CIeNET

QUALITY
INTEGRITY
LEADERSHIP

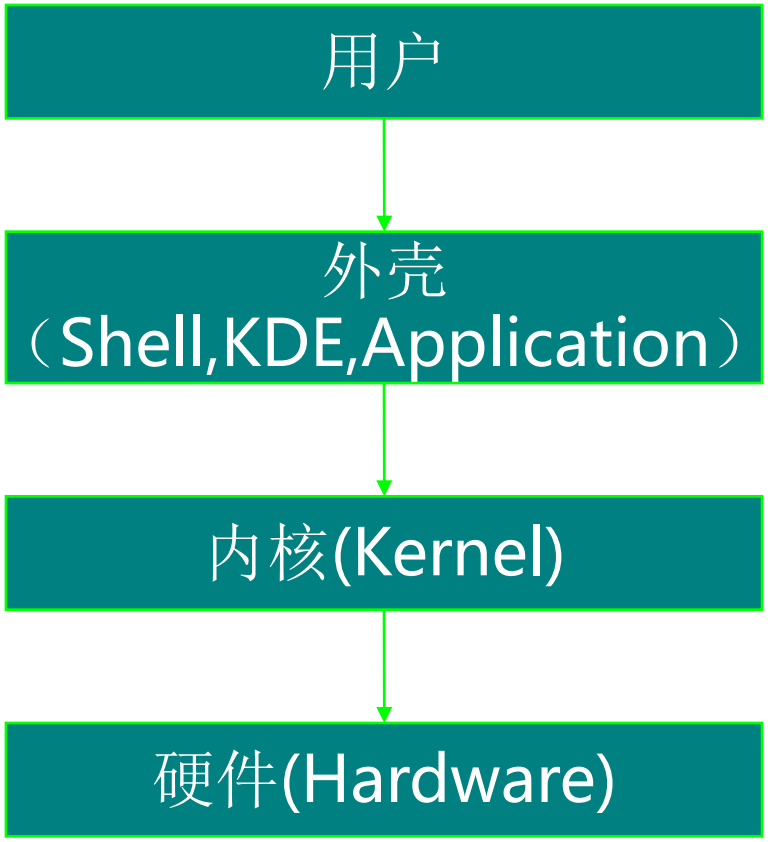
Shell 编程基础

- Shell概述
- Shell编程基础
- 进程管理
- Vi 编辑器
- sed &awk

Shell 概述

什么是shell？

- 命令解释器, 俗称外壳程序
- 用户通过shell向kernel发送指令，kernel再向硬件发送指令
- Shell版本众多，例如常听到的Bourne Shell (sh) 、在Sun中默认的C Shell、 商业上常用的KSHell以及TCSH 等，每一种Shell都各有其特点。
- Linux 使用的是Bourne Again Shell（简称 bash），这个 Shell是Bourne Shell的增强版本，是基于GNU的架构下发展出来的。



Shell概述 （特点）

- Shell是解释性的，多数高级语言是编译性的；
- Shell语言与高级语言处理的对象不同；
- Shell与系统有密切的关系；
- Shell易编写、调试、灵活性较强，但速度低；
- Shell作为命令级语言，命令组合功能很强。

shell常用命令格式

Shell命令的一般格式如下：命令名【选项】【参数1】【参数2】...
【选项】是对命令的特别定义，以减号(-)开始，多个选项可以用一个减号(-)连起来，如ls -l -a 与ls -la 相同。【参数】提供命令运行的信息，或者是命令执行过程中所使用的文件名。使用分号(;)可以将两个命令隔开，这样可以实现一行中输入多个命令。命令的执行顺序和输入的顺序相同。

shell下的常用命令 - 目录和文件操作

查看当前目录相对于根目录的位置

pwd

查看当前目录内容

ls

以长格式查看当前目录内容。对应每个文件的条目将包括连接数目、所有者、大小、最后修改时间、权限等内容

ls -l

改变当前目录。目的目录名可用相对路径表示，也可以用绝对路径表示。

cd [目的目录名]

转移到上一级目录

cd ..

