的版本有不同的安装源，在更新了/etc/apt/sources.txt后使用 |
| sudo apt-get upgrade | 进行系统升级 |
| sudo apt-cache search | 搜索软件名 |

第二种方式，通过dpkg安装deb包的形式

|  |  |
| --- | --- |
| dpkg -i package.deb | 安装包 |
| dpkg -r package | 删除包 |
| dpkg -P package | 删除包（包括配置文件） |
| dpkg -L package | 列出与该包关联的文件 |
| dpkg -l package | 显示该包的版本 |
| dpkg –unpack package.deb | 解开 deb 包的内容 |
| dpkg -S keyword | 搜索所属的包内容 |
| dpkg -l | 列出当前已安装的包 |
| dpkg -c package.deb | 列出 deb 包的内容 |
| dpkg –configure package | 配置包 |

第三种方式使用make install,源码安装有三个步骤

1. 配置

./configure –prefix=路径，来指定安装位置

1. 编译

make install，来进行编译

1. 安装

make install 软件名

1. centOS的集中安装方式
2. rpm
3. yum install
4. make install
5. tcpdump

语法：

tcpdump [-adeflnNOpqStvx][-c<数据包数目>][-dd][-ddd][-F<表达文件>][-i<网络界面>][-r<数据包文件>][-s<数据包大小>][-tt][-T<数据包类型>][-vv][-w<数据包文件>][输出数据栏位]

|  |  |
| --- | --- |
| -a | 尝试将网络和广播地址转换成名称。 |
| -c<数据包数目> | 收到指定的数据包数目后，就停止进行倾倒操作。 |
| -d | 把编译过的数据包编码转换成可阅读的格式，并倾倒到标准输出。 |
| -dd | 把编译过的数据包编码转换成C语言的格式，并倾倒到标准输出。 |
| -ddd | 把编译过的数据包编码转换成十进制数字的格式，并倾倒到标准输出。 |
| -e | 在每列倾倒资料上显示连接层级的文件头。 |
| -f | 用数字显示网际网络地址。 |
| -F<表达文件> | 指定内含表达方式的文件。 |
| -i<网络界面> | 使用指定的网络截面送出数据包。 |
| -l | 使用标准输出列的缓冲区。 |
| -n | 不把主机的网络地址转换成名字。 |
| -N | 不列出域名。 |
| -O | 不将数据包编码最佳化。 |
| -p | 不让网络界面进入混杂模式。 |
| -q | 快速输出，仅列出少数的传输协议信息。 |
| -r<数据包文件> | 从指定的文件读取数据包数据。 |
| -s<数据包大小> | 设置每个数据包的大小。 |
| -S | 用绝对而非相对数值列出TCP关联数。 |
| -t | 在每列倾倒资料上不显示时间戳记。 |
| -tt | 在每列倾倒资料上显示未经格式化的时间戳记。 |
| -T<数据包类型> | 强制将表达方式所指定的数据包转译成设置的数据包类型。 |
| -v | 详细显示指令执行过程。 |
| -vv | 更详细显示指令执行过程。 |
| -x | 用十六进制字码列出数据包资料。 |
| -w<数据包文件> | 把数据包数据写入指定的文件。 |

1. 查看系统版本lsb\_release

LSB是Linux Standard Base的缩写， lsb\_release命令 用来显示LSB和特定版本的相关信息。如果使用该命令时不带参数，则默认加上-v参数

|  |  |
| --- | --- |
| -v | 显示版本信息 |
| -i | 显示发行版的id |
| -d | 显示该发行版的描述信息 |
| -r | 显示当前系统是发行版的具体版本号 |
| -c | 发行版代号 |
| -a | 显示上面的所有信息 |
| -h | 显示帮助信息 |

1. 使用sed命令替换当前目录下所有文件的某个值

sed –i “s/172.17.0.2/198.16.1.1/g” `grep -rl “172.17.0.2” ./ `

1. 查看进程号

ps –ef |grep java

ps –u 用户名| grep java

1. 根据进程号罗列出进程加载的jar包

lsof -p pid 显示出所有该进程加载的文件

1. 管道 |

命令1 |命令2:将命令1的标准输出作为命令2的标准输入，并将最终处理结果打印在屏幕上

管道命令只处理前一个命令正确输出，不处理错误输出。

管道命令右边命令，必须能够接收标准输入流命令才行。

find . –name “\*.log” | xargs rm –f

ll | grep “Exited”

