

一 vim(文本编辑器)

1 什么是 VIM

是一个类似 vi 的文本编辑器，不过在 vi 的基础上增加了很多新特性，vim 被认为是类 vi 编辑器中最好用的一个。

2 为什么要学习 VIM

在 vi 基础上增加了一些小功能，可以快速排查问题

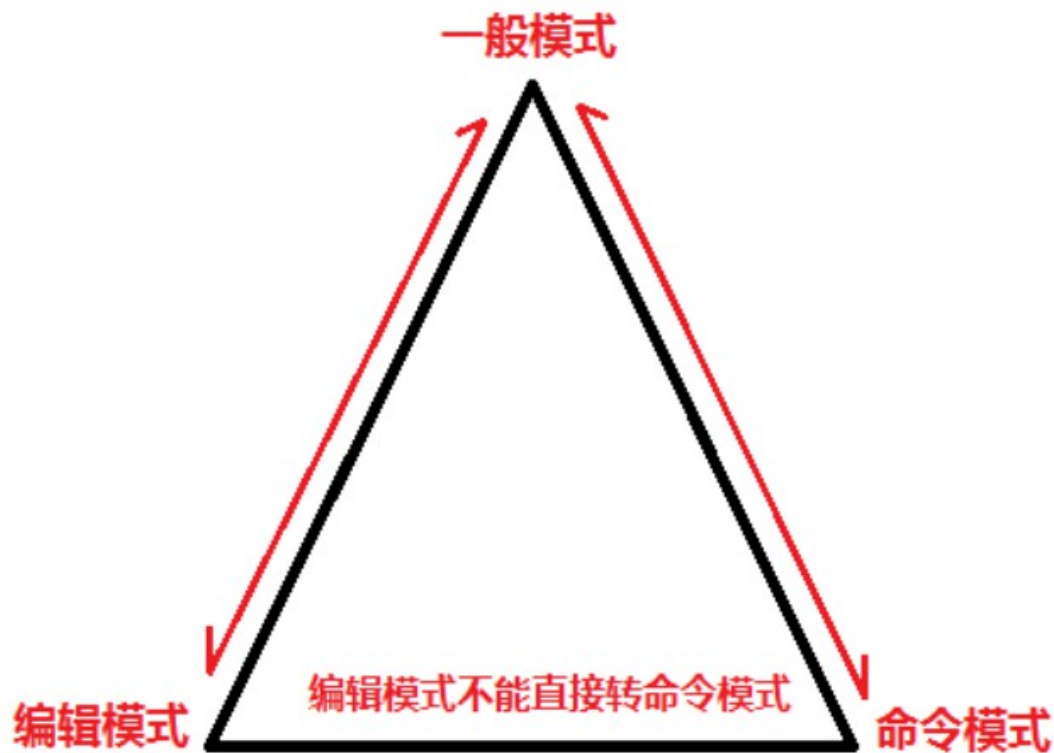
很多系统内建 vi 编辑器，其它文本编辑器不一定有，很多软件会主动调用 vi 的接口

3 Vim 的三种模式

一般模式、编辑模式、命令模式

一般模式是 vim 打开文件后默认进入的模式，这个模式不能插入字符，但可以设定 vim 的工作方式

三种模式转换图



4 一般模式快捷键

一般模式(默认模式)的快捷键:

h 或←光标左移一个字符。如果是 20h, 表示左移 20 个字符。

j 或↓光标下移一个字符 同上

k 或↑光标上移一个字符 同上

l 或→光标右移一个字符 同上

[Ctrl]+[f]屏幕向下移动一页 Page Down!!

[Ctrl]+[b]屏幕向上移动一页 Page Up !!

0 或[Home]移动到此行最前面字符处!!

\$或[End]移到光标所在行的行尾!!

H 光标移到当前屏幕最上方行的第一个字符!!

M 光标移到当前屏幕中间行的第一个字符!!