创建目录

mkdir [新目录名]

例：

mkdir /var/ftp

shell下的常用命令-目录和文件操作

- 删除目录命令

rmdir [目录名]

例：

rmdir /var/ftp

递归删除一个目录中的所有文件和该目录本身。

rm -r [目录名]

复制文件

cp [源文件名] [目标文件名]

重命名文件

mv [源文件名] [目标文件名]

例：

mv /etc/rc.d/rc3.d/K50xinetd /etc/rc.d/rc3.d/S50xinetd

删除文件

rm [文件名]

不需确认地删除多个文件

rm -f [带通配符的文件名]

shell下的常用命令 - 目录和文件操作

- **ln [源文件名] [目标文件名]**
  如下面的例子。源文件abc.png存储在/root下，硬链接myabc.png存储在用户workman的家目录下。
  做了这个链接之后，用户root和workman就可以使用不同的文件名来读写同一个图形文件。即使root把abc.png删除，这个图形文件仍然存在，只有当workman把myabc.png也删除后，这个文件才会彻底从计算机上消失。
  ln /root/abc.png /home/workman/myabc.png
  软链接文件。也就是符号链接。可用此法创建文件的快捷方式。
  ln -s [源文件或文件夹名] [目标名]
  按文件名查找文件。
  find / -name nametofind -print
  改变文件所有者。
  例：
  chown workman.workgroup /data/pub/upload
  改变文件访问权限
  例：
  chmod -R 755 /usr/local/LumaQQ
  查看一个文件有多少行
  wc -l usr.bin
  查看一个文件有多少字节
  wc -c usr.bin
  查看文本文件的内容
  cat usr.bin

shell下的常用命令 - 用户管理

- 改变身份。如果不指定用户名，则默认将用户身份换至root。从root身份切换到其他任何身份都不需要口令。

  su [username]

  添加用户

  例：

  adduser -c "MyW" -d /home/workman -g root workman

  设置用户口令

  例：**passwd workman**

  安装和卸载软件

  安装rpm软件

  例：**rpm -Uvh kdevelope-3.1-1.i386.rpm**

  卸载rpm软件

  例：**rpm -e kdevelope**

  配置、编译、安装、卸载源码发布的软件包。

  ./configure

  make

  make install

  make clean

  卸载源码发布的软件包

  make uninstall

shell下的常用命令 - 文件的打包和解包

解tar包

例：

tar -xvzf gaim-0.77.tar.gz

tar -jxvf XXX.tar.bz2

生成tar包

例：

tar cvf - /etc | gzip-9c > backup.tar.gz

生成rar包

rar a test.rar file1.txt

rar a test.rar dir1

解rar包

rar x test.rar

解zip包

unzip lumaqq_p3.zip

Shell配置文件

shell在启动时会读取一些配置文件。下面是bash的配置文件，实际上他们也是shell脚本。/etc目录下的是系统级的配置文件，用户家目录下的两个配置文件是用户的个性化。

① **/etc/profile** 设置系统级的环境变量 PATH、HOME、HISTSIZE等。第一次登录时运行该脚本。

② **~/.bash_profile** 用户可在这里设置环境变量或启动程序。当用户登录时,该文件仅仅执行一次 。

③ **~/.bashrc** 用户可在这里设定别名和函数。启动shell时都会执行该配置文件。

④ **/etc/bashrc** 启动shell时运行该脚本。

- ~/.bash_logout 退出shell时执行。

Shell语法

- 1. 如果Script的第一个非空白字符不是"#"，则它会使用Bourne Shell。
- 2. 如果Script的第一个非空白字符是"#"时，但不以"#!"开头时，则它会使用C Shell。
- 3. 如果Script以“#!”开头，则“#!”后面所写的就是所使用的Shell，而且要将整个路径名称指出来。
- 建议使用第三种方式指定Shell ，以确保所执行的就是所要的。Bourne Shell的路径名称为/bin/sh ，而C Shell则为/bin/csh。

Shell语法 （变量）

- 用户自定义变量
- 特殊变量
- 系统环境变量
- 数组变量

Shell语法 （变量）

用户自定义变量

- 用户定义的变量由字母或下划线打头，由字母、数字或下划线序列组成，并且大小写字母意义不同。变量名长度没有限制。
- 在使用变量值时，要在变量名前加上前缀“$”
- 如果用双引号“”将值括起来，则括起来的字符串允许出现空格、制表符和换行符的特殊字符，而且允许有变量替换
- 如果用单引号‘’将值括起来，则括起来的字符串允许出现空格、制表符和换行符的特殊字符，但不允许有变量替换
- 引用变量的值时，可以用花括号{}将变量名称括起来，使变量名称与它的后续字符分隔开

Shell语法 （变量）

特殊变量

$* 这个程序的所有参数
$# 这个程序的参数个数
$$ 这个程序的PID
$! 上一个后台指令的PID
$? 上一个指令的返回值

shell特殊符号

shell使用两种引号和反斜线

- 双引号“”（让引号内的变量生效）
- 单引号‘’（引号内的变量当作一个字符使用）
- 反斜线\（禁用特殊字符功能\$）
- 后引号 ｀ ｀（让引号内的命令生效）
  - Var=hello
  - Echo “var is $var”
  - Echo ‘var is $var’
  - Echo “var is \$var”
  - Echo `ls`