1. 重定向> 和 >>
2. du:显示每个文件和目录磁盘使用空间：

du命令 也是查看使用空间的，但是与df命令不同的是Linux du命令是对文件和目录磁盘使用的空间的查看，还是和df命令有一些区别的，它的语法格式为：du [option] 文件

Option:

|  |  |
| --- | --- |
| -a或-all | 显示目录中个别文件的大小。 |
| -b或-bytes | 显示目录或文件大小时，以byte为单位。 |
| -c或--total | 除了显示个别目录或文件的大小外，同时也显示所有目录或文件的总和。 |
| -k或--kilobytes | 以KB(1024bytes)为单位输出。 |
| -m或--megabytes | 以MB为单位输出。 |
| -s或--summarize | 仅显示总计，只列出最后加总的值。 |
| -h或--human-readable | 以K，M，G为单位，提高信息的可读性。 |
| -x或--one-file-xystem | 以一开始处理时的文件系统为准，若遇上其它不同的文件系统目录则略过。 |
| -L<符号链接>或--dereference<符号链接> | 显示选项中所指定符号链接的源文件大小。 |
| -S或--separate-dirs | 显示个别目录的大小时，并不含其子目录的大小。 |
| -X<文件>或--exclude-from=<文件> | 在<文件>指定目录或文件。 |
| --exclude=<目录或文件> | 略过指定的目录或文件。 |
| -D或--dereference-args | 显示指定符号链接的源文件大小。 |
| -H或--si | 与-h参数相同，但是K，M，G是以1000为换算单位。 |
| -l或--count-links | 重复计算硬件链接的文件。 |

1. df：显示磁盘的相关信息，以磁盘分区为单元，用于显示磁盘分区上的可使用的磁盘空间。默认显示单位为KB。可以利用该命令来获取硬盘被占用了多少空间，目前还剩下多少空间等信息，语法如下：df (选项) (参数)

|  |  |
| --- | --- |
| -a或--all | 包含全部的文件系统； |
| --block-size=<区块大小> | 以指定的区块大小来显示区块数目； |
| -h或--human-readable | 以可读性较高的方式来显示信息； |
| -H或--si | 与-h参数相同，但在计算时是以1000Bytes为换算单位而非1024Bytes； |
| -i或--inodes | 显示inode的信息； |
| -k或--kilobytes | 指定区块大小为1024字节； |
| -l或--local | 仅显示本地端的文件系统； |
| -m或--megabytes | 指定区块大小为1048576字节； |
| --no-sync | 在取得磁盘使用信息前，不要执行sync指令，此为预设值； |
| -P或--portability | 使用POSIX的输出格式； |
| --sync | 在取得磁盘使用信息前，先执行sync指令； |
| -t<文件系统类型>或--type=<文件系统类型> | 仅显示指定文件系统类型的磁盘信息； |
| -T或--print-type | 显示文件系统的类型； |
| -x<文件系统类型>或--exclude-type=<文件系统类型> | 不要显示指定文件系统类型的磁盘信息； |
| --help | 显示帮助； |
| --version | 显示版本信息。 |

参数

文件：指定文件系统上的文件。

大小格式

显示值以 --block-size 和 DF\_BLOCK\_SIZE，BLOCK\_SIZE 和 BLOCKSIZE 环境变量中的第一个可用 SIZE 为单位。 否则，单位默认为 1024 个字节（如果设置 POSIXLY\_CORRECT，则为512）。

SIZE是一个整数和可选单位（例如：10M是10 \* 1024 \* 1024）。 单位是K，M，G，T，P，E，Z，Y（1024的幂）或KB，MB，...（1000的幂）。

1. free
2. top显示或管理执行中的程序

top命令 可以实时动态地查看系统的整体运行情况，是一个综合了多方信息监测系统性能和运行信息的实用工具。通过top命令所提供的互动式界面，用热键可以管理。类似于windows中的任务管理器。

