RHCSA 系列（四）：编辑文本文件及分析文本

2015-9-16 10:11 评论: 1 收藏: 4

原文：http://www.tecmint.com/rhcsa-exam-how-to-use-nano-vi-editors/
译文：LCTT https://linux.cn/article-6229-1.html

作者： Gabriel Cánepa
译者： FSSlc

作为系统管理员的日常职责的一部分，每个系统管理员都必须处理文本文件，这包括编辑已有文件（大多可能是配置文件），或创建新的文件。有这样一个说法，假如你想在 Linux 世界中挑起一场圣战，你可以询问系统管理员们，什么是他们最喜爱的编辑器以及为什么。在这篇文章中，我们并不打算那样做，但我们将向你呈现一些技巧，这些技巧对使用两款在 RHEL 7 中最为常用的文本编辑器: nano（由于其简单和易用，特别是对于新手来说）和 vi/m（由于其自身的几个特色使得它不仅仅是一个简单的编辑器）来说都大有裨益。我确信你可以找到更多的理由来使用其中的一个或另一个，或许其他的一些编辑器如 emacs 或 pico。这完全取决于你自己。

RHCSA: 使用 Nano 和 Vim 编辑文本文件 – Part 4

使用 Nano 编辑器来编辑文件

要启动 nano，你可以在命令提示符下输入 `nano` ，或可选地跟上一个文件名（在这种情况下，若文件存在，它将在编辑模式中被打开）。若文件不存在，或我们省略了文件名， nano 也将在编辑模式下开启，但将为我们开启一个空白屏以便开始输入：

Nano 编辑器

正如你在上一张图片中所见的那样， nano 在屏幕的底部呈现出一些可以通过指定的快捷键来触发的功能（^，即插入记号，代指 Ctrl 键)。它们中的一些是：

- Ctrl + G: 触发一个帮助菜单，带有一个关于功能和相应的描述的完整列表；

Nano 编辑器帮助菜单

- Ctrl + O: 保存更改到一个文件。它可以让你用一个与源文件相同或不同的名称来保存该文件，然后按 Enter 键来确认。

Nano 编辑器的保存更改模式

- Ctrl + X: 离开当前文件，假如更改没有被保存，则它们将被丢弃；
- Ctrl + R: 通过指定一个完整的文件路径，让你选择一个文件来将该文件的内容插入到当前文件中；

Nano: 插入文件内容到主文件中

上图的操作将把 `/etc/passwd` 的内容插入到当前文件中。

- Ctrl + K: 剪切当前行；
- Ctrl + U: 粘贴；

- Ctrl + C: 取消当前的操作并返回先前的屏幕；

为了轻松地在打开的文件中浏览， nano 提供了下面的功能：

- Ctrl + F 和 Ctrl + B 分别先前或向后移动光标；而 Ctrl + P 和 Ctrl + N 则分别向上或向下移动一行，功能与箭头键相同；
- Ctrl + space 和 Alt + space 分别向前或向后移动一个单词；

最后，

- 假如你想将光标移动到文档中的特定位置，使用 Ctrl + _ (下划线) 并接着输入 X,Y 将准确地带你到 第 X 行，第 Y 列。

在 nano 中定位到具体的行和列

上面的例子将带你到当前文档的第 15 行，第 14 列。

假如你可以回忆起你早期的 Linux 岁月，特别是当你刚从 Windows 迁移到 Linux 中，你就可能会同意：对于一个新手来说，使用 nano 来开始学习是最好的方式。

使用 Vim 编辑器来编辑文件

Vim 是 vi 的加强版本，它是 Linux 中一个著名的文本编辑器，可在所有兼容 POSIX 的 *nix 系统中获取到，例如在 RHEL 7 中。假如你有机会并可以安装 Vim，请继续；假如不能，这篇文章中的大多数（若不是全部）的提示也应该可以正常工作。

Vim 的一个出众的特点是可以在多个不同的模式中进行操作：

- Command Mode 命令模式 将允许你在文件中跳转和输入命令，这些命令是由一个或多个字母组成的简洁且大小写敏感的组合。假如你想重复执行某个命令特定次数，你可以在这个命令前加上需要重复的次数（这个规则只有极少数例外）。例如， `yy` （或 `Y` ，yank 的缩写）可以复制整个当前行，而 `4yy` （或 `4Y` ）则复制整个从当前行到接下来的 3 行(总共 4 行)。
- 我们总是可以通过敲击 `Esc` 键来进入命令模式（无论我们正工作在哪个模式下）。
- Ex Mode 末行模式 中，你可以操作文件（包括保存当前文件和运行外部的程序或命令）。要进入末行模式，你必须从命令模式中（换言之，输入 `Esc` + `:`)输入一个冒号(`:`)，再直接跟上你想使用的末行模式命令的名称。
- Insert Mode 插入模式 ，可以输入字母 `i` 进入，然后只需要输入文字即可。大多数的击键结果都将出现在屏幕中的文本中。

