

# 第二章 UNIX的软件工具

- 2.1 Unix软件开发工具简介
- 2.2 BACKUS系统
- 2.3 vi
- 2.4 sed
- 2.5 awk
- 2.6 make
- 2.7 SVN

## ➤2.1 Unix软件开发工具简介

### ➤文本编辑工具(text editor, 编辑器, 编辑程序)

软件工具分类	适合于字符终端	适合于图形终端	功能特点
行文本编辑	ed, ex		面向行
交互文本编辑	vi, vim, emacs	xeditor 等	面向字符
字符流编辑	sed, awk		面向整个文件
部分编辑功能	tr, dd, sort, uniq 等		非编辑工具, 具有可利用的编辑功能

## ➤ 2.1 Unix软件开发工具简介

### ➤ 文本格式化工具(document generation)

- 撰写学术论文(paper and thesis)

tex, latex, postscript (ps), pdf

### ➤ 软件维护工具(software maintenance)

- 软件编译和生成，或工程组织等过程的自动辅助工具

make

- 版本管理工具(version controller)

SCCS, CVS, SVN等

## ➤ 2.1 Unix软件开发工具简介

### ➤ 软件调试查错工具(debugging tools)

功能：设置断点；跟踪运行；单步或多步；监视数据变化；设定循环次数；条件运行等

- 适合于字符终端

**adb, sdb, gdb**

- 适合于图形终端

**dbx, dbxtool, wdb, ddd, ...**

### ➤ 编译器生成工具(compiler generation)

- 词法编译器生成工具(lexical generation)

**lex**

- 语法编译器生成工具(yet another compiler compiler)

**yacc**

# 第二章 UNIX的软件工具

➤2.1 Unix软件开发工具简介

➤2.1 BACKUS系统

➤2.3 vi

➤2.3.1 vi概述

➤2.3.2 vi的工作状态/模式

➤2.3.3 编辑对象及定位

➤2.3.4 编辑类命令

➤2.3.5 全局操作命令

➤2.3.6 块操作命令

➤2.3.7 杂类命令

➤2.3.8 vi命令综合示例

## ➤ 2.3.1 vi概述

### ➤ 文本编辑器与vi

由比尔·乔伊(Bill Joy)始创的vi(Visual display editor, 全屏幕编辑器)是UNIX中最常用的全屏幕文本编辑器。

vim(作者Bram Moolenaar布莱姆·莫利纳罗)和emacs(作者Richard Stallman, 理查·斯托曼)。



Bill Joy



Bram Moolenaar



Richard Stallman<sup>6</sup>

## ➤ 2.3.1 vi概述

### ➤ 启动vi的方式

`vi [file [...]]`

例如, `vi`

例如, `vi a.txt`

例如, `vi a.txt b.txt`

对0到多个文件`file`进行编辑

编辑一个无名文件(空文件)

