# linux 基础

# vim编辑器

文本编辑器 纯ASCII码的编辑器

字处理器 wps word 等，有多余的修饰符

Linux基本原则，配置文件都是纯文本。

**nano** 全屏编辑器

**sed** 流处理编辑器

**vi** visual interface (可视化接口) 遵循GPL协定的开源软件

**vim** （VI iMproved） vi增强

vim 是全屏编辑器，又是一个**模式化**的编辑器。

## 打开

**vim /xxx/xxx/file**

如果file存在则打开此文件，如果不存在新建此文件。

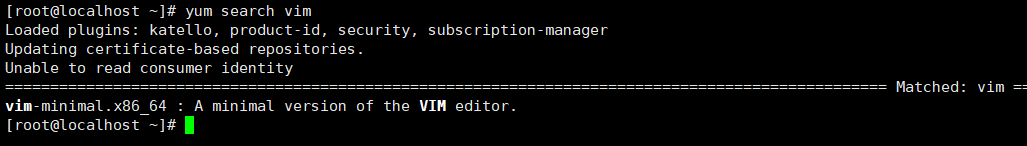
whereis vim

which vim

yum search vim

yum search vim-minimal.x84\_64

如果没有vim 编辑器



**vim +n file** 让光标在第n行



**vim + file** 让光标在最后一行



**vim + /pattern** 第一次被pattern匹配到的行的行首



## vim 模式

**1.编辑模式（命令模式）**

**2.输入模式**

**insert append o**

**3.末行模式**

**可以在：后面输入命令，对文本进行操作**

**模式转换**：

**编辑模式---> 输入模式**

小写

**i** (insert) 当前光标所在字符的**前面**转为输入模式

**a** (append) 当前光标所在字符的**后面**转为输入模式

**o** 当前光标所在行的**下方新建一行**，并转为输入模式

大写

**I** 当前光标所在行的**行首**转为输入模式

**A** 当前光标所在行的**行尾**转为输入模式

**O** 当前光标所在行的**上方新建一行**，并转为输入模式

**输入模式 ---> 编辑模式**

**ESC 键**

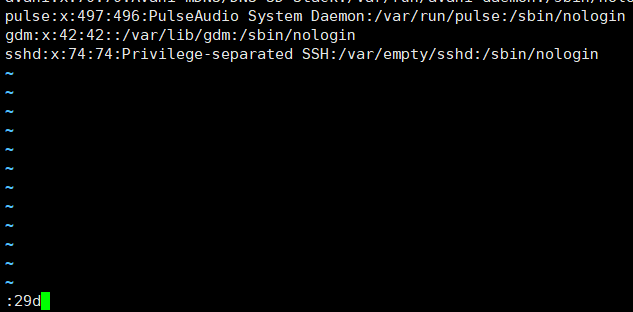
