

效率为王：终端管理工具 Tmux

简介

Tmux 是一款非常实用的终端复用器，用来管理一个终端窗口中运行的多个终端会话。它通过会话，窗口，面板的形式高效且有序的管理你所有的工作终端。此外，它还可以通过将终端会话置于后台运行，在需要时按需接入，以及将会话共享给其他人，是远程办公和结对编程的利器。无论是前端还是后端开发工程师，运维人员，都值得将其加入个人的日常工具列表。

安装

在 MacOS 下，安装 tmux 非常简单，使用 [homebrew](#) 便可以安装最新的版本：

```
brew install tmux
```

对于Linux，大部分发行版都有打包 tmux，可以通过包管理器安装，比如在 Ubuntu 下，可以使用 apt 安装。

```
apt install tmux
```

在windows下想使用 tmux 有两种方法：

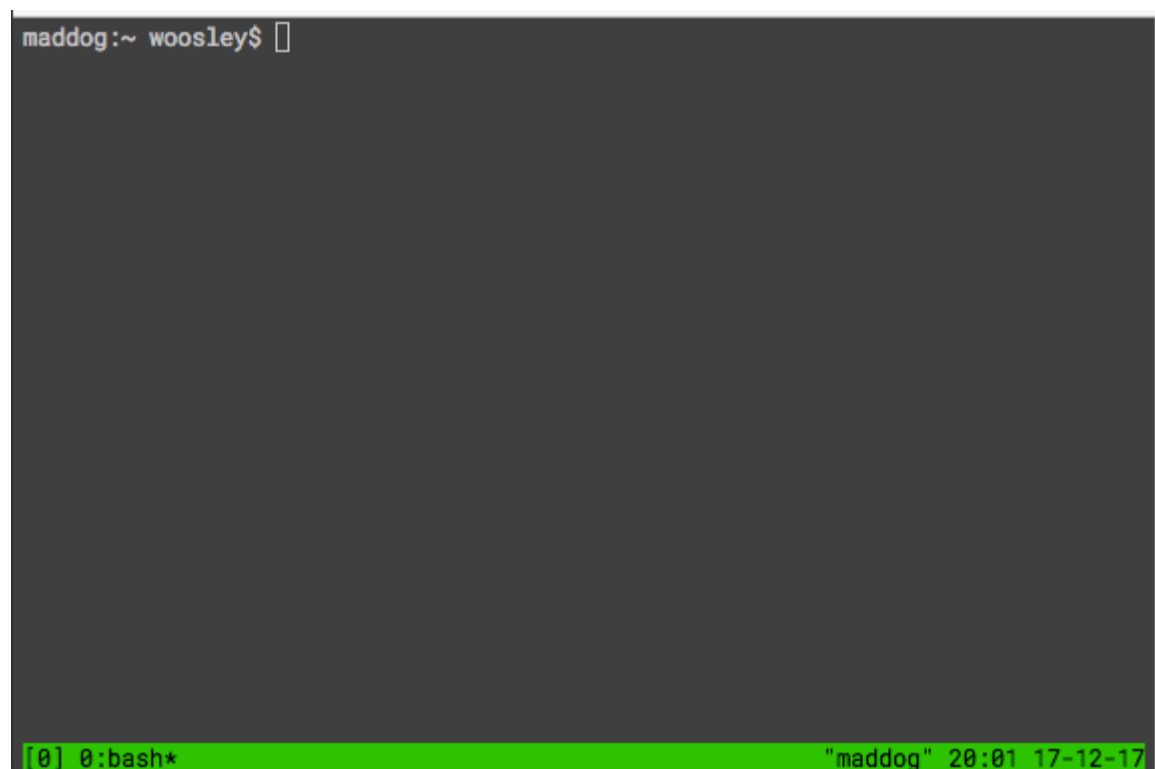
- window10 加入了 linux 子系统的功能，用户可以在windows下使用 Ubuntu linux。安装 tmux 的方法和原生 Ubuntu 完全一样。
- 对于 windows7 以及更加古老的版本，个人推荐使用 [Cygwin](#)，这是一个在windows下模拟 linux 的工具，提供了大部分*nix软件的安装，里面就包括了 tmux。

初次使用

初次使用tmux，只需要在终端下面键入命令：

```
tmux
```

默认情况下会启动一个新的会话（session）。这时候窗口显示如下：



```
maddog:~ woosley$ [0] 0: bash* "maddog" 20:01 17-12-17
```

