

17 配置Nano和终端

更新时间：2019-07-04 18:13:18



“

我们有力的道德就是通过奋斗取得物质上的成功；这种道德既适用于国家，也适用于个人。

——罗素

”

内容简介

1. 前言
2. 通过 `.nanorc` 来配置 Nano
3. 通过 `.bashrc` 配置终端
4. 总结

1. 前言

上一课Nano，初学者的文本编辑器很简单。这一课比上一课略微难一些，不过还是保持轻松。

这一课我们学习用 Nano 编辑器来编辑一些配置文件，例如配置 Nano 本身，还有配置终端。

2. 通过 `.nanorc` 来配置 Nano

现在，既然我们已经初步了解了如何使用 Nano（当然了，要熟练使用还需要多实践咯）。

我们也看到，Nano 实在很易用，只需要熟悉一些常用的键盘快捷键组合就可以了。

在上一课的开始处，我们说了要学习用文本编辑器修改一些配置文件。

Nano 也有一个配置文件，用于设置 Nano 的一些选项。这个文件叫做 `.nanorc`

注意，`.nanorc` 的最前面有一个点，表明这是一个隐藏文件。像这样的配置文件，如果用 `ls -l` 命令是列不出来的，需要用 `ls -a` 来列出。

一般 Linux 中的配置文件大多以点开头，而且多以 `rc` 结尾。比如 Vim 的配置文件 `.vimrc`；Bash Shell 的配置文件 `.bashrc` 等等。

那这个 `rc` 是什么意思呢？当然了，不理解 `rc` 什么意思也没有关系，不过我们来满足一下自己的好奇心：

Linux 或 Unix 的许多程序在启动时，都需要 `rc` 后缀的初始文件或配置文件。

`rc`，它是 `runcomm` 的缩写，即 `run command`（运行命令）的简写。`rc` 是取自 `runcom`，来自麻省理工学院在 1965 年发展的 `CTSS` 系统。相关文献曾记载这一段话：“具有从档案中取出一系列命令来执行的功能；这称为 `run commands`，又称为 `runcom`，而这种档案又称为一个 `runcom (a runcom)`。”

`rc` 是很多脚本类文件的后缀，这些脚本通常在程序的启动阶段被调用，通常是 Linux 系统启动时。如 `.bashrc` 是当 Linux 的 Bash Shell（之后的课程会学习这种脚本）启动后所运行的脚本。

每个 Linux 的用户都可以在自己的家目录创建 `.nanorc` 这个文件，在每次 `nano` 启动前，它会读取此配置文件。

我的用户名是 `oscar`，所以我的 `.nanorc` 文件应该是 `/home/oscar/.nanorc`

但是，我查找了，在我的家目录中，并没有 `.nanorc` 这个文件。在你的情况，可能有，也可能没有。但是不要紧。因为如果在你的家目录没有 `.nanorc`，那么 `nano` 会用全局的配置文件。

创建 `.nanorc`

如果你的家目录里也没有 `.nanorc`，那么你可以创建一个。怎么创建呢？很简单：

```
nano .nanorc
```

在这个 `.nanorc` 文件中，你可以输入你的配置信息。

每一行一句配置语句，配置语句是以 `set`（用于激活。`set` 是英语“放置，设置”的意思）或 `unset`（用于关闭）开头，后接你要配置的项目。例如：

```
set mouse
```

这句话就用于激活鼠标（`mouse` 是英语“鼠标”的意思）。有了这句话，那么每次 `Nano` 启动时都会激活鼠标操作了，我们启动 `Nano` 就不必写 `-m` 这个参数了，是不是很方便？