**想要进入末行模式，需要从编辑模式进入，输入模式无法进入末行模式。**

**编辑模式 ---> 末行模式**

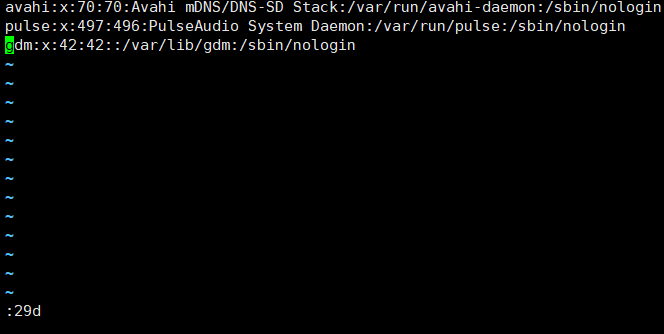
**末行模式功能非常强大**

编辑模式进入末行模式输入 **：**

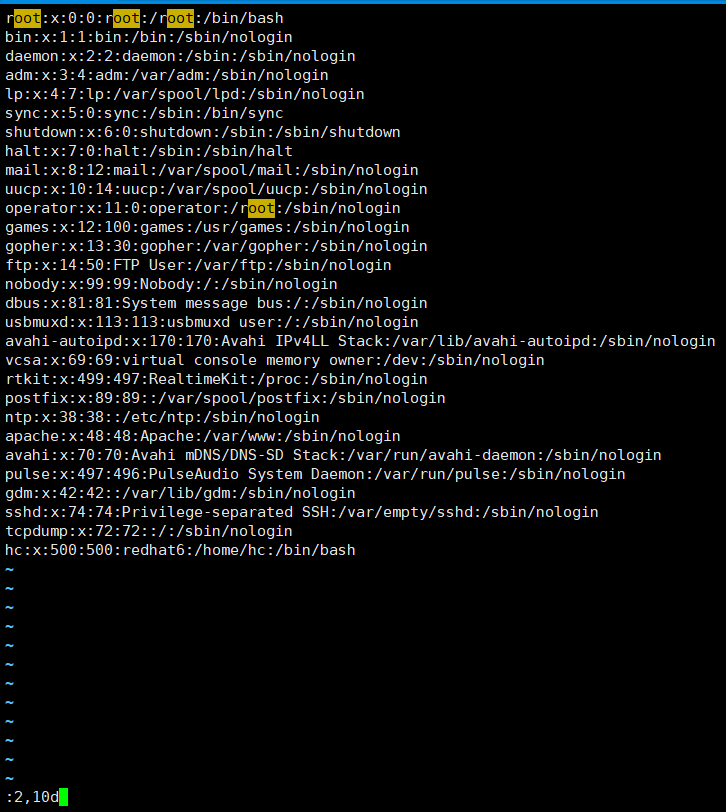
vim passwd 进入末行模式



删除第29行 sshd 敲enter回车 就删除了

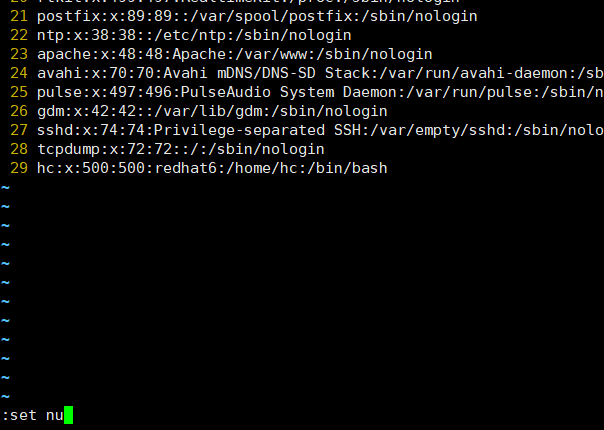


**2，10d** 删除第2行到第10行



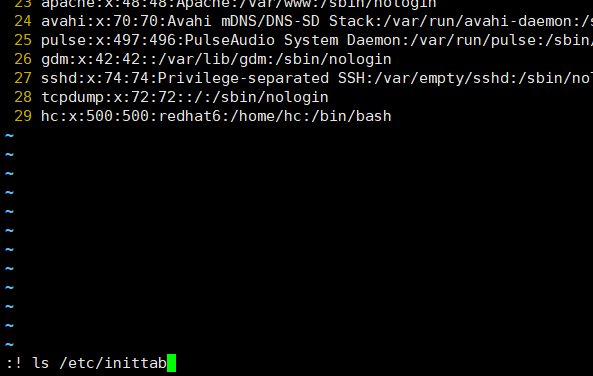
还可以显示行号

**set nu**



还可以使用命令查看其他文件

:! ls /etc/inittab



enter后已进入到编辑模式



**末行模式--->编辑模式**

**ESC键**

## 关闭文件

**末行模式关闭文件：**

**:q** 退出并不会保存内容

**:wq (:X)** 保存并退出

**:q!** 不保存并退出

**:w** 保存 ，如果是一个只读文件没有w权限，是保存不了的

**:w!** 强行保存 ,只有管理员有这个权限

**编辑模式下关闭文件：**

**ZZ （大写） shift zz**

## 移动光标（编辑模式下）

**逐个字符移动**

**h** 向左移动

**l**  向右移动

**j** 向下移动

**K** 向上移动

**数字 h**  3h 向左移动3个字符

**逐个单词移动**

**w** 移动到下一个单词词首

**e** 跳到当前或者下一个单词词尾

**b** 跳到当前或者前一个单词的词首

数字 w 3w

**行内跳转**

0 跳转当前行首(绝对行首 空白字符也算)

^ 跳转当前行首(行首非空白字符)

