

一文搞定我的环境配置

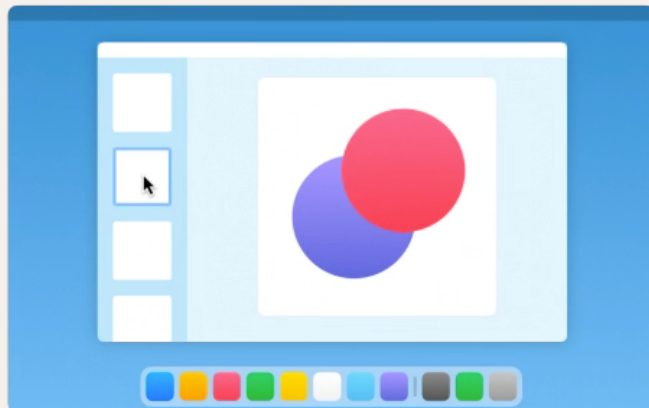
新的工作环境，新的电脑，一定要把拾掇好，这样才能有好心情，效率才会高，才有创造性产出。

一、macOS 洁癖配置

1.1 触摸板开启轻点点按

老习惯了，之前老版本的 Mac 不支持 forcetouch，通过轻点一下来达到点击效果，我个人非常喜欢这种方式，不需要蛮力点击，非常优雅，forcetouch 优势是有震动反馈，对于我来说可有可无。

触控板



光标与点按

滚动缩放

更多手势

跟踪速度



点按



静默点按



用力点按和触感反馈



点按，然后用力地按下以进行快速查看、查询以及变速媒体控制。

查询与数据检测器

单指用力点按

辅助点按

双指点按或轻点

轻点来点按



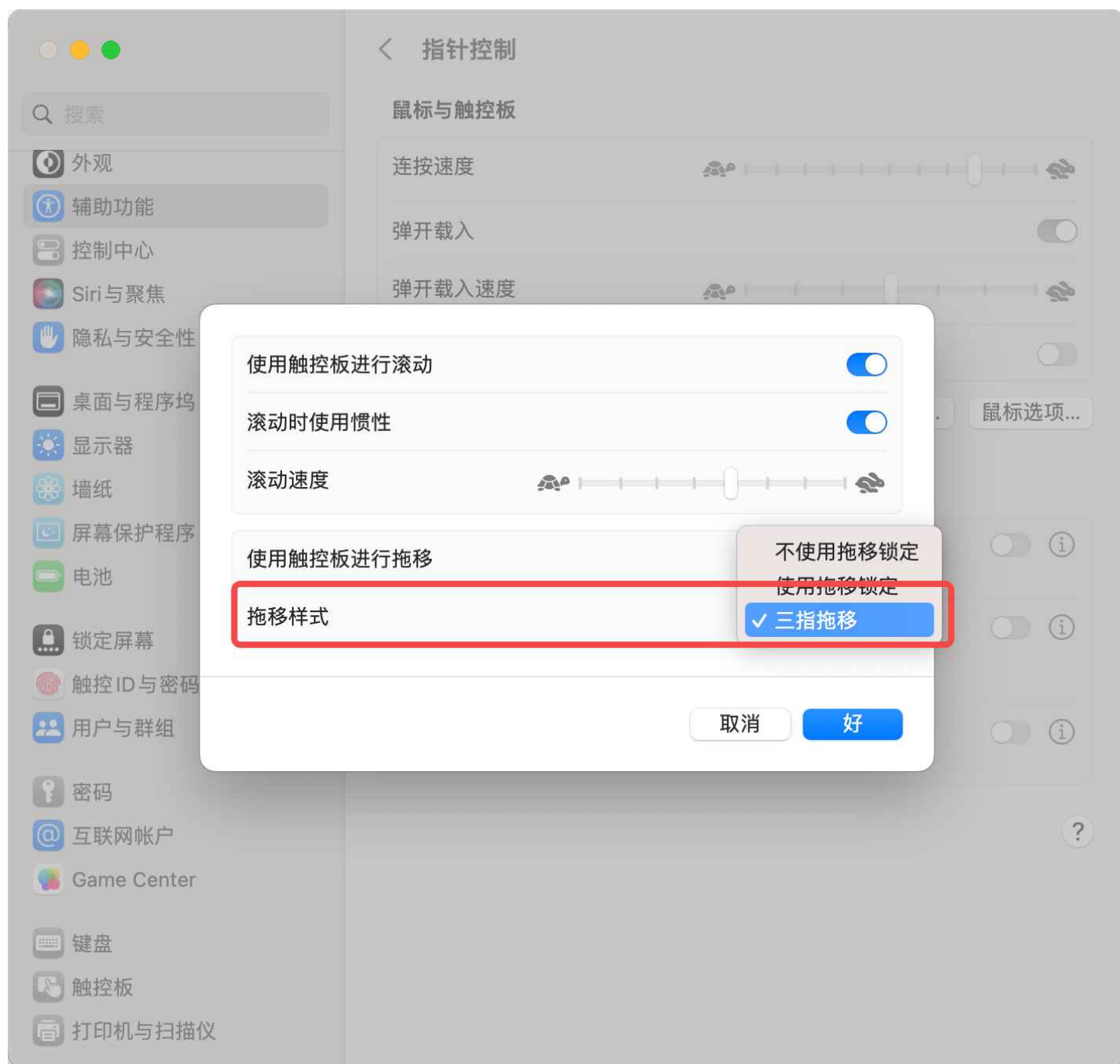
单指轻点

设置蓝牙触控板...