语法：top 选项

|  |  |
| --- | --- |
| -b | 以批处理模式操作； |
| -c | 显示完整的治命令； |
| -d | 屏幕刷新间隔时间； |
| -I | 忽略失效过程； |
| -s | 保密模式； |
| -S | 累积模式； |
| -i<时间> | 设置间隔时间； |
| -u<用户名> | 指定用户名； |
| -p<进程号> | 指定进程； |
| -n<次数> | 循环显示的次数。 |

top交互命令

在top命令执行过程中可以使用的一些交互命令。这些命令都是单字母的，如果在命令行中使用了-s选项， 其中一些命令可能会被屏蔽。

|  |  |
| --- | --- |
| h | 显示帮助画面，给出一些简短的命令总结说明； |
| k | 终止一个进程； |
| i | 忽略闲置和僵死进程，这是一个开关式命令； |
| q | 退出程序； |
| r | 重新安排一个进程的优先级别； |
| S | 切换到累计模式； |
| s | 改变两次刷新之间的延迟时间（单位为s），如果有小数，就换算成ms。输入0值则系统将不断刷新，默认值是5s； |
| f或者F | 从当前显示中添加或者删除项目； |
| o或者O | 改变显示项目的顺序； |
| l | 切换显示平均负载和启动时间信息； |
| m | 切换显示内存信息； |
| t | 切换显示进程和CPU状态信息； |
| c | 切换显示命令名称和完整命令行； |
| M | 根据驻留内存大小进行排序； |
| P | 根据CPU使用百分比大小进行排序； |
| T | 根据时间/累计时间进行排序； |
| w | 将当前设置写入~/.toprc文件中。 |

实例

top - 09:44:56 up 16 days, 21:23, 1 user, load average: 9.59, 4.75, 1.92

Tasks: 145 total, 2 running, 143 sleeping, 0 stopped, 0 zombie

Cpu(s): 99.8%us, 0.1%sy, 0.0%ni, 0.2%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st

Mem: 4147888k total, 2493092k used, 1654796k free, 158188k buffers

Swap: 5144568k total, 56k used, 5144512k free, 2013180k cached

解释：

top - 09:44:56[当前系统时间],

16 days[系统已经运行了16天],

1 user[个用户当前登录],

load average: 9.59, 4.75, 1.92[系统负载，即任务队列的平均长度]

Tasks: 145 total[总进程数],

2 running[正在运行的进程数],

143 sleeping[睡眠的进程数],

0 stopped[停止的进程数],

0 zombie[冻结进程数],

Cpu(s): 99.8%us[用户空间占用CPU百分比],

0.1%sy[内核空间占用CPU百分比],

0.0%ni[用户进程空间内改变过优先级的进程占用CPU百分比],

0.2%id[空闲CPU百分比], 0.0%wa[等待输入输出的CPU时间百分比],

0.0%hi[],

0.0%st[],

Mem: 4147888k total[物理内存总量],

2493092k used[使用的物理内存总量],

1654796k free[空闲内存总量],

158188k buffers[用作内核缓存的内存量]

Swap: 5144568k total[交换区总量],

56k used[使用的交换区总量],

5144512k free[空闲交换区总量],

2013180k cached[缓冲的交换区总量]

1. rename

使用字符串替换的方式批量改变文件名，一般如果为单个文件重命名使用mv，但是对多个文件的重命名则使用rename

语法：

rename 参数

参数：

原字符串：将文件名需要替换的字符串；

目标字符串：将文件名中含有的原字符替换成目标字符串；

文件：指定要改变文件名的文件列表。

示例：

将以.log.bak结尾的文件重命名为.log结尾

rename .log.bak .log \*.log.bak

rename支持通配符：

? 可替代单个字符

\* 可替代多个字符

[charset] 可替代charset集中的任意单个字符

文件夹中有这些文件foo1, ..., foo9, foo10, ..., foo278

如果使用rename foo foo0 foo?，会把foo1到foo9的文件重命名为foo01到foo09，重命名的文件只是有4个字符长度名称的文件，文件名中的foo被替换为foo0。