$ 绝对行尾

**行间跳转**

#G 5G 跳转到第5行的行首

G 跳到最后一行

**行首行尾跳转**

**shift + {** 跳到文件开头

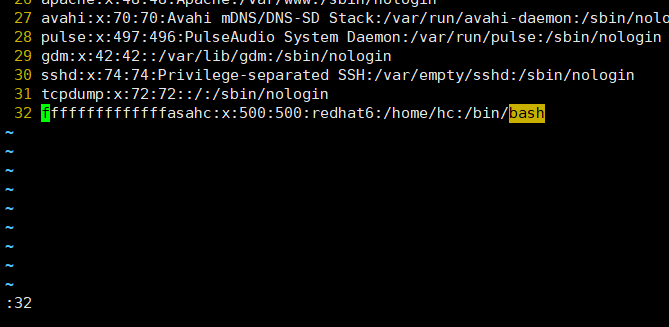
**shift + }**  跳到文件行尾

**段落跳转**

**shift +(**  跳到下一句开头

**shift +)**  跳到上一句开头

末行模式跳转



## 翻屏（编辑模式下）

**ctr+f 向下翻一屏**

**ctr+b 向上翻一屏**

**ctr+d 向下翻半屏**

**ctr+u 向上翻半屏**

## 删除单个字符（编辑模式下）

**x** 删除光标所在向后的单个字符

**#x** 逐个删除光标所在先后的n个字符

## 删除命令（编辑模式下）

**d + 编辑模式下的跳转命令**，就删除到位置**。**

**d$** 从光标当前位置删除到行尾

**d ctrl {** 从光标当前位置删除到文件末尾

**d ctrl (** 从光标当前位置删除到句末

**d0** 从光标当前位置删除到行首

**de** 从光标当前位置删除一个单词

**dj d↓** 从光标的位置向下删除两行

**dK d↑** 从光标的位置向上删除两行

**dh d←**

**dl d→**

**dw** 从光标处一个单词一个单词的向后删除

**#dw** 从光标处向后逐个删除n个单词

**db** 从光标处向前一个单词一个单词删除

**#db** 从光标处向前逐个删除n个单词

**dd** 删除光标所在行

**#dd** 从光标处向后删除n行

**末行模式下**

**10,20d** 删除10行到第20行

**:.,+9d** 从光标处向下删除(. 表示当前行) 10行

**:.,$** 从光标处删除到行尾

**:.,$-2d** 从光标处删除指定行

**删除的内容会保存在缓存中，可以撤销，最后一次删除可以粘贴**

## 粘贴命令（编辑模式下）

**p (小写) 如果删除的或者复制的为整行的内容，则粘贴在光标所在的下方，如果复制的内容为非整行，则粘贴在光标所在字符的后面**

**P (大写) 如果删除的或者复制的为整行的内容，则粘贴在光标所在的上方，如果复制的内容为非整行，则粘贴在光标所在字符的前面**

## 复制命令 （编辑模式下）

**y** 用法同d命令

**yy 复制1行**

**3yy 复制3行**

## 先删除内容在转入输入模式

**c 用法同d命令**

**cc 一行**

**ce， cw ， c (上下左后) ， ch， ck， cl， cj， c0， c^，**

**c ctrl ( ，**

**c ctrl ) ，**

**c ctrl { ，**

**c ctrl }**

## 替换单个字符

**r**  替换光标出单个字符

**r** s 将光标处字符替换成s

#r s 把光标处后面的n个字符替换成s 很少用

