

# 把vim打造一个比Source Insight更强更好用的IDE

笨叔叔

《奔跑吧linux内核》配置指南已经上线  
<https://shop115683645.taobao.com/>

# Linux从业人员大部分人都使用VIM

```
figo@figo-OptiPlex-9020: ~/work/test1/runninglinuxkernel/runninglinuxkernel_4.0
File Edit View Search Terminal Tabs Help

figo@figo-OptiPlex-9020: ~/wo... x figo@figo-OptiPlex-9020: ~/wo... x figo@figo-OptiPlex-9020: ~/wo... x figo@figo-OptiPlex-9020: ~/wo... x figo@figo-OptiPlex-9020: ~/wo... x figo@figo-OptiPlex-9020: ~/.vim x figo@figo-OptiPlex-9020: ~ x +

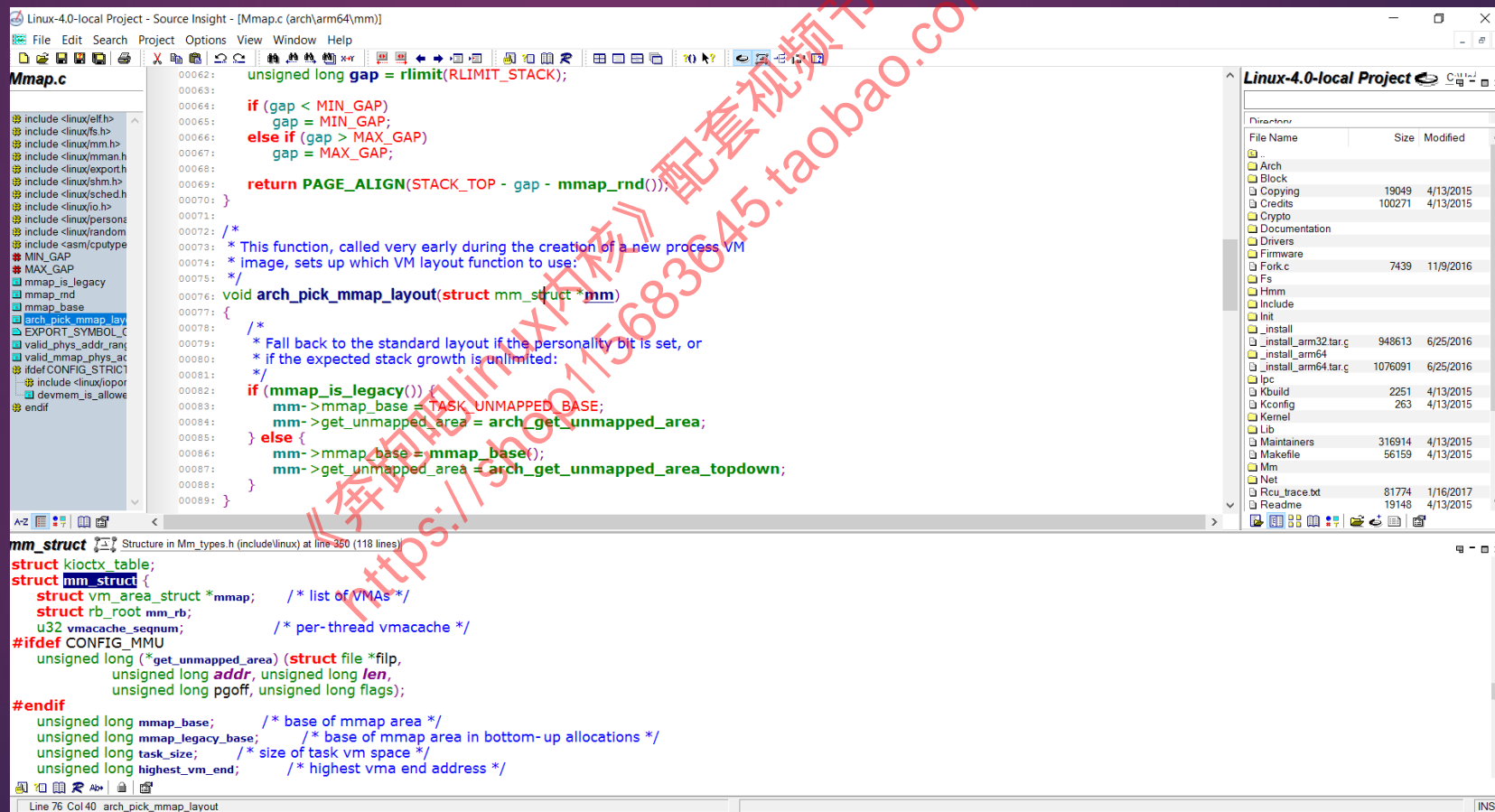
1 " Press ? f 2554 */
2 2555 static int do_anonymous_page(struct mm_struct *mm, struct vm area struct *vma,
3 >----- 2556 unsigned long address, pte_t *page_table, pmd_t *pmd,
4 2557 unsigned int flags)
5 .. (up a di 2558 {
6 <nuxkernel 2559 struct mem_cgroup *memcg;
7 > _install 2560 struct page *page;
8 > _install 2561 spinlock_t *ptl;
9 > _install 2562 pte_t entry;
10 > arch/ 2563 vma->
11 > block/ 2564 address
12 > crypto/ 2565 pte address_space
13 > Documenta 2566 address_space::
14 > drivers/ 2567 /* C alignas( expression )
15 > firmware/ 2568 if ( alignof( type )
16 > fs/ 2569 alloc_context
17 > include/ 2570 alloc_context::
18 > init/ 2571 /* U auto
19 > ipc/ 2572 if ( basename( const char * filename )
20 > kernel/ 2573 bcmp( const void * __s1, const void * __s2, size_t __n )
21 > kmodules/ 2574 bcopy( const void * __src, void * __dest, size_t __n )
22 > lib/ 2575 bool
23 > [x]mm/ 2576 bzero( void * __s, size_t __n )
24 > module_te 2577 char
25 > net/ 2578 char16_t
26 > samples/ 2579 } char32_t
27 > scripts/ 2580 check_sync_rss_stat( struct task_struct *task )
28 > security/ 2581 /* A class
29 > sound/ 2582 if ( clear_page_mlock( struct page *page )
30 > tools/ 2583 const
31 > usr/ 2584 page constexpr
32 > virt/ 2585 if ( const_cast<type>( expression )
<nuxkernel 4.0 INSERT 1:memory. decltype( expression )
-- INSERT --
```