如果使用rename foo foo0 foo??，foo01到foo99的所有文件都被重命名为foo001到foo099，只重命名5个字符长度名称的文件，文件名中的foo被替换为foo0。

如果使用rename foo foo0 foo\*，foo001到foo278的所有文件都被重命名为foo0001到foo0278，所有以foo开头的文件都被重命名。

如果使用rename foo0 foo foo0[2]\*，从foo0200到foo0278的所有文件都被重命名为foo200到foo278，文件名中的foo0被替换为foo。

rename支持正则表达式：

字母的替换

rename "s/AA/aa/" \* # 把文件名中的AA替换成aa

修改文件的后缀

rename "s//.html//.php/" \* # 把.html 后缀的改成 .php后缀

批量添加文件后缀

rename "s/$//.txt/" \* # 把所有的文件名都以txt结尾

批量删除文件名

rename "s//.txt//" \* # 把所有以.txt结尾的文件名的.txt删掉

1. liunx查找文件并带目录复制到指定文件夹

mkdir --parents ../shell && find . -type f -name "\*.sh" -exec cp -r --parents {} ../shell \;

1. for

shell中的for循环：

bat中的for循环：可以在windows下通过help for查看

bat中的for循环语法

for /d %variable in (set) do command [command-parameters]

如果集中包含通配符，则指定与目录名匹配，而不与文件名匹配。

for /r [[drive:]path] %variable in (set) do command [command-parameters]

检查以 [drive:]path 为根的目录树，指向每个目录中的 FOR 语句。如果在 /R 后没有指定目录规范，则使用当前目录。如果集仅为一个单点(.)字符，则枚举该目录树。

for /l %variable in (start,step,end) do command [command-parameters]

该集表示以增量形式从开始到结束的一个数字序列。因此，(1,1,5)将产生序列1 2 3 4 5，(5,-1,1)将产生序列(5 4 3 2 1)

for /f ["option="] %variable in (file-set) do command [command-parameters]

for /f ["option="] %variable in ("string") do command [command-parameters]

for /f ["option="] %variable in ('command') do command [command-parameters]

|  |
| --- |
| rem 循环删除.svn文件  rem 在cmd中可以使用  for /f "delims=" %f in ('dir /b /s "\*.svn"') do del %f;  rem 在.bat文件中则要使用  for /f "delims=" %%f in ('dir /b /s "\*.svn"') do del %%f; |

fileset 为一个或多个文件名。继续到 fileset 中的下一个文件之前，每份文件都被打开、读取并经过处理。处理包括读取文件，将其分成一行行的文字，然后将每行解析成零或更多的符号。然后用已找到的符号字符串变量值调用 For 循环

以默认方式，/F 通过每个文件的每一行中分开的第一个空白符号。跳过空白行。您可通过指定可选 "options" 参数替代默认解析操作。这个带引号的字符串包括一个

或多个指定不同解析选项的关键字。这些关键字为:

eol=c - 指一个行注释字符的结尾(就一个)

skip=n - 指在文件开始时忽略的行数。

delims=xxx - 指分隔符集。这个替换了空格和跳格键的

默认分隔符集。

tokens=x,y,m-n - 指每行的哪一个符号被传递到每个迭代

的 for 本身。这会导致额外变量名称的分配。m-n

格式为一个范围。通过 nth 符号指定 mth。如果

符号字符串中的最后一个字符星号，

那么额外的变量将在最后一个符号解析之后

分配并接受行的保留文本。

usebackq - 指定新语法已在下类情况中使用:

在作为命令执行一个后引号的字符串并且一个单

引号字符为文字字符串命令并允许在 file-set

中使用双引号扩起文件名称。

某些范例可能有助:

FOR /F "eol=; tokens=2,3\* delims=, " %i in (myfile.txt) do @echo %i %j %k

会分析 myfile.txt 中的每一行，忽略以分号打头的那些行，将每行中的第二个和第三个符号传递给 for 函数体，用逗号和/或空格分隔符号。请注意，此 for 函数体的语句引用 %i 来获得第二个符号，引用 %j 来获得第三个符号，引用 %k来获得第三个符号后的所有剩余符号。对于带有空格的文件名，您需要用双引号将文件名括起来。为了用这种方式来使