## 替换模式

**R 或者 两下insert键**

直接把光标处后面的字符替换为标准输入的字符

## 撤销编辑操作

**u**

撤销前一次对文件的编辑操作

**#u**

撤销最近n次对文件的编辑操作

**u不停** 撤销到缓冲区缓存的操作次数为止

## 取消撤销

如果把不想撤销的给撤销了，想要取消撤销

**ctrl + r**

重复前一次编辑的操作

**.**

## 可视化模式

v 选中具体到字符（字符选取）

V 选中具体到行 （矩形选取）

选中后使用命令编辑

## 查找

/pattern

?pattern

n 向下找匹配的pattern

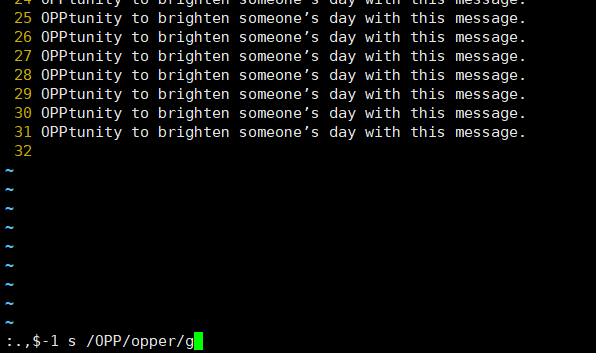
N 向上

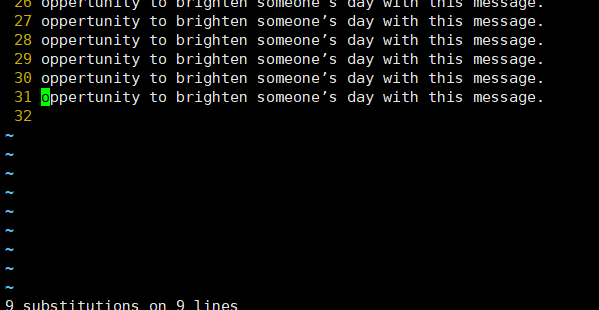
:/want

## 查找并替换

在末行模式下使用s命令

和sed命令用法相同





: .,$-1 s /OPP/opper/g

**.,$-1** 从光标处到倒数第一行

**1，$ == %**  表示全文 查找并替换

**% s /OPP/opper/g** 全文查找OPP并替换为opper

## 编辑多个文件

vim file1 file2 file3

默认打开file1

在file1的末行模式下输入**next** 显示file2

在file2的末行模式下输入**prev 显示** file1

**next** 切换到下一个文件

**prev** 切换到上一个文件

**last** 切换到最后一个文件

**first** 切换到最前一个文件

退出

**:qa** 全部退出

## 分屏显示文件

**ctrl+w s** 水平拆分

**ctrl+w v** 垂直拆分

**ctrl+w ↓↑** 移动到哪个窗口

**ctrl+w ←→** 移动到哪个窗口

**:qa** 关闭全部窗口

## 分窗口显示或者编辑多个文件

vim -o file1 file2 水平分割显示多个文件

vim -O file1 file2 垂直分割显示多个文件

:qa 关闭全部窗口

## 当前文件中的内容存为另外一个文件

末行模式下使用w命令

**:w /xxx/xxx/file**

**:addr1,addr2 w /xxx/xxx/file**

## 另外一个文件的内容填充在当前文件中

末行模式使用r命令 read