2554 \*/  
2555 static int do\_anonymous\_page(struct mm\_struct \*mm, struct vm area struct \*vma,  
2556 unsigned long address, pte\_t \*page\_table, pmd\_t \*pmd,  
2557 unsigned int flags)  
2558 {  
2559 struct mem\_cgroup \*memcg;  
2560 struct page \*page;  
2561 spinlock\_t \*ptl;  
2562 pte\_t entry;  
2563 vma->  
2564 address  
2565 pte address\_space  
2566 address\_space::  
2567 /\* C alignas( expression )  
2568 if ( alignof( type )  
2569 alloc\_context  
2570 alloc\_context::  
2571 /\* U auto  
2572 if ( basename( const char \* filename )  
2573 bcmp( const void \* \_\_s1, const void \* \_\_s2, size\_t \_\_n )  
2574 bcopy( const void \* \_\_src, void \* \_\_dest, size\_t \_\_n )  
2575 bool  
2576 bzero( void \* \_\_s, size\_t \_\_n )  
2577 char  
2578 char16\_t  
2579 } char32\_t  
2580 check\_sync\_rss\_stat( struct task\_struct \*task )  
2581 /\* A class  
2582 if ( clear\_page\_mlock( struct page \*page )  
2583 const  
2584 page constexpr  
2585 if ( const\_cast<type>( expression )  
decltype( expression )