编辑一个文件`a.txt`

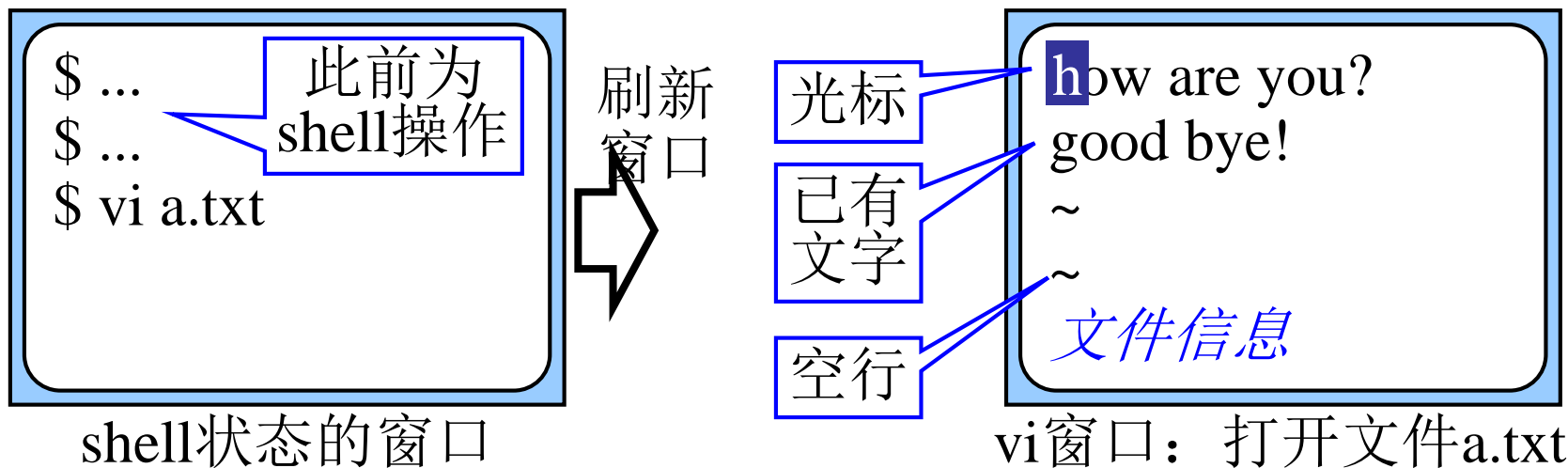
编辑两个文件`a.txt`和`b.txt`

### ➤ 编辑已有的文件

如果`a.txt`是非空文件, 在shell状态下执行命令:

`$ vi a.txt`

将启动vi, 刷新shell状态的窗口, 转为显示vi窗口。



## ➤ 2.3.1 vi概述

### ➤ 编辑一个空文件

如果在shell状态下执行命令：

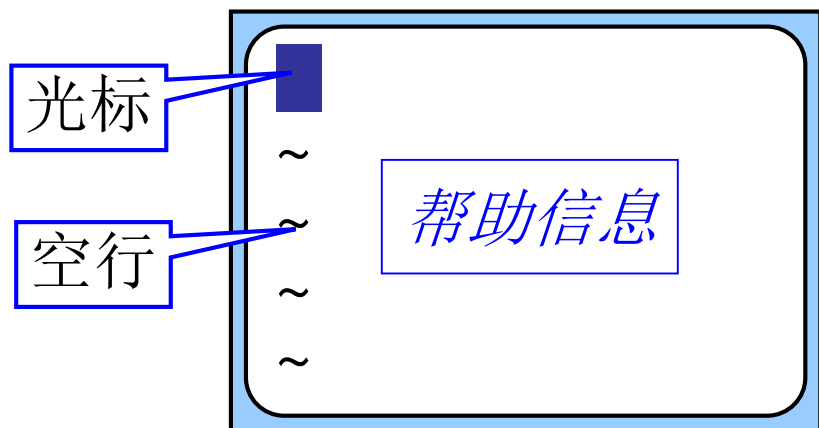
`$ vi`

编辑无名文件

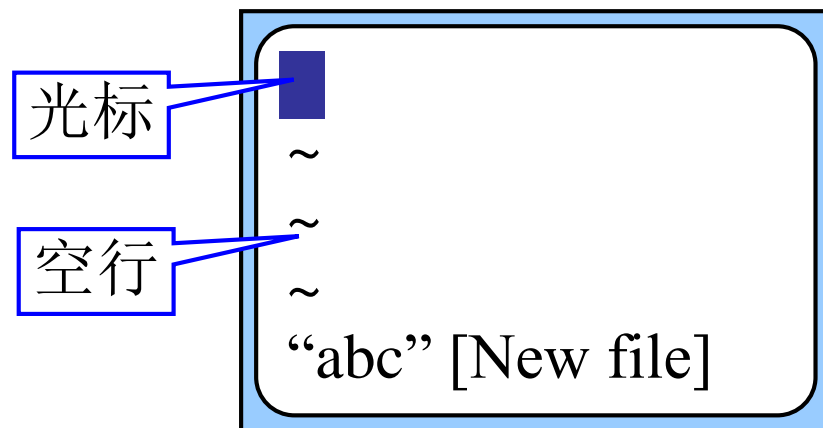
或者 `$ vi abc`

编辑新文件abc

shell状态的窗口将刷新为vi的空白窗口，左侧一列显示~，表示空行。



vi窗口：打开无名文件



vi窗口：打开新文件abc



## ➤ 2.3.2 vi的工作状态/模式

所有的ASCII字符都是vi的命令，也就是说，至少有100多条命令，包括字母数字、标点符号、以及各种控制字符，而且大写字母与小写字母分别代表不同的命令。**因此在vi中不要随意按键。**

排序: (例: 1-5行排序: 光标放在1行上, 然后!5G, 然后sort)  
 字体设定: (set guifont=Raize\ Bold\ 13)  
 键绑定: (如: map h <Insert>; map <F2> h{ } 按下F2后进入Insert模式并打印{ })