可以看到 Tmux 在默认终端上面启动了一个新的界面。包括原来的 shell 窗口和下方的状态栏。

状态栏显示了当前 tmux session 的基本信息。

- [0] 代表当前 session 的名字
- 0: bash 代表当天 session 的第一个窗口，名字为 bash
- 其他部分为当前主机名，以及当前时间。

作为一款基于终端的工具，tmux 所有的操作都使用键盘快捷键来进行，熟悉了 tmux 的快捷键之后，我们可以把自己的双手从鼠标完全解放出来，对提高工作生产力有极大的作用。

tmux 的快捷键使用 Prefix + key 的形式。如果使用过 Gnu screen，那么对这种模式应该很熟悉，不同的是，Gnu screen 默认的 prefix 是 ctrl + a，而 tmux 的默认快捷键是 ctrl + b，按键方法为同时按下 ctrl 键和 b 键。这个组合是可以定制的，在本文中，我们用 prefix 来代表这个按键组合。

现在我们可以使用快捷键创建一个新的窗口，按下 prefix + c（同时按下 Ctrl + b，放开，然后按下 c）。效果如下图所示：

```
maddog:~ woosley$ [
[0] 0:bash- 1:bash* "maddog" 20:17 17-12-17
```

此时整个窗口没有太大的变化，只是下面的状态栏多了一个 **1:bash**，表示当前启动了两个窗口。

现在可以试着按下 `prefix + n`，看 `tmux` 如何在不同的窗口间转跳。

服务器和会话

`tmux` 本质上可以说是一个服务器，当 `tmux` 命令运行的时候，后台运行了一个 `tmux` 服务，并启动一个会话，会话和服务器之间通过 `Unix socket` 来通信。

默认情况下启动的 `tmux` 会话通过数字命名，比如第一个会话为 0。可以通过

```
tmux new -s session_name
```

来创建一个命名的 `tmux` 会话，比如 `tmux new -s test`。启动会话之后，`tmux` 自动连接到此会话之上。我们可以脱离会话，这样可以将会话里面运行的任务置于后台，在需要的时候重新连接。