apply to pud\_range(st  
check\_stack\_guard pag  
check\_sync\_rss\_stat(s  
check\_sync\_rss\_stat(s  
clear\_gigantic\_page(s  
clear\_huge\_page(struc  
copy\_one\_pte(struct m  
copy\_page\_range(struc  
copy\_pmd\_range(struc  
copy\_pte\_range(struc  
copy\_pud\_range(struc  
copy\_user\_gigantic\_pa  
copy\_user\_huge\_page(s  
cow\_user\_page(struct  
disable\_randmaps(char  
do\_anonymous\_page(str  
do\_cow\_fault(struct m  
do\_fault(struct mm\_st  
do\_fault\_around(struc  
do\_numa\_page(struct m  
do\_page\_mkwrite(struc  
do\_read\_fault(struct  
do\_set\_pte(struct vm  
do\_shared\_fault(struc  
do\_swap\_page(struct m  
fault\_around\_bytes\_ge  
fault\_around\_bytes\_se  
fault\_around\_debugfs(  
follow\_pfn(struct vm\_  
follow\_phys(struct vm\_  
follow\_pte(struct mm\_  
free\_pgd\_range(struc  
Tagbar Name memory.c

# Source Insight有什么地方吸引我们？

- 查找函数或者变量的定义
- 查找哪些函数调用了该函数
- 代码补全功能



# vim设计理念

- vim的设计理念是把整个文本编辑器都用键盘的来操作而不需要使用鼠标
  - 减少使用鼠标
  - 减少敲击键盘
  - 减少手指移动
  - 减少目光移动

《奔跑吧linux内核》配套视频节目已经上线  
<https://shop115683645.taobao.com/>

# vi / vim 键盘图

**Esc**  
命令  
模式

~ 转换 大小写	! 外部 过滤器 <sup>2</sup>	@ 运行 宏	# prev ident	\$ 行尾	% 括号 匹配	^ "软" 行首	& 重复 :s	* next ident	( 句首	) 下一 句首	"soft" bol down	+ 后一行 行首
\. 跳转到 标注	1	2	3	4	5	6	7	8	9	0 "硬" 行首	- 前一行 行首	= 自动 <sup>3</sup> 格式化
Q 切换至 ex模式	W 下一 单词	E 词尾	R 替换 模式	T back 'till	Y 拷贝 行	U 撤消 行内命令	I 到行首 插入	O 分段 (前)	P 粘贴 (前)	{ 段首	}	段尾
q 录制 宏	w 下一 单词	e 词尾	r 替换 字符	t 'till	y 拷贝 <sup>1,3</sup>	u 撤消 命令	i 插入 模式	o 分段 (后)	p 粘贴 <sup>1</sup> (后)	[ 杂项	]	杂项
A 在行尾 附加	S 删除行 并插入	D 删除 至行尾	F 行内字符 反向查找	G 文尾/ 行号	H 游标 上行	J 合并 两行	K 帮助	L 屏幕 底行	:	ex 命令	" 寄存器 <sup>1</sup> 标识	行首/ 列
a 附加	s 删除字符 并插入	d 删除 <sup>1,3</sup>	f 行内字符 查找	g 附加 命令 <sup>6</sup>	h ←	j ↓	k ↑	l →	;	重复 u/T/f/F	' 跳转到标 注的行首	\. 未用!
Z 退出 <sup>4</sup>	X 退格	C 修改 至行末	V 可视 行模式	B 前一 单词	N 查找 上一处	M 屏幕 中间行	< 反缩进 <sup>3</sup>	> 缩进 <sup>3</sup>	?	向前 搜索		
Z 附加 命令 <sup>5</sup>	X 删除 (字符)	c 修改 <sup>1,3</sup>	v 可视 模式	b 前一 单词	n 查找 下一处	m 设置 标注	, 反向 u/T/f/F	.	重复 命令	/	向后 搜索	

**动作** 移动光标, 或者定义操作的范围

**命令** 直接执行的命令,  
红色命令 进入编辑模式

**操作** 后面跟随表示操作范围的指令

**extra** 特殊功能,  
需要额外的输入

q 后跟字符参数

w,e,b命令

小写(b): quux(foo, bar, baz);

大写(B): quux(foo, bar, baz);

主要ex命令:

:w (保存), :q (退出), :q! (不保存退出)

:ef (打开文件 f),

:%s/x/y/g ('y' 全局替换 'x'),

:h (帮助 in vim), :new (新建文件 in vim),

其它重要命令:

CTRL-R: 重复 (vim),

CTRL-F/-B: 上翻/下翻,

CTRL-E/-Y: 上滚/下滚,

CTRL-V: 块可视模式 (vim only)

可视模式:

漫游后对选中的区域执行操作 (vim only)