我们也可如法炮制，使得我们不用每次启动 `Nano` 都加上 `-i` 和 `-A` 参数：

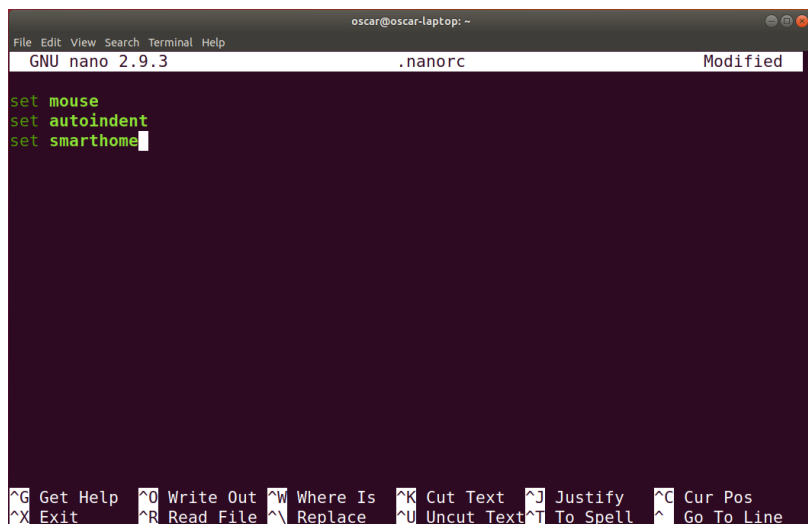
```
set autoindent
```

这句是用于激活自动缩进，相当于 `-i` 参数的作用。

```
set smarthome
```

这句用于激活智能 Home 键，相当于 `-A` 参数的作用。

如下图所示：



```
oscar@oscar-laptop: ~
GNU nano 2.9.3 .nanorc Modified

set mouse
set autoindent
set smarthome

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace  ^U Uncut Text ^T To Spell  ^_ Go To Line
```

如果要保存文件，只要按下 `Ctrl+O`，它会提示你文件名是 `.nanorc`。因为我们已经指定了文件名，直接按下回车，这三行就写入到了 `.nanorc` 中。可以看到终端提示：`[Wrote 3 lines]`，表示“写入了 3 行”。

如果你完成了配置，那么可以按下 `Ctrl+X` 来退出 Nano。

下次你再启动 Nano 的时候，你会发现：鼠标被激活了，自动缩进也激活了，智能 Home 键也激活了。

配置文件可以大大提高我们的工作效率。

全局的 `nanorc` 和语法高亮

在每个用户的家目录中的 `.nanorc` 这个文件非常实用，因为它可以帮助你设置自己的 Nano 选项。

但是，如果你的 Linux 系统中有几十个用户，你想要为所有这些用户都激活 Nano 的鼠标操作，难道你要登录每一个用户的账户，然后在他们各自的 `.nanorc` 中添加 `set mouse` 这句话么？那上百个用户呢？岂不是要累坏了。

Linux 系统的开发者早就想到了这一点。事实上，Nano 有一个全局的配置文件，是为系统上所有用户所公共调用的，也叫 `nanorc`，但是在 `/etc` 中，是 `/etc/nanorc`。这回 `nanorc` 前面没有点了。

这个全局的 Nano 配置文件只能被 root 用户修改，因为是在系统文件夹 `/etc` 中。

在我们以前的课程中，我们介绍过 `/etc` 这个目录，它里面存放系统的配置文件。一般为所有用户共用。

因此，如果我们要修改这个文件，建议用 `sudo` 命令：

```
sudo nano /etc/nanorc
```

输入你的用户密码，就打开了这个配置文件，如下图：

```
oscar@oscar-laptop: ~
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/nanorc

## Sample initialization file for GNU nano.
##
## Please note that you must have configured nano with --enable-nanorc
## for this file to be read! Also note that this file should not be in
## DOS or Mac format, and that characters specially interpreted by the
## shell should not be escaped here.
##
## To make sure an option is disabled, use "unset <option>".
##
## For the options that take parameters, the default value is given.
## Other options are unset by default.
##
## Quotes inside string parameters don't have to be escaped with
## backslashes. The last double quote in the string will be treated as
## its end. For example, for the "brackets" option, ""')>]}" will match
## ", ', ), >, ], and }".

## When soft line wrapping is enabled, make it wrap lines at blanks
## (tabs and spaces) instead of always at the edge of the screen.
[ Read 270 lines ]

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

可以看到这个配置文件内容就很多了，当然也有很多是注释，也就是以 # 开头的。你可以向下滚动这个文件，会发现现有 270 行（我的情况）。

在这个配置文件里，有所有可以放置在你自己的 .nanorc 中的语句，比如 set autoindent。但是这些配置语句都是以 # 开头，就是说默认是注释掉的，也就是在全局说来，这些配置语句不生效。如果你在 /etc/nanorc 中把那些配置语句前面的 # 去掉，就会对全局用户生效了。