L 光标移动到当前屏幕最下方行第一个字符!!

G 移到此文件最后一行!!!

nG 移到第 n 行

gg 相当于 1G, 即移到第一行!!!

n[Enter]光标下移 n 行

/word 向下查找单词 “word” (! ! !)

? word 向上查找单词 “word” (! ! !)

n 表示重复前一个查找操作

N 与 n 相反 (反向查找)

yy 复制光标所在行 (! !)

nyy 复制光标所在向下 n 行(n 为数字)

y1G 复制光标所在行到第一行所有数据

yG 复制光标所在行到最后一行所有数据

y\$复制光标所在处到同行最后一个字符

y0 复制光标所在处到同行第一个字符

p 将已复制的数据粘贴到光标所在下一行

P 将已复制的数据粘贴到光标所在上一行

u 复原前一个操作 (类似于 windows 中的 ctrl+z) !!!

Ctrl+r 恢复一个操作。

x 向后删除一个字符

nx 向后删除 n 个字符(n 为数字)

X 向前删除一个字符

dd 删除光标所在行 (! ! !)

ndd 删除光标所在行以下 n 行(n 为数字,包含当前行在内)

d1G 删除光标所在行到第一行所有数据 (包括所在的行)

dG 删除光标所在行到最后一行 (! !)

d\$或 d end 删除光标所在处到同行最后一个字符 (! !)

d0 或 d home 删除光标所在处到同行第一个字符。 (! !)

r 替换光标所在处字符一次

R 一直替换光标所在处文字直到按下 Esc(!!!)

5 命令模式快捷键

如何进入命令模式:

: ? /	三个符号任意都可以进入命令模式
-------	-----------------

:w [filename] 另存为 filename

:r [filename] 读取 filename 指定文件中的内容到光标所在的行。

:n1,n2 w [filename] 将 n1 到 n2 行另存为 filename

:! command 临时切换到命令行模式下执行 command 命令。

例如 “:!find / -name Hello.java” 即可在 vim 当中执行命令。

:wq 保存后离开

:q 不保存离开(未改可以离开，如果修改了需要 q!强制离开)

:q! 不保存强制离开

:set nu 显示行号 (number)

:set nonu 取消显示行号 (noNumber)

:s/word1/word2/g 在当前行将 word1 替换成 word2 (! !)

:%s/word1/word2/g 在当前文件将 word1 替换成 word2 (! !)

**:n1,n2s/word1/word2/g 在 n1 到 n2 行查找 word1 替换成 word2
(n1、n2 为数字)**

:10,\$ s/word1/word2/g 从第一行到最后一行查找 word1 替换成 word2

:%s/word1/word2/gc 同上，在替换前确认是否替换。(!!!) 只能单行确认，需要逐个确认。

替换为 b (y/n/a/q/l/^E/^Y)?

y 表示 yes, n 表示 no, a 表示 all(限光标当前到最后一行), q 表示 quit, l 表示替换后移动光标到行首, ^E(Ctrl+E)表示向下翻, ^y(Ctrl+Y)表示向上翻。

6 编辑模式快捷键

进入编辑模式：

i 从光标所在处插入(!!!)

I 从所在行第一个非空白字符处插入(! !)

a 从光标所在下一个字符处插入

A 从光标所在行最后一个字符处插入 (! !)

o 在光标所在处下一行插入新的一行(! !)

O 在光标所在处上一行插入新的一行 (! !)