备注:

(1) 在 拷贝/粘贴/删除 命令前使用 "x (x=a..z,\*)  
使用命令的寄存器('剪贴板')

(如: "ay\$ 拷贝剩余的行内容至寄存器 'a')

(2) 命令前添加数字  
多遍重复操作  
(e.g.: 2p, d2w, 5i, d4j)

(3) 重复本字符在光标所在行执行操作  
(dd = 删除本行, >> = 行首缩进)

(4) ZZ 保存退出, ZQ 不保存退出

(5) zt: 移动光标所在行至屏幕顶端,  
zb: 底端, zz: 中间

(6) gg: 文首 (vim only),  
gf: 打开光标处的文件名 (vim only)

## vim工作模式

- 命令模式（**Command mode**）：用户打开vim的时候便进入到命令模式。在命令模式下输入的键盘动作会被vim识别成命令，而非输入字符。比如这时输入i，vim识别的是一个i命令。用户可以输入命令来控制屏幕光标的移动，文本的删除或者复制某段区域等，也可以进入底行模式或者插入模式。
- 插入模式（**Insert mode**）：在命令模式下输入i命令就可以进入插入模式，按ESC键可以回到命令行模式。要想在文本中输入字符，必须是在插入模式下。
- 底行模式（**Last line mode**）：在命令模式下按下“：”就进入底线模式。在底行模式下可以输入单个或者多个字符的命令。比如“:q”退出vim编辑器。

# vim第一层功力：入门

《奔跑吧linux内核》配套视频节目已经上线  
<https://shop115683640.taobao.com/>



- i → Insert 模式，按 ESC 回到 Normal 模式.
- x → 删当前光标所在的一个字符。
- :wq → 存盘 + 退出 (:w 存盘, :q 退出)
- dd → 删除当前行，并把删除的行存到剪贴板里
- p → 粘贴剪贴板
- hjkl -> 光标键 (←↓↑→) 使用注:j 就像下箭头。
- :help <command> → 显示相关命令的帮助。

《奔跑吧linux》  
https://shop115669543.taobao.com



# 训练使用hjk1键来操作光标

技巧：把光标键 映射成Nop  
修改 ~/.vimrc 文件：

```
noremap <Up> <Nop>  
noremap <Down> <Nop>  
noremap <Left> <Nop>  
noremap <Right> <Nop>
```

## vim第二层功力：熟悉

《奔跑吧linux内核》配套视频节目已经上线  
<https://shop115683640.taobao.com/>

# 插入命令

插入命令：

a → 在光标后插入

o → 在当前行后插入一个新行

O → 在当前行前插入一个新行

CW → 替换从光标所在位置后到一个单词结尾的字符

《奔跑吧兄弟》配套视频节目已经上线  
<https://shop116829.taobao.com/>

# 简单的光标移动

0 → 数字零，到行头

^ → 到本行第一个不是blank字符的位置（所谓blank字符就是空格，tab，换行，回车等）

\$ → 到本行行尾

g\_ → 到本行最后一个不是blank字符的位置。

/pattern → 搜索 pattern 的字符串（如果搜索出多个匹配，可按n键到下一个）

# 拷贝粘贴撤销

- P → 粘贴 ( p/P都可以 , p是表示在当前位置之后 , P表示在当前位置之前 )
- yy → 拷贝当前行
- u : 撤销
- <C-r> → redo

# 文件操作

- :e <path/to/file> → 打开一个文件
- :w → 存盘
- :saveas <path/to/file> → 另存为 <path/to/file>
- :wq → 保存并退出
- :q! → 退出不保存 :qa! 强行退出所有的正在编辑的文件，就算别的文件有更改。
- :bn 和 :bp → 你可以同时打开很多文件，使用这两个命令来切换下一个或上一个文件