```
oscar@oscar-laptop: ~
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/nanorc

## Use auto-indentation.
# set autoindent

## Back up files to the current filename plus a tilde.
# set backup

## The directory to put unique backup files in.
# set backupdir ""

## Use bold text instead of reverse video text.
# set boldtext

## The characters treated as closing brackets when justifying
## paragraphs. They cannot contain blank characters. Only closing
## punctuation, optionally followed by closing brackets, can end
## sentences.
# set brackets ""')>]}"

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

按 Ctrl+O 来保存修改，然后 Ctrl+X 退出。当然，也可以直接 Ctrl+X，它问你是否保存修改，输入 y（表示 yes，“是”）或 n（表示 no，“不是”）或 Ctrl+C 取消。

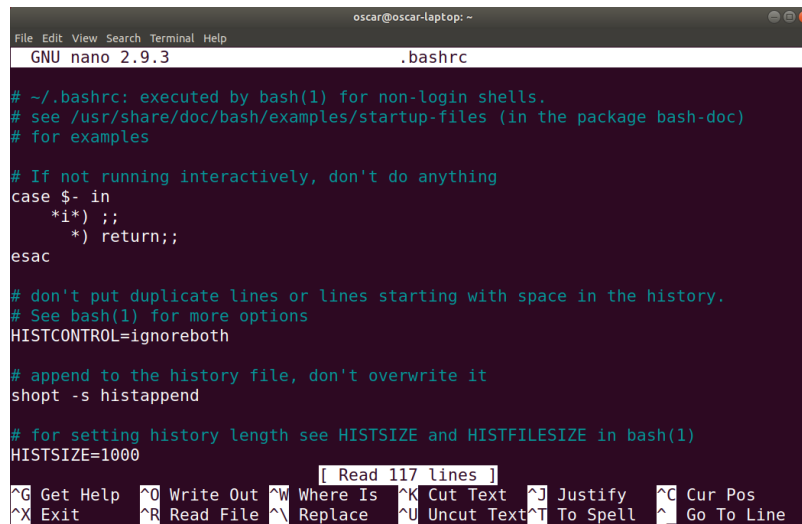
3. 通过 .bashrc 配置终端

对于 Nano，我们有一个配置文件叫 .nanorc。其实对于我们的终端，也有一个配置文件，叫做 .bashrc，这是用户个人的终端配置文件。在我的情况，位于 /home/oscar/.bashrc。

这个文件一般来说是默认存在的。不像我们的 .nanorc 可能还要自己创建。

我们打开家目录下的终端配置文件看看，只要输入以下命令：

```
nano ~/.bashrc
```

A screenshot of a terminal window on a Linux system. The window title is 'oscar@oscar-laptop: ~'. The editor is 'GNU nano 2.9.3' editing the '.bashrc' file. The content of the file is as follows:

```
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# If not running interactively, don't do anything
case $- in
  *i*) ;;
  *) return;;
esac

# don't put duplicate lines or lines starting with space in the history.
# See bash(1) for more options
HISTCONTROL=ignoreboth

# append to the history file, don't overwrite it
shopt -s histappend

# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000
```

The bottom status bar of the nano editor shows: '[Read 117 lines]', '^G Get Help', '^O Write Out', '^W Where Is', '^K Cut Text', '^J Justify', '^C Cur Pos', '^X Exit', '^R Read File', '^_ Replace', '^U Uncut Text', '^T To Spell', and '^_ Go To Line'.

