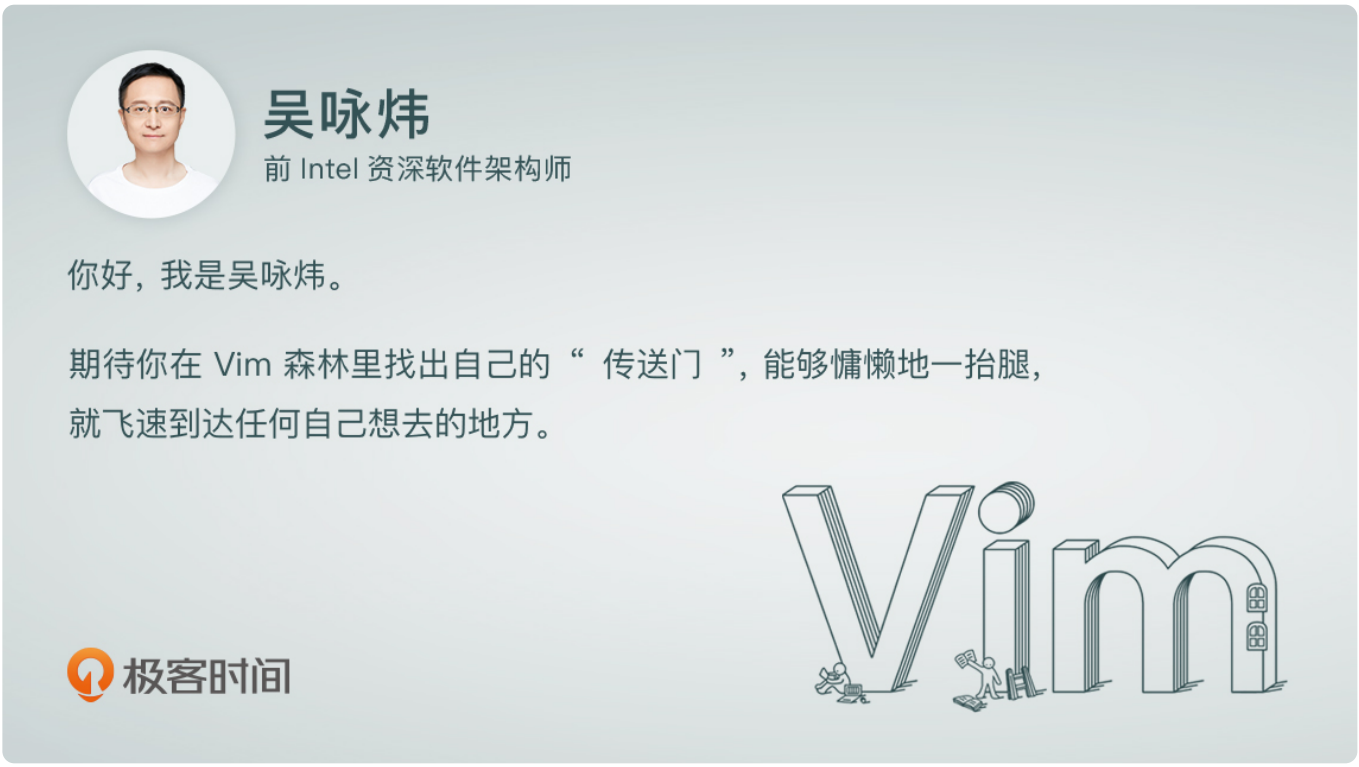
结束语 | Vim 森林探秘，一切才刚刚开始

2020-09-11 吴咏炜

Vim 实用技巧必知必会 进入课程 >



讲述：吴咏炜
时长 07:24 大小 6.80M

你好，我是吴咏炜。

我们的课程到这里就结束了，而你的学习旅程，到这儿只能算是一个小小的休息站。

学习的度

对于一个持续发展了 30 年的编辑器，我们显然不可能在一门短小精悍的课程里完整地覆盖它的所有功能。不过，我从来就没打算介绍 Vim 的一切。如果把 Vim 比作一片大森林，我只是一个导游，为你制定了一条旅游路线，带你绕过沼泽地和陷阱，攀上了几座峰顶，让你能够领略到若干美景。如果想在林中长久地居住下去，熟稔各条秘径，你仍然要靠自己去探索。