1.2 触摸板开启三指拖移

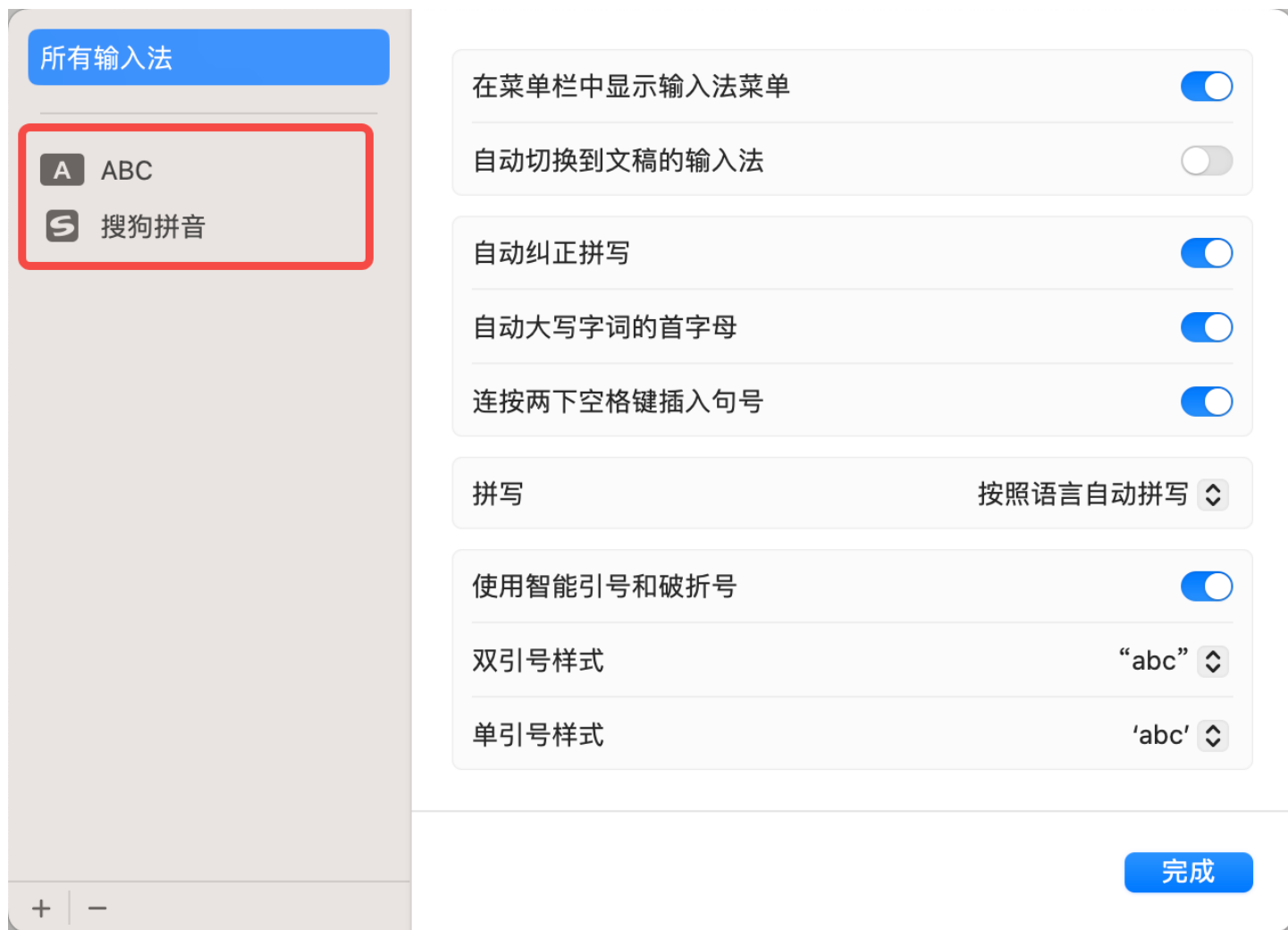
配置路径：**辅助功能 -> 指针控制 -> 触控板选项 -> 拖移样式**，也是老习惯，没有 forcetouch 之前，通过三个手指在触摸板上移动来控制拖移非常舒服，而且可接力，新的 forcetouch 点下去拖移，我是不太喜欢用，原因是不可接力，同时摩擦力太大，废手。



1.3 输入法配置

macOS 自带的输入法挺“简洁”的，本土化没做好，只能用搜狗拼音输入法

<https://pinyin.sogou.com/mac/>，为了进一步洁癖，移除不必要的输入法，只保留 **ABC** 和 **搜狗拼音**，切换输入法的更快。



