git的安装使用

1 git的安装

配置用户名和邮箱。

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

添加中文支持。

- git bash 终端输入命令：`git config --global core.quotepath false`
- 在git bash的界面中`右击`空白处，弹出菜单，选择`Options->Text->Locale改为zh_CN, Character set改为UTF-8`

2 ssh配置

2.1 查看密钥

win查看`C:\Users\yourname\`该目录下面有没有`.ssh`文件夹。查看是否存在以下文件：`id_rsa`（私钥）、`id_rsa.pub`（公钥）。如果有，进行步3，如果没有进行步骤2。

2.2 生成密钥

在`git bash`中执行

```
ssh-keygen -t ed25519 -C "your_email@example.com"
# 如果上面不行，执行下面的
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

一路回车。

生成成功后，你会看见`Your public key is: /c/Users/YourName/.ssh/id_ed25519.pub`。

2.3 复制粘贴密钥

前面得到了`.pub`这个文件，复制其内容，粘贴到github的ssh配置那里，github->个人设置->ssh配置。

3 创建远程仓库

1. 首先在github创建一个仓库，❌ **不要勾选** “Add a README file”、“Add .gitignore”、“Choose a license”。
2. 在本地创建项目文件夹，使用`git init`初始化项目
3. 加入一些文件
4. 将文件加入暂存区

```
git add main.c
git add src/
git add *.h
git add .   # 大部分情况
```

5. 提交文件保存

```
git commit -m "描述你的更改，例如：添加传感器初始化代码"
```

6. 连接远程仓库

最好选用ssh连接远程仓库。

```
git remote add origin git@github.com:yourname/reponame.git
```

7. push到远程

注意需要提前配置好git的用户名和邮箱。

注意需要提前配置好ssh。

注意避免当前分支和要Push的远程分支名不同。

```
# 1. 重命名当前分支 master 为 main
git branch -m master main
# 2. 推送到远程 main
git push -u origin main # 第二次及以后只需要git push
```

4 git的常用命令

当然可以！下面是一份 **Git 常用命令速查表**，按使用场景分类，适合日常开发（包括 STM32 项目管理），简洁清晰，建议收藏 ✅

4.1 配置类

命令	说明
`git config --global user.name "Your Name"`	设置全局用户名
`git config --global user.email "you@example.com"`	设置全局邮箱
`git config --global core.editor "code --wait"`	设置默认编辑器（如 VS Code）
`git config --global init.defaultBranch main`	设置默认分支名为 main
`git config --list`	查看所有配置

💡 首次使用 Git 必须配置 `user.name` 和 `user.email`

4.2 仓库初始化 & 克隆

命令	说明
`git init`	初始化当前目录为 Git 仓库
`git clone https://github.com/user/repo.git`	克隆 HTTPS 仓库
`git clone git@github.com:user/repo.git`	克隆 SSH 仓库（需配置 SSH）

4.3 查看状态 & 差异

命令	说明
`git status`	查看工作区状态（哪些文件被修改/未跟踪）
`git diff`	查看**工作区 vs 暂存区**的差异
`git diff --cached`	查看**暂存区 vs 最后一次提交**的差异
`git log`	查看提交历史（简洁版）
`git log --oneline`	单行显示提交历史
`git log --graph --all --oneline`	图形化查看分支历史

4.4 暂存 & 提交（核心！）

命令	说明
`git add <file>`	将指定文件加入暂存区
`git add .`	添加当前目录所有更改（包括新文件）
`git add -u`	添加所有**已跟踪文件**的修改和删除（不包括新文件）
`git commit -m "提交信息"`	提交暂存区内容到本地仓库
`git commit -am "提交信息"`	**跳过 add**，直接提交所有已跟踪文件的修改（不包括新文件）

⚠️ `git commit -a` 不会添加新文件（**untracked files**）！

4.5 远程仓库操作

命令	说明
`git remote -v`	查看远程仓库地址
`git remote add origin <url>`	添加远程仓库（首次）
`git remote set-url origin <new-url>`	修改远程仓库地址（如 HTTPS → SSH）
`git push -u origin main`	首次推送并设置上游分支
`git push`	推送当前分支到远程（后续使用）
`git pull`	拉取远程更新并合并（= `fetch` + `merge`）
`git fetch`	仅下载远程更新，不自动合并

🔑 推荐：首次推送用 `git push -u origin main`，之后只需 `git push`

4.6 分支管理

命令	说明
`git branch`	列出本地分支（* 表示当前分支）
`git branch -m old new`	重命名当前分支
`git checkout -b dev`	创建并切换到 dev 分支
`git switch -c dev`	（Git 2.23+）同上，更语义化
`git checkout main`	切换到 main 分支
`git merge dev`	将 dev 分支合并到当前分支
`git branch -d dev`	删除已合并的 dev 分支
`git branch -D dev`	强制删除 dev 分支

4.7 撤销 & 回退

命令	说明
`git restore <file>`	丢弃工作区修改（恢复到暂存区状态）
`git restore --staged <file>`	从暂存区撤回（取消 add）
`git reset --hard HEAD~1`	**危险！** 回退到上一次提交（丢弃最近一次提交及所有更改）
`git reset --soft HEAD~1`	回退提交，但保留更改在暂存区
`git revert <commit-id>`	新建一个“反向提交”来撤销某次更改（安全，适合已推送的提交）

⚠️ `reset --hard` 会**永久删除代码**，慎用！

4.8 标签（Tag）

命令	说明
`git tag v1.0`	创建轻量标签
`git tag -a v1.0 -m "Release 1.0"`	创建带注释的标签
`git push origin v1.0`	推送单个标签
`git push origin --tags`	推送所有标签

4.9 忽略文件（.gitignore）

在项目根目录创建 `.gitignore` 文件，例如：

```
# 编译产物
*.bin
*.hex
*.elf
Debug/
Release/

# IDE
.project
.cproject
.settings/
.metadata/

# OS
.DS_Store
Thumbs.db
```

✅ `.gitignore` 只对**未跟踪文件**生效。已跟踪的文件需先 `git rm -r --cached .` 再提交。

4.10 小技巧

- **查看帮助**：`git help <command>`（如 `git help commit`）
- **自动补全**：Git Bash 支持命令/分支名自动补全（按 Tab）
- **别名设置**（可选）：

```
git config --global alias.st status
git config --global alias.co checkout
git config --global alias.br branch
git config --global alias.ci commit
git config --global alias.unstage 'restore --staged'
```