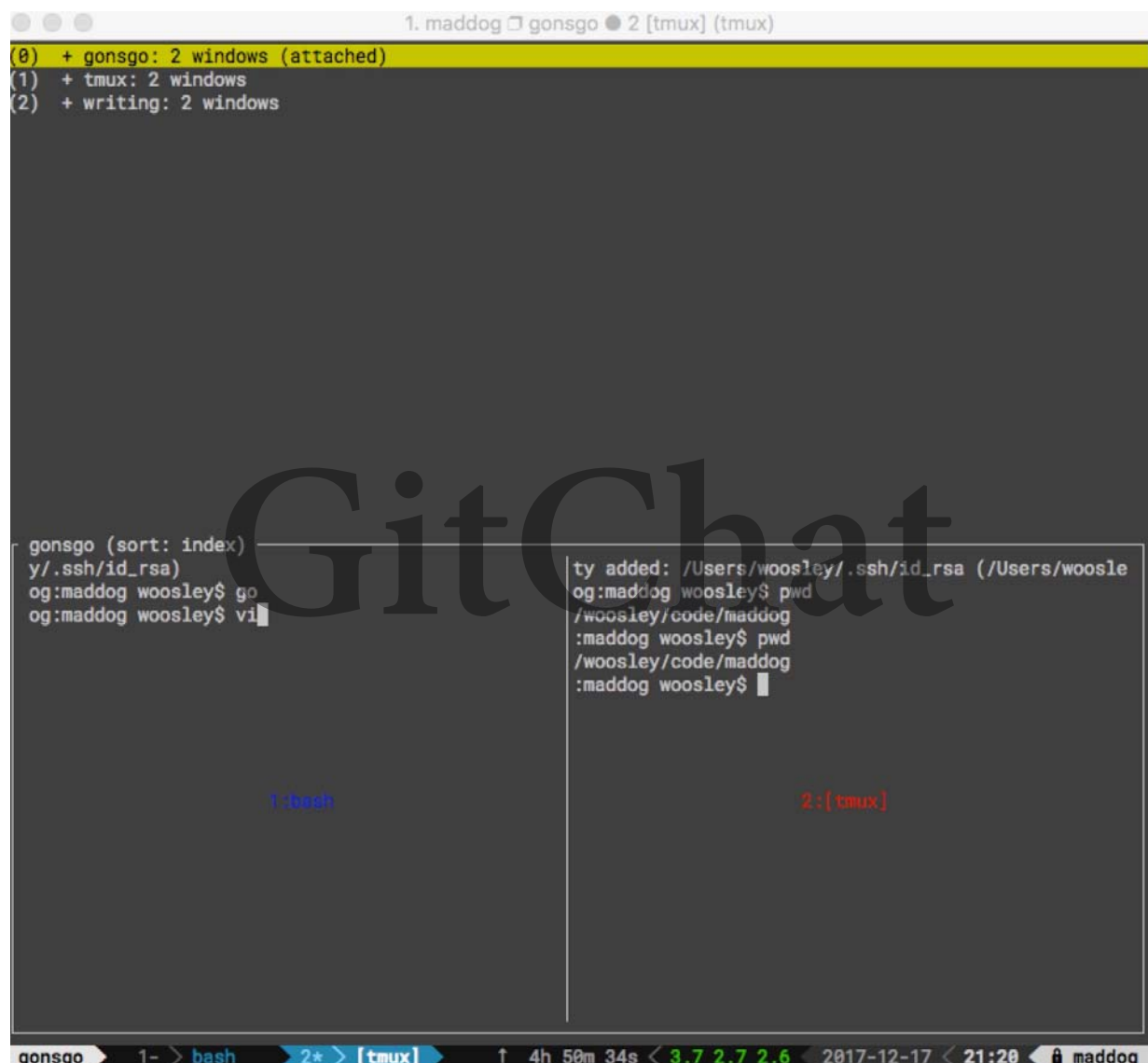
脱离会话的快捷键为 `prefix + d`，脱离会话之后系统回到之前的终端。用这种方法，我们可以方便地在远程主机上启动一个长期运行的 `tmux` 会话，运行我们想长期运行的程序。我们可以用 `tmux ls` 查看当前所有创建的 session，比如：

```
tmux ls
learn: 1 windows (created Sun Dec 17 21:03:20 2017) [80x24]
test: 1 windows (created Sun Dec 17 20:59:44 2017) [80x24]
```

这里我创建了两个 tmux 会话，名字分别为 test 和 learn。

重新连接tmux的命名为 `tmux attach`。在没有任何参数的情况下默认连接最新创建的会话。可以添加参数 `-t $name`，连接名字为 `$name` 的tmux 会话。比如 `tmux a -t learn`。同时这里显示了一个tmux的小技巧，很多tmux的命令可以缩写，这里将 `attach` 缩写成为了 `a`。

在 `tmux session` 里面，可以通过 `prefix + s` 选择并快速切换 tmux 会话，如下图所示。tmux 会弹出一个会话的选择列表，可以通过方向键选择我们想打开的会话。



如果在创建了会话之后想重命名当前会话，可以使用 `prefix + $`，在底部弹出的输入框里面输入想要的名字即可。这里也可以使用tmux的命令模式。按下 `prefix + :`，在输入框里输入 `:rename-session new-session` (支持 tab 补全)，便可以重命名当前的会话。

注意当服务器重启之后，tmux 的会话信息会丢失。要持久化保存 tmux 会话信息，在本文的 `tmux` 插件部分会介绍一款简单易用的插件 **tmux-resurrect**

窗口

之前我们演示了如何创建新的窗口，并使用 `prefix + n`，表示选择下一个窗口。在不同窗口之间移动，除了 `prefix + n` 之外，还可以用数字键，选择第 `N` 个窗口，注意 `tmux` 中窗口的序号是从 0 开始，因此 `prefix + 1` 表示选择第二个窗口。`prefix + p` 表示跳转到前一个窗口。