用双引号，还需要使用 usebackq 选项，否则，双引号会被理解成是用作定义某个要分析的字符串的。

%i 在 for 语句中显式声明，%j 和 %k 是通过tokens= 选项隐式声明的。可以通过 tokens= 一行指定最多 26个符号，只要不试图声明一个高于字母"z"或"Z"的变量。请记住，FOR 变量是单一字母、分大小写和全局的变量；而且，不能同时使用超过 52 个。

还可以在相邻字符串上使用 FOR /F 分析逻辑，方法是，用单引号将括号之间的 file-set 括起来。这样，该字符串会被当作一个文件中的一个单一输入行进行解析。

最后，可以用 FOR /F 命令来分析命令的输出。方法是，将括号之间的 file-set 变成一个反括字符串。该字符串会被当作命令行，传递到一个子CMD.EXE，其输出会被捕获到内存中，并被当作文件分析。如以下例子所示:

FOR /F "usebackq delims==" %i IN (`set`) DO @echo %i

会枚举当前环境中的环境变量名称

1. groupadd创建一个新的工作组，新工作组的信息将被添加到系统文件中,groupdel用于删除一个工作组。

groupadd 选项 参数

选项：

|  |  |
| --- | --- |
| -g | 指定新建工作组的id； |
| -r | 创建系统工作组，系统工作组的组ID小于500； |
| -K | 覆盖配置文件“/ect/login.defs”； |
| -o | 允许添加组ID号不唯一的工作组。 |

参数：组名，指定工作组名称，例如oracle

例如：建立一个新组，并设置组ID加入系统

gorupadd –g 2300 oracle

此时在/etc/passwd文件中产生一个组ID（GID）是2300的项目

1. groupdel用于删除指定的工作组

groupdel命令 用于删除指定的工作组，本命令要修改的系统文件包括/ect/group和/ect/gshadow。若该群组中仍包括某些用户，则必须先删除这些用户后，方能删除群组。

例如：

groupdel oracle

1. useradd创建的新的系统用户

useradd命令 用于Linux中创建的新的系统用户。useradd可用来建立用户帐号。帐号建好之后，再用passwd设定帐号的密码．而可用userdel删除帐号。使用useradd指令所建立的帐号，实际上是保存在/etc/passwd文本文件中。

在Slackware中，adduser指令是个script程序，利用交谈的方式取得输入的用户帐号资料，然后再交由真正建立帐号的useradd命令建立新用户，如此可方便管理员建立用户帐号。在Red Hat Linux中， adduser命令 则是useradd命令的符号连接，两者实际上是同一个指令。

useradd(选项)(参数)

|  |  |
| --- | --- |
| -c<备注> | 加上备注文字。备注文字会保存在passwd的备注栏位中； |
| -d<登入目录> | 指定用户登入时的启始目录； |
| -D | 变更预设值； |
| -e<有效期限> | 指定帐号的有效期限； |
| -f<缓冲天数> | 指定在密码过期后多少天即关闭该帐号； |
| -g<群组> | 指定用户所属的群组； |
| -G<群组> | 指定用户所属的附加群组； |
| -m | 自动建立用户的登入目录； |
| -M | 不要自动建立用户的登入目录； |
| -n | 取消建立以用户名称为名的群组； |
| -r | 建立系统帐号； |
| -s<shell> | 指定用户登入后所使用的shell； |
| -u<uid> | 指定用户id。 |

例如：

useradd oracle –u 2300 –g oracle –s /bin/bash –md /home/oracle

需要说明的是，设定ID值时尽量要大于500，以免冲突。因为Linux安装后会建立一些特殊用户，一般0到499之间的值留给bin、mail这样的系统账号。

1. netstat查看端口号

netstat –ano | grep 2210

1. jstack和jmap

jstack –p procid

jmap

1. 查找类interactive.class包含在哪个jar包中：

find . –name “\*.jar” |awk ‘{print “jar –tf ”$1}’ | sh –x | grep interactive.class

1. 查找哪个jar包引用了interactive.class类

find . –name “\*.jar” | awk ‘{print “jdeps –v ”$1}’ | sh –x | grep interactive.class