现在，让我们看看如何在 vim 中执行在上一节列举的针对 nano 的相同的操作。不要忘记敲击 Enter 键来确认 vim 命令。

为了从命令行中获取 vim 的完整手册，在命令模式下键入 `:help` 并敲击 Enter 键：

vim 编辑器帮助菜单

上面的部分呈现出一个内容列表，这些定义的小节则描述了 Vim 的特定话题。要浏览某一个小节，可以将光标放到它的上面，然后按 `Ctrl + ]` （闭方括号）。注意，底部的小节展示的是当前文件的内容。

1、 要保存更改到文件，在命令模式中运行下面命令中的任意一个，就可以达到这个目的：

```
:wq!
:x!
ZZ (是的，两个 ZZ,前面无需添加冒号)
```

2、 要离开并丢弃更改，使用 `:q!` 。这个命令也将允许你离开上面描述过的帮助菜单，并返回到命令模式中的当前文件。

3、 剪切 N 行：在命令模式中键入 `Ndd` 。

4、 复制 M 行：在命令模式中键入 `Myy` 。

5、 粘贴先前剪贴或复制过的行：在命令模式中按 `P` 键。

6、 要插入另一个文件的内容到当前文件：

```
:r filename
```

例如，插入 `/etc/fstab` 的内容，可以这样做：

在 vi 编辑器中插入文件的内容 <http://www.tecmint.com/wp-content/uploads/2015/03/Insert-Content-vi-Editor.png>

在 vi 编辑器中插入文件的内容

7、 插入一个命令的输出到当前文档：

```
:r! command
```

例如，要在光标所在的当前位置后面插入日期和时间：

在 vi 编辑器中插入时间和日期

在另一篇我写的文章中，(**LFCS 系列（二） <http://www.tecmint.com/vi-editor-usage/>**)，我更加详细地解释了在 vim 中可用的键盘快捷键和功能。或许你可以参考那个教程来查看如何使用这个强大的文本编辑器的更深入的例子。

使用 grep 和正则表达式来分析文本

到现在为止，你已经学习了如何使用 nano 或 vim 创建和编辑文件。打个比方说，假如你成为了一个文本编辑器忍者 – 那又怎样呢？ 在其他事情上，你也需要知道如何在文本中搜索正则表达式。

正则表达式（也称为 "regex" 或 "regexp"） 是一种识别一个特定文本字符串或模式的方式，使得一个程序可以将这个模式和任意的文本字符串相比较。尽管利用 grep 来使用正则表达式值得用一整篇文章来描述，这里就让我们复习一些基本的知识：

1、 最简单的正则表达式是一个由数字和字母构成的字符串（例如，单词 "svm"） ，或者两个（在使用两个字符串时，你可以使用 `|` (或) 操作符）：

```
# grep -Ei 'svm|vmx' /proc/cpuinfo
```

上面命令的输出结果中若有这两个字符串之一的出现，则标志着你的处理器支持虚拟化:

正则表达式示例

2、 第二种正则表达式是一个范围列表，由方括号包裹。

例如， `c[aeiou]t` 匹配字符串 cat、cet、cit、cot 和 cut，而 `[a-z]` 和 `[0-9]` 则相应地匹配小写字母或十进制数字。假如你想重复正则表达式 X 次，在正则表达式的后面立即输入 `{X}` 即可。

例如，让我们从 `/etc/fstab` 中析出存储设备的 UUID：

```
# grep -Ei '[0-9a-f]{8}-([0-9a-f]{4}-){3}[0-9a-f]{12}' -o /etc/fstab
```

从一个文件中析出字符串

方括号中的第一个表达式 `[0-9a-f]` 被用来表示小写的十六进制字符， `{8}` 是一个量词，暗示前面匹配的字符串应该重复的次数（在一个 UUID 中的开头序列是一个 8 个字符长的十六进制字符串）。