我们还可以使用快捷键 `prefix + w` 来弹出一个虚拟的窗口列表，然后使用方向键来选择所需要打开的窗口。

使用 `prefix + ,`，可以用来重命名当前的窗口，对应的命令模式为 `rename-window`。

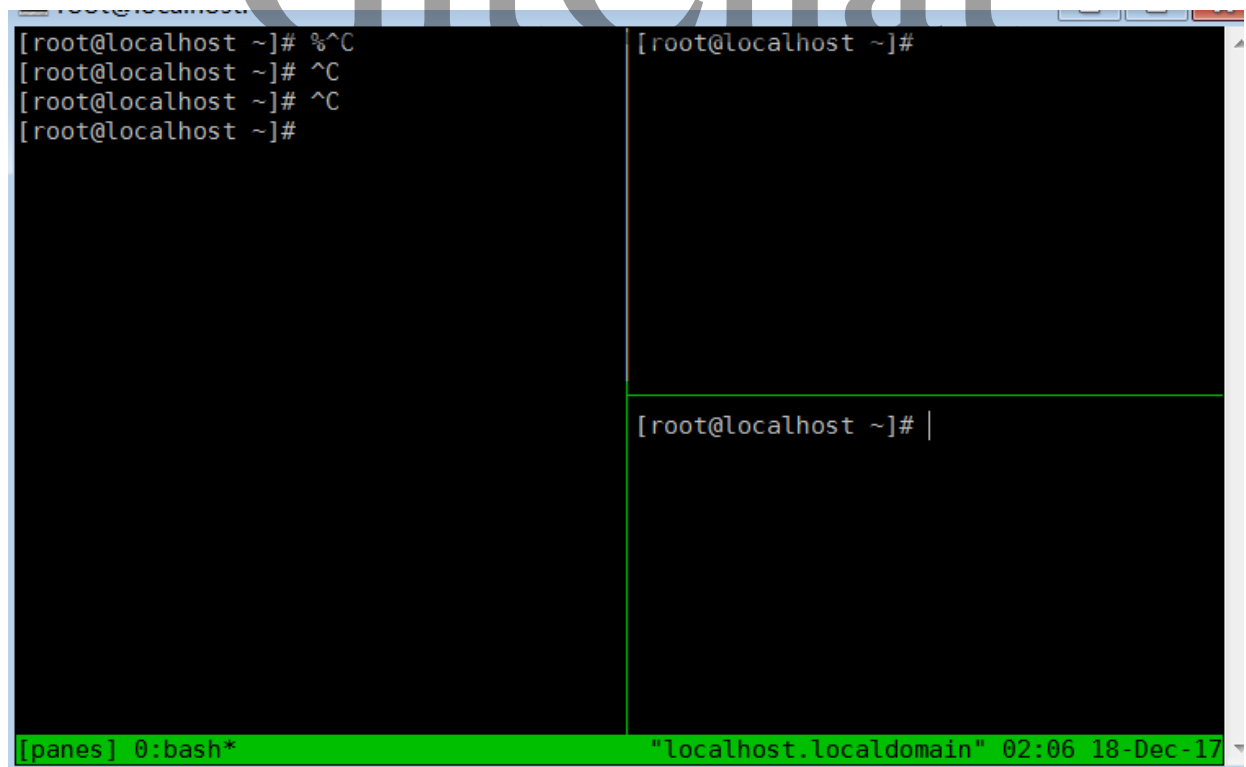
键入 `exit`，会退出当前窗口，但是有的时候窗口可能会卡死，此时我们可以使用 `prefix + &`，在输入确认之后，强制杀死当前窗口。

面板

窗口可以分割为更小的面板，配合大屏显示器使用，非常有黑客的感觉。首先我们使用：

```
tmux new -s pane
```

创建一个新的session，然后键入 `prefix + %`，然后键入 `prefix + "`，可以得到如下这样一个被分割的窗口：



当前光标所在的面板被高亮了出来。我们可以继续键入 `prefix + %` 和 `prefix + "` 查看继续分割面板的效果。要在不同的面板之间跳转，使用快捷键 `prefix + o`。如果要想上下左右的移动到不同的面板，使用快捷键 `prefix + 方向键`。

在默认情况下，tmux 平均分割一个面板。我们可以通过 `prefix + ctrl + 方向键` 来调整面板的大小。

有的时候我们可能需要将一个面板放大，占满整个窗口。我们可以使用 `prefix +!`，将面板转化为一个新的窗口；或者也可以使用 `prefix + z`，使当前面板最大化为窗口，并暂时隐藏其他的面板。