1.4 访达 Finder 配置

点击 访达 -> 设置



- 开启 Finder 默认定位到自己用户目录
- 配置侧边栏自己需要的，取消勾选：影片、音乐、最近项目，勾选：用户目录
- 显示所有文件扩展名

访达设置



通用



标签



边栏



高级

在桌面上显示这些项目：

- ☐ 硬盘
- ☒ 外置磁盘
- ☒ CD、DVD 和 iPod
- ☐ 已连接的服务器

开启新“访达”窗口时打开：



my



☒ 在标签页（而不是新窗口）中打开文件夹

访达设置



通用



标签



边栏




高级

在边栏中显示这些项目：

个人收藏

- ☐ 最近使用
- ☒ 隔空投送
- ☒ 应用程序


☒  桌面

☒  文稿

☒  下载


☐  影片


☐  音乐

☐  图片

☒  my

iCloud


☐  iCloud 云盘

☐  共享

位置


☐  MyAir


☒  硬盘

☒  外置磁盘


☒  CD、DVD 和 iOS 设备

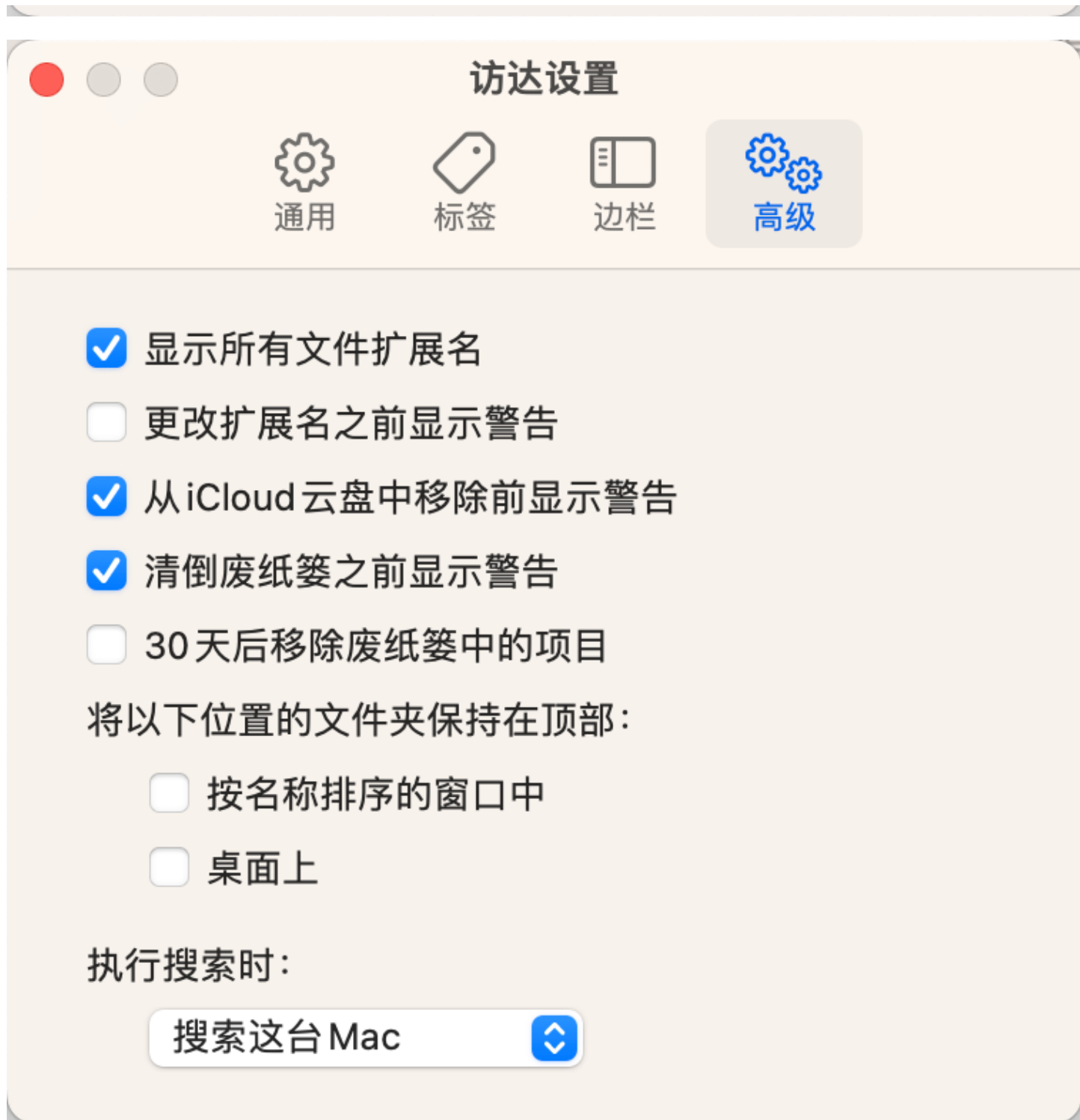
☒  云端储存空间

☒  Bonjour 电脑

☒  已连接的服务器

标签

☐  最近使用的标签



1.5 显示器配置

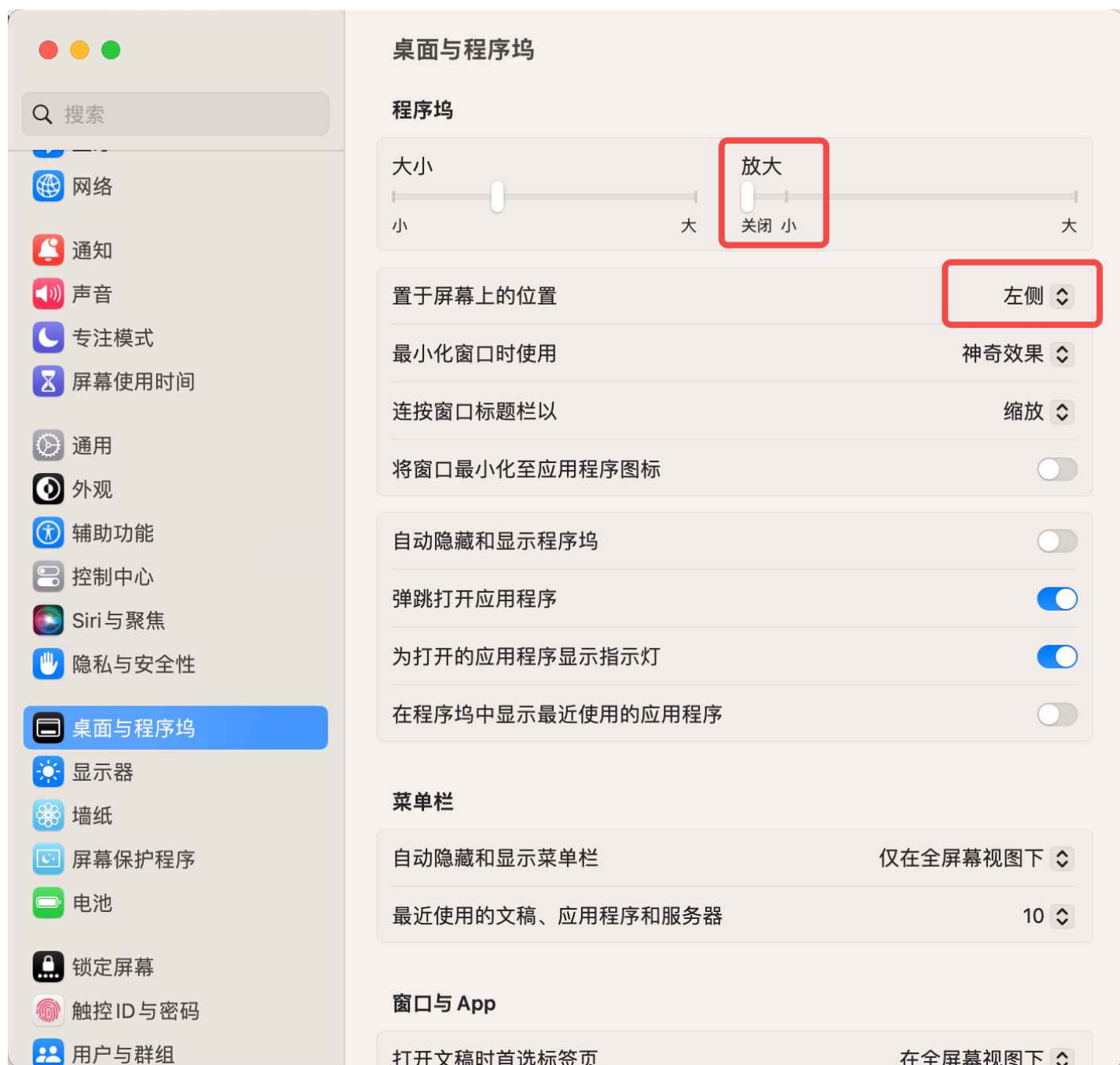
新款 MBP 16 寸屏幕非常大，字显示的很小，我一般将显示器缩放到 1536 * 960，对于 13 寸，空间也太小，我一般缩放到更多空间 1680 * 1050，根据自己的眼球喜好来配置吧。