vim第三层功力：更快

《奔跑吧linux内核》配套视频节目已经上线  
<https://shop115683640.taobao.com/>



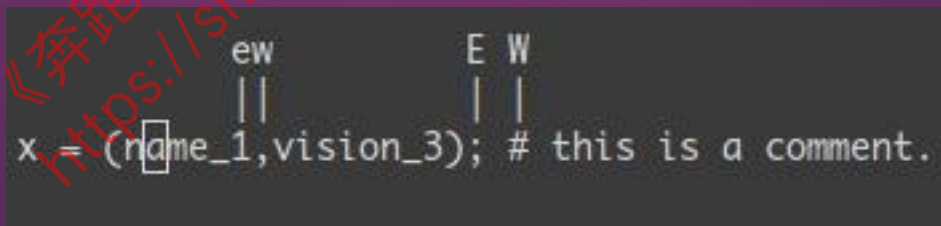
# 重 复

- . → (小数点) 可以重复上一次的命令
- N<command> → 重复某个命令N次

《奔跑吧Linux内核》配套视频节目已经上线  
<https://shop115683745.taobao.com/>

# 更快的光标移动

- NG → 到第 N 行（另外: N 到第N行，如 :137 到第137行）
- gg → 到第一行。（相当于1G，或 :1）
- G → 到最后一行。
- w → 到下一个单词的开头。
- e → 到下一个单词的结尾。
- % : 匹配括号移动，包括 (, {, [.
- #: 匹配光标当前所在的单词，移动光标到下一个（或上一个）匹配单词（\*是下一个，#是上一个），有点类似 source insight 那个 HightLight word



```
x = (name_1, vision_3); # this is a comment.
```

# 命令和光标移动命令的组合

- 很多命令都可以和这些移动光标的命令连动:

<start position><command><end position>

例子: 0y\$ :

0 → 先到行头

y → 从这里开始拷贝

\$ → 拷贝到本行最后一个字符

《奔跑吧linux内核》配套视频节目已经上线  
<https://shop115683645.taobao.com/>

# vim第四层功力：超能力

《奔跑吧linux内核》配套视频节目已经上线  
<https://shop115683649.taobao.com/>

# 光标移动的高级用法

- 0 → 到行头
- ^ → 到本行的第一个非blank字符
- \$ → 到行尾
- g\_ → 到本行最后一个不是blank字符的位置。
- fa → 到下一个为a的字符处，你也可以fs到下一个为s的字符。
- t, → 到逗号前的第一个字符。逗号可以变成其它字符。
- 3fa → 在当前行查找第三个出现的a。
- F 和 T → 和 f 和 t 一样，只不过是相反方向。

```
0  ^  fi  t)  4fi  g_  $  
|  |  |  |  |  |  |  
x = (name_1, vision_3); #this is a comment.  
~
```

# 可视化操作

- 按v启用可视模式，可以按单个字符选择内容，移动光标可以选择。
- 按V启用可视模式，立刻选中光标所在行，按单行符选择内容，移动光标可以选择。
- 按CTRL+V启用可视中的列块模式，可以在列方向上选择单个字符，移动光标可以选择
- CTRL+v,启用块可视模式，可以选中某一个矩形块，对于有规律的表格可以用这个功能。

《奔跑吧linux内核》可视化目录已发布  
<https://shop115683645.taobao.com/>

# 分屏管理

- :split → 创建分屏 (:vsplit创建垂直分屏)
- <C-w><dir> : dir就是方向，可以是 hjkl 或是 ←↓↑→ 中的一个，其用来选择分屏。
- <C-w>\_ (或 <C-w>|) : 最大化尺寸 (<C-w>| 垂直分屏)
- <C-w>+ (或 <C-w>-) : 增加尺寸

