gvim

1. windows gvim

1.1 _vimrc

```
set nocompatible
source $VIMRUNTIME/vimrc_example.vim
"source $VIMRUNTIME/mswin.vim
"behave mswin
set encoding=utf-8

set tabstop=4
set shiftwidth=4
set expandtab
set autoindent
```

1.2 菜单语言切换为英文

在_vimrc文件末尾加入以下内容

```
set langmenu=en_US
let $LANG= 'en_US'
source $VIMRUNTIME/delmenu.vim
source $VIMRUNTIME/menu.vim
```

1.3 gvim支持中文显示

```
set fileencodings=utf-8,gbk,gb2312,gb18030
" set termencoding=utf-8
"set fileformats=unix
"set encoding=utf-8
```

1.4 保持linux操作方式，比如支持Ctrl-V 选择

注释以下行

```
"source $VIMRUNTIME/mswin.vim
```

1.5 tab宽度及其它

```
set expandtab
```

```
set shiftwidth=4
set tabstop=4
set noundofile
set cursorline
set cursorcolumn
set nobackup
```

2. 用法技巧等

2.1 tab转空格

```
:retab
```

2.2 显示match匹配行数

```
:%s/xxx//gn

[g] 替换行中出现所有匹配，否则只替换第一个。
[n] 意思是显示匹配数，但是不做实际的替换操作; 如果有g选项的话，则计算所有的匹配数。
[i] 匹配时忽略大小写, set ignorecase在这不起作用
[I] 匹配时不忽略大小写， , set ignorecase在这不起作用
```

2.3 输入时不自动换行

```
set textwidth=0
```

默认情况下，其width值为78。设置为0就相当于关闭这个功能，设置成正值就可以让 vim 在指定长度上折行。

2.4 gvim启动时窗口默认最大化

linux环境，在.vimrc文件中加入以下内容：

```
if has("gui_running")
  " GUI is running or is about to start.
  " Maximize gvim window (for an alternative on Windows, see simalt below).
  set lines=999 columns=999
else
  " This is console Vim.
  if exists("+lines")
    set lines=50
  endif
  if exists("+columns")
    set columns=100
  endif
```

```
endif
```

以上的数值可以根据实际情况再作微调。

如果只是想把gvim窗口设置到合适大小，可以根据情况自行调整lines和columns的大小。

windows环境，在_vimrc文件中加入以下内容:

```
autocmd GUIEnter * simalt ~x
```

2.5 gvim窗口初始位置

有时候可能会有调整gvim窗口刚打开时的窗口初始位置，比如固定为左上角，右上角，居中等等，可以通过下面的方式去设置它的初始窗口位置。

```
# 在.vimrc文件写入如下语句，将gvim窗口放在指定位置，x y 分别为想设置的x轴y轴坐标
:winpos x y
```

2.6 vim打开文件时回到上次浏览位置

```
"remember last update or view postion"
 " Only do this part when compiled with support for autocommands
 if has("autocmd")
 " When editing a file, always jump to the last cursor position
 autocmd BufReadPost *
 \ if line("'\"") > 0 && line ("'\"") <= line("$") |
 \ exe "normal g'\"" |
 \ endif
 endif
```

以下是来自于gvim help '“ last-position-jump
This autocommand jumps to the last known position in a file just after opening it, if the '” mark is set: >

```
au BufReadPost * if line("'\"") > 1 && line("'\"") <= line("$") | exe
"normal! g`\"" | endif
```

2.7 显示当前文件的完整路径

先按下数字键1，然后再按Ctrl + G

```
CTRL-G          打印当前文件名

{count}CTRL-G   1CTRL-G 打印当前文件全路径名
                If the count is higher than 1 the current buffer number is
also given.
```

2.8 vimdiff 忽略空格

https://www.dazhuanlan.com/2019/12/07/5deb081359393/

只需在vimrc文件中加入以下：

```
set diffopt+=iwhite
```

或者

```
if &diff
    set diffopt+=iwhite
endif
```

或者

```
if &diff
  map gs :call IwhiteToggle()<CR>
  function! IwhiteToggle()
    if &diffopt =~ 'iwhite'
      set diffopt-=iwhite
    else
      set diffopt+=iwhite
    endif
  endfunction
 endif
```