Shell中的特殊符号

- 文件通配符
  - * -- 通配文件名中的任意长度字符串(包括空字符)
  - ? -- 通配文件名中的任意一个字符
  - [list] -- 通配文件名中的一个字符，该字符是列表中的任一个
- 命令结束/连接符 -- cmd1 ; cmd2
- 逻辑运算符 -- cmd1 && cmd2 、cmd1 || cmd2

Shell语法（变量）

系统环境变量

- /etc/profile：定义系统全局的工作环境，用户主目录下的.profile：定义该用户的工作环境
- 主要环境变量有：

HOME	用户主目录	PATH	搜索路径
PS1	shell提示符	PWD	用户当前所处的目录
MAIL	邮箱的路径	TERM	使用的终端类型

Shell 环境变量

- 环境变量的形式通常如：name=value
- Linux内核并不查看这些字符串，他们的解释完全依赖于各个应用程序。例如，shell就使用了大量的环境变量，其中一些在登录时自动设置，如HOME和USER等，有些则由用户设置。
- 我们在shell中设置环境变量以控制shell的动作。例如设置了MAIL就告诉shell应该到哪里去取邮件。
- 显示变量值的命令：echo
  - 用法：echo $variable
  - 例如:在环境变量前加上$就代表使用该变量的值

```
echo $PATH
echo $HOME
```

- 查看当前所有Linux环境变量的命令：env

Shell语法 （变量）

数组变量

- 在bash中，多个元素的设置方式如下：

  格式: name=(value1 value2 …valuen)

例: array=(a b c d e)

这样:$array[0]=a $array[3]=d

Shell算术运算

- + - * / % 分别对应加、减、乘、除、取余
- 只需将特定的算术表达式用 “$((” 和 “))” 括起来。a=$((4-2)) a的值为2

```
a=10
b=2
echo $(($a+$b))
echo $(($a-$b))
echo $(($a*$b))
echo $(($a/$b))
echo $(($a%$b))
```

Shell变量赋值

- 把命令的执行结果赋值给变量（例如ls），用`command`，或$(command)
- 注意：`是反引号(backquote), 不是'单引号(quote)

```
将ls的输出结果作为变量a的值
#a=`ls` 等价于 a=$(ls)
打印变量
#echo  $a
```

Shell语法 （条件控制）

- If、then、else语法

```
if 条件1
then 命令1
elif 条件2
then 命令2
else 命令3
fi
```

例子：

```
#!/bin/sh
if [ $# -ne 2 ]; then
   echo "Not enough parameters"
   exit 0
fi
if [ $1 -eq $2 ]; then
   echo "$1 equals $2"
elif [ $1 -lt $2 ]; then
   echo "$1 littler than  $2"
elif [ $1 -gt $2 ]; then
   echo "$1 greater than  $2"
fi
```

判断条件

- -f file 判断文件
- -d file 判断目录
- -r file 判断可读
- -w file 判断可写
- -x file 判断可执行
- -s file 判断文件非空
- -z string 空字符串
- -n string 字符非空

- string != string2 字符不同
- string == string2 字符相同
- var1 -eq var2 数值相等
- var1 -gt var2 数值大于
- var1 -lt var2 数值小于
- var1 -ge var2 大于等于
- var1 -le var2 小于等于
- var1 -ne var2 数值不等

Shell语法 （条件控制）

- case语法

```
case value in
    pattern1)
command11
                ...

command1n;;
    pattern2)     command21
                    ...
            command2n;;
   patternn)     commandn1
            ...
        commandnn;;
esac
```

例子：

```
#! /bin/sh
read op
case $op in
    C)
      echo "your selection is Copy"
    ;;
    D)
      echo "your selection is Delete"
    ;;
    B)
      echo "your selection is Backup"
    ;;
    *)
      echo "invalide selection"
esac
```

Shell语法 （条件控制）

for语法

1. for variable in arg1 arg2 … argn
   do
   command
   …
   command
   done

2. for ((初始值; 限制值; 执行步骤))
   do
   程式段
   done

例子:

```
#!/bin/bash
for DAY in Sunday Monday Tuesday
Wednesday Thursday Friday
Saturday
do
  echo "The day is : $DAY"
done
```