《奔跑吧Linux》已上架视频节目已经上线  
<https://shop115683655.taobao.com/>



# vim第五层功力：打造超级IDE

《奔跑吧linux内核》配套视频节目已经上线  
<https://shop115683605.taobao.com/>

```
figo@figo-OptiPlex-9020: ~/work/test1/runninglinuxkernel/runninglinuxkernel_4.0
File Edit View Search Terminal Tabs Help
figo@figo-OptiPlex-9020: ~/wo... x figo@figo-OptiPlex-9020: ~/wo... x figo@figo-OptiPlex-9020: ~/wo... x figo@figo-OptiPlex-9020: ~/wo... x figo@figo-OptiPlex-9020: ~/wo... x figo@figo-OptiPlex-9020: ~/.vim x figo@figo-OptiPlex-9020: ~ x
1 " Press ? f 2554 */
2 2555 static int do_anonymous_page(struct mm_struct *mm, struct vm area struct *vma,
3 >----- 2556 unsigned long address, pte_t *page_table, pmd_t *pmd,
4 2557 unsigned int flags)
5 .. (up a di 2558 {
6 <nuxkernel 2559 struct mem_cgroup *memcg;
7 > _install 2560 struct page *page;
8 > _install 2561 spinlock_t *ptl;
9 > _install 2562 pte_t entry;
10 > arch/ 2563 vma->
11 > block/ 2564 address
12 > crypto/ 2565 pte address_space
13 > Documenta 2566 address_space::
14 > drivers/ 2567 /* C alignas( expression )
15 > firmware/ 2568 if ( alignof( type )
16 > fs/ 2569 alloc_context
17 > include/ 2570 alloc_context::
18 > init/ 2571 /* U auto
19 > ipc/ 2572 if ( basename( const char * filename )
20 > kernel/ 2573 bcmp( const void * __s1, const void * __s2, size_t __n )
21 > kmodules/ 2574 bcopy( const void * __src, void * __dest, size_t __n )
22 > lib/ 2575 bool
23 > [x]mm/ 2576 bzero( void * __s, size_t __n )
24 > module_te 2577 char
25 > net/ 2578 char16_t
26 > samples/ 2579 } char32_t
27 > scripts/ 2580 check_sync_rss_stat( struct task_struct *task )
28 > security/ 2581 /* A class
29 > sound/ 2582 if ( clear_page_mlock( struct page *page )
30 > tools/ 2583 const
31 > usr/ 2584 page constexpr
32 > virt/ 2585 if ( const_cast<type>( expression )
<nuxkernel 4.0 INSERT 1:memory. decltype( expression )
-- INSERT --
```

```
-apply to pud_range(st
-check_stack_guard pag
-check_sync_rss_stat(s
-check_sync_rss_stat(s
-clear_gigantic_page(s
-clear_huge_page(struc
-copy_one_pte(struct m
-copy_page_range(struc
-copy_pmd_range(struc
-copy_pte_range(struc
-copy_pud_range(struc
-copy_user_gigantic_pa
-copy_user_huge_page(s
-cow_user_page(struct
-disable_randmaps(char
-do_anonymous_page(str
-do_cow_fault(struct m
-do_fault(struct mm_st
-do_fault_around(struc
-do_numa_page(struct m
-do_page_mkwrite(struc
-do_read_fault(struct
-do_set_pte(struct vm_
-do_shared_fault(struc
-do_swap_page(struct m
-fault_around_bytes_ge
-fault_around_bytes_se
-fault_around_debugfs(
-follow_pfn(struct vm_
-follow_phys(struct vm_
-follow_pte(struct mm_
-free_pgdn_range(struc
Tagbar Name memory.c
```

## 插件一： vundle

- 可以在.vimrc中跟踪和管理插件，自动更新插件等
- 安装：
  - `git clone https://github.com/VundleVim/Vundle.vim.git ~/.vim/bundle/Vundle.vim`
- 在vimrc中添加 “Plugin xxx”
- 运行命令 “:PluginInstall”，就会从网络上下下载插件并安装。

《奔跑吧Linux内核》网络视频节目已正式上线  
<https://shop115683645.taobao.com/>

## 插件二： ctags

- ctags工具全称Generate tag files for source code，它扫描指定的源文件，找出其中包含的语法元素，并把找到的相关内容记录下来，这样在代码浏览和查找的时候就可以利用这些记录来方便实现查找和跳转功能。
- 使用如下命令可安装。
  - `sudo apt-get install ctags`
- 生成索引文件： `# ctags -R`
- 常用查找命令：
  - `Ctrl + ]` 跳转到光标处的函数或者变量的定义所在的地方。
  - `Ctrl + T` 返回到跳转之前的地方

## 插件三：cscope

- 安装： `sudo apt-get install cscope`
- 对源代码生成索引库： `cscope -Rbq`
- vim支持8种cscope的查询功能，如下：
  - ✓s: 查找C语言符号，即查找函数名、宏、枚举值等出现的地方
  - ✓g: 查找函数、宏、枚举等定义的位置，类似ctags所提供的功能
  - ✓d: 查找本函数调用的函数
  - ✓c: 查找调用本函数的函数
  - ✓t: 查找指定的字符串
  - ✓e: 查找egrep模式，相当于egrep功能，但查找速度快多了
  - ✓f: 查找并打开文件，类似vim的find功能
  - ✓i: 查找包含本文件的文件