Esc 命令模式																							
~ 转换大小写	! 外部过滤器	@ 执行寄存器	# 反向查找	\$ 行末	% 括号匹配	^ 行首	& :s//~	*	( 句首	) 下一句首	_ 前一行首	+ 次行首	= 自动格式化										
1	2	3	4	5	6	7	8	9	0	- 前一行首	= 自动格式化												
Q 切换至ex模式	W 下一单词	E 单词尾	R 替换模式	T 替换字符	Y 复制行	U 撤销	I 到行首插入	O 分段(前)	P 粘贴(前)	{ 段首	}	段尾											
q	w 下一单词	e 单词尾	r 替换字符	t 替换字符	y 复制	u 撤销	i 插入模式	o 分段(后)	p 粘贴(后)	[ 段首	]	段尾											
A 在行末附加	S 删除行并插入	D 删除至行末	F 行内字符反向查找	G 文件尾/行号	H 画面顶	J 合并两行	K 画面底	L 画面底	:	命令	"	跳转到位置标记	行首/列										
a 附加	s 删除字符并插入	d 删除	f 行内字符查找	g 附加命令	h ←	j ↓	k ↑	l →	;	重复	'	跳转到位置标记	\										
Z 退出	X 退格	C 修改至行末	V visual行模式	B 前一单词	N 查找上一除	M 设定标记	< 反缩进	> 缩进	:	命令	"	跳转到位置标记	行首/列										
Z 附加命令	X 删除字符	C 修改	V visual模式	b 前一单词	n 查找下一除	m 设定标记	< 反缩进	> 缩进	:	命令	"	跳转到位置标记	行首/列										

**动作** 移动光标或欲定义操作的范围

**指令** 直接执行的指令  
红色指令进入编辑模式

**操作** 后接用以表示操作范围的指令

**extra** 特殊功能需额外输入

主要ex指令: :w (保存)  
:q (退出)  
:e (打开文件) <Tab>  
:%s/x/y/g (以y全文替换x)  
:h (帮助)

其它重要指令:  
Ctrl-R: 重做  
Ctrl-F/-B: 向前(下)翻页/向后(上)翻页  
Ctrl-E/-Y: 向前(下)一列/向后(上)一列  
Ctrl-V: 切换visual模式

(1) 在复制/粘贴/删除 指令前使用 "x (x=a...z) 使用指令的寄存器(如: "ay\$ 复制该行目前位置到行尾的内容到寄存器'a')  
(2) 命令前添加数字 重复指定次数的操作 (如: 2p, d2w, 5i, d4j)  
(3) 重复光标所在字符处指定的查找 (dd = 删除本行, >>> 行首缩进)  
(4) ZZ 保存并退出  
(5) zt: 移动游标所在行至画面顶端, zb: 底端, zz: 中央  
(6) gg: 文件首; ddp/P: 交换上下两行 daw: 删除单词; 全选: ggVG

w,e,b指令

b(小写): quux(**foo**, **bar**, **baz**);

B(大写): quux(**foo**, **bar**, **baz**);