.bashrc 文件比较复杂，初看可能会有点眼花缭乱的感觉。我们暂时不会深入学习它的语法，上面我们说了 .bashrc 是 Bash 的配置文件，而 Bash 是一种 Shell。

我们以后的课程会重点来学习 Shell。暂时只要了解 Shell 是外壳程序（Shell 是英语“外壳”的意思）是用于解释我们输入终端的各种命令。

Shell 是一个用户跟操作系统之间的一个命令解释器，也就是用户与 Linux 操作系统之间沟通的桥梁。

Bash 是最常用的一种 Shell 程序，Ubuntu 和大部分常见的 Linux 发行版默认的 Shell 程序就是 Bash。macOS 的默认 Shell 也是 Bash。

.bashrc 就是 Bash 这个 Shell 程序的配置文件。

所以 bashrc 本身的语法也是 Bash 的语法，是一种脚本语言。

我们在之前的课程中讲过，我们可以通过配置文件来修改我们的命令行提示符：

```
oscar@oscar-laptop:~$
```

上面是我的用户 oscar 目前的命令行提示符。你的情况肯定和我不一样，我们在以前的课程中也解释过命令行提示符各个部分的含义。

在 .bashrc 文件中，我们可以修改命令行提示符的样式，如果你觉得目前的命令行提示符太繁琐太长了，你可以把它改短一些、简洁一些。

把我们的 .bashrc 文件向下拉，可以看到有好几行类似这样的：

```
oscar@oscar-laptop: ~
File Edit View Search Terminal Help
GNU nano 2.9.3 .bashrc

# (ISO/IEC-6429). (Lack of such support is extremely rare, and such
# a case would tend to support self rather than setaf.)
color_prompt=yes
else
color_prompt=
fi
fi

if [ "$color_prompt" = yes ]; then
PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[$
else
PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
fi
unset color_prompt force_color_prompt

# If this is an xterm set the title to user@host:dir
case "$TERM" in
xterm*|rxvt*)
PS1="\[\e]0;${debian_chroot:+($debian_chroot)}\u@\h: \w\a\]$PS1"
*)
;;
esac

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

上图中，我们看到有 PS1 开头的行，这个就是设置命令行提示符样式的。

\u 表示用户名（u 是 user 的首字母，表示“用户”。例如我的用户名是 oscar），\h 是电脑的名称（h 是 hostname 的首字母，表示“主机名”，我的情况是 oscar-laptop），@ 就是分割用户名和电脑名的那个@号等等。

如果你学过 Shell 语法，那么你可以试着修改。

暂时我们不带大家修改了，因为比较复杂，对于初学者来说，不要把 .bashrc 文件搞乱了为好。

创建别名

别名是这样一些命令：在你运行时转换为其它命令。就好比我的英文名是 oscar，那别人呼叫 oscar 的时候，我知道其实是叫我。

别名的英语是 alias。如果我们向下查找 .bashrc 文件，会发现有 alias 开头的行，如下所示：

```
oscar@oscar-laptop: ~
File Edit View Search Terminal Help
GNU nano 2.9.3 .bashrc

alias ll='ls -aLF'
alias la='ls -A'
alias l='ls -CF'

# Add an "alert" alias for long running commands. Use like so:
# sleep 10; alert
alias alert='notify-send --urgency=low -i "${[ $? = 0 ]} && echo terminal || echo $'

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
. ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
```

也不难理解：

- 当我们在终端输入比如 ll（两个小写的 L），其实就是等同于 ls -aLF；
- 当我们在终端输入比如 la，其实就是等同于 ls -A；
- 等等。

所以说，别名设置得好，可以降低我们的工作量，因为输入 ll 总比输入 ll -aLF 简单吧。

在 `.bashrc` 中创建别名的语法是这样的：