Vim 的作者 Bram 这么告诫人们不要走到两个极端上去 [Moolenaar 2007]：

> 你需要马上把文本准备好。所以没有时间读文档或学习新命令。——**你会一直使用原始的命令。**
> 你想学编辑器提供的所有功能，并在任何时候都能使用最高效的命令。——**你会浪费很多时间学习很多你永远不会用到的东西。**

前者的问题很明显，如果你不学习，那你只能使用初级的功能，所以效率一定很低。后者的问题可能不那么明显了：实际上，除了多花时间之外，你很难培养出良好的习惯，形成“肌肉记忆”。而这，恰恰是高效工作的关键之一——不需要想，就知道怎么做，从而可以把头脑和精力投入到更重要的问题上。

在这个课程里，我也只是告诉你基本的原则和技巧，并培养你基本的编辑习惯。回头，在遇到实际问题时，你会需要使用搜索引擎、讨论组等工具来找到问题的答案。

学习、积累和分享

我学习 Vim，原本是处于 Linux 下开发的需要，而后慢慢成为一种习惯。回过头来看，Vim 就和 Unix 一样，老而弥坚，经久不衰，不管时代怎样变化，它们却一直没有过时。而同时代我用过的其他 Windows 上的编辑器，如 EditPlus 和 UltraEdit，现在虽然都还在，但是应该已经很少有人提起了吧……

经过几年的学习和使用，我居然也就可以给别人分享我的经验了。我先是在 IBM developerWorks 上发了一系列的三篇 Vim 文章，后面又在 SHLUG（Shanghai Linux User Group）的活动中分享了我的 Vim 使用经验。事后，我在网上看到别人觉得我这个分享做得最好，也是非常的欣慰。

时间快进到今年的年初，我做完了我在极客时间的第一门课程《现代 C++ 实战 30 讲》。当编辑问我还有什么其他可分享的课程时，我立刻想到了 Vim。于是，就有了这个课程。我很喜欢知识分享的过程，因为准备的过程同时也是自我梳理和刷新的过程，毕竟要给别人讲，就不能像自己用的时候那样不求甚解了。让我没预料到的是，在课程中我还向某些积极的同学学了一两手（为了不让你们太骄傲，我就不点名了😉）。知识分享真是一种非常好的活动，于人于己都非常有利。建议大家在工作中也可以多多考虑😇。它唯一的缺点，就是会让自己变得非常忙碌、周末没有休息时间而已😂。

高效编辑的诀窍

现在回到 Vim。我们来考虑一下这个“元”问题：**怎么样可以进行高效的编辑?**

事实上，Bram 早就回答过这个问题了。这跟一般的效率改进计划没有本质的区别。我们要做的是：

1. 发现低效的根源
2. 找出更快的方法
3. 形成新的习惯

Vim 的功能也是围绕着这样的模式开发出来的。

比如，我们要找出当前文件中某个符号的使用，在任何编辑器里，都会提供一个搜索的功能。但每次都使用搜索，实际上也是有点麻烦的。Vim 里的 * 搜索键和搜索加亮，这两个你目前应当已经习以为常的功能，就是为了解决这种低效而诞生的。而对你，现在更快的方法，就是在配置中启用搜索加亮（目前我们的基本配置里已经启用），并使用 * 来搜索光标下的单词。

又比如，我们有一个非常长的函数名，打起来又费力又容易出错。这时候，我们需要找出更快的方法，那就是自动完成。Vim 内置的改进方法是 <C-N> 和 <C-P> 命令。而我们学到现在的就该知道，YCM 还提供了更现代的模糊完成引擎。用好内置命令，或安装合适的插件，就是我们需要形成的新习惯。

说到这儿，你也看到了，不仅我们的课程不是高效编辑的终点，而且就连 Vim 也不是。Vim 一直在发展，但有人嫌 Vim 发展得不够激进，另外搞了 Neovim。我们这个课程完全没有讨论 Neovim，主要是不想在一个已经很复杂的课题上再增加复杂性（同时也是因为 Neovim 虽然看起来势头不错，但远没到可以尘埃落定、一定会在将来替代 Vim 的程度，再加上两者并不完全兼容……反过来，Neovim 倒是刺激 Vim 更快地添加新功能了）。