1.6 程序坞 Dock 配置

配置路径：设置 -> 桌面与程序坞，尽量少一些花哨，提高工作效率

- 取消放大效果
- 放在屏幕左侧
- 整理 Dock 中驻留的程序，只留常用的



二、工作环境配置

2.1 字体安装

- [Fira Code](#) 编程字体，有连字符，也用于 terminal 显示
- [Operator Mono](#) 编程字体，比较骚气，保持年轻的字体，需要先安装 FiraCode

```

1  # This is bogus code by Doug Wilson (from Tosh
2  Typeface:Operator Mono SSm()
3      except Exception as e:
4          print("Export.py Error: %s" % e)
5  class ExportInDesignTaggedText( object ):
6      def __init__( self ):
7          windowWidthResize = 100 1/2 1/2 3/4 5/8 # user c
8          self.w = findfloat.FloatingWindow(
9              "Export InDesign", # window title
10         # UI elements:
11         self.w.runButton = vanilla.Button((sp, sp*3+edV
12             if sender.get():
13                 self.w.breakCheck.set(True)

```

- [阿里巴巴普惠体](#) 免费中文字体，画架构图、写 ppt 时用起来很好看

预览

汉体书写信息技术标准相容
 档案下载使用界面简单
 支援服务升级资讯专业制作
 创意空间快速无线上网
 (一)(二)(三)(四)(五)(六)(七)(八)(九)(十)
 A a B b C c A a B b C c

- [霞鹜文楷](#) 开源中文字体，写 ppt 时用起来好看

落霞与孤鹜齐飞
秋水共长天一色

〔唐〕王勃《滕王阁序》

「霞鹜文楷」 / LXGW WenKai

FONTWORKS 开源日文字体 [Klee One](#) 衍生 CJK 字体

2.2 Xcode 安装

开发时有很多工具依赖 Xcode Command line developer tools，所以是必装的，直接用以下命令搞定：

```
1 xcode-select --install
```



但如果是搞 iOS、macOS 开发等，直接安装 xcode.app 吧（8G 左右）。

2.3 Github 不稳定解决方案（非靠谱）

github 经常网络不通，原因是 IP 经常变，DNS 缓存的历史 IP 地址没及时更新，所以思路是不通的时候我们自己手动更新 IP。

Step 1: 在 <https://www.ipaddress.com/ip-lookup> 这个地方依次获取以下网址所对应的 IP 地址

```
1 github.com
2 github.global.ssl.fastly.net
3 codeload.github.com
4 assets-cdn.github.com
5 api.github.com
6 www.google-analytics.com
7 live.github.com
8 codeload.github.com
```

Step 2: 将获取的 IP 地址写到 `/etc/hosts` 文件中（一般只需要找 [github.com](https://www.github.com) 的地址就可以提速了）

```
1 140.82.114.3 github.com
```

```
2 140.82.112.9 codeload.github.com
3 140.82.114.5 api.github.com
4 140.82.112.25 live.github.com
5 140.82.112.9 codeload.github.com
6 151.101.1.194 github.global.ssl.fastly.net
7 151.101.65.194 github.global.ssl.fastly.net
8 151.101.129.194 github.global.ssl.fastly.net
9 151.101.193.194 github.global.ssl.fastly.net
10 185.199.108.153 assets-cdn.github.com
11 185.199.110.153 assets-cdn.github.com
12 185.199.111.153 assets-cdn.github.com
13 185.199.109.153 assets-cdn.github.com
14 142.250.11.100 www.google-analytics.com
15 142.250.11.101 www.google-analytics.com
16 142.250.11.102 www.google-analytics.com
17 142.250.11.113 www.google-analytics.com
18 142.250.11.138 www.google-analytics.com
19 142.250.11.139 www.google-analytics.com
```

Step 3: 刷新 DNS 缓存

```
1 sudo killall -HUP mDNSResponder # Sierra、El Capitan 或更新版本的 macOS
```

看起来靠谱的方案：

- <https://zhuanlan.zhihu.com/p/549428861>
- <https://www.cnblogs.com/kewei/p/15047759.html>
- <https://github.com/azhai/githubdns>