二 压缩打包的概念

1 压缩：

指通过某些算法，将文件尺寸进行相应的缩小，同时不损失文件的内容。

2 打包：

指将多个文件（或目录）合并成一个文件，方便传递或部署。

在 Linux 系统中，文件的后缀名不重要，但是针对于压缩文件的后缀名是必须的，因为可以让其他的程序员根据文件的后缀名使用对应的算法进行解压。

3 Linux 常见的压缩文件后缀名：

*.gz	gzip 程序压缩的文件
*.bz2	bzip2 程序压缩的文件
*.tar	tar 命令打包的数据,并没有压缩过
*.tar.gz	tar 程序打包的文件,并且经过 gzip 的压缩
*.tar.bz2	tar 程序打包的文件,并且经过 bzip2 的压缩

三 压缩解压(gzip、bzip2)

1 gzip:

压缩/解压命令

选项:

-c :	将压缩的数据输出到标准输出 (stdout) 上
-d :	解压缩
-t :	可以用来检验一个压缩文件的一致性,看看文件有无错误
-v :	可以显示出原文件/压缩文件的压缩比等信息
-(1,2,...,9):	压缩等级,1 最快,但是压缩比最差; 9 最慢,但是压缩比最好, 默认是 6。
-l :	查看压缩文件的压缩比: <code>gzip -l *.gz</code>

案例:

```
cp /root/install.log /home/gzip
```

```
1, gzip -c install.log //将压缩的数据输出到标准输出
```

```
2, gzip -v install.log //压缩完显示
```

这时发现源文件不在了, 如果想保留源文件, 可以用数据重导向技术

```
3, gzip -d install.log.gz //解压
```

```
4, gzip -c install.log > install.log.gz
```

```
5, gzip -t install.log.gz //检查文件是否有误
```

```
6, gzip -c9v install.log //提高压缩比 (文件如果本身很小可能体现不出来)
```

练习:

在/tmp 文件夹下创建 part1/gzip

将/root/anaconda-ks.cfg 文件拷贝到/tmp/part1/gzip

将拷贝后的文件进行 gzip 压缩,并显示压缩信息。

将压缩后文件的名称改为 mygzip01.gz

2 bzip2:

压缩/解压命令:

选项:

-c :	将压缩的过程产生的数据输出到标准输出 (stdout)
-d :	解压缩
-k :	保留源文件,而不会删除原始的文件
-v :	可以显示出原文件/压缩文件案的压缩比等信息;
-(1,2,...,9):	与 gzip 同样的,都是在计算压缩比的参数,-9 最佳,-1 最快

通过 bzip2 命令压缩 install.log 文件

```
#bash
bzip2 -kv install.log
```

3 bzip2 和 gzip 的对比

gzip 拥有更快的压缩性能。压缩更快
bzip2 拥有更高的压缩比。压缩文件更小
单纯从压缩比方面来说，那么 bzip2 > gzip
压缩性能方面来说 gzip>bzip2

4 查看压缩文件中的内容：

cat：可以用来查看文本文件中的内容。
zcat：可以用来查看 gzip 算法压缩的压缩文件内容。
bzcata：可以用来查看 bzip2 算法压缩的压缩文件内容。

四 打包解包(tar)

可以将一个文件/夹打包成一个文件。可以结合 gzip、bzip2 的算法对包文件进行相应的压缩和解压。

1 语法：

压缩：tar [选项] newFileName.tar.gz sourceFileName
解压：tar [选项] fileName.tar.gz [-C /path]

2 选项：

-c：	建立打包文件,
-t：	查看打包文件的内容含有哪些文件
-x：	解打包或解压缩的功能,可以搭配-C(大写)在指定目录解开
-j：	通过 bzip2 的支持进行压缩/解压缩:此时文件最好为 *.tar.bz2

-z :	通过 gzip 的支持进行压缩/解压缩:此时文件最好为 *.tar.gz
-v :	在压缩/解压缩的过程中,将正在处理的文件名显示出来
-f filename:	-f 后面跟处理文件的全名称 (路径+文件名+后缀名)
-C 目录:	这个选项用在 解压的时候 ,若要在特定目录解压,可以使用这个选项

注:

使用命令进行打包、压缩的时候,使用了什么算法,文件后缀名就一定要与其对应。

3 案例:

压缩:

- 1、使用 gzip 的算法进行打包压缩。

```
# bash
tar -zcvf install.log.tar.gz install.log
```

注意 tar 的语法, tar -zcvf newFile sourceFile

- 2、使用 bzip2 的算法进行打包压缩。

```
# bash
tar -jcvf install.log.tar.bz2 install.log
```

- 3、如果想要压缩指定目录中的内容是,可以考虑使用绝对路径。

```
# bash
tar -zcvf [path]/newFileName.tar.gz [path]/sourceFile
```

解压:

- 1、将一个压缩包文件解压到当前目录下

```
# bash
tar -zxvf install.log.tar.gz
执行完成之后,文件会在当前的目录下。
```

- 2、将一个压缩包文件解压到指定目录下

```
# bash
tar -zxvf install.log.tar.gz -C /
```

- 3、只解压包中的某个文件

```
# bash
tar -zxvf etc.tar.gz etc/shells
```


4、配置jdk 环境变量：

```
# bash
tar -zxvf jdk-8u131-linux-x64.tar.gz
cd jdk1.8.0_131
pwd # 复制路径
vim /etc/profile # profile 文件是系统环境变量的配置文件
                在该文件的最后一行添加内容：
                export JAVA_HOME=/home/software/jdk1.8.0_131
                export PATH=$JAVA_HOME/bin:$PATH
                保存退出
source /etc/profile
                使环境变量生效
```

五 软件管理(rpm)

最初只有.tar.gz 的打包文件，用户必须编译每个他想要在 Linux 上运行的软件。用户们普遍认为系统很有必要提供一种方法来管理这些安装在机器上的软件包，当 Debian 诞生时，这样一个管理工具也就应运而生，它被命名为 dpkg。稍后 RedHat 才决定开发自己的“rpm”包管理系统。

1 优点：

自带编译后的文件，免除用户对软件编译的过程
可以自动检测文件系统(硬盘)的容量、系统的版本。避免软件被错误的安装。
自带软件的版本信息、帮助文档、用途说明等信息。

2 缺点：

无论安装还是卸载，RPM 都有一个恶心人的依赖关系。
安装的软件需要依赖，那么优先安装依赖。

卸载的软件存在依赖，那么优先卸载依赖。

3 默认路径：

/etc	一些配置文件放置的目录,例如/etc/crontab
/usr/bin	一些可执行文件
/usr/lib	一些程序使用的动态链接库
/usr/share/doc	一些基本的软件使用手册与说明文件
/usr/share/man	一些 man page（Linux 命令的随机帮助说明）文件

4 安装：

语法：rpm -ivh packageName.rpm

选项：

i	表示安装
v	表示处理过程
h	显示处理进度(进度条)

案例：

软件包在资料中提供：

X:\课件 V4.0 提供的资料\rpm\

安装软件：

单个安装：

```
# bash
```

```
rpm -ivh pack1.rpm
```

多个安装：

```
# bash
```

```
rpm -ivh pack1.rpm pack2.rpm *.rpm
```

安装网络上的 RPM 包

```
# bash
```

```
rpm -ivh "https://网络地址/package.rpm"
```

5 查询：

rpm -[选项]

选项：

-q :	仅查询,后面接的软件名称是否有安装
-qa :	列出所有的,已经安装在本机 Linux 系统上面的所有软件名称 !!!
-ql :	列出该软件所有的文件与目录所在完整文件名 !!
-qc :	列出该软件的所有配置文件 !
-qd :	列出该软件的所有说明文件
-qR :	列出和该软件有关的相依软件所含的文件