抛开 Neovim 不谈，我们还有插件：正是这些辛勤的插件作者，才使得 Vim 真正更为强大，成为一个特别高效的编辑器。就像上面讨论的，有了 YCM，我们找到了一个比 Vim 内置功能更加简单、直观、高效的方法。我们也应该去用好这样的新工具，提升自己的工作效率。

要提高自己的编辑效率，需要时时刻刻注意自己有哪儿效率特别低，出现了不必要的重复，然后找到更好的办法来改进，并确保自己形成新的习惯。

这个新的习惯是什么呢？可以是掌握了 Vim 里的一个之前不熟悉的功能，可以是用上了一个新的插件，也可以是……为 Vim 社区贡献一个新的插件！

这也恰恰是我在开篇词里提到的“懒惰”。所谓懒惰，就是我们要不让自己做低效的重复工作。而要不做低效的重复工作，我们就需要开动自己的脑子，监测自己的工作方式，找出问题点，予以改进，并坚持下去。懒惰的手段就是高效；反过来说也可以，高效的目的就是懒惰。

期待你在 Vim 森林里找出自己的“传送门”，能够慵懒地一抬腿，就飞速到达任何自己想去的地方。

我们后会有期！

《Vim 实用技巧必知必会》课程结束了，这里有一份毕业问卷，题目不多，希望你能花两分钟填一下。十分期待能听到你说一说，你对这个课程的想法和建议。

参考资料

作为写作的基本规矩，我最后列一下我参考过的资料（插件和之前文中直接给出的链接则不再重复）。希望它们也能帮到你，让你在下面的旅途中再上一层楼：

Allen, Leo. 2016. “Why Vim is so much better than Atom”.

Arthur, Barry. 2014. “Learning the tool of Vim”.

Bringhurst, Robert. 2012. *The Elements of Typographic Style*, 4th ed. Hartley & Marks Publishers.

Irwin, Conrad. 2013. “Bracketed paste mode”.

Kochkov, Anton. 2019. “Terminal Colors”.

Leonard, Andrew. 2000. “BSD Unix: Power to the people, from the code”.

Moolenaar, Bram. 2000. “The continuing history of Vim”.

Moolenaar, Bram. 2002. “Vim, an open-source text editor”.

Moolenaar, Bram. 2007. “7 habits for effective text editing 2.0”. YouTube video.

Moolenaar, Bram. 2018. “Vim: Recent developments”.

Neil, Drew. 2015. *Practical Vim*, 2nd ed. O’Reilly. 中文版：杨源、车文隆译《Vim 实用技巧》，人民邮电出版社，2014（第一版），2016（第二版）。

Neil, Drew. 2018. *Modern Vim: Craft your development Environment with Vim 8 and Neovim*. Pragmatic Bookshelf. 中文版：死月译《精通 Vim：用 Vim 8 和 Neovim 实现高效开发》，电子工业出版社，2020。

Ornbo, George. 2019. “Vim: you don’t need NERDtree or (maybe) netrw”.

Osipov, Ruslan. 2018. *Mastering Vim*. Packt. 中文版：王文涛译《Vim 8 文本处理实战》，人民邮电出版社，2020。

Robbins, Arnold, Elbert Hannah, and Linda Lamb. 2008. *Learning the vi and Vim Editors*, 7th ed. O’Reilly.

Salus, Peter H. 1994. *A Quarter Century of UNIX*. Addison-Wesley.

Schneider, Peter A. 2018. Answer to “How do I disable the weird characters from ‘bracketed paste mode’ on the Mac OS X default terminal?”.

Stack Overflow. 2015. “2015 developer survey”, section “Technology > Text editor”.

Stack Overflow. 2019. “Developer survey results 2019”, section “Technology > Development environments and tools”.

Target, Sinclair. 2018. “Where Vim came from”.

Vance, Ashlee. 2003. “Bill Joy’s greatest gift to man – the vi editor”.

Vim Online.

Wikipedia. “Berkeley Software Distribution”.

Wikipedia. “Bill Joy”.

Wikipedia. “ex (text editor)”.

Wikipedia. “Version 6 Unix”.

Wikipedia. “vi”.

Wikipedia. “Vim”.

Wikipedia. “Visual editor”.