Shell语法 （条件控制）

- while语法
  - 只要测试条件保持为真时就继续执行。一旦条件为假，它就终止执行。

```
while command
      do
      command
done
```

例子：

```
#! /bin/sh
num=1
while [ $num -le 10 ]
do
   square=`expr $num \* $num`
   echo $square
   num=`expr $num + 1`
done
```

- until语法
  - 只要测试条件保持为假时就继续执行。一旦条件为真，它就终止执行。

```
until condition
   do
         command block
   done
```

```
#! /bin/sh
num=10

until [ $num -le 0 ]
do
   square=`expr $num \* $num`
   echo $square
   num=`expr $num - 1`
done
```

Shell语法 （条件控制）

1.break 是用来打断循环，也就是“强迫结束”循环。
若 break 后面指定一个数值 n 的话，则"从里向外"打断第 n 个循环.
默认值为 break 1，也就是打断当前的循环。
在使用 break 时需要注意的是，它与 return 及 exit 是不同的：

1.break 是结束 loop
2.return 是结束 function
3.exit 是结束 script/shell

2.continue是强迫进入下一次循环动作。
简单的看成：continue 到 done 之间的句子略过而返回循环顶端...
continue 后面也可指定一个数值 n，以决定继续哪一层(从里向外计算)的循环，默认值为 continue 1，也就是继续当前的循环。

Shell语法 （函数）

- 函数的定义：

  函数名 ()
  {
  　命令序列
  }

- 函数的调用：不带()

  函数名　参数1　参数2 …

例子

```
add(){
    result=0
    for n in $*
    do
        result=$(($result+$n))
    done
    return $result
}
```

执行:

```
add 1 2 3
echo $?
echo $result
```

输出结果都为 6

- shell中是不能直接获得函数的返回值的，如果你要用函数返回值，只能用全局变量传输。
- shell中的变量默认都是全局的，除非你在前面加了local修饰符。
- shell会把函数返回值放在$?全局变量中，你可以用$? 来取得前个函数调用的返回值。

  $*可以获得函数的所有输入参数，$1表示第一个参数，以此类推。

- 用bash内嵌的read命令
- 功能：读取来自键盘的输入，并传给变量
- 用法：read [选项] 变量名
- 选项：
- -p 字符串 后面接字符串给出提示
- -t 时间 后接等待的秒数，即不会一直等待用户输入

```
#echo “input your name”
#read  name
上面两行可以用下面一行代替
#read -p “input your name” name

#echo “my name is $name”
```

- bash [-nvx] script
- 参数说明

  -n 不执行脚本，只检查脚本的语法，若没有语法错误，则不会显示任何信息

  -v 执行脚本前，把每行脚本都输入到屏幕

  -x 执行脚本，显示脚本每一行脚本的执行情况。这是非常有用的参数。

进程管理

进程分类

前台(交互)进程
后台(守护)进程

进程管理相关的命令-ps

- ps [参数]
- -A:显示所有进程
- a:显示属于用户的当前活动的进程
- e:显示进程序环境变量
  - 普通用户不能查看其它用户的环境变量
- l:按长格式显示输出
- u:显示用户名和进程开始时间
- f:显示进程的全部信息
- t 终端名:与某个终端相连的所有进程

ps 各字段含义

- UID:进程所有者
- PID:进程ID
- %CPU:进程使用CPU的时间比
- %MEM:使用内存的百分比
- SIZE:使用虚拟内存大小
- RSS:常驻内存大小(Kbyte)
- TTY:与进程有关的终端
- STAT:状态