**:r /xxx/xxx/file**

## 跟shell交互

**:! commond**

## 高级话题

**1显示行号**

**:set number 简写成 set nu**

**不显示行号**

**:set nonu**

**2忽略大小写**

**:set ignorecase**

**简写 :set ic**

**不忽略大小写**

**:set noignorecase**

**简写 :set noic**

**3设定自动缩进**

**:set autoindent**

**简写 :set ai**

**4** **高亮显示搜索到的文本或不高亮显示**

:set hlsearch

:set nohlsearch

5 **语法高亮**

**:syntax on**

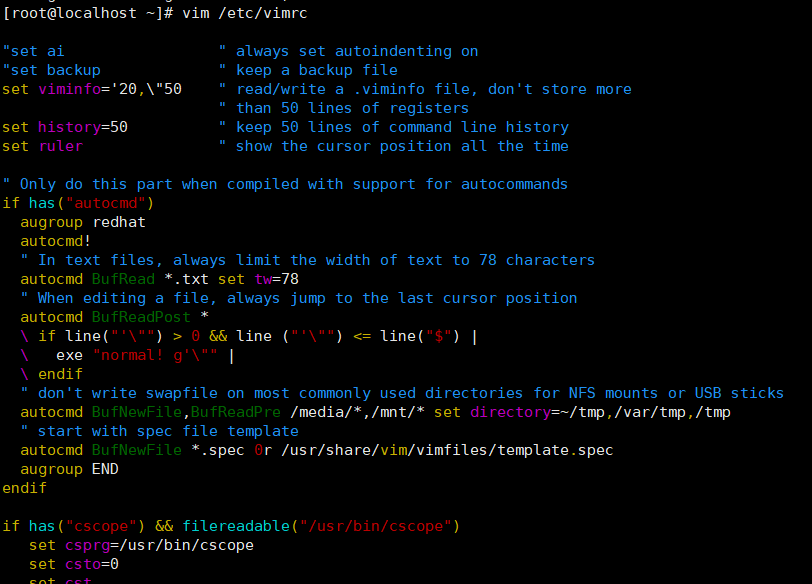
**:syntax off**

这些设定只对当前vim进程有效，如果要想长期有效，需要设定配置文件。

## 配置文件

全局，对所有用户都有效

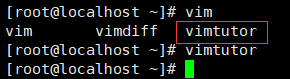
**/etc/vimrc**

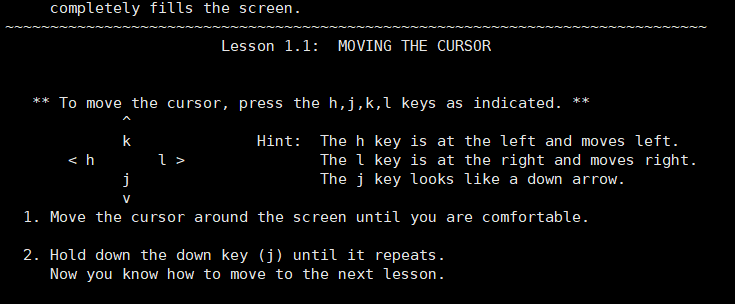


以用户家目录中的 .vimrc 针对用户个人配置为准

~/.vimrc 是隐藏的

## vim学习帮助





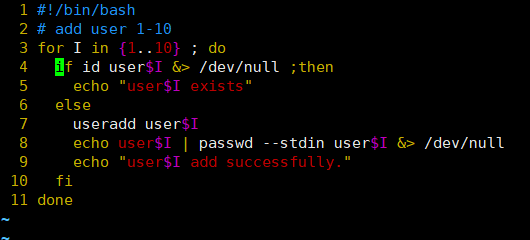
## 从错误中恢复

如果正在编辑的文件突然非法退出，那没有保留到有.xxx.swp文件

使用 vim -r xxx 就可以从.swp 文件恢复回来

## vim 编写脚本

### 添加10个用户



#!/bin/bash

# add user 1-10

for I in {1..10} ; do

if id user$I &> /dev/null ;then

echo "user$I exists"

else

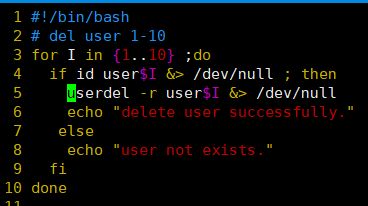
useradd user$I

echo user$I | passwd --stdin user$I &> /dev/null

echo "user$I add successfully."

fi

done



#!/bin/bash

# del user 1-10

for I in {1..10} ;do

if id user$I &> /dev/null ; then

userdel -r user$I &> /dev/null

echo "delete user successfully."