提建议

上一篇 拓展5 | 其他插件和技巧：吴咏炜的箱底私藏

精选留言 (8)

写留言

我来也
2020-09-11

老师辛苦了！

本专栏令我收获颇丰！
还有很多地方有待我去实践。
...
展开

作者回复: 谢谢。

你几乎每讲都是第一个学习完，还能提出很多好建议——厉害！

1 4

doge
2020-09-11

老师的课程让我受益良多，虽然一直用vim，也一直沿用github上高手们共享的配置，但还没真正仔细的研究过相关功能实现的方式以及脚本的写法，看了老师的教学和评论高手的分享，对vim的理解更深入了些，对高效编辑的一些思考也多了一些。按照老师的脚本也算是自我定制出了比较满意的一个vim版本。评论的朋友说的好，选好一个编辑器，然后一直用下去，好东西还是需要多多打磨的。最后再次感谢老师系统性的分享。

展开

作者回复: 是的，Vim是一个可以长期积累、持续使用的环境。

1

Sochooligan
2020-09-11

感谢老师的深厚经验和精彩分享！断续在看，很有必要再精读几遍。虽然现在用的是emacs、spacemacs用的是emacs的按键，但编辑器（IDE）面对的问题都是类似的，很多解决问题的方法都是通用的。关于编辑器我有一点体会是：你最终会在Vim和Emacs之间不再纠结（也许还有sublime，atom，vs code等），选择一个自己的最爱，把使用时所有遇到的问题，都在这个编辑器里配好、改进，并一直用下去，用到最好。选你所爱，爱你...

展开

作者回复: 欢迎Emacs党。不过，这两个要精通一个都不容易啊，再加设计哲学都有点区别。双修还是很费力气的，当然，搞好了就很牛。:-)

1

pyhhou
2020-09-13

到这里真是有些不舍，感觉自己每一讲都能学到很多新知识，了解自己之前不曾了解的东西（可能是自己对 VIM 的了解不够[illegible]）。回看这个专栏，自己的收获真的不小，比如掌握了 vim 脚本的基本配置方法，了解了一些高效便捷的指令组合，也跟着老师知道了很多很便捷的插件，另外自己还尝试阅读了一些 VIM 源码。让我收获最大的还是专栏通过各种常见的例子，很清楚地展示一些指令和插件的应用场景，知道了一个指令或插件为...

展开

newcode
2020-09-11

时间过得真快，“一切才刚刚开始”。

展开

作者回复: 结束才是开始。:-)

YouCompleteMe
2020-09-11

虽然之前看过《Vim实用技巧》/《Vim8文本实战处理》，但是这个专栏才真正打通了我使用Vim的任督二脉。以前自己的vimrc参考了Github上一些Vim插件作者的配置，很多配置只是人云亦云，现在有了一份自己的精简的vimrc，对其中的每一项配置都了然于胸。对于发现不高效的地方，还可以自己写VimL改进效率。感谢老师的辛苦付出，收获颇丰，期待老师的下一门课程。...

展开

作者回复: 很高兴对你有帮助。能把控细节，也确实是我对工具的期望。

一只呆子
2020-09-11

我从毕业工作开始，到现在使用VIM已经十年了，也是和老师一样，先是被逼着CentOS环境开发使用
购买老师的课程也是想知道老师是怎么使用VIM的，有哪些我个人不知道的高效技巧
从9.9购买课程，到9.10看完课程，今天9.11课程完结
时间虽短，收获也是颇为丰富...

展开

作者回复: 学习飞速啊。厉害！

qinsi
2020-09-11

个人觉得Vim作为IDE而言功能还是薄弱了些，毕竟原本定位就只是编辑器。而很多传统的使用场景如运维连上远程服务器改配置，或是在服务器上进行简单的远程开发和调试等，也随着自动化运维以及WebIDE技术的发展逐渐减少。但Vim的高效编辑键位作为一种遗产保留了下来，在很多主流IDE中都可以通过插件方式支持。既然离不开现有的IDE，仅仅

通过熟练掌握Vim的键位也可以显著提高效率。但很多操作是键位表上没有的，这门课...

展开

作者回复: 也是一种用法吧……对我来说，用Vim配插件可以更好地做到跟编译环境一致的代码理解。如果做Linux开发，用Windows的IDE，虽然也能配到完全理解代码，我感觉这个配置反而也很复杂。而且在Vim里更能控制一切，开销还能比较低。