案例 1：查找是否安装 jdk

```
# rpm -qa |grep jdk
```

案例 2：查找所有系统已经安装的包，并只查看前 3 个

```
# rpm -qa |head -n 3
```

案例 3：查询 lrzsz 所包含的文件及目录

```
# rpm -ql lrzsz
```

案例 4：查看 lrzsz 包的相关说明

```
# rpm -qi lrzsz
```

列出 iptables 的配置文件

```
# rpm -qc iptables
```

案例 7：查看 apr 需要的依赖

```
# rpm -qR apr
```

6 卸载：

rpm -e package_Name # package_Name 需要通过 qa 的选项来查询出来。

--nodeps

7 RPM 升级与更新

`rpm -Uvh <package_name>` (不管有没有都安装最新版)

-Uvh 后面接的软件如果没有安装过, 系统会直接安装,若后面接的软件安装过但版本较旧,则更新至新版

```
[root@localhost soft]# rpm -Uvh jdk-8u111-linux-x64.rpm
```

Preparing...

```
##### [100%]
```

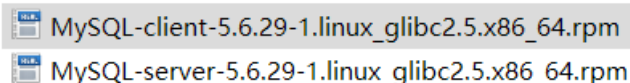
```
package jdk1.8.0_111-2000:1.8.0_111-fcs.x86_64 is already
installed
```

`rpm -Fvh <package_name>` (只有安装才更新)

-Fvh 如果后面接的软件并未安装到 Linux 系统上,则该软件不会被安装,只有已安装的软件才会被升级

六 通过 RPM 安装 mysql

1. 下载 mysql 安装包



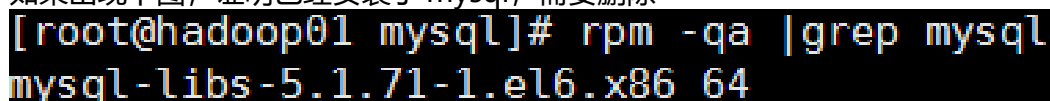
```
MySQL-client-5.6.29-1.linux_glibc2.5.x86_64.rpm
MySQL-server-5.6.29-1.linux_glibc2.5.x86_64.rpm
```

1. 确认当前虚拟机之前是否有安装过 mysql

执行: `rpm -qa` 查看 linux 安装过的所有 rpm 包

执行: `rpm -qa | grep mysql`

如果出现下图, 证明已经安装了 mysql, 需要删除



```
[root@hadoop01 mysql]# rpm -qa | grep mysql
mysql-libs-5.1.71-1.el6.x86_64
```

1. 删除 mysql

执行: `rpm -ev --nodeps mysql-libs-5.1.71-1.el6.x86_64`

此时, 再执行: `rpm -qa | grep mysql` 发现没有相关信息了

1. 新增 mysql 用户组，并创建 mysql 用户

```
groupadd mysql
```

```
useradd -r -g mysql mysql
```

1. 安装 mysql server rpm 包和 client 包，执行：

```
rpm -ivh MySQL-server-5.6.29-1.linux_glibc2.5.x86_64.rpm
```

```
rpm -ivh MySQL-client-5.6.29-1.linux_glibc2.5.x86_64.rpm
```

1. 安装后，mysql 文件所在的目录

Directory	Contents of Directory
/usr/bin	Client programs and scripts
/usr/sbin	The mysqld server
/var/lib/mysql	Log files, databases
/usr/share/info	MySQL manual in Info format
/usr/share/man	Unix manual pages
/usr/include/mysql	Include (header) files
/usr/lib/mysql	Libraries
/usr/share/mysql	Miscellaneous support files, including error messages, character set files, sample configuration files, SQL for database installation
/usr/share/sql-bench	Benchmarks

1. 修改 my.cnf,默认在/usr/my.cnf，执行：vim /usr/my.cnf，添加如下内容：

```
[client]
```

```
default-character-set=utf8
```

```
[mysql]
```

```
default-character-set=utf8
```

```
[mysqld]
```

```
character_set_server=utf8
```