## ➤ 2.3.2 vi的工作状态/模式

### ➤ vi的工作状态及工作模式

vi有三种工作状态，vi状态、ex状态以及open状态。

open状态不常用，因此不予介绍。

vi状态有两种工作模式，vi状态的命令模式和vi状态的插入模式，简称命令模式和插入模式（[上机指南二](#)）。

#### ➤ 命令模式

启动vi后总是先处于命令模式。在命令模式中，没有待命符，不能直接输入文件的内容。键入的字符均视为命令，不显示，而只是执行。

可以从命令模式转换为插入模式或者ex状态。

#### ➤ 插入模式

在命令模式中使用插入类命令，将转入插入模式，随后键入的字符(文本)才是文件的内容(以下用[下划线](#)表示)。插入文本完毕，必须按<ESC>回到命令模式。

## ➤ 2.3.2 vi的工作状态/模式

### ➤ vi的工作状态及工作模式

➤ **ex**状态可称为**ex**模式或者底行模式。

1. 在命令模式中按Q键，可从vi状态(的命令模式)转到ex状态。  
ex状态的特征是在vi窗口的底行显示待命符“:”。

执行ex命令的完整形式为：

: *ex 命令* <CR>

以回车键表示结束

在ex状态中键入

: vi <CR>

“:”是ex的待命符

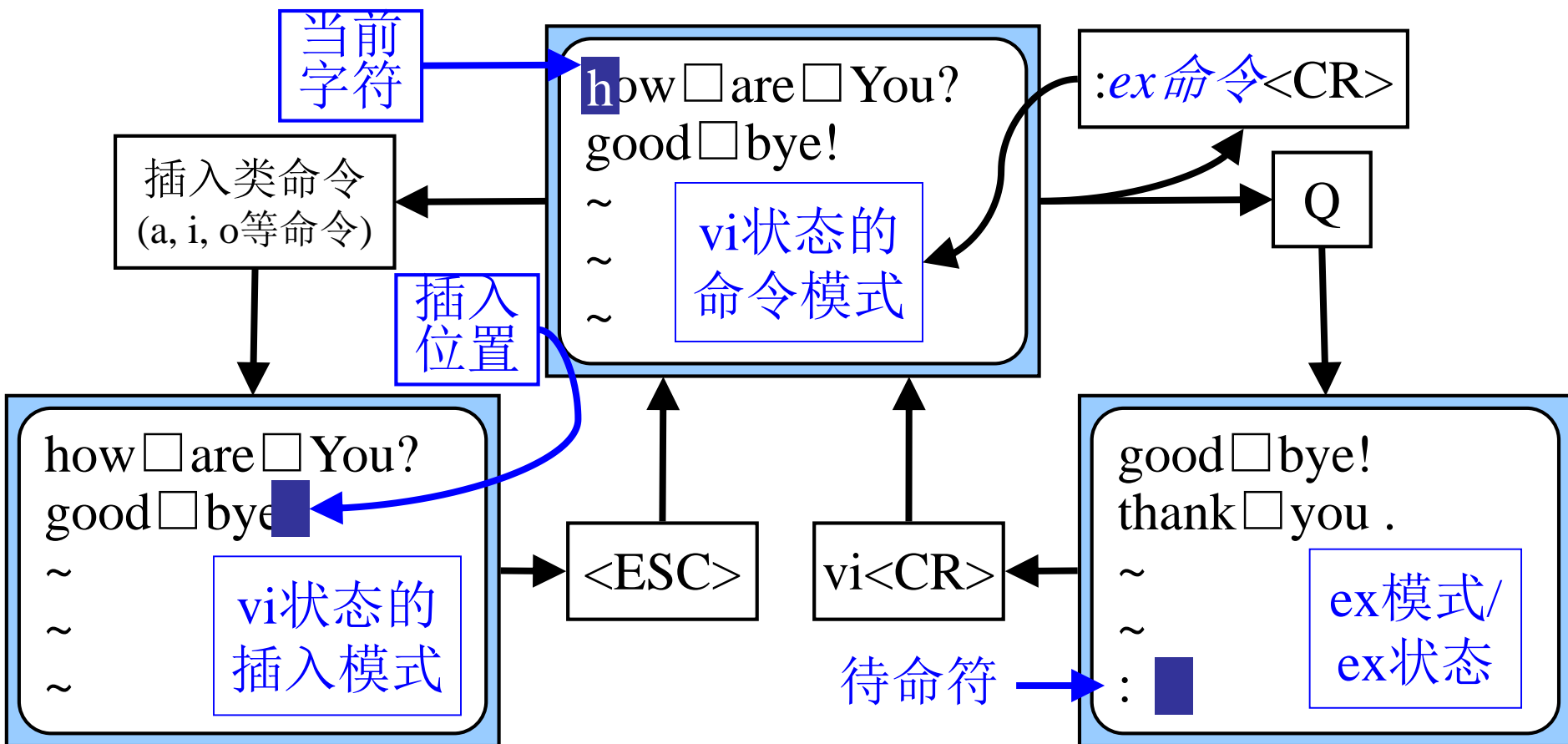
将从ex状态转到vi状态(的命令模式)。

2. 可以在vi状态(的命令模式)中按“: *ex 命令* <CR>”，执行一个ex命令。

可见，vi状态和ex状态可以相互转换，但仅限于命令模式与ex状态之间的相互转换。

## ➤ 2.3.2 vi的工作状态/模式

### ➤ vi工作状态/模式的转换



## ➤2.3.2 vi的工作状态/模式

### ➤写文件命令

<code>:w&lt;CR&gt;</code>	将当前文件存盘
<code>:w□<i>file</i>&lt;CR&gt;</code>	将当前文件另存为 <i>file</i>
<code>:w !□<i>file</i>&lt;CR&gt;</code>	另存为一个已存在的 <i>file</i> (强制覆盖 <i>file</i> )

### ➤退出vi的命令

<code>:q&lt;CR&gt;</code>	已存盘或者没有修改文件时退出vi
<code>:q !&lt;CR&gt;</code>	放弃所作的修改，退出vi
<code>:wq&lt;CR&gt;</code>	存盘并且退出vi
<code>ZZ</code>	存盘并且退出vi(不需要回车)