在圆括号中，量词 `{4}` 和连字符暗示下一个序列是一个 4 个字符长的十六进制字符串，接着的量词 `({3})` 表示前面的表达式要重复 3 次。

最后，在 UUID 中的最后一个 12 个字符长的十六进制字符串可以由 `[0-9a-f]{12}` 取得， `-o` 选项表示只打印出在 `/etc/fstab` 中匹配行中的匹配的(非空)部分。

3、 POSIX 字符类

字符类	匹配 ...
[:alnum:]	任意字母或数字 [a-zA-Z0-9]
[:alpha:]	任意字母 [a-zA-Z]
[:blank:]	空格或制表符
[:cntrl:]	任意控制字符 (ASCII 码的 0 至 32)
[:digit:]	任意数字 [0-9]
[:graph:]	任意可见字符
[:lower:]	任意小写字母 [a-z]
[:print:]	任意非控制字符
[:space:]	任意空格
[:punct:]	任意标点字符
[:upper:]	任意大写字母 [A-Z]
[:xdigit:]	任意十六进制数字 [0-9a-fA-F]
[:word:]	任意字母，数字和下划线 [a-zA-Z0-9_]

例如，我们可能会对查找已添加到我们系统中给真实用户的 UID 和 GID（参考“RHCSA 系列（二）：如何进行文件和目录管理 <https://linux.cn/article-6155-1.html> ”来回忆起这些知识）感兴趣。那么，我们将在 `/etc/passwd` 文件中查找 4 个字符长的序列：

```
# grep -Ei [[:digit:]]{4} /etc/passwd
```

在文件中查找一个字符串

上面的示例可能不是真实世界中使用正则表达式的最好案例，但它清晰地启发了我们如何使用 POSIX 字符类来使用 grep 分析文本。

总结

在这篇文章中，我们已经提供了一些技巧来最大地利用针对命令行用户的两个文本编辑器 nano 和 vim，这两个工具都有相关的扩展文档可供阅读，你可以分别查询它们的官方网站(链接在下面给出)以及使用“RHCSA 系列（一）：回顾基础命令及系统文档 <https://linux.cn/article-6133-1-rel.html> ”中给出的建议。

参考文件链接

- http://www.nano-editor.org/ <http://www.nano-editor.org/>
- http://www.vim.org/ <http://www.vim.org/>

---

via: http://www.tecmint.com/rhcsa-exam-how-to-use-nano-vi-editors/ <http://www.tecmint.com/rhcsa-exam-how-to-use-nano-vi-editors/>

作者：Gabriel Cánepa <http://www.tecmint.com/author/gacanepa/> 译者：FSSIc <https://github.com/FSSIc> 校对：wxy <https://github.com/wxy>

本文由 LCTT <https://github.com/LCTT/TranslateProject> 原创翻译，Linux中国 <file:///root/github/my-notes/Manual/Linux.cn/RHCSA/RHCSA%E7%B3%BB%E5%88%974%E7%BC%96%E8%BE%91%E6%96%87%E6%9C%AC%E6%96%87%E4%BB%B6%E5%8F%8A%E5%88%86%E6%9E%90%E6%96%87%E6%9C%AC-%E6%8A%80%E6%9C%AF%20.html> 荣誉推出

原文：http://www.tecmint.com/rhcsa-exam-how-to-use-nano-vi-editors/ <http://www.tecmint.com/rhcsa-exam-how-to-use-nano-vi-editors/>

作者： Gabriel Cánepa

译文：LCTT <http://lctt.github.io/> https://linux.cn/article-6229-1.html <https://linux.cn/article-6229-1.html> 译者： FSSlc

发表评论

验证码 换一个 评论

热点评论

来自上海的 QQ Browser 9.1|Windows 7 用户 2015-9-17 10:16

圣战、、、、

赞 6

体验环境

本文导航

相关阅读

RHCSA

- RHCSA 系列（三）：如何管理 RHEL7 的用户和组 2015-9-10
- RHCSA 系列（五）：RHEL7 中的进程管理：开机，关机 2015-9-18
- RHCSA 系列（八）：加固 SSH，设定主机名及启用网络服务 2015-9-23
- RHCSA 系列（九）：安装、配置及加固一个 Web 和 FTP 服务器 2015-9-24
- RHCSA 系列（十）：Yum 包管理、Cron 自动任务计划和监控系统日志 2015-9-26
- RHCSA 系列（十一）：使用 firewalld 和 iptables 来控制网络流量 2015-9-29