1. 将 mysqld 加入系统服务，并随机启动

执行：cp /usr/share/mysql/mysql.server /etc/init.d/mysqld

说明：/etc/init.d 是 linux 的一个特殊目录，放在这个目录的命令会随 linux 开机而启动。

1. 启动 mysqld，执行：service mysqld start

```
[root@hadoop01 mysql]# cp /usr/share/mysql/mysql.server /etc/init.d/mysqld
[root@hadoop01 mysql]# service mysqld start
Starting MySQL.... SUCCESS!
```

1. 查看初始生成的密码，执行：vim /root/.mysql_secret。这个密码随机生成的

```
# The random password set for
: UCgUjqWmTcBNctCJ
```

1. 修改初始密码

第一次安装完 mysql 后，需要指定登录密码

执行：mysqladmin -u root -p password root 此时，提示要输入初始生成的密码，拷贝过来即可

1. 进入 mysql 数据库

执行：mysql -u root -p

输入：root 进入

执行：\s 查看 mysql 数据配置信息

13. 如果安装失败按以下操作重新安装

1-rpm -qa |grep -i mysql 通过 rpm -e 删除

2-find / -name mysql 查到所有的 mysql 相关的目录和文件全部删掉。

3-将/usr 下面的 my.cnf 这个文件删掉。

然后按笔记一步一步重新装。

七 软件管理(yum)

yum 的由来，是因为 rpm 的缺点所导致，因为 rpm 无论安装还是卸载都需要解决依赖关系，并且比较繁琐，所以诞生 yum 的技术。

yum 通过分析 rpm 的信息来进行软件的安装、升级、卸载。

优点:	可以一键解决 rpm 的依赖关系。
缺点:	yum 的所有执行操作全都都需要 repo 文件(YUM 源)。 使用 yum 安装软件，中招几率高达 90%。

所有的 yum 源都存放在/etc/yum.repos.d/目录下。

工作环境中，一般都会屏蔽系统自带的 yum 源，而选择权威机构的 yum 源。

1 yum 的查询:

search	查询某个软件名称或者是描述的关键字
list	列出目前 yum 所管理的所有的软件名称与版本,有点类似 rpm -qa

2 yum 的安装:

```
yum install package_Name -y
```

案例:

```
# bash
```

```
yum install lrzsz
```

期间会提示 y/N

输入 y 即可。

3 yum 的卸载:

```
yum remove package_Name
```

案例:

```
# bash
```

```
yum remove lrzsz
```

4 yum 的更新:

`yum update package_Name`

yum 安装、卸载、更新的过程中出现的 y/N, 可以通过在命令的结尾出 -y, 表示全部过执行 yes 操作。

yum 客户端运行机制

客户端每次使用 yum 调用 install 或者 search 的时候, 都会去解析/etc/yum.repos.d/下面所有以.repo 结尾的文件, 这些配置文件指定了 yum 服务器的地址。

yum 需要定期去 “更新” yum 服务器上的 rpm “清单”, 然后把 “清单” 下载保存到 yum 自己的 cache 里面, 根据/etc/yum.conf 里配置(默认是在/var/cache/yum/\$basearch/\$releasever 下、即/var/cache/yum/x86_64/6), 每次调用 yum 安装包的时候都会去这个 cache 目录下去找 “清单”, 根据 “清单” 里的 rpm 包描述从而来确定安装包的名字, 版本号, 所需要的依赖包等, 如果 rpm 包的 cache 不存在, 就去 yum 服务器下载 rpm 包安装。

3.清理 yum 缓存, 并生成新的缓存

`yum clean all`

`yum makecache`

加入 hadoop 组件相关 yum 源

1, 查看当前系统中 yum 支持的所有软件包中是否存在 hadoop

`[root@tedu yum.repos.d]# yum list|grep hadoop` #发现没有

2, 如果想要当前系统的 yum 支持 hadoop 软件包, 需要本地/etc/yum.repos.d 下创建 cloudera-cdh5.repo 文件

http://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh

[centos6 系列更换阿里 yum 源](#)