### ➤读文件命令

<code>:e□<i>file</i>&lt;CR&gt;</code>	读入文件 <i>file</i> ，作为当前的编辑文件。
<code>:n&lt;CR&gt;</code>	如果使用命令“vi file1 file2 ...”启动vi，表示编辑多个文本文件，n表示编辑下一文件(next file)。

## ➤2.3.3 编辑对象及定位

### ➤vi编辑的对象

#### ➤ 字符(由光标位置确定)

当前字符(光标所在位置)，前一字符，后一字符，文件首，文件尾，行首，行尾

#### ➤ 词word

由字母、数字和下划线组成，或者由标点符号组成。

#### ➤ 词WORD

由非空字符组成，空字符作为WORD的分隔符。

例如： 2004_Mar_1	表示1个word，或者1个WORD
2004-Mar-1	表示5个word，或者1个WORD
%d%d,n,m	表示8个word，或者1个WORD
%d+-%d12	表示4个word，或者1个WORD

## ➤2.3.3 编辑对象及定位

### ➤定位命令

#### ➤按字符移动光标

**h**或者←键： 向左移动一格

**l**或者→键： 向右移动一格

**k**或者↑键： 移到上一行同一格，若上一行短，则移到行尾

**j**或者↓键： 移到下一行同一格，若下一行短，则移到行尾

#### ➤按词(word/WORD)移动光标

**w**: 向后移到下一个word首      **W**: 向后移到下一个WORD首


**b**: 向前移到前一个word首      **B**: 向前移到前一个WORD首

**e**: 向后移到下一个word尾      **E**: 向后移到下一个WORD尾

## ➤ 2.3.3 编辑对象及定位

### ➤ 定位命令

#### ➤ 行内移动光标

- 0: 移到本行第一个非空字符
-  ^: 移到本行第一个字符(包括空字符)
- \$: 移到本行行尾(最后一个字符)

#### ➤ 跨行移动光标

<CR>: 移到下一行首

^F: 向下翻页

G: 移到最后一行行首

^B: 向上翻页

1G: 移到第一行行首



## ➤2.3.4 编辑类命令

### ➤插入类命令

#### ➤ 插入类命令的用法

<插入类命令>插入内容<ESC>

<插入类命令>和<ESC>是不显示的，插入内容是用户键入的。

#### ➤ 插入类命令

i 在当前字符前插入

I 在行首插入

a 在当前字符后插入

A 在行尾插入

o 在当前行以下插入

O 在当前行以上插入

## ➤ 【例2-5】插入类命令示例

插入前	插入命令及插入内容	插入后
<b>h</b> ow are you Bye	<u>i</u> <u>Good⊕thank!</u> <ESC> i命令： 在当前字符前插入	Good thank!how are you Bye
<b>h</b> ow are you Bye	<u>i</u> <u>Good⊕thank!⊕</u> <ESC> i命令： 在当前字符前插入	Good thank! how are you Bye
<b>h</b> ow are you Bye	<u>a</u> <u>Good⊕thank!</u> <ESC> a命令： 在当前字符后插入	hGood thank!ow are you Bye
<b>h</b> ow are you Bye	<u>o</u> <u>Good⊕thank!</u> <ESC> o命令： 在当前行以下插入	how are you Good thank! Bye
<b>h</b> ow are you Bye	<u>O</u> <u>Good⊕thank!</u> <ESC> O命令： 在当前行以上插入	Good thank! how are you Bye

## ➤ 2.3.4 编辑类命令

### ➤ 替代类命令

替代命令s, cw, cW, c\$, C和cc均以<ESC>为结束符。

**rx** 将当前字符替换为**x**，r命令不需要<ESC>

**[n]s** 将**n**个字符替换为插入内容，**n**缺省为1

**cw 或 cW** 将当前字符到word/WORD尾替换为插入内容

**c\$或C** 将当前字符到行尾的内容替换为插入内容

**cc** 将当前行替换为插入内容

## 【例2-6】 替代命令示例

替换前	替换命令及作用	替换后
Good Day May-1 OK?	<u>rr</u> 或者 <u>sr</u> <ESC> 将当前字符y改为r	Good Day Mar-1 OK?
Good Day May-1 OK?	cW <u>r-20</u> <ESC> 将y-1改为r-20	Good Day Mar-20 OK?
Good Day May-1 OK?	c\$ <u>r-20 right?</u> <ESC> 或者 <u>Cr-20 right?</u> <ESC> 将y-1 OK?改为r-20 right?	Good Day Mar-20-right?
Good Day May-1 OK?	cc <u>Buy!&lt;CR&gt;Do it!</u> <ESC> 将当前行改为两行	Good Day Buy! Do it!

