emacs最简单入门：只要记住一个按键就行了： M-x . 按住Alt键再按x，在窗口最底下出现M-x的提示，在这里输入left-几个字符，按Tab键，出现两行提示 left-char, left-word，再输入一个c，再按Tab键，这时自动补全为left-char，按回车，如果有内容的话，当前的光标点向左移动，就像按了一个向左的方向键一样。  就这样？就这样！

就这样，发挥你的想象力，想象你现在需要做什么，把它翻译成英文单词，按 M-x，先输入几个字母，按Tab，看是否存在你想象中的指令。要取消当前操作的指令，多按几次ESC键或Ctrl-g键就取消了。

现在假设下emacs有100条指令吧，你如果想做什么事都去执行指令的话，虽然也可以，但是总比不上为常用的指令设置一个快捷键方便。现在看下默认的各种快捷键：

先来看几个简单的 forward-char  backward-char  previous-line  next-line , 很直白的命令，向前后移一个字符或向上下移一行，对应4个方向键。默认绑定的快捷键就是Ctrl加指令首字母 C-f/b/p/n。简洁直白合理极致吧。 复制一些内容到窗口上去，多按几次按键，多按几次手指上的肌肉就不会忘记了。就像用了十多年的五笔输入法一样，看着键盘时一个字也打不也来了，字根键码早忘光了。只有两只手都放在键盘上时，看着汉字，手指自然就打出来了。

再看下 forward-word  backward-word ，向前后一个单词，默认绑定快捷键为Alt加指令首字母 M-f/b。也很合理吧。

如果要快速移动到当前行首行尾，或者是一句话的首尾部呢？ move-beginning-of-line move-end-of-line  backward-sentence  forward-sentence ，默认绑定的快捷键是 C-a/e  M-a/e，好像也还算合理吧。

好了，只要理解了emacs里做任何操作都是执行LISP函数，就算是入门了。emacs被称为复杂难学是因为默认设置了很多快捷键，不容易一下记住。现在只要记住用M-x来执行命令，快捷键一律忽视，等熟悉了LISP指令及对应的功能，想为它设置一个快捷键时，再看看默认设置的快捷键是否合你的胃口，改或不改你决定。

对emacs感兴趣或者不感兴趣现在就可以决定了。什么学emacs需要很强的耐心那些都是胡扯，我敢打赌只要有好奇心的人就会继续往下看。

现在可以打开帮助菜单下的教程，跟着操作一遍，学习一些最常用的快捷键了。多动手，肌肉记忆比大脑记忆好用。

----入门教程结束----

想象一下你使用其它编辑器时使用到的各种功能快捷键，把它和emacs默认设置的快捷键比较一下，哪种设置比较合理。

emacs里的各种快捷键设置都是无害的，意思就是你改掉一个快捷键对应的功能，命令却还在，永远可以用M-x来直接执行。如果是其它编辑器就比较麻烦，可能这个操作的功能就找不回来了。

一些默认快捷键

C-代表按住Ctrl键

M-代表按住Alt键，也可以用按一下ESC放开代替

最先要记住的

M-x <cmd>      输入指令执行，在输入时用Tab可以自动补全或列出项目

C-g            取消当前操作指令

C-h k <key>    查看当前按键绑定的指令及介绍

移动插入点（光标）

C-n/p/b/f      到 下一行、上一行、前一字符、后一字符

M-b/f          到 前、后一单词

C-a/e          到 行首、末

M-a/e          到 句首、末

M-         到 文件首、末

C-v/M-v        下、上翻屏

C-l        循环 将当前光标行显示在窗口下、中、上位置

C-u <num> <xx> 重复执行之后的 xx 命令 num 次，不输入num默认为4

C-u C-v        向下滚动4行

编辑

Back/C-d       删除 前、后字符

M-Back/M-d     移除 前、后单词

C-k/M-k        移除到 行末、句末

C-/            undo/redo

C-x u          undo/redo

C-w            剪切

C-y        召回（粘贴）（粘贴移除的、复制、剪切的内容）

C-c        复制（emacs一般也默认定制了与OS的剪切复制粘贴相同的快捷键）

查找

C-s

  C-s 查找下一个

  C-r 查找上一下

  C-g 一次返回，二次结束查找

  回车    停止到当前查找到的内容

替换

M-%

标记

M-@

文件操作

C-x C-c        退出emacs

C-x C-f        打开文件

C-x C-s        保存文件

C-x C-v        在当前缓冲区重新打开一个文件，可以用于清除undo历史!!

C-x 1          最大化当前缓冲，关闭其它

C-x 2/3        垂直、水平创建新缓冲区

C-x o          切换到其它缓冲区

C-x C-b        列出缓冲区

C-x b          提示输入缓冲区名称，切换当前窗口的缓冲区

C-x k          删除当前缓冲区

C-j        换行回车，有些模式下比较直接回车好用，有增加缩进之类的功能

------

理解一些概念

buffer

  缓冲区，内存中的数据，打开文件时读入内容到内存，修改后保存才会修改到磁盘上的文件。

window

  窗口，缓冲区的一个展示区域

frame

  OS上的一个emacs程序窗口

mode 模式

  mode有主模式，辅助模式，主模式就像vim中设置当前filetype一样，当前编辑的是ruby或者python源代码，根据不同文件类型，设置按Tab缩进时缩进不同的字符数，把不同的词当作关键字时行语法高亮等等，同一时间只能选一个主模式。试着多开几个窗口，执行text-mode ruby-mode python-mode试下。emacs打开文件时会自动根据文件扩展名或内容检查并设置主模式。

  辅助模式就是窗口是否要自动换行，是否显示行号等，辅助模式可以多个同时启用。

undo

  在 EMACS 中想要将已经做过的动作放弃，以恢复原状。

  EMACS 所提供的 undo，可以连续恢复最近使用过的指令。

  undo 的顺序是最新使用过的指令最先被 undo ，第二次使用 undo 则恢复第二新的指令，任何指令的输入（除了 undo 本身之外） 都会使指令输入的顺序重整， 这也同时影响 undo 的顺序。

  使用 undo 有一个限制，就是 undo 只能 undo 对缓冲区内容造成改变的指令。对于只是改变游标动作的指令，是无法以 undo 来恢复旧观。若所有修改过内容的指令，都以 undo 恢复原状后， 再一次使用 undo 的指令，echo area 会出现如下的讯息：No further undo information.

当使用了 undo 之后，还想要在 redo 这个已被 undo 的动作时， 有一个技巧可以达成如此的效果。

 1.首先键入一个不会改变缓冲区内容的指令（如光标移动的指令），使原来

 2.存放指令的顺序因新指令的加入而改变。

 3.再使用一次 undo 的指令，就可以达到 redo 的效果了。

undo的行为很难描述，示例：

 新建一个文件

 执行几个修改内容的指令 <1回车2回车3回车>， 这时undo历史里有6条记录

 undo顺序为654321, 继续undo则提示No further undo information

 按下左右方向键（虽然移不动光标），这时undo历史里有12条记录了，前6条undo就变成redo了，顺序123456654321

 再按下方向键，undo历史记录数量再翻倍

emacs的undo不会丢失任何操作，你先做一些操作，undo，再做另一些操作，如果是其它编辑器，第一次undo的动作就失踪了，emacs里却一直存在着，原因同上。多试几次就明白了。

----完----