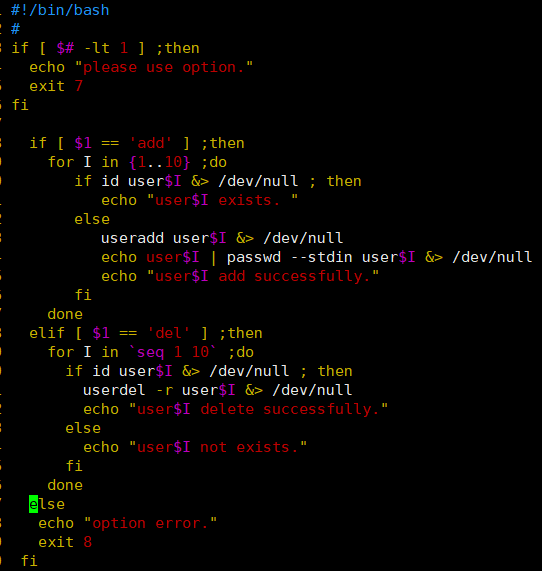
else

echo "user not exists."

fi

done

### 接受参数添加用户的删除和添加



#!/bin/bash

#

if [ $# -lt 1 ] ;then

echo "please use option."

exit 7

fi

if [ $1 == 'add' ] ;then

for I in {1..10} ;do

if id user$I &> /dev/null ; then

echo "user$I exists. "

else

useradd user$I &> /dev/null

echo user$I | passwd --stdin user$I &> /dev/null

echo "user$I add successfully."

fi

done

elif [ $1 == 'del' ] ;then

for I in `seq 1 10` ;do

if id user$I &> /dev/null ; then

userdel -r user$I &> /dev/null

echo "user$I delete successfully."

else

echo "user$I not exists."

fi

done

else

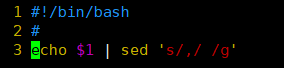
echo "option error."

exit 8

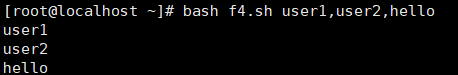
fi

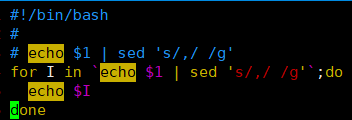
### 添加指定用户

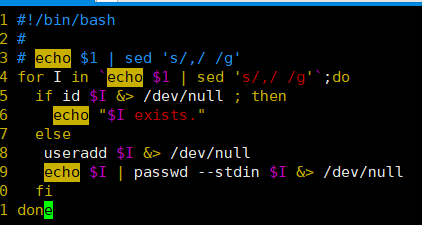




$I 引用下标，代表真实的参数







#!/bin/bash

#

# echo $1 | sed 's/,/ /g'

for I in `echo $1 | sed 's/,/ /g'`;do

if id $I &> /dev/null ; then

echo "$I exists."

else

useradd $I &> /dev/null

echo $I | passwd --stdin $I &> /dev/null

fi

done

### 选项添加指定用户



#!/bin/bash

#

if [ $1 == '--add' ]; then

for I in `echo $2 | sed 's/,/ /g'`; do

if id $I &> /dev/null ; then

echo "$I exists,not add"

else

useradd $I &> /dev/null

echo $I | passwd --stdin $I &> /dev/null

fi

done

elif [ $1 == '--del' ]; then

for I in `echo $2 | sed 's/,/ /g'`; do

if id $I &> /dev/null; then

userdel $I &> /dev/null

echo "$I delete sucessfully."