## ➤ 2.3.4 编辑类命令

### ➤ 删除类命令

#### ➤ 删除字符的命令

`[n]``x`            删除`n`个字符，`n`缺省为1

`d[n]``w`            删除`n`个word（包括其后的空格），`n`缺省为1

`d[n]``W`            删除`n`个WORD（包括其后的空格），`n`缺省为1

`d``$`                删除到行尾

`dd`                删除当前行

`[n]``dd`            删除包括当前行在内的以下`n`行，`n`缺省为1

`:[i [j]]``d``<CR>`    删除第`i`行至第`j`行

## 【例2-7】删除字符命令示例

删除前	删除命令及作用	删除后
Good Day is 2005-May <sup>y</sup> -1 OK?	x 删除当前字符 <sup>y</sup>	Good Day is 2005-Ma <sup>1</sup> -1 OK?
Good Day is 2005-May <sup>y</sup> -1 OK?	5x 删除5个字符“y-1 O”	Good Day is 2005-Ma <sup>K</sup> ?
Good Day is 2005-May <sup>y</sup> -1 OK?	d2w 删除2个word“y-”	Good Day is 2005-Ma <sup>1</sup> OK?
Good Day is 2005-May <sup>y</sup> -1 OK?	dW 删除WORD“y-1 ”	Good Day is 2005-Ma <sup>O</sup> K?
Good Day is 2005-May <sup>y</sup> -1 OK?	d\$ 删除到行尾	Good Day is 2005-Ma <sup>a</sup>

## 【例2-8】 行删除命令示例

删除前	删除命令及作用	删除后
how are you Good thank! Bye	dd 删除第2行(当前行)	how are you thank! Bye
how are you Good thank! Bye	3dd 向下删除3行	how are you
how are you Good thank! Bye	:2,3d<CR> 删除第2行和第3行	how are you Bye

## ➤2.3.5 全局操作命令

### ➤搜索命令

**/[*str*][/]<CR>** 从当前字符起向下搜索匹配*str*的字符串。  
向下搜索遇到文件尾时将自动接着从文件首继续，直到当前字符。

**?[*str*][?]<CR>** 从当前字符起向上搜索匹配*str*的字符串。  
向上搜索遇到文件首时将自动接着从文件尾继续，直到当前字符。

以上命令中的第二个斜杠或问号可以省略。

若不是第一次搜索，可省略*str*。

**N** 反向搜索*str*。等价于?**<CR>**或者/**<CR>**。

**n** 按前一次搜索命令的搜索方向再一次搜索*str*



## ➤2.3.5 全局操作命令

### ➤正则表达式

以上匹配`str`的字符串可以有多种表述方式，称为正则表达式：

· 除了换行符，可以替代任何一个字符的通配符。

`[xyz]` 方括号中用`xyz`的枚举方式表述的一个字符的通配符。方括号是必需的符号。枚举方式可以采用`a-b`的形式，表示按照ASCII的顺序从字符`a`到`b`之间的任何一个字符，例如，`[0-9]`表示一个数字，`[0-9_]`表示一个数字或者下划线。

`x*` 字符`x`的0到多次重复，`x`代表用以上任何一种表述的一个字符。例如`a*`可以匹配`aaa`和`a`等。

`^` 不表示一个实际的字符，仅代表行首。  
例如，`^Lead`表示匹配位于行首的字符串`Lead`。

`$` 不表示一个实际的字符，仅代表行尾。  
例如，`Tail$`表示匹配位于行尾的字符串`Tail`。  
`^$`表示空行，表示在行首和行尾之间没有字符。

## ➤ 2.3.5 全局操作命令

### ➤ 正则表达式

`\c`            如果 `c` 是特殊字符(如 `.` 和 `*` 等), 需用反斜杠 `\` 转义。  
例如, 匹配特殊字符 `.` (dot), `\`, `/` 和 `*` 时应该表述为  
`\.`, `\/`, `\\` 和 `\*`。

