六：Git 配置和帮助

一般 git 工具安装好了之后，我们配置 git 的工作环境，Git 提供了一个叫做 git config 的工具专门用来配置或读取相应的工作环境变量。而正是由这些环境变量，决定了 Git 在各个环节的具体工作方式和行为。这些变量可以存放在以下三个不同的地方：

- /etc/gitconfig 文件：系统中对**所有用户**都普遍适用的配置。若使用 `git config --system` ，读写的就是这个文件。
- ~/.gitconfig 文件：用户目录下的配置文件只适用于该用户。若使用 `git config --global` ，读写的就是这个文件。
- 当前项目的 Git 目录中的配置文件（也就是工作目录中的 .git/config 文件）：这里的配置仅仅针对当前项目有效。每一个级别的配置都会覆盖上层的相同配置，所以 .git/config 里的配置会覆盖 /etc/gitconfig 中的同名变量。

在 Windows 系统上，Git 会找寻用户主目录下的 .gitconfig 文件。主目录即 $HOME 变量指定的目录，一般是 C:\Users\meng.li.ssh。此外，Git 还会尝试找寻 /etc/gitconfig 文件，需要看当初 Git 装在什么目录，就以此作为根目录来定位。

- **用户信息**

首先需要配置的是你个人的用户名和电子邮件，这个配置是非常重要的，每次git提交都会引用这两条信息，说明是谁提交了更新。

```
$ git config --global user.name "xxx"
$ git config --global user.email xxx@example.com
```

`-- global` 选项代表更改的配置文件是位于用户主目录下的那个，以后所有的项目默认使用这里的配置用户和邮箱信息。如果你的某个项目需要使用特定的用户名或者邮箱，则只需要去掉 --global 重新配置即可，新的设定保存在当前项目的 .git/conifg 文件里。

- **文本编辑器**

git 需要你输入一些额外信息的时候（如 git commit）会自动调用一个外部编辑器给你使用，默认会使用操作系统指定的默认编辑器，一般可能是 Vi 或者 Vim。 如果你偏好使用其他的编辑器，如 Emacs 的话，则可以重新设置：

```
$ git config --global core.editor emacs
```

- **差异分析工具**

还有个比较常用的是，在解决合并冲突时使用哪种差异分析工具，比如改用 vimdiff 的话：

```
$ git config --global merge.tool vimdiff
```

- **颜色配置**

在操作 git 的时候，我们运用 git branch 查看分支或者 使用 git diff 查看修改变化，看到的颜色是同一个颜色，我们可以进行如下配置，修改颜色。

```
git config --global color.diff auto //git diff要显示的颜色
git config --global color.status auto //git status 要显示的颜色
git config --global color.branch auto //git branch 要显示的颜色
git config --global color.log auto //git log 要显示的颜色
```

上面的命令可以用一条命名实现：

```
git config --global color.ui true
```

- **查看配置信息**

要查看配置信息，可以使用 git config --list

```
xxx@xxxxxxx:~/Android/Code/mocor_sc7731eo1$ git config --list
user.name=xxx
user.email=xxx@example.com
color.diff=auto
color.status=auto
color.branch=auto
color.log=auto
```

查看全局配置可以使用

```
git config --global --list
```

有时候会看到重复的变量名，说明它们来自不同的配置文件(如：/etc/gitconfig 和 ~/.gitconfig),不过最终git实际会采用后面一个。

- **获取帮助**

想了解 git 的各种工具怎么使用，可以阅读它的使用帮助，有三种方法获取帮助

```
$ git help <verb>
$ git <verb> --help
$ man git-<verb>
```

如获取 config 命名怎么用，运行

```
$ git help config
```

```
limeng@revoserverx:~/Android/Code/mocor_sc7731eo1$ git help config
GIT-CONFIG(1)                                                                 Git Manual

NAME
       git-config - Get and set repository or global options

SYNOPSIS
       git config [<file-option>] [type] [-z|--null] name [value [value_regex]]
       git config [<file-option>] [type] --add name value
       git config [<file-option>] [type] --replace-all name value [value_regex]
       git config [<file-option>] [type] [-z|--null] --get name [value_regex]
       git config [<file-option>] [type] [-z|--null] --get-all name [value_regex]
       git config [<file-option>] [type] [-z|--null] --get-regexp name_regex [value_regex]
       git config [<file-option>] [type] [-z|--null] --get-urlmatch name URL
       git config [<file-option>] --unset name [value_regex]
       git config [<file-option>] --unset-all name [value_regex]
       git config [<file-option>] --rename-section old_name new_name
       git config [<file-option>] --remove-section name
       git config [<file-option>] [-z|--null] -l | --list
       git config [<file-option>] --get-color name [default]
       git config [<file-option>] --get-colorbool name [stdout-is-tty]
       git config [<file-option>] -e | --edit
```

- **git命名别名**

如果我们想偷懒，少敲几个字符，我们可以使用 git config 为命令设置别名。

```
$ git config --global alias.co checkout
$ git config --global alias.br branch
$ git config --global alias.ci commit
$ git config --global alias.st status
```

现在如果要提交的话，只需要输入 git ci

使用这种方式我们可以简写一些比较复杂的命令，如查看分支合并和提交信息的命令

```
git config --global alias.mergelog 'log --pretty=oneline --abbrev-commit --graph'
```

现在我们直接输入 `git mergelog` 就可以查看