此方法只能缓解，治标还是要架梯子。

2.4 Terminal 配置

以下配置将配出如图效果：

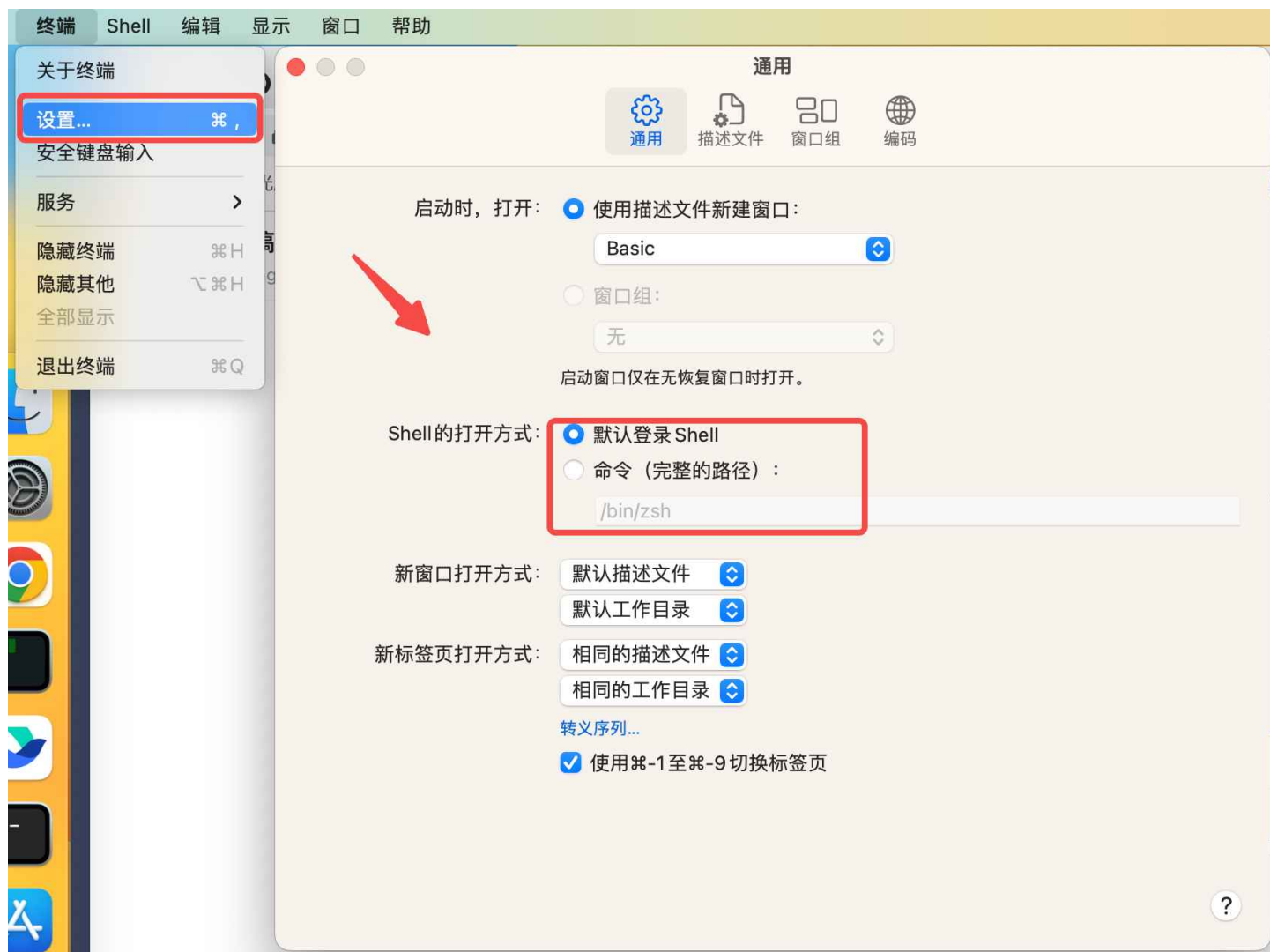


2.4.1 使用 zsh

zsh 比 bash 更高效，将当前的 shell 切换至 zsh

```
1 chsh -s /bin/zsh
```

在终端 -> 设置中可查看是否生效



2.4.2 配置描述文件

配置路径：**终端 -> 设置 -> 描述文件**，基于 Pro 主题复制一个描述文件，取名为 MyPro，并进行以下配置

- 光标设为竖条，并闪动
- 使用 Fira Code 字体，大小 16
- 取消勾选可听报警声（有时候按键会发出没必要的报警声，比较吵）

描述文件



通用



描述文件



窗口组



编码

文本

窗口

标签页

Shell

键盘

高级

背景



颜色与效果

图像:

没有背景图像



字体

Fira Code Regular 16

更改...