强制退出一个面板的快捷键为 `prefix + x`。

小结

会话+窗口+面板的组合是提升我们工作效率的一个强力组合。一个推荐的使用方法为对不同的项目建立不通的会话，使用窗口来分割一个项目里面的不同工作内容，然后使用面板来适用大屏开发。下面的一个截屏是我工作所建立的所有会话列表：

```
(0) + aws: 2 windows
(1) + chat: 2 windows
(2) + gogate: 2 windows
(3) + gonsgo: 2 windows
(4) + k8s: 1 windows
(5) + netdata: 2 windows
(6) + puppeteer: 2 windows
(7) + seeds: 3 windows
(8) + tmux: 4 windows (attached)
(9) + work: 2 windows
```

chat (sort: index)

```
st GoNSGo]# ^C
st GoNSGo]# ^C
st GoNSGo]# :|
```

1:weechat

2:bash

[02:35] [13] [core] 1:weechat [H: 3(1,

tmux < [tmux] 1.4 0.4 0.3 2017-12-18 < 02:35 < localhost.localdomain

不同的项目被我放到了不同的会话里面，当我需要转跳到某一个项目的时候，使用 `prefix + s` 转跳到对应会话，之前项目所配置好的环境立刻就恢复了。

tmux 配置

tmux 比 screen 更加流行的一个原因就是默认配置的情况下它已经足够好用了。当然，我们也可以通过配置文件对 tmux 进行个性化配置。它的默认配置文件为 `~/.tmux.conf`，如果需要使用其他的文件，可以使用 `tmux -f` 选项，读入另外一个配置文件。

默认prefix

我们可以通过配置文件更改 tmux 的默认 prefix。个人使用的 prefix 为 `ctrl-z`，习惯了 `screen` 的人可以配置为 `ctrl-a`。这里以 `ctrl-z` 为例，用文本编辑器打开 `~/.tmux.conf`，在里面加入内容：

```
set -g prefix C-z
unbind-key C-b
```

这里 `-g` 表示全局设置，应用于我们创建的所有会话。这是 `ctrl-b` 可以被释放出来组合，这里通过 `unbind-key C-b` 实现。

要使这个设置生效，我们应该重新加载 `.tmux.conf`。键入 `prefix + :` 打开命名模式，在输入框内输入 `source ~/.tmux.conf`。此时我们就可以使用新的 prefix 了。

快速重载配置文件

修改完配置文件之后再使用命令行模式重载实在太麻烦了，我们可以自定义一个快捷键，简化这个操作。在 `~/.tmux.conf` 里面加入：

```
bind-key r source-file ~/.tmux.conf\; display-message "Config
reloaded"
```

我们最后一次使用 `prefix + : + source ~/.tmux.conf` 的方式重新加载配置文件。之后就可以用 `prefix + r` 的方式来做这件事了。在 `.tmux.conf` 中加入。

```
bind e new-window -n ".tmux.conf" "vim ~/.tmux.conf"
```

这个 `prefix + e` 的组合可以让你迅速的打开 `.tmux.conf` 并进行配置修改，配置完成之后使用 `prefix + r` 的方式重新加载，整个操作在几秒内即可完成。

更改默认序号

由于 tmux 的窗口和面板默认序号都是从0开始，我们可以更改这个设置，使默认序号从1开始。

```
set -g base-index 1
setw -g pane-base-index 1
```

更改分割面板的快捷键

tmux 垂直和水平分割面板的快捷键分别为 `prefix + %` 和 `prefix + "`。这两个按键比较难记忆，我们可以将其更改为 `prefix + |` 和 `prefix + -`。将以下配置加入 `~/.tmux.conf`。

```
bind-key | split-window -h
bind-key - split-window
```

更改面板间移动的快捷键

大部分键盘操作工具，比如vim，都使用 `jkh` 来进行上下左右的移动操作。我们可以将面板间移动的操作绑定到对应的按键。配置如下：

```
bind-key l select-pane -R
bind-key h select-pane -L
bind-key j select-pane -D
bind-key k select-pane -U
```

面板大小调整

我们用类似移动键的方式来进行面板的大小调整。将快捷键重新定义为 `JKHL`。配置如下：

```
bind-key L resize-pane -R 5
bind-key H resize-pane -L 5
bind-key K resize-pane -U 5
bind-key J resize-pane -D 5
```

鼠标模式

鼠标模式有时候也可能非常有用，比如你可能想用鼠标来选中一个面板或者窗口，用鼠标调整面板大小，或者用鼠标滚轮来向上滚动浏览历史。

开启/关闭鼠标模式的配置为 `set -g mouse on/off`，需要在命令模式下敲入这串字符。我们可以更进一步，通过绑定到快捷键 `prefix +m` 来触发鼠标模式的开关。配置如下：

```
bind m run 'old=$(tmux show -gv mouse);new=""; if [ "$old" = "on" ];
then new="off"; else new="on"; fi; tmux set -g mouse $new; tmux
display "mouse: $new"
```