else

echo "$I not exists,not delete."

fi

done

else

echo "error option."

fi

# 组合条件测试

**条件测试类型**

**整数测试**

**-eq -lt -le -ge -gt -ne**

**字符（串）测试**

**== ！= > < -n（空为真） -z (不空为真)**

**文件测试**

**-e -f -d -r -w -x**

**组合条件测试**

**-a : 与关系**

**-o : 或关系**

**! : 非关系**

**条件测试的表达式**

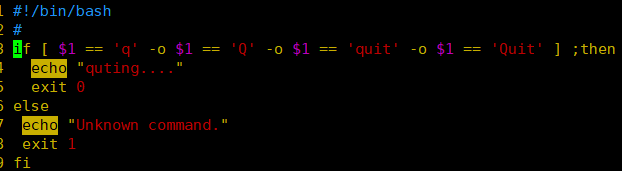
**[ expression ]**

**[[ expression ]]**

**test expression**

$#大于等于1 小于等于4 两种方法都可以

**if [ $# -gt 1 -a $# -lt 4 ]** ==  **if [ $# -gt 1 ] && [ $# -lt 4 ]**



#!/bin/bash

#

if [ $1 == 'q' -o $1 == 'Q' -o $1 == 'quit' -o $1 == 'Quit' ] ;then

echo "quting...."

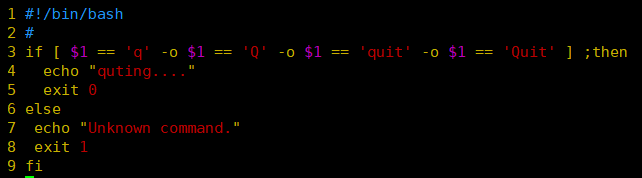
exit 0

else

echo "Unknown command."

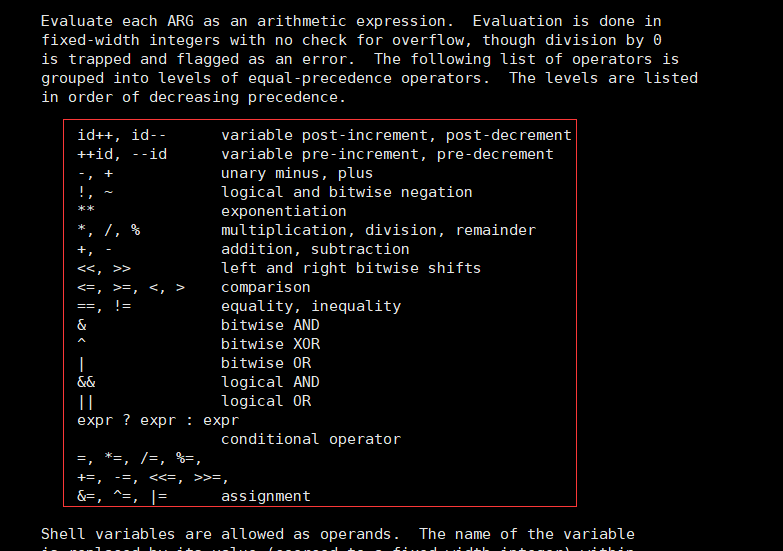
exit 1

fi



不让高亮末行模式下输入 **:noh**

let 运算



let sum+=I

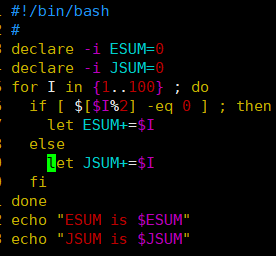
let i++

let i--

let ++i

let --i

100以内的偶数和和奇数和



#!/bin/bash

#

declare -i ESUM=0

declare -i JSUM=0

for I in {1..100} ; do

if [ $[$I%2] -eq 0 ] ; then

let ESUM+=$I

else

let JSUM+=$I

fi

done

echo "ESUM is $ESUM"

echo "JSUM is $JSUM"