进程管理相关的命令-kill

- 信号：传递给进程的一个异常事件
  信号共有31个

格式： kill [信号] PID

常用信号定义：

-SIGSTOP:暂停某个进程

-SIGCONT:继续暂停的进程

-SIGHUP:挂起一个进程

-SIGTERM:软中断，正常退出一个软件

-SIGKILL:杀死一个进程

-SIGTSTP:键盘停止信号，ctrl+z

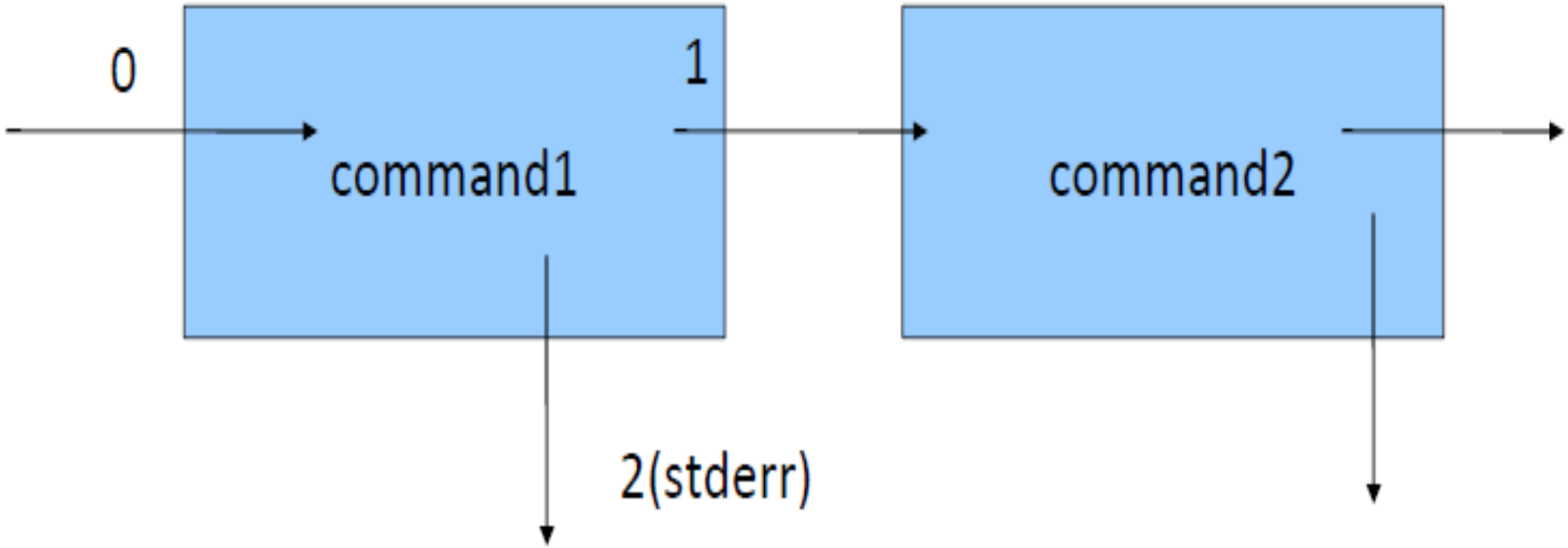
进程管理相关的命令-进程的挂起和恢复

- 进程的中止（挂起）和终止
  - 挂起（Ctrl+Z）
  - 终止（Ctrl+C）
- 进程的恢复
  - 恢复到前台继续运行（fg）
  - 恢复到后台继续运行（bg）
- 查看被挂起的进程（jobs）