注意这是一行配置，细心的读者可能已经发现，这其实是一段 shell 脚本，在鼠标模式关闭的情况下打开它，反之亦然。

状态栏的配置

默认tmux的状态栏是窗口下方的一段绿色长条，比如下图：



它分为三个部分：

- 最右边的会话名字
- 中间的窗口列表
- 左边的主机信息

这些显示的信息，以及字体前景色，背景色都是可以灵活配置的。开源的好处就是社区已经有许多成熟的解决方案，这里面最受欢迎的是powerline。

powerline 用Python编写，安装使用Python 的包管理工具 pip

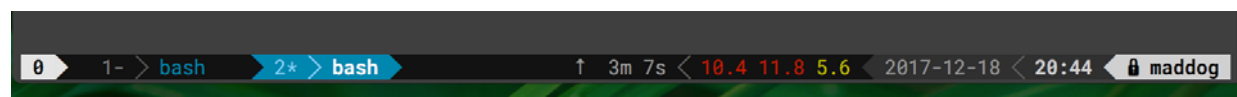
```
pip install powerline-status
```

此外我们还需要安装powerline使用的字体，可以在<https://github.com/powerline/fonts> 下载并安装。

对tmux使用powerline，只需将如下配置加入 .tmux.conf， prefix + r 重新加载即可：

```
source "${repository_root}/powerline/bindings/tmux/powerline.conf"
```

其中 repository_root 是 powerline 的安装路径。以下是作者安装powerline之后状态栏的一个截屏：



复制模式

tmux 初学者经常遇到的一个问题就是发现无法使用滚动键向上滚动查看终端的历史。要做到这一点，必须进入 tmux 的复制模式。

按下 prefix + [，tmux就进入了复制模式，再按回车键退出。默认情况下 tmux 保存 1000 行终端历史。我们可以通过。

```
set -g history-limit 10000
```

调整历史记录大小。

tmux 复制模式可以使用两种键盘模式，Vi 和 Emacs，默认为 Emacs，我们通过：

```
set -g mode-keys vi
```

更改为 Vi 的键盘模式。在 Vi 模式下，可以通过 jkhl 来上下左右移动光标。

要复制文本，我们先进入复制模式，将光标移动到指定位置，按下空格键，然后移动光标来选择文本，被选择的文本会高亮显示，最后按回车键，表示选择完毕。

此时敲入命令 `tmux list-buffer`，我们可以看到tmux缓存区保存的文本，使用 `tmux paste-buffer` 或者快捷键 `prefix +]` 可以粘贴缓存区里面的内容。

更加高级的是 tmux 维护一个缓冲区的栈，每复制一次，tmux在这个栈顶部创建了一个新的缓冲区。现在我们可以多复制几次文本，然后键入命令 `tmux list-buffers`，这时可以看到一个缓冲列表。再敲入命令 `tmux choose-buffer`，tmux会创建一个选择框，我们可以选择需要粘贴的文本，然后回车，对应文本就插入到了光标位置。

vim用户可以重新配置这些命令和快捷键，让使用起来更加熟悉。

```
bind-key -Tcopy-mode-vi 'v' send -X begin-selection
bind-key -Tcopy-mode-vi 'y' send -X copy-selection
unbind p
bind p paste-buffer
bind b choose-buffer
```

现在在复制模式下可以使用 v 和 y 来选择文本，使用 `prefix + p` 粘贴文本，使用 `prefix + b` 来选择缓冲（将prefix设置成为 `ctrl + b` 的用户可自行选择其他快捷键）。

其他配置

为了使tmux更好的工作，作者在这里还贴出一些其他的 tmux 基本配置。

```
set -g default-terminal "screen-256color" # 颜色支持
setw -q -g utf8 on # utf8 支持
set -q -g status-utf8 on # tmux < 2.2
setw -g automatic-rename on # 自动重命名窗口
set -g renumber-windows on # 关闭窗口的时候重新计算窗口index
set -g display-time 4000 # tmux 消息提示时间为4秒
```

session 共享和结对编程

目前很多企业都在推行结对编程，两个人同一工作台前开发软件。当开发人员在同一个办公室的情况下，实施结对编程比较简单，但是如果开发人员处于异地的状态，实施结对编程就必须有屏幕共享的软件。而 **tmux** 基于终端的会话共享可以在即使网络状况不佳的情况下提供良好的结对编程体验。