1.首先备份原来的 cent os 官方 yum 源

```
cp /etc/yum.repos.d/CentOS-Base.repo /etc/yum.repos.d/CentOS-Base.repo.bak
```

2.获取阿里的 yum 源覆盖本地官方 yum 源

```
wget -O /etc/yum.repos.d/CentOS-Base.repo  
http://mirrors.aliyun.com/repo/Centos-6.repo
```

八 RPM 和 YUM 的取舍

如果安装、卸载、更新的软件是单个独立的离线安装包，那么建议使用 RPM 的方式进行安装、卸载、更新。

如果安装一个软件时，发现此软件有众多的依赖环境，那么首选就是 yum 的方式进行处理。

例如：

安装一个 jdk，那么首选 rpm 的方式。

安装 tomcat 的话就可以考虑使用 yum

九 命令执行判断

命令执行判断

\$?:命令回传值

命令回传值\$?的两种用法：与 && 或 ||

&&:

cmd1 && cmd2 若 cmd1 运行完毕且正确运行 (\$? =0) ，则开始运行 cmd2；若 cmd1 运行完毕且为错误 (\$? !=0) ，则 cmd2 不运行；

|| :

cmd1 || cmd2 若 cmd1 进行完毕且正确运行 (\$?=0) , 则 cmd2 不运行; 若 cmd1 运行完毕且为错误 (\$?!=0) ,则开始运行 cmd2;

不管与还是或, 运行正确回传值均为 0,不同的是与的时候运行 cmd2,而或的时候不运行 cmd2; 若运行错误, 则回传值均为非 0,但与的时候不运行 cmd2,而或的时候运行 cmd2。。

举例:

如果/tmp/test 存在, 则创建/tmp/test/demo

```
# bash
```

```
mkdir -p /tmp/test && touch /tmp/test/demo
```

如果/tmp/test1 不存在, 则删除/tmp/test/demo

```
# bash
```

```
cd /tmp/test1 || rm -rf /tmp/test/demo
```

十 数据重定向

- [数据重定向](#)
- 数据重定向就是将某个命令执行后应该要出现在屏幕上的数据, 给他传输到其他地方,

- 通常执行一条命令的时候会有标准输出和标准错误输出

- 标准输出是指命令执行之后, 传回正确信息的输出目标

```
[root@localhost ~]#ll /media
```

```
total0
```

- 标准错误输出是命令执行失败后,所传回错误信息的输出目标

```
[root@localhost~]#ll m
```

```
ls:can not access m :No such file or directory
```

标准输入 (stdin) :编号为 0 使用<或<<

标准输出 (stdout) :编号为 1 使用>或>>

标准错误输出 (stderr) :编号为 2 使用>或>>

1>:以覆盖的方法,将正确的数据输出到文件;

1>>:以累加的方法,将正确的数据输出到文件;

2>:以覆盖的方法,将错误输出的数据输出到文件;

2>>:以累加的方法,将错误输出的数据输出到文件;

案例:

某一条命令执行后会有标准输出和标准错误输出, 将标准输出的内容输出到文件中。

```
# bash
```

```
ll /root /roo 1>fileName
```

某一条命令执行后会有标准输出和标准错误输出, 将标准错误输出的内容输出到文件中。

```
# bash
```

```
ll /root /roo 2> fileName
```

某一条命令执行后会有标准输出和标准错误输出, 将标准输出和标准错误输出的内容输出到文件中。

```
# bash
```

```
ll /root /roo > fileName 2>&1
```

还是上面的案例, 只不过要求结果文件不保存

```
# bash
```

```
ll /root /roo > /dev/null 2>&1
```

>>用法同上

标准输入案例

打印文本中的行数

```
wc -l < 文本
```

并且可以将打印出来的重定向到新的文件中

```
wc -l< 文本 > count
```

利用标准输入编写文件

```
cat >> demo.txt<< "abc"
```

```
cat >> 123.txt << abc
```

```
cat > 123.txt << abc
```