文本



平滑文本



使用粗体字



允许闪烁文本



显示 ANSI 颜色



对粗体文本使用亮丽颜色



文本



粗体文本



所选内容

ANSI 颜色



标准



明亮

光标



块



下划线



| 竖条

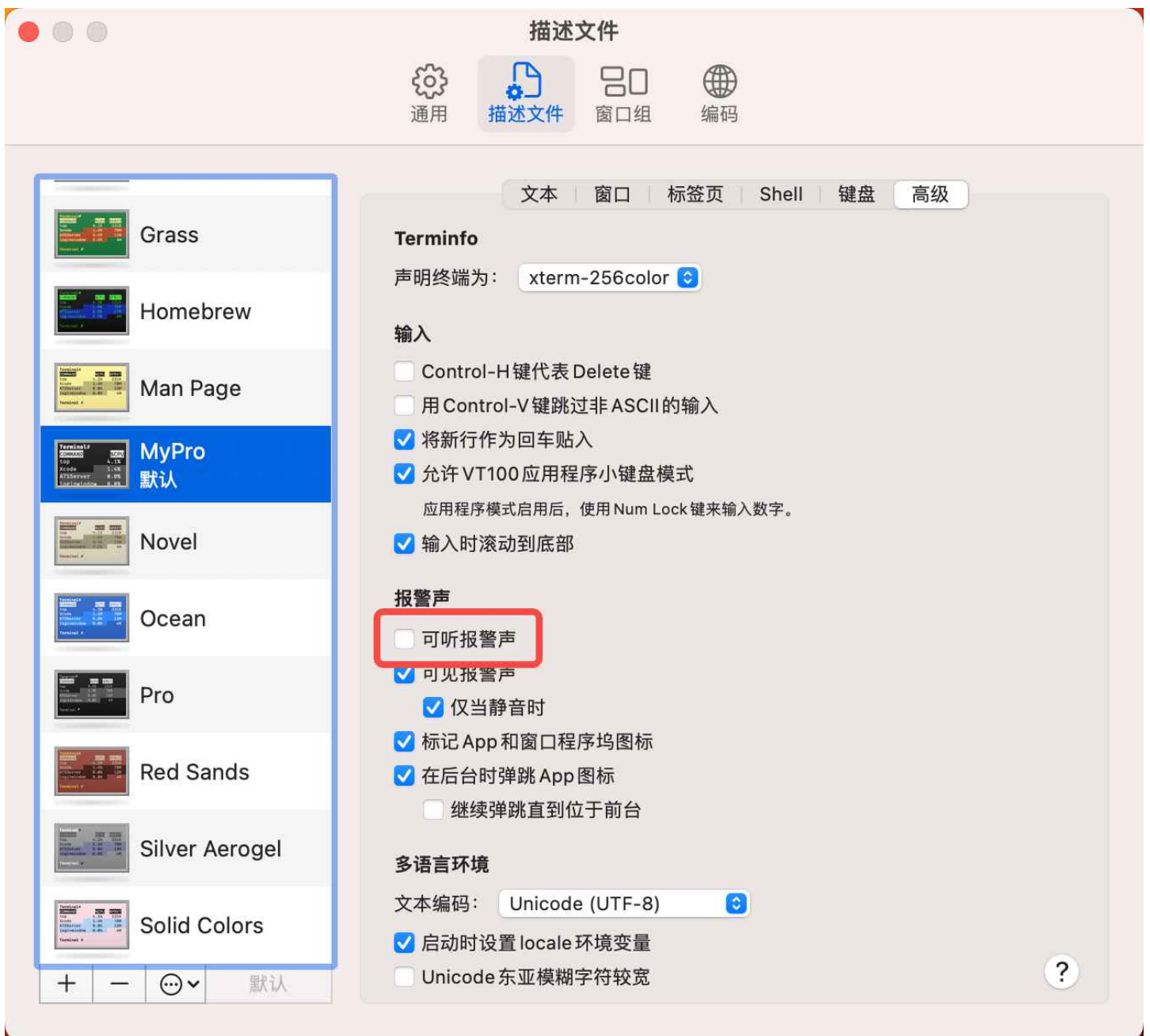


闪烁光标



光标





重启终端, 输入 `echo -e "\xee\x82\xa0"` 命令测试一下支持连字符的字体是否已经生效:



应当输出一个**分支符号**的字符。

2.4.3 安装 oh-my-zsh

oh-my-zsh 带火了 zsh，是业内最流行的 zsh 的配置，执行以下命令安装：

```
1 sh -c "$(curl -fsSL https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/too
```

如果不想用这么重的 oh-my-zsh，看看这个帖子 <https://zhuanlan.zhihu.com/p/454191603>

2.4.4 配置 .zshrc

以上步骤会自动在用户目录创建 .zshrc 文件，我们继续配置一下主题，并添加一些好用的插件。

使用 **spaceship** 主题，克隆 spaceship-prompt 到 ZSH_CUSTOM 目录，并创建符号链接 `spaceship.zsh-theme` 到 oh-my-zsh 自定义主题目录下：

```
1 # 进入 ZSH_CUSTOM 目录
2 cd ${ZSH_CUSTOM:~/.oh-my-zsh/custom}
3 # 克隆 spaceship-prompt
4 git clone https://github.com/spaceship-prompt/spaceship-prompt.git spaceship-pro
5 # 进入 themes 目录
6 cd themes
7 # 创建符号链接
8 ln -s ../spaceship-prompt/spaceship.zsh-theme spaceship.zsh-theme
```

安装 **zsh-syntax-highlighting** 和 **zsh-autosuggestions** 插件：

```
1 git clone https://github.com/zsh-users/zsh-syntax-highlighting ${ZSH_CUSTOM:~/.
2 git clone https://github.com/zsh-users/zsh-autosuggestions ${ZSH_CUSTOM:~/.oh-m
```

编辑 `~/.zshrc` 文件：

```
1 # 将主题设为 spaceship
```

```
2 ZSH_THEME="spaceship"
3 # 开启需要的插件
4 plugins=(git fnm node brew zsh-syntax-highlighting zsh-autosuggestions)
```

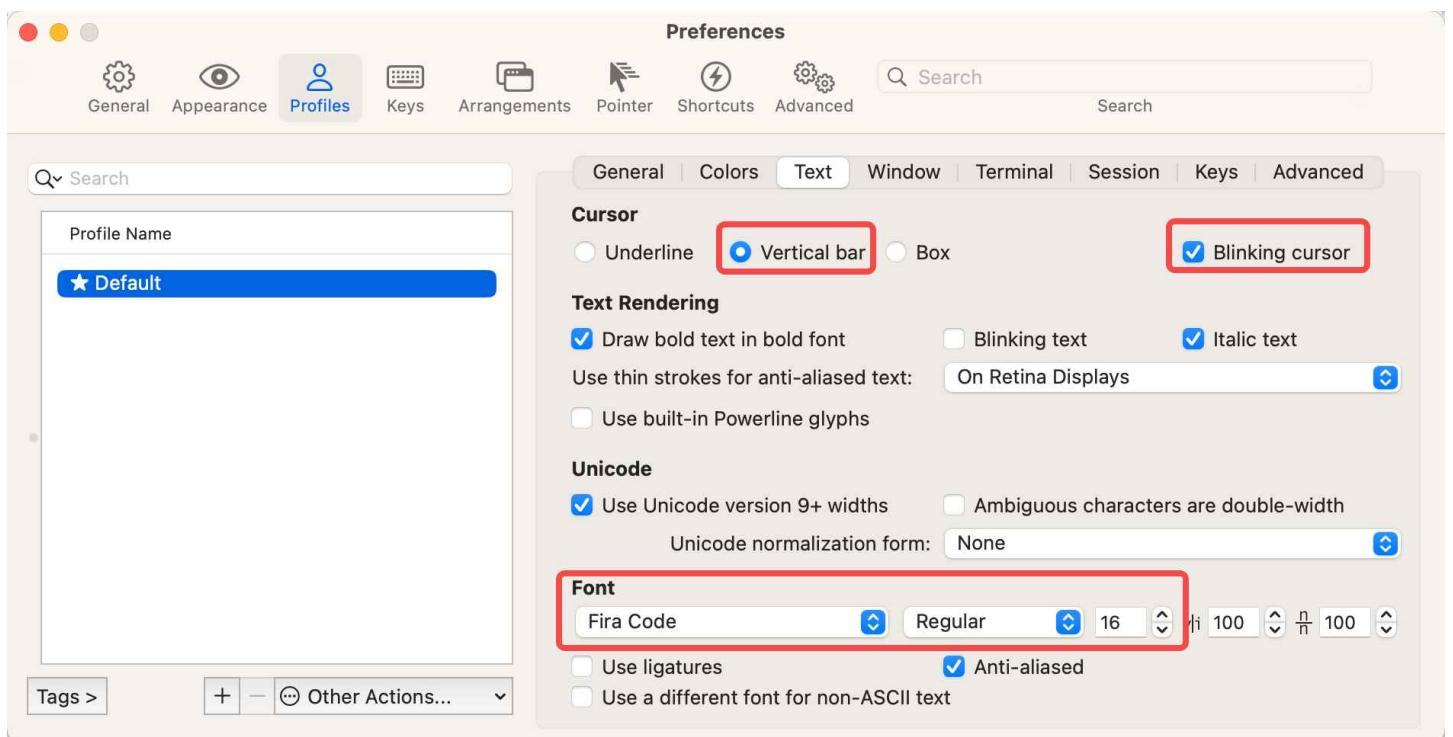
重启 Terminal 或新建一个 Tab (COMMAND + T) 即可体验最终效果。