或者以下面的命令方式启动比较:

```
vimdiff -c 'set diffopt+=iwhite'
```

2.9 批处理命令执行

比如以下的命令

```
:%s/^\w\+/cc/
:%s/good/bad/
```

假设是期望依次执行上述语句，可以将其保存到一个临时文件，比如abc.vim 然后在vim窗口中执行`:source abc.vim`，即可以把上述命令依次执行,这样可以减少一些人工操作，而又不用使用其它的脚本语言来处理。

2.10 在每行内容前插入行号

:set number 命令可以在 vim 中显示行号，但是如何让 vim 在真正的文件中添加行号呢？比如说当你要打印代码的时候，或者将代码粘贴到网上的时候，在代码中添加行号会更利于浏览，废话说了这么多，其实只要一行命令就可以搞定。

```
"===== 正向顺序
:g/^/exe ":s/^/" . line(".")

"====== 反向顺序, 行中的70为首行号, 可以根据情况自行修改。
:g/^/exe ":s/^/" . 70-line(".")



"===== 也可以加入printf 修改行号占用的宽度, 比如占用4位宽, 这样显得比较对齐。
:g/^/exe ":s/^/" . printf("%4d ", line("."))


"===== 选中范围行首插入行号, 行号后面再补一个空号, 下面g前面有空格和无空格都可以
:'<,'>g//exe ":s/^/" . line(".") . " "
:'<,'> g//exe ":s/^/" . line(".") . " "

"===== 10行到30行首插入行号, 行号后面再补一个空号, 下面g前面必须要加入空格
:10,30 g//exe ":s/^/" . line(".") . " "


"===== 语法解释：匹配到^的每一行都执行 exe :s/^/xxx/ 其中xxx是line() 或者printf计算后的字符。


" ========== 以下这种方式也可以, 显得好像更简单一些
:%s/^/\=printf('%04d',line('.'))/
:%s/^/\=printf('%04d',300 - line('.'))/
:%s/^/\=printf('%04d',line('.') - 20)/
```

2.11 表达式替换

比如一个文件内容为：

```
AAAA
AAAA
AAAA
```

需要替换为如下样式：

```
BBBB_1
BBBB_2
BBBB_3
```

注意看，123分别是行号，可以采用以下命令进行vim替换操作.

```
:%s@AAAA@\="BBBB_".line(".")@
```

这里用到的=，后面就是跟需要的表达式就行。

2.12 iskeyword

参考： https://zhuanlan.zhihu.com/p/115200953

```
iskeyword, 简写为isk
set iskeyword
set isk
```

使用 `set iskeyword`, 显示当前vim word 分词，方便搜索当前word, 不会乱分词。

注意，有的语法加载会修改这个分词，导致使用gvim造成不便。

通过执行`verbose set iskeyword`， 可以显示是谁修改了iskeyword

如果发现iskeyword被修改，可以单独再把它修改回来，这样避免后面的不便。

3. vim正则表达式

```
Vim中的正则表达式

. 匹配任意一个字符
[abc] 匹配方括号中的任意一个字符。可以使用-表示字符范围， 如[a-z0-9]匹配小写字母和阿拉伯数字
[^abc] 在方括号内开头使用^符号，表示匹配除方括号中字符之外的任意字符
\d 匹配阿拉伯数字，等同于[0-9]
\D 匹配阿拉伯数字之外的任意字符，等同于[^0-9]
\x 匹配十六进制数字，等同于[0-9A-Fa-f]
\X 匹配十六进制数字，等同于[^0-9A-Fa-f]
\w 匹配单词字母，等同于[0-9A-Za-z_]
\W 匹配单词字母之外的任意字符，等同于[^0-9A-Za-z_]
\t 匹配<TAB>字符
\s 匹配空白字符，等同于[ \t]
\S 匹配非空白字符，等同于[^ \t]
\a 所有的字母字符. 等同于[a-zA-Z]
\l 小写字母 [a-z]
\L 非小写字母 [^a-z]
\u 大写字母 [A-Z]
\U 非大写字母 [^A-Z]

表示数量的元字符
* 匹配0-任意个
\+ 匹配1-任意个
\? 匹配0-1个
\{n,m} 匹配n-m个
\{n} 匹配n个
\{n,} 匹配n-任意个
\{,m} 匹配0-m个
\_. 匹配包含换行在内的所有字符
\{-} 表示前一个字符可出现零次或多次，但在整个正则表达式可以匹配成功的前提下，匹配的字符数越
```