## 插件四：Tagbar

- tagbar插件可以把源代码文件生成一个大纲，包括类、方法、变量以及函数名等，可以选中快速跳转到目标位置
- 在.vimrc文件中添加：
  - `Plugin 'majutsushi/tagbar' " Tag bar"`
- 重启vim，输入然后运行命令“`:PluginInstall`”完成安装

《奔跑吧Linux内核》视频节目已经上线  
<https://shop115683645.taobao.com/>

## 插件⑤：NeredTree

- nerdtree插件可以显示树形目录。、
- 安装nerdtree插件，在.vimrc文件中添加：
  - Plugin 'scrooloose/nerdtree'
- 重启vim，输入然后运行命令“:PluginInstall”完成安装。





## 如何使用vim阅读内核代码？

- 在vimrc中配置vim启动的时候自动加载tags和cscope的索引库文件
- 使用make 命令来生成ctags和cscope，下面以ARM vexpress平台为例：
  - ✓ #export ARCH=arm
  - ✓ #export SUBARCH=arm
  - ✓ #export CROSS\_COMPILE=arm-linux-gnueabi-
  - ✓ #make vexpress\_defconfig
  - ✓ #make tags cscope TAGS //生成tags,cscope, TAGS等索引文件

Enjoy VIM!

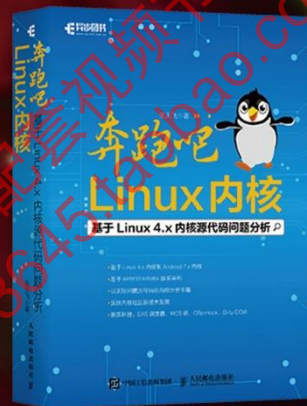
《奔跑吧Linux内核》配套视频节目已经上线  
<https://shop115683640.taobao.com/>

# 奔跑吧linux内核教学视频

旗舰篇一次订阅，持续更新

配套视频 **旗舰篇**

第**1**季  
内存管理



官方淘宝店



微信公众号

课程优势

1. 最有深度和广度的 Linux 内核视频
2. 手把手解读 Linux 内核代码
3. 紧跟 Linux 内核社区技术热点
4. 一键订阅，持续更新
5. 图书 + 视频，全新学习模式
6. 笨叔叔的 VIP 私密微信群答疑