2.4.5 安装 iTerm2 (可选)

免费终端工具，比系统自带的 Terminal 更具可配置性，可玩性较高，下载地址：

<https://iterm2.com/>，然后做一些额外配置：

- 光标设为 Vertical Bar，并闪烁
- 使用 Fira Code 字体，大小改为 16
- 静默响铃



General | Colors | Text | Window | **Terminal** | Session | Keys | Advanced

Scrollback lines: 1,000

☒ Save lines to scrollback when an app status bar is present

☒ Save lines to scrollback in alternate screen mode

Terminal Emulation

Character encoding: Unicode (UTF-8)

Report terminal type: xterm-256color

ENQ answer back:

☒ Enable mouse reporting

☒ Report mouse wheel events

☒ Report mouse clicks & drags

☐ Terminal may report window title

☐ Disable session-initiated printing

☐ Disable save/restore alternate screen

☒ Disable session-initiated window resizing

☒ Terminal may enable paste bracketing

Notifications

☒ Silence bell

☐ Flash visual bell

☒ Notification Center Alerts

☒ Show bell icon in tabs

Environment

☒ Set locale variables automatically

Shell Integration ?

☒ Insert newline before start of command prompt if needed

☒ Show mark indicators

2.5 Git 配置

2.5.1 Git 全局配置

写入以下内容到 ~/.gitconfig 文件中：

```
1 [user]
2   name = your-name
3   email = your-email
4 [alias]
5   ci = commit
6   co = checkout
7   br = branch
8   st = status
```

```
9 re = reset
10 rv = revert
11 go = checkout -b
12 reci = commit --amend
13 unadd = reset HEAD
14 pl = pull -r
15 lg = log HEAD --stat --graph --name-status --pretty=format:'-----'
```

2.5.2 Github 配置

提交代码到 github 上需要配置 SSH Key，简单步骤如下：

生成 SSH key `~/.ssh/id_ed25519`

```
1 ssh-keygen -t ed25519 -C "your_email@example.com"
```

在 `~/.ssh/config` 中写入以下内容：

```
1 Host github.com
2   AddKeysToAgent yes
3   UseKeychain yes
4   IdentityFile ~/.ssh/id_ed25519
```

拷贝 SSH 公钥内容到剪贴板：

```
1 pbcopy < ~/.ssh/id_ed25519.pub
```

在 <https://github.com/settings/keys> 点击 **New SSH key**，粘贴内容并添加

Title

Key type

Authentication Key

Key

Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

Add SSH key

2.6 Homebrew 安装

brew 全称 [Homebrew](#) 是 macOS 上的软件包管理工具，类似 linux 的 apt-get/yum。

执行以下命令安装：

```
1 /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD)
```

检查 brew 安装：

```
1 brew doctor
```

查看安装的包：

```
1 brew list
```

2.7 Node.js 环境配置

2.7.1 安装 fnm

fnm 是使用 rust 开发的 node.js 版本管理器，跨平台，速度快，安装方式如下：

在 <https://github.com/Schniz/fnm/releases> 页面下载最新包，解压之后将 fnm 放置在 `/usr/local/bin` 中，并加上执行权限 `chmod +x fnm`。

执行以下命令查看安装情况：

```
1 fnm --version # output: fnm 1.33.1
```

在 profile (`~/.zshrc`, `~/.bashrc`) 文件中添加，以便把需要的位置添加到 PATH 中：

```
1 eval "$(fnm env --use-on-cd)"
```

2.7.2 设置公用 npm 全局模块

使用 fnm 时，默认的 prefix 是当前激活的 node 版本的安装路径，切换 node 版本之后，之前安装过的全局模块需要再重新安装，非常不方便，**解决步骤如下：**

配置 prefix：

```
1 npm config set prefix ~/.npm-global
```

在当前用户目录建立全局 npm 模块目录

```
1 mkdir $HOME/.npm-global
```

并在此目录中新建 env 文件，内容如下：

```
1 #!/bin/sh
2 case ":
```



```
3  ${PATH}:" in
4      *:"$
5  HOME/.npm-global/bin":*)
6      ;;
7      *)
8          # Prepending path in case a system-installed rustc needs to be override
9          export PATH="
10 $HOME/.npm-global/bin:$
11 PATH"
12      ;;
13 esac
```

在 ~/.zshenv 文件中添加：

```
1 . "$HOME/.npm-global/env"
```

重启 terminal 使之生效。

这样无论使用哪个版本的 node，安装全局模版时都会安装到统一的目录 `~/.npm-global`，就可避免在切换版本后需要重新安装。

2.7.3 安装 yarn/pnpm

yarn 和 pnpm 都是比较好用的 node 包管理器，都支持 monorepo，建议都装上：

```
1 npm install --global yarn
2 npm install --global pnpm
```

2.7.4 切换国内源

国内源速度比较快，执行以下命令进行配置：

```
1 npm config set registry https://r.cnpmjs.org/
```

2.8 Go 环境 李子昊

2.8.1 方法一：安装包安装

进入[golang官网](https://golang.google.cn)，下载自己所需的版本安装包，下载完成根据指引完成安装步骤，默认安装位置为'/usr/local/go'。

2.8.2 方法二：压缩包安装

也可以使用终端命令行获取golang包。

```
1 wget -c https://golang.google.cn/dl/go1.21.4.darwin-amd64.tar.gz
```

解压到安装目录'/usr/local/go'。

```
1 sudo tar -xzvf go1.21.4.darwin-amd64.tar.gz -C /usr/local
2 #查看安装目录
3 cd /usr/local/go
4 ls
```