我们先来看一下最基本的会话共享机制。

假设有主机Foo，程序员 A ssh 连接到了这台主机，并使用 **tmux** 开始了会话 **pairing**。

```
tmux new -s pairing
```

A 想把这个会话共享给开发人员 B，那么他只需要让 B 登录同一台主机的同一个用户，B 就可以用命令。

```
tmux attach -t pairing
```

attach 到同一个会话。此时A 和 B 看到的就是一会话的统一窗口，两人的操作也会完全同步到各自的屏幕上。

这种方法一个不那么完美的地方就是A和B看到的屏幕永远是完全一致的，有的时候我们可能需要让A和B能够同时做不同的事情，同步窗口的结果但不必保持窗口显示的同步。要做到这一点，只需要 B 使用命令：

```
tmux new -s test -t pairing
```

创建一个新的会话并将其加入到 **pairing** 会话即可。这样 A 和 B 看到的窗口结果是一样，但是两人都可以独立输入而不会互相打扰。

如果想在共享服务器的登录权限的情况下共享 **tmux** 会话，可以尝试使用 **tmate**。它是一款 **tmux** 的 **fork**，可以在无需登录主机的情况下只读的共享你的 **tmux** 会话。详细使用读者可以自行参考相关网站。

插件管理

tmux 官方支持一系列的插件，可以在 <https://github.com/tmux-plugins> 找到。注意大部分插件都需要 **tmux 1.9+** 版本。

tpm

首先需要提及的是插件管理工具 **tpm**。它可以用来方便的安装和删除插件。安装 **tpm** 的方法为

```
git clone https://github.com/tmux-plugins/tpm ~/.tmux/plugins/tpm
```

在 `~/.tmux.conf` 中加入这些配置：

```
#tmux 插件列表
set -g @plugin 'tmux-plugins/tpm'
set -g @plugin 'tmux-plugins/tmux-sensible'

#将这一行插入.tmux.conf最底部
run '~/.tmux/plugins/tpm/tpm'
```

然后使用 `prefix + r` 重新加载tmux配置，就可以使用**tmp**了。

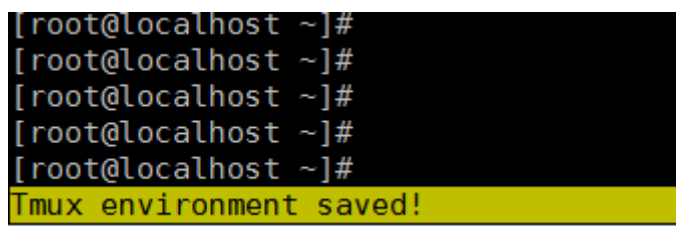
当在 `.tmux.conf` 里面加入了新的插件时，我们可以使用 `prefix + I` 安装插件，使用 `prefix + U` 更新插件。删除插件的快捷键为 `prefix + alt + u`。

会话保持：tmux-resurrect

tmux-resurrect 是一款轻量级的会话保持插件。它可以用来在服务器重启之后重新加载之前保存的tmux会话。

安装 **tmux-resurrect**，首先将 `set -g @plugin 'tmux-plugins/tmux-resurrect'` 加入到 `.tmux.conf`中，然后按 `prefix + I` 安装。

保存 tmux 会话的方法为 `prefix + C-s`。保存成功之后会出现如下提示

A terminal window screenshot showing a series of shell prompts. The last line of the screenshot is highlighted in yellow and reads "Tmux environment saved!".

```
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]#
Tmux environment saved!
```

此时我们如果重启机器，然后打开一个新的 tmux 会话，那么可以使用快捷键 `prefix + C-r` 恢复保存的会话信息。

tmuxinator是另外一款 **tmux** 的会话保持工具，它通过编辑和读取配置文件的形式进行会话的保持。这里留给读者自行参考。

结束语

到此为止，本篇关于tmux的介绍就结束了。正如文章开头所说，tmux可以大幅度的提高工作效率，希望读者都能喜欢上这一款优秀的终端管理工具。

GitChat