## ➤2.3.5 全局操作命令

### ➤d命令

d命令即行删除命令(delete line)，基本格式为：

:*[g[!]]*[/*pattern*/]d<CR>

### ➤s命令

s命令即行替换命令(line substitute)，基本格式为：

:*[[g]/pattern/]*s/*[str]/new\_str/[cmd]*<CR>

## ➤2.3.5 全局操作命令

### ➤d命令

d命令即行删除命令(delete line)，基本格式为：

`:[g[!]][/pattern/]d<CR>`

d命令的格式

`:/pattern/d<CR>`      删除第一个匹配*pattern*的行

`:g/pattern/d<CR>`      删除所有匹配*pattern*的行

`:g!/pattern/d<CR>`      删除所有不匹配*pattern*的行

*/pattern/*用以描述需要删除的范围(又称行地址address)，一般格式为：

*/pattern1/*      匹配*pattern1*的行

或 *addr1*

*/pattern1/, /pattern2/*      从匹配*pattern1*到匹配*pattern2*之间的行

或 *addr1, addr2*

省略*pattern*      删除当前行

## 【例2-9】d命令示例

<b><code>:/The/d&lt;CR&gt;</code></b>	删除第一个含字符串 <b>The</b> 的行
<b><code>:g/[0-9]/d&lt;CR&gt;</code></b>	删除所有含有数字的行
<b><code>:g!/a-z/d&lt;CR&gt;</code></b>	删除所有不含小写字母的行
<b><code>:/Start/,/End/d&lt;CR&gt;</code></b>	删除从含有 <b>Start</b> 到含有 <b>End</b> 的行
<b><code>:g/^#/d&lt;CR&gt;</code></b>	删除所有行首为 <b>#</b> 的行(shell的注释行)
<b><code>:g/\/\*/d&lt;CR&gt;</code></b>	删除所有含 <b>/*</b> (特殊字符)的行(C注释行)
<b><code>:g/^\$/d&lt;CR&gt;</code></b>	删除所有空行(在 <b>^</b> 和 <b>\$</b> 之间没有字符)
<b><code>:\$d&lt;CR&gt;</code></b>	删除最后一行，这里 <b>\$</b> 表示最后一行
<b><code>:/^#/, \$d&lt;CR&gt;</code></b>	从第一个shell注释行删除到最后一行

## ➤2.3.5 全局操作命令

### ➤s命令

s命令即行替换命令(line substitute)，基本格式为：

`:[g]/pattern/]s/[str]/new_str/[cmd]<CR>`

s命令的格式：

`:/pattern/s/str/new_str/<CR>`

寻找匹配*pattern*的行，将匹配*str*的字符串替换为新的字符串*new\_str*(只替换该行中的第一个*str*)。

`:g/pattern/s/str/new_str/<CR>`

寻找所有匹配*pattern*的行，将*str*替换为*new\_str*。

`:/pattern/s//new_str/<CR>`

表示*str*与*pattern*相同，即寻找*pattern*，用*new\_str*替换*pattern*。

`:s//new_str/<CR>`

表示*str*与前一次搜索的字符串相同(字符串存在搜索缓冲区中)。

`:g/s//new_str/<CR>`

寻找所有*pattern*，用*new\_str*替换*pattern*。

## ➤2.3.5 全局操作命令

### ➤辅助命令(辅助功能)

在s命令 “: [[g]/*pattern*/]s/[*str*]/*new\_str*/[*cmd*]<CR>” 中含有 *cmd*。

用 *cmd* 表示s命令的辅助功能，包括：

- g            在匹配 *pattern* 的行内替换所有匹配 *str* 的字符串
- p            显示最后一个替换的行
- gp          替换所有匹配 *str* 的字符串，并显示所有替换过的行

### 【例2-10】s命令示例

:if□[/⊕

寻找“if□[”

:g//s//if□(/g⊕

在所有行中用“if□(”替换所有的“if□[”

等价于

:g/if□[/s//if□(/g⊕

:3,50s/ABC/123/

在第3行到第50行中用123替换一个ABC

:3,50s/ABC/123/gp

在第3行到第50行中用123替换所有的ABC，并且显示最后一个被替换的行

## ➤ 2.3.6 块操作命令

### ➤ 块操作命令

#### ➤ vi块操作命令

**pick:**    `["x][n]yw`或者`["x][n]yW`      复制 $n$ 个词存入缓冲区  
(从当前位置开始, 包括其后的空格)

**pick:**    `["x][n]yy`      复制含当前行在内的 $n$ 行存入缓冲区。

**cut:**      `["x][n]dd`      剪切含当前行在内的 $n$ 行存入缓冲区。

其中, $n$ 缺省为1。字符 $x$ 表示定义有名缓冲区, 省略 $x$ 表示存取无名缓冲区。

已经介绍过的删除类命令都可以作为剪取命令。

**put:**      `["x]p`或者`["x]P`      粘贴缓冲区内容  
若块的内容以行为单位, 则向下( $p$ )或者向上( $P$ )粘贴。  
若块的内容以字符为单位, 则向右( $p$ )或者向左( $P$ )粘贴。



## 【例2-13】块操作命令示例

将第2行的you复制到第1行的问号前，操作过程为

yw/?/⊕P

how are ?  
thank□you .  
good bye!