```
alias name='command'
```

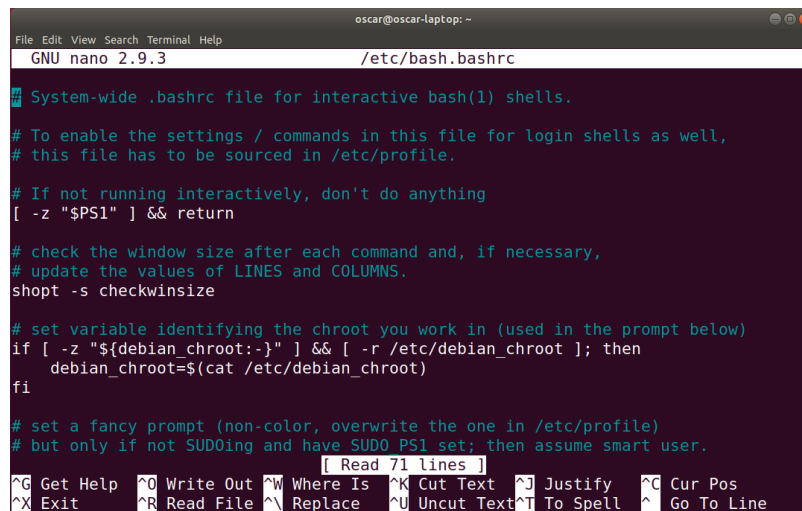
`name` 就是别名的名字，而 `command` 就是替换别名的实际终端命令。

全局的 `bashrc`

之前的 Nano 有全局的配置文件：`/etc/nanorc`。

我们的终端所有的 Bash 也有它的全局配置文件：`/etc/bash.bashrc`。我们用 Nano 来打开它看看：

```
sudo nano /etc/bash.bashrc
```



对于每个用户来说，家目录下的 `.bashrc` 文件的优先级比系统的 `/etc/bash.bashrc` 文件高。例如同样的配置选项，如果 `.bashrc` 和 `/etc/bash.bashrc` 不同，那么以 `.bashrc` 的为准。同样的原则也适用于其它配置文件，例如 `.nanorc` 和 `/etc/nanorc`。

profile 配置文件

在我们的家目录下，其实还有一个 `.profile` 文件，而且它也有对应的全局 `profile` 文件，是 `/etc/profile`：

```
nano .profile
```



```
oscar@oscar-laptop: ~
GNU nano 2.9.3 .profile

#!/usr/bin/env bash
# ~/.profile: executed by the command interpreter for login shells.
# This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
# exists.
# see /usr/share/doc/bash/examples/startup-files for examples.
# the files are located in the bash-doc package.

# the default umask is set in /etc/profile; for setting the umask
# for ssh logins, install and configure the libpam-umask package.
#umask 022

# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin if it exists
[ Read 27 lines ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

```
sudo nano /etc/profile
```

```
oscar@oscar-laptop: ~
GNU nano 2.9.3 /etc/profile

#!/usr/bin/env bash
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

if [ "${PS1:-}" ]; then
    if [ "${BASH:-}" ] && [ "$BASH" != "/bin/sh" ]; then
        # The file bash.bashrc already sets the default PS1.
        # PS1='\h:\w\$ '
        if [ -f /etc/bash.bashrc ]; then
            . /etc/bash.bashrc
        fi
    else
        if [ "`id -u`" -eq 0 ]; then
            PS1='# '
        else
            PS1='$ '
        fi
    fi
fi
[ Read 27 lines ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

profile 在英语中是“外观，轮廓”的意思。那么这个 profile 文件和 bashrc 有什么区别呢？

简单来说是这样的：

- profile 这个配置文件是用户登录的终端配置文件，也就是我们以前学过的 tty1~tty6 这 6 个命令行终端（没有图形界面的，分别通过 Ctrl + Alt + F1~F6 进入）。profile 是这些需要登录的、非图形界面的终端配置文件；
- bashrc 这个配置文件是无需用户登录的终端，也就是我们一直在使用的终端形式——图形化的终端情况。这种终端是读取 .bashrc 为配置文件的。

有一点需要记住：profile 文件会调用 .bashrc，所以其实我们修改了 .bashrc，也就是间接修改了 profile 文件。因为 profile 文件会用 profile 本身的配置再加上 .bashrc 的配置。

在我们修改了 .bashrc 和 profile 文件后，默认是在用户下次登录系统时才能生效。但是我们可以用 source 命令来使改动立即生效：

```
source .bashrc
```



```
source .profile
```

小结

我们可以用 Nano 来修改 `.bashrc` 这个终端的配置文件，也可以修改 `.nanorc` 这个 Nano 的配置文件。这样我们就可以定制我们的软件了。

今天的课就到这里，一起加油吧！



16 Nano , 初学者的文本编辑器

18 软件安装, 如虎添翼