并发执行

- 程序进程合并 — (cmd; cmd; cmd)
- 后台执行 — cmd &

管道




```
ls /usr/bin | sort | more
```

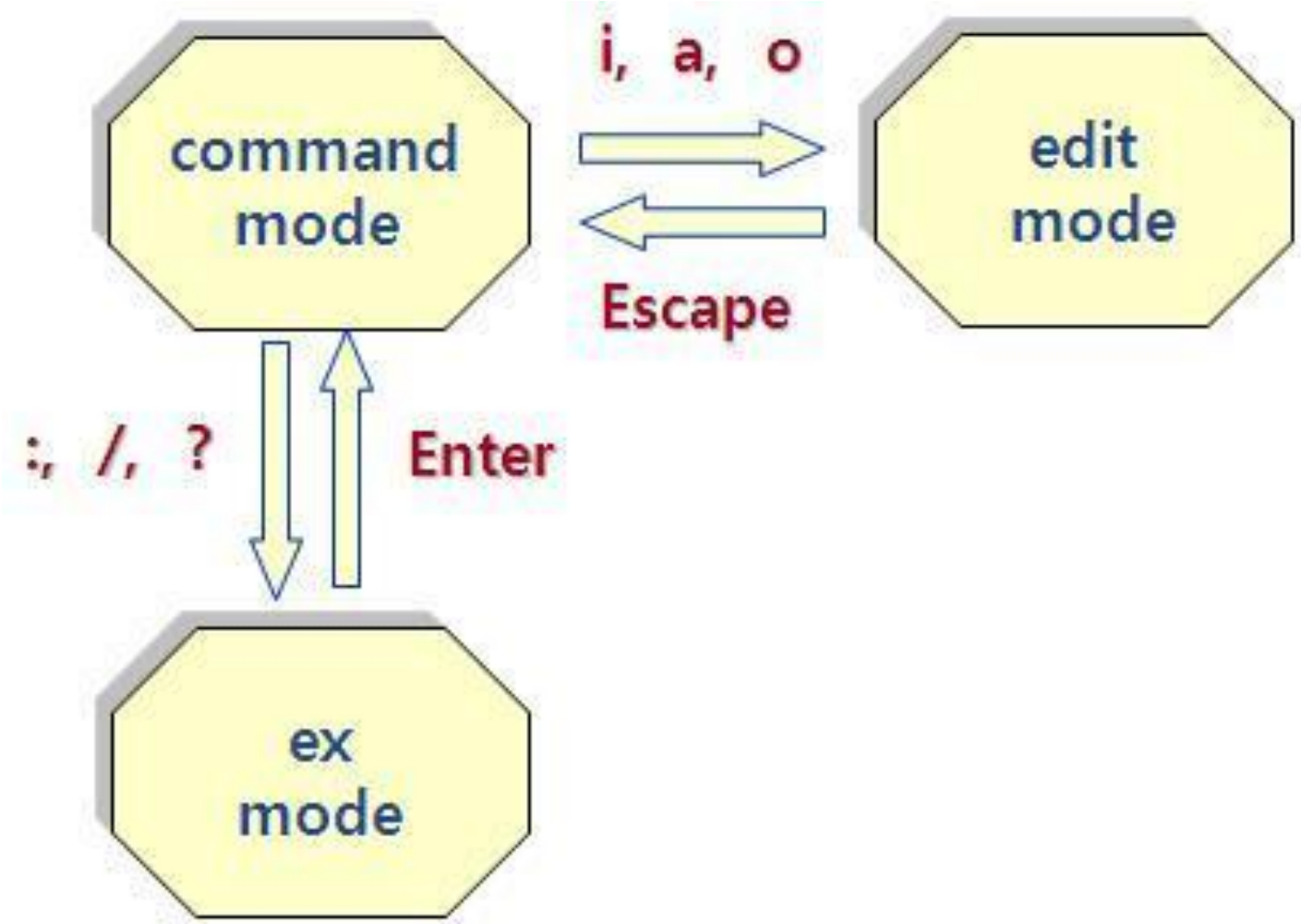
Vi 简介

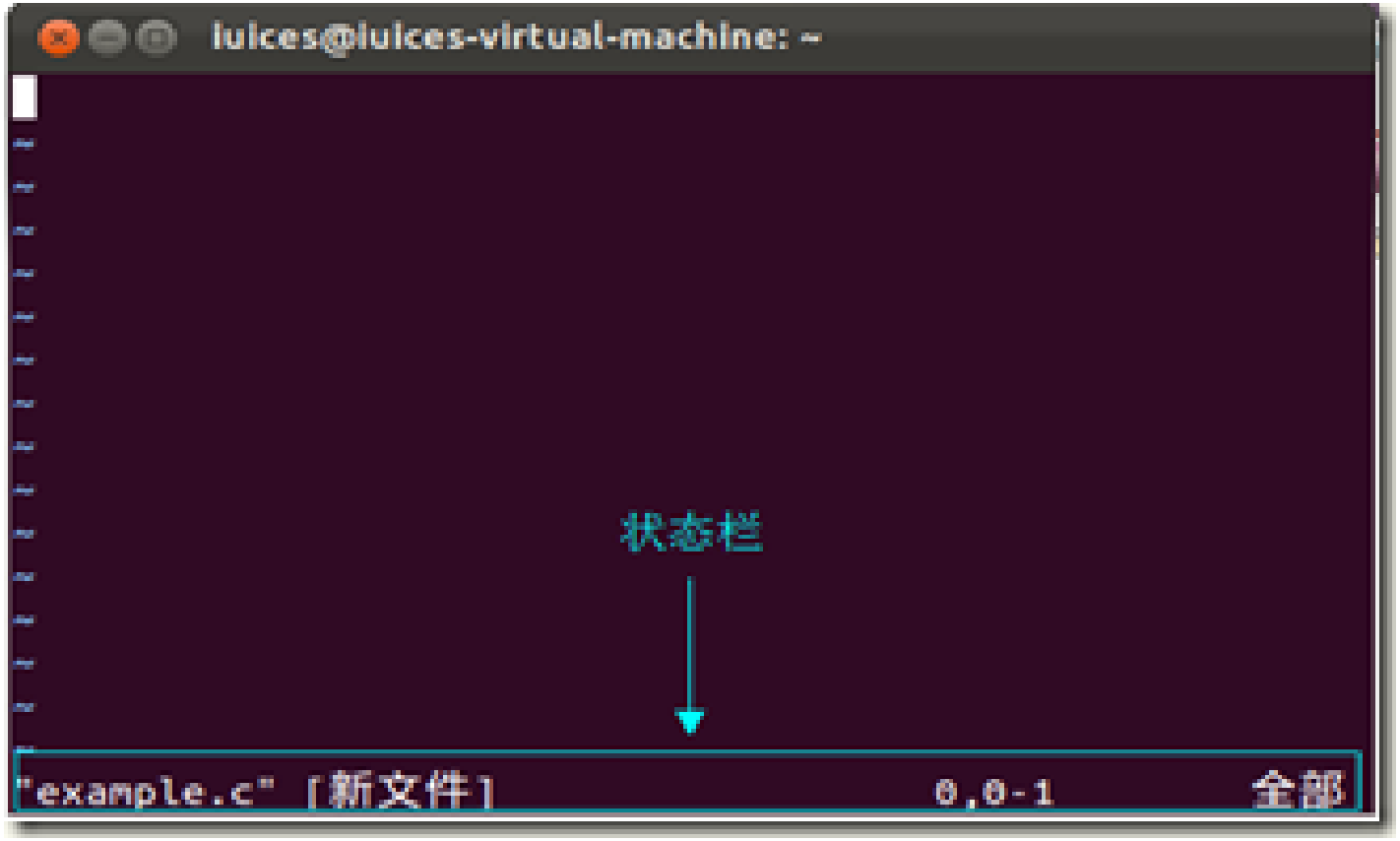
vi文本编辑器

- Linux/Unix下的配置文件都是文本文件
- vi是使用最广泛的编辑器
- vi分为三种工作模式：
  - 一般模式
  - 编辑模式
  - 命令模式

vi文本编辑器

- vi的一般模式：
  - 以vi处理一个文件的时候，一进入该文件就是一般模式了（vi命令启动后的默认状态）。在这个模式中，可以使用上下左右键来移动光标，也可以使用删除字符或删除整行来处理文件内容，也可以使用复制、粘贴。
- vi的编辑模式：
  - 在一般模式中可以处理删除、复制、粘贴等动作，但是却无法编辑。
  - 按下i, I, o, O, a, A等字母之后才会进入编辑模式，进入该模式后，左下方会出现INSERT或 REPLACE等字样。
  - 按下ESC就可以回到一般模式


command mode
i, a, o
edit mode
Escape
:, /, ?
Enter
ex mode


iuices@iuices-virtual-machine: ~
状态栏
"example.c" [新文件]