~

~

待编辑文本

how are ?  
thank□you.  
good bye!

~

~

执行yw/?/⊕后

how are you?  
thank□you.  
good bye!

~

~

执行P命令后

将第2行移到第3行(本行与下一行交换)，操作过程为

⊕ddp

how are you?  
thank□you .  
good bye!

~

~

待编辑文本

how are you?  
good bye!  
~

~

~

执行⊕dd后

how are you?  
good bye!  
thank□you .

~

~

执行p命令后

## 【例2-14】块操作命令示例

将当前行(第3行)复制到第1行，操作过程为  
yy1GP

```
how are you?  
good bye!  
thank you .  
~  
~
```

待编辑文本

```
how are you?  
good bye!  
thank you .  
~  
~
```

执行yy1G后

```
thank you .  
how are you?  
good bye!  
thank you .  
~
```

执行P命令后

利用有名缓冲区将“thank”复制到第3行尾，操作过程为  
”ayw ⊕ ⊕ \$”ap

```
thank you .  
how are you?  
good bye!  
thank you .  
~
```

待编辑文本

```
thank you .  
how are you?  
good bye!  
thank you .  
~
```

执行”ayw ⊕ ⊕ \$”后

```
thank you .  
how are you?  
good bye!thank you .  
thank you .  
~
```

执行”ap命令后

## ➤2.3.7 杂类命令

- . 重复执行编辑类命令(插入、删除和替代命令)
- u 撤消刚才执行的编辑类命令,
- ^r 恢复刚才撤消的命令。
- ^L 控制键CTRL+L, 刷新屏幕。
- J 将下一行连到本行尾, 等价于将换行符改为空格。
- :sh<CR> 转到shell状态, 将显示shell的待命符, 以便执行shell命令, 结束时需要用exit命令返回vi。
- :!*command*<CR> 执行1个shell命令*command*, 随后返回vi。

## ➤2.3.7 杂类命令

例如，将文本中的fi替换为endif，重复执行替换的命令过程为：

/fi ⊕	搜索fi(省略了第二个斜杠/), 等价于/fi/<CR>
2s <b>endif</b> <ESC>	用endif替代两个字符fi(2s和<ESC>不显示)
n	搜索下一个fi
.	再次执行替换操作

例如，文本为：

按J键后成为：

this□line ⊕  
is□complete. ⊕

this□line is□complete. ⊕  
将换行符改为空格

## ➤ 2.3.8 vi命令综合示例

文本原始内容为

```
are how you  
jenny, Good mornjng  
fjne, Thank you
```

1) **dww**      删除are, 移到y

```
how you  
jenny, Good mornjng  
fjne, Thank you
```

3) **/j/<CR>n**      定位第2个j

```
how are you  
jenny, Good mornjng  
fjne, Thank you
```

如何将文件改成如下内容?

```
jenny, Good morning  
how are you  
fine, Thank you, and you?
```

2) **P**      粘贴are

```
how are you  
jenny, Good mornjng  
fjne, Thank you
```

4) **rin**      将j改为i, 再定位j

```
how are you  
jenny, Good morning  
fine, Thank you
```

## ➤ 2.3.8 vi命令综合示例

5) . 重复ri, 改fjne为fine

```
how are you
jenny, Good morning
fine, Thank you <==
```

6) 1G 定位到文件首

```
how are you <==
jenny, Good morning
fine, Thank you
```

7) dd 删除第1行

```
jenny, Good morning <==
fine, Thank you
```

8) p 向下粘贴删除的行

```
jenny, Good morning
how are you <==
fine, Thank you
```

9) G\$ 定位到最后一行行尾

```
how are you
jenny, Good morning
fine, Thank youu <==
```

10) a,and you?<ESC> 插入

```
jenny, Good morning
how are you
fine, Thank you, and you?
```

全部命令: dwwP/j/<CR>nrin.1GddpG\$a,and you?<ESC>

# 作业

➤ 习题: 2-1.1, 2-1.2, 2-1.3