第一季内存管理旗舰篇课程目录

课程名称	时长	课程名称	时长
序言一：Linux内核学习方法论	0:09:13	奔跑2.4.2Linux内核中的伙伴系统和碎片化	0:11:14
序言二：学习前准备		奔跑2.4.3Linux的页面分配器	0:21:37
序言2.1Linux发行版和开发板的选择	0:13:56	2.5slab分配器	
序言2.2搭建Qemu+gdb单步调试内核	0:13:51	奔跑2.5.1slab原理和核心数据结构	0:18:36
序言2.3搭建Eclipse图形化调试内核	0:10:59	奔跑2.5.2Linux内核中slab机制的实现	0:16:56
实战运维1：查看系统内存信息的工具（一）	0:20:19	2.6vmalloc分配	
实战运维2：查看系统内存信息的工具（二）	0:16:32	奔跑2.6vmalloc分配	0:15:48
实战运维3：读懂内核log中的内存管理信息	0:25:35	2.7VMA操作	
实战运维4：读懂procmeminfo	0:27:59	奔跑2.7VMA操作	0:16:42
实战运维5：Linux运维能力进阶路线图	0:09:40	2.8malloc分配器	
实战运维6：Linux内存管理参数调优（一）	0:19:46	奔跑2.8.1malloc的三个迷惑	0:17:41
实战运维7：Linux内存管理参数调优（二）	0:31:20	奔跑2.8.2内存管理的三个重要的函数	0:17:38
实战运维8：Linux内存管理参数调优（三）	0:22:58	2.9mmap分析	
运维高级如何单步调试RHEL—CENTOS7的内核一	0:15:45	奔跑2.9mmap分析	0:23:14
运维高级如何单步调试RHEL—CENTOS7的内核二	0:41:28	2.10缺页中断处理	
vim:打造比sourceinsight更强更好用的IDE（一）	0:24:58	奔跑2.10.1缺页中断一	0:31:07
vim:打造比sourceinsight更强更好用的IDE（二）	0:20:28	奔跑2.10.2缺页中断二	0:16:58
vim:打造比sourceinsight更强更好用的IDE（三）	0:23:25	2.11page数据结构	
实战git项目和社区patch管理		奔跑2.11page数据结构	0:29:41
2.0Linux内存管理背景知识介绍		2.12反向映射机制	
奔跑2.0.0内存管理硬件知识	0:15:25	奔跑2.12.1反向映射机制的背景介绍	0:19:01
奔跑2.0.1内存管理总览一	0:23:27	奔跑2.12.2RMAP四部曲	0:07:31
奔跑2.0.2内存管理总览二	0:07:35	奔跑2.12.3手撕Linux2.6.11上的反向映射机制	0:07:35
奔跑2.0.3内存管理常用术语	0:09:49	奔跑2.12.4手撕Linux4.x上的反向映射机制	0:10:08
奔跑2.0.4内存管理究竟管些什么东西	0:28:02	2.13回收页框	
奔跑2.0.5内存管理代码框架导读	0:38:09	奔跑2.13页面回收一	0:16:07
2.1Linux内存初始化		奔跑2.13页面回收二	0:11:41
奔跑2.1.0DDR简介	0:06:47	2.14匿名页面的生命周期	制作中会更新
奔跑2.1.1物理内存三大数据结构	0:19:39	2.15页面迁移	制作中会更新
奔跑2.1.2物理内存初始化	0:11:13	2.16内存规整	制作中会更新
奔跑2.1内存初始化之代码导读一	0:43:54	2.17KSM	制作中会更新
奔跑2.1内存初始化之代码导读二	0:23:31	2.18DirtyCOW内存漏洞	制作中会更新
奔跑2.1代码导读C语言部分（一）	0:27:34	2.19内存数据结构和API总结	制作中会更新
奔跑2.1代码导读C语言部分（二）	0:21:28	2.20Meltdown漏洞分析	
2.2页表的映射过程		奔跑2.20.1Meltdown背景知识	0:10:13
奔跑2.2.0ARM32页表的映射	0:08:54	奔跑2.20.2CPU体系结构之指令执行	0:11:25
奔跑2.2.1ARM64页表的映射	0:10:58	奔跑2.20.3CPU体系结构之乱序执行	0:11:03
奔跑2.2.2页表映射例子分析	0:11:59	奔跑2.20.4CPU体系结构之异常处理	0:03:48
奔跑2.2.3ARM32页表映射那些奇葩的事	0:09:42	奔跑2.20.5CPU体系结构之cache	0:10:56
2.3内存布局图		奔跑2.20.6进程地址空间和页表及TLB	0:17:39
奔跑2.3.1内存布局	0:10:35	奔跑2.20.7Meltdown漏洞分析	0:06:04
奔跑2.3.2内存布局二	0:13:30	奔跑2.20.8Meltdown漏洞分析之x86篇	0:12:07
2.4分配物理内存		奔跑2.20.9ARM64上的KPTI解决方案	0:25:39
奔跑2.4.1伙伴系统原理	0:10:10	2.21spectre漏洞分析	制作中会更新
		2.22异构内存管理	制作中会更新
		2.23Hugepage巨页	制作中会更新
		2.24运维人员必会的内存调优	制作中会更新
		2.25实战内存漏洞	制作中会更新
		2.26从内存管理代码中学会的优化技巧	制作中会更新

后期课程不定期更新中，等您来提交topic

截至18年5月已录制完成约20小时，后续精彩视频不断

规划中

第二季

虚拟化

第三季

Linux 内核和应用开发调试必杀技

第四季

进程管理和调度 / 中断 / 锁等

第五季

红帽系列



震撼上线

# 全书配套视频

第1季 内存管理篇



更多高清和精彩节目尽在淘宝：  
[shop115683645.taobao.com](https://shop115683645.taobao.com)