0,0-1
全部

vi文本编辑器

- vi的进入编辑模式的命令详解
  - i 从光标所在位置前开始插入文本
  - I 将光标移到当前行行首，然后在其前插入文本
  - a 用于在当前光标位置之后插入文本
  - A 把光标移到行尾，从那里开始插入文本
  - o 在光标所在行的*下面*新开一行，并将光标置于新行行首，等待输入文本
  - O 在光标所在行的上面新开一行，并将光标置于新行行首，等待输入文本

vi文本编辑器

- vi的命令模式
  - 在一般模式当中，输入 : 或 / 就可以将光标移动到最底下那一行，从而进入命令模式。
  - 在这个模式当中，可以进行搜索、保存、退出、读取另外一个文件、显示行号等动作。

vi文本编辑器

- vi的举例：

  输入 vi test.txt

  按下i，进入编辑模式

  输入hello

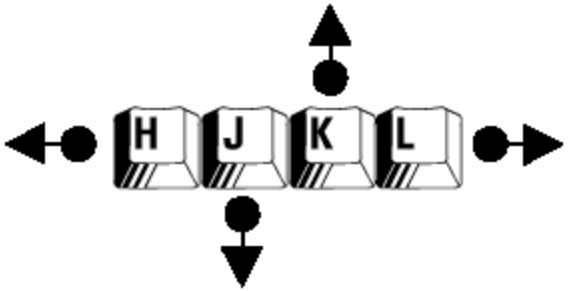
  按下Esc进入一般模式

  输入:wq回车保存退出

vi文本编辑器

- vi一般模式下常用命令（1/3）：

h j k l	用于光标左移、下移、上移、右移
[Ctrl] + [f]	屏幕向前移动一页
[Ctrl] + [b]	屏幕向后移动一页
0	移到行的第一个字符处
$	移到行的最后一个字符处
G	移到文件的最后一行
nG	移动到文件的第n行
n<Enter>	光标向下移动n行
nH	光标移到当前屏幕的第n行行首