少越好 --- 非贪婪匹配
\= 匹配一个可有可无的项
_s 匹配空格或断行
_[]
* 匹配 * 字符
\. 匹配 . 字符
\/ 匹配 / 字符
\\ 匹配 \ 字符
\[匹配 [字符

表示位置的符号
$ 匹配行尾
^ 匹配行首
\< 匹配单词词首
\> 匹配单词词尾

替换变量
在正规表达式中使用 和 符号括起正规表达式，即可在后面使用\1□\2等变量来访问 和 中的内容

懒惰模式
\{-n,m} 与\{n,m}一样，尽可能少次数地重复
\{-} 匹配它前面的项一次或0次, 尽可能地少
\| "或"操作符
\& 并列

函数式
:s/替换字符串/\=函数式
在函数式中可以使用 submatch(1)□submatch(2) 等来引用 \1□\2 等的内容，而submatch(0)可以引用匹配的整个内容

与Perl正则表达式的区别

Vim语法 Perl语法 含义
\+ + 1- 任意个
\? ? 0-1 个
\{n,m} {n,m} n-m 个
和 (和) 分组

例如：
1,去掉所有的行尾空 格：“ :%s/\s\+$//”□“% ”表示在整个文件范围内进行替换□“\s”表示空白字符（空格和制表符），“ \+ ”对前面的字符匹配一次或多次（越多越好□□“___FCKpd___0rdquo;匹配行尾（使用“___FCKpd___0rdquo;表示单纯的 “___FCKpd___0rdquo;字符）；被替换的内容为空；由于一行最多只需替换一次，不需要特殊标志。这个还是比较简单的。(/<Space><Tab>)
2,去掉所有的空白 行：“ :%s/\s*\n\+/\r/ ”。这回多了“”□“”□“\n”□“\r”和 “ *”□“* ”代表对前面的字符（此处为“\s”□匹配零次或多次（越多越好；使用“ * ”表示单纯的“ * ”字符□□“\n”代表换行符□“\r”代表回 车符，“ ”和“ ”对表达式进行分组，使其被视作一个不可分割的整体。因此，这个表达式的完整意义是，把连续的换行符（包含换行符前面可能有的连续 空白字符）替换成为一个单个的换行符。唯一很特殊的地方是，在模式中使用的是“\n”□而被替换的内容中却不能使用“\n”□而只能使用“\r”□原因是 历史造成的，详情如果有兴趣的话可以查看“ :help NL-used-for-Nul”□
3,去掉所有的“ // ”注 释：“ :%s!\ s*//.*!! ”。首先可以注意到，这儿分隔符改用了“ ！”，原因是在模式或字符串部分使用了“ / ”字符，不换用其他分隔符的话就得在每次使用“ / ”字 符本身时写成“ \/ ”，上面的命令得写成“ :%s/\s*\/\/.*// ”，可读性较低。命令本身倒是相当简单，用过正

则表达式的人估计都知道“ . ”匹 配表示除换行符之外的任何字符吧。
4,去掉所有的“ ”注 释：“ :%s!\s*/*_.\{-}*/\s*! !g”这个略有点复杂了，用到了几个不太常用的 Vim 正则表达式特性。“ _. ”匹配包含换行在内的所有字符；“ \{-} ”表示前一个字符可出现零次或多次，但在整个正则表达式可以匹配成功的前提下，匹配的 字符数越少越好；标志“g”表示一行里可以匹配和替换多次。替换的结果是个空格的目的是保证像“intmain() ”这样的表达式在替换之后仍然是合法 的。

```
:g/^\s*$/d      删除只有空白的行
:s/\w\+\s\+\w\+/\2\t\1 将 data1 data2 修改为 data2 data1
:%s/\w\+, \w\+/\2 \1/  将 Doe, John 修改为 John Doe
:%s/\<id\>/\=line(".") 将各行的 id 字符串替换为行号
:%s/\<\w\+/\=(line(".")-10) "." submatch(1)
将每行开头的单词替换为(行号-10).单词的格式,如第11行的word替换成1. word
排序 :/OB/+1,$!sort
```