

Vim 使用手册

这份文档基于你的自定义 .vimrc 配置。在这个配置中，我们将 **Leader 键** 映射为了 **空格键 (Space)**，这是所有组合键的核心。

1. 核心操作 (Core)

最基础的模式切换与文件保存，彻底告别难以触达的按键。

目标操作	快捷键	说明
退出插入模式	jj	最常用！ 手指不用离开主键盘区，秒退回普通模式 (代替 Esc)
快速保存	空格 + w	保存当前文件 (:w)
快速退出	空格 + q	关闭当前窗口 (:q)
Leader 键	空格	后续所有带 <Leader> 的操作都指按一下空格

2. 文件与工程导航 (Navigation)

像使用 VS Code 一样在项目文件之间穿梭。

全局模糊搜索 (FZF)

场景：你知道文件名（比如 trap.c），想立刻打开它，不想在文件夹里一层层找。

- **快捷键：**Ctrl + p
- **操作：**
 1. 按下快捷键，底部弹出一个列表。
 2. 输入文件名的一部分（如 sysfile）。
 3. Enter 打开选中文件。
 4. Ctrl + t 在新标签页打开；Ctrl + v 垂直分屏打开。
- **特性：**底层使用 ripgrep，自动忽略 .git、.o、.d 等垃圾文件，速度极快。

目录树浏览 (NERDTree)

场景：刚接手代码，想看看 kernel 目录下到底有哪些文件。

- **开关目录树：**空格 + e

- **定位当前文件：** 空格 + v (在左侧目录树中自动高亮显示你当前正在编辑的文件)
- **常用操作**（在目录树窗口中）：
 - o：打开文件。
 - i：水平分屏打开。
 - s：垂直分屏打开。
 - C：将目录树根目录切换到当前节点。

 **代码大纲 (Tagbar)**

场景： proc.c 有几百行，你想快速找到 allocproc 函数在哪里。

- **开关大纲：** F8 或 空格 + t
- **说明：** 屏幕右侧会出现当前文件的 宏 (Macros)、结构体 (Structs)、函数 (Functions) 列表。
- **操作：** 光标移动到函数名上，按 Enter 直接跳转。

3. 窗口管理 (Window Management)

在分屏之间切换不再痛苦。

 **开启与关闭分屏**

命令	说明	示例
:vsp [文件名]	垂直分屏 (Vertical Split)。左右两边对照代码神器。	:vsp proc.c (右边打开 proc.c)
:sp [文件名]	水平分屏 (Split)。上下分屏。	:sp (上下查看同一个文件)
:close 或 :q	关闭 当前 光标所在的分屏窗口。	-
:only	关闭 除当前窗口外 的所有其他分屏。	-

 **窗口切换快捷键**

基于配置文件的优化按键（无需按 Ctrl+w）：

目标操作	快捷键	说明
光标移到左窗口	Ctrl + h	代替了原生的 Ctrl+w h

光标移到下窗口	Ctrl + j	代替了原生的 Ctrl+w j
光标移到上窗口	Ctrl + k	代替了原生的 Ctrl+w k
光标移到右窗口	Ctrl + l	代替了原生的 Ctrl+w l

4. 高效编辑 (Editing)

基于插件的代码编辑神技。

一键代码格式化 (自定义脚本)

场景：从网页复制的代码缩进乱了，或者行尾有一堆红色的多余空格。

- **快捷键：** 空格 + f
- **功能：**
 1. **自动缩进：** 重新排列全文缩进 (C 语言/Python 标准)。
 2. **清理空格：** 删除行尾多余的空白字符。
 3. **统一格式：** 将所有 Tab 强制转换为 4 个空格。

快速注释 (Commentary)

- **注释当前行：** gcc
- **注释选中的多行：** 进入 Visual 模式选好行，按 gc。

符号包裹修改 (Surround)

场景：C 语言中修改括号、引号极其常用。

原始代码	想要变成	快捷键命令	助记 (Mnemonic)
"Hello"	'Hello'	cs"	C hange S urround " to '
"Hello"	Hello	ds"	D elete S urround "
Hello	"Hello"	ysiw"	Y ou S urround I nnner W ord with "
(x + y)	[x + y]	cs([C hange S urround (to [

5. 阅读与搜索 (Reading)

针对内核源码阅读的优化。

 **搜索与跳转**

- **取消搜索高亮**：空格 + Enter (回车)
 - *说明*：当你用 / 搜索完后，满屏黄色很不舒服，按这个快速清除。
- **代码定义跳转** (原生 + Ctags)：
 - Ctrl +]： **跳转**到函数/变量定义处。
 - Ctrl + t： **返回**跳转前的位置。

 **相对行号的使用**

- **显示逻辑**：当前行显示绝对行号，其他行显示距离当前行的行数。
- **实战用法**：
 - 想删除下面第 5 行代码？看一眼标号是 5，直接按 5dd。
 - 想向下移动 8 行？直接按 8j。

6. 剪贴板与寄存器 (Registers)

Vim 的“复制粘贴”比你想象的要强大，因为它有多个“剪贴板”（称为寄存器）。

 **常用寄存器**

符号	名称	用途
""	无名寄存器 (默认)	只要你 d (删除) 或 y (复制) 了内容，都会自动存到这里。按 p 默认粘贴这里的内容。
"+	系统剪贴板	连接 Windows 的桥梁 。复制到这里的内容 ("+y) 可以去 Windows 粘贴；Windows 复制的内容可以用 "+p 粘贴进来。
"0	专用复制寄存器	只存你 复制 (Yank) 的内容， 不会被删除 (Delete) 操作覆盖。

 **实战场景**

1. **场景：复制 Linux 代码到 Windows**
 - **全文件复制**：gg"+yG (跳到头 -> 选系统板 -> 复制 -> 到尾)

- **选区复制**：按 v 选中代码 -> 按 "+y
- 2. **场景：防止“删除”覆盖了“复制”**
 - **问题**：你复制了一行代码，然后去删掉了另一行准备替换，结果按 p 粘贴出来的却是刚才删除的那行（气死人！）。
 - **解决**：使用 "Op。因为刚才的删除操作污染了默认寄存器，但你的复制内容还安全地躺在 0 号寄存器里。
- 3. **查看所有寄存器**
 - 命令：:reg (可以看到你刚才复制过的所有历史片段)

7. 界面说明 (UI)

- **底部状态栏 (Airline)**:
 - **蓝色/绿色/橙色条**：代表当前是 普通/插入/可视 模式。
 - **最右侧**：显示 Ln: 10 Col: 5 (行号:列号)。
 - **警告**：如果状态栏出现橙色/红色，通常意味着文件包含 **混合缩进 (Mixed-indent)** 或 **尾部空格 (Trailing)**，请按 空格 + f 修复。
- **配色 (Gruvbox)**:
 - 暖色调深色背景，专为长时间阅读 C 语言设计，高对比度但护眼。

8. 遇到问题？

- **粘贴代码缩进错乱**:
 - 临时解决：输入 :set paste -> 粘贴 -> :set nopaste。
 - 终极解决：使用一键格式化 空格 + f。
- **Tagbar 报错**:
 - 确保安装了 ctags：sudo apt install universal-ctags。
- **FZF 搜出垃圾文件**:
 - 你的配置已强制使用 ripgrep 过滤，如果还搜得到，请检查项目根目录下的 .gitignore 文件。