2.8.3 方法二：使用brew安装

上文已经安装了Homebrew的可以使用brew命令安装golang，也是默认安装到'/usr/local/go'。

```
1 brew install go
```

2.8.4 配置环境

确认go安装目录正确，接下来需要配置环境，创建并修改'~/.bash_profile'。

```
1 vim ~/.bash_profile
2
3 #写入下述内容
4 PATH=$PATH:/usr/local/mysql/bin
5 export GOROOT=/usr/local/go
6 export GOBIN=$GOROOT/bin
7 export PATH=$PATH:$GOBIN
8 export GOPATH=/Users/ileecury/go #写自己的go工作区路径
9 export GOPROXY=https://goproxy.cn
```

重要的两个：

- GOROOT：golang的安装目录。

- GOPATH：golang的工作区路径，每个人都有自己的工作区，go项目必须放在GOPATH下，其中包含go的源码文件以及go的可以执行文件以及包文件，三个子目录：bin、pkg、src。**GOPATH可以有多个。**

编辑'~/.zshrc'以及'~/.bashrc'。

```
1 vim ~/.zshrc
2 #都可以编辑，取决于你用的终端是zsh还是bash
3 vim ~/.bashrc
4
5 #写入以下内容
6 source ~/.bash_profile
```

终端重新读取配置。

```
1 source ~/.zshrc
```

查看是否配置成功。

```
1 go version
2 #
3 go env
```

2.8.5 使用brew切换go版本



注意，一定是使用brew安装的go。

先解绑当前的go版本。

```
1 brew unlink go
```

查看可以安装的go版本列表。

```
1 brew search go
2 #
3 ...
```

```
4 go@1.16 go@1.17 go@1.18 go@1.19 go@1.20
```

安装想要的版本。

```
1 brew install go@1.20
```

绑定当前版本。

```
1 brew link go@1.20
```

重新读取配置。

```
1 source ~/.bash_profile
```

检测版本是否切换成功。

```
1 go version
```

2.9 Rust 环境（可选） 赵康

方式一：使用 Rustup（推荐）

```
1 curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

选择1继续安装

```
glancake — curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh — curl — rustup-init — 80x24
Current installation options:

↔ 中文（简体） 英语 日语
default host triple: x86_64-apple-darwin
default toolchain: stable (default)
profile: default
modify PATH variable: yes

1) Proceed with installation (default)
2) Customize installation
3) Cancel installation
>1
```

方式二：brew安装

```
1 brew install rust
```

检查是否安装成功

```
1 rustc -V      # 注意的大写的 V
2 cargo -V
```

2.10 Tree 安装（可选）

[tree](#) 是一个小巧的字符目录树生成工具，方便写技术文档，使用 brew 命令安装：

```
1 brew install tree
```

使用效果如图：

```
my@MyAir - ..ckages/engine

lowcode-engine/packages/engine on main is 1.1.0 via v18.15.0
→ tree
.
├── README-zh_CN.md
├── README.md
├── babel.config.js
├── build.json
├── build.plugin.js
├── build.test.json
├── build.umd.json
├── jest.config.js
├── package.json
├── src
│   ├── engine-core.ts
│   ├── index.ts
│   ├── inner-plugins
│   │   ├── builtin-hotkey.ts
│   │   ├── component-meta-parser.ts
│   │   ├── default-panel-registry.tsx
│   │   └── setter-registry.ts
│   └── modules
│       ├── classes.ts
│       ├── designer-types.ts
│       ├── live-editing.ts
│       ├── lowcode-types.ts
│       ├── shell-model-factory.ts
│       ├── skeleton-types.ts
│       └── symbols.ts
└── tsconfig.json

4 directories, 23 files

lowcode-engine/packages/engine on main is 1.1.0 via v18.15.0
→
```

2.11 Vi 编辑器美化（可选）

经常改一些配置文件需要直接在 terminal 里使用 vi 编辑器来修改，所以适当美化一下，还是很有必要的。

todo...

WIP: 2.12 SSH 免密登录服务器（可选）

自己在阿里云有个人服务器，每次登录使用密码一是记不住密码、二是不安全，我们可以如下配置实现免密登录。

Step1: 配置服务器 sshd

以 root 用户登录，更改 ssh 配置文件 `/etc/ssh/sshd_config`，去除以下配置的注释：

```
1 RSAAuthentication yes # 启用rsa认证
2 PubkeyAuthentication yes # 启用公钥私钥配对认证方式
3 AuthorizedKeysFile .ssh/authorized_keys # 公钥文件路径，相对于登录的用户目录
```

重启 SSH 服务：

```
1 systemctl restart sshd // 重启ssh服务
```

Step2: 配置本地 ssh

生成公钥私钥对（和 [Github 配置](#) 类似）

```
1 ssh-keygen -t rsa -C "myair@host"
```

一路默认回车，将在 `~/.ssh` 下生成 `id_rsa` 和 `id_rsa.pub`

这里使用的 `rsa` 还可以指定其他算法如：`rsa` | `dsa` | `ecdsa` | `ed25519` | `rsa1z`，不同版本的系统支撑的算法可能不一样。

把 `~/.ssh/id_rsa.pub` 拷贝到服务器

```
1 ssh-copy-id -i /root/.ssh/myair_rsa.pub myair@192.168.11.20
```

拷贝 SSH 公钥内容到剪贴板：

```
1 pbcopy < ~/.ssh/myair_ed25519.pub
```

验证

```
[root@client /]#ssh root@192.168.11.20 #server ip
```

Step3: 配置服务器 alias

```
1 HOST hostalias
2   HostName 10.10.10.10
3   User yourname
4   GSSAPIAuthentication yes
5   GSSAPIDelegateCredentials no
6   PasswordAuthentication no
```

三、应用

3.1 VSCode 编辑器



The screenshot shows the VS Code editor interface. The top bar indicates the active file is 'Python'. The editor area displays the following content in a settings.json file:

```
1 # settings.json
2 "editor.fontFamily": "'Operator Mono Light', Fira Code",
3
```

On the right side of the editor, there is a watermark that reads: 微信号: zeropython

copilot

git graph

Material Icon Theme

git lens

3.2 Chrome

3.3 飞书