vi文本编辑器

- vi一般模式下常用命令（2/3）：
  - x　向后删除一个字符
  - nx　向后删除n个字符
  - X　向前删除一个字符
  - dd　删除光标所在的行
  - ndd　删除光标所在行的向后n行
  - yy　复制光标所在行
  - nyy　复制光标所在行的向后n行
  - p　粘贴到光标的下一行
  - P　粘贴到光标的上一行

vi文本编辑器

- vi一般模式下常用命令（3/3）：

/word	在光标之后查找word
?word	在光标之前查找word
n	向相同的方向移动到有word的地方
N	向相反的方向移动到有word的地方
u	撤销上次操作
.	重做
ZZ	存盘退出
ZQ	不保存退出

vi文本编辑器

- vi命令模式下常用命令（1/3）：

:n	跳到第n行
:q	退出，如果有修改没有保存会提示
:wq	存盘退出
:q!	不保存退出
:w	存盘
:w file	将当前编辑内容保存到file
:r file	读取另外一个文件file的数据，插入到光标所在行的后面
:f newfile	将当前文件重命名到newfile

vi文本编辑器

- vi命令模式下常用命令（2/3）：

  :n1,n2s/word1/word2/g

  在第n1与n2行之间寻找word1这个字符串，并将该字符串替换为word2

  :1,$s/word1/word2/g

  从第一行到最后一行寻找 word1 字符串，并将该字符串替换为word2（. 可以表示当前行，$可以表示最后一行）

  :1,$s/word1/word2/gc

  从第一行到最后一行寻找word1字符串，并将该字符串替换为word2。且在替换前显示提示字符给使用者确认。

vi文本编辑器

- vi命令模式下常用命令（3/3）：

:n1,n2 co n3	第n1行到n2行的内容拷贝到n3后
:n1,n2 m n3	第n1行到n2行的内容移动到n3后
:n1,n2 d	删除第n1行到n2行的内容
:!Cmd	运行Shell命令Cmd
:r !Cmd	将命令行运行的结果写入当前位置
:n1,n2 w !Cmd	将第n1行到n2行的内容作为Cmd的输入运行命令，如果没有指定n1和n2，则将整个文件作为输入。
:set nu	显示行号
:set nonu	隐藏行号

Sed—字符流编辑器

- 命令格式 sed <表达式> [file]
- 基本表达式
- [address-range]p 打印指定的行
- [address-range]d 删除指定的行
- s/pattern1/pattern2/ 用pattern2字符串替换pattern1
- [address-range]s/pattern1/pattern2/ 指定行数(同上)
- [address-range]y/pattern1/pattern2/ 逐个字母替换
- **g 整行内全局操作，加在表达式的尾部**

e.g.: $ sed –n ‘1p’ /etc/passwd

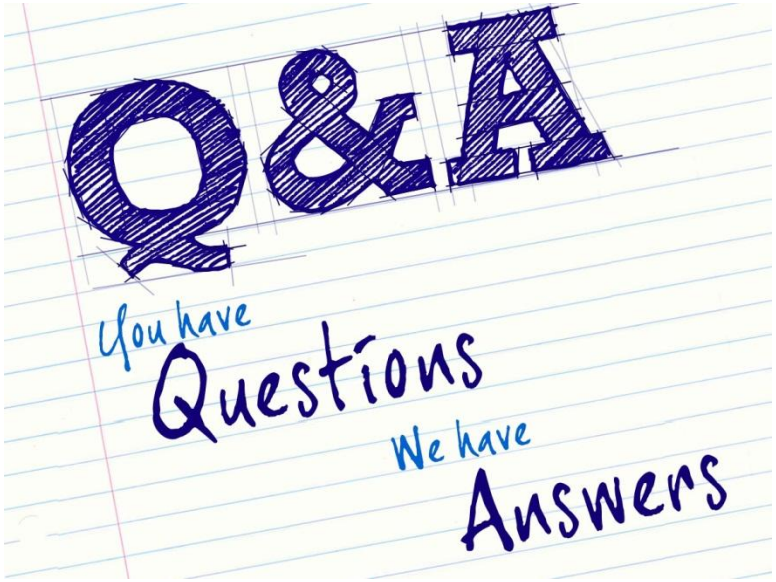
$ sed ‘1,10d’ /etc/passwd

$ sed ‘s/192\.168\./192.169./g’ /etc/hosts

awk

- awk /nawk/gawk pattern {action}
- 逐行处理输入信息，对复合pattern要求的行执行action动作，可自动识别空白字符，将整行分割成不同的字段。

- e.g.: awk '/101/' file
- du –s /home/* | awk '{ if ($1 > 100) print $1,$2}'


Q&A
You have
Questions
We have
Answers