测试

昨天，我开始创建一个 Shadowsocks Outline 的自动配置工具，目的是将其打造成一个供他人使用的 Python 项目。我开发了一个脚本，通过解码来自 `ssconfig` 文件的 Shadowsocks URL，更新 `config.yaml` 文件中的 Shadowsocks 代理配置。此外，我还创建了另一个脚本，使用 `gsutil` 将订阅文件上传到 Google Cloud Storage，供客户端使用。

我使用了 Windsurf，一个 AI 代码编辑器，来辅助完成这项工作。然而，它在处理 Python 单元测试中的模拟依赖时遇到了困难。

回顾 Yin Wang 分享的测试经验，我想起了他在 Google 工作时的经历。他在那里开发了一个 Python 解释器，并为公司的代码搜索功能编制索引。他的同事坚持要写测试，这让他感到厌烦。他认为编写优雅的代码比测试更重要，并且他的同事只理解表面的东西，而没有抓住本质。

我意识到自己的错误；AI 没有指出这一点。我应该确保库的主要代码是牢固的，然后再关注测试。这个原则同样适用于概念验证项目。在以前的工作中，比如启动一个微服务，测试应该在微服务有一些 API 或功能之后编写。

如果 Windsurf 能很好地处理测试部分，我就不会有这个抱怨了。然而，这里有两个不同的问题：实施测试的时机和正确编写测试的方法。目前，我们专注于前者。这两个问题在一定程度上是相关的。如果一个 AI 代码编辑器或人类发现编写测试代码很容易，测试的时机可能显得微不足道。然而，编写测试所需的努力与编写主要代码相当，这使得时机成为一个重要的考虑因素。

从协作的角度来看，测试的方法可以有所不同。对于个人项目，我可能会在创建测试之前编写大量代码。然而，在团队工作时，通常最好为每个代码片段或功能编写测试。但这并不总是正确的；这取决于团队如何协作。更准确地说，测试应该为团队成员共享的代码编写。目标是确保代码质量，因此在交付代码之前，每个团队成员都可以自由选择他们的测试时机。

在一次过去的工作经历中，我与另外三名后端工程师合作开发了一个功能，花了半年时间才交付。从测试的角度来看，本文讨论的观点可能解释了为什么当时的开发速度较慢。

从协作的角度来看，负责主要代码的人也应该负责相关的测试。任务应尽量减少交织，每个团队成员的责任应清晰明确。

回到测试本身，AI 代码编辑器在这方面也缺乏优化，突显了改进的空间。这个原则不仅限于软件工程；它也适用于硬件和其他领域。测试是一种优化形式，正如那句话所说，“过早的优化是万恶之源”。

记住工作的主要目标是至关重要的。虽然流程和程序是不可避免的，但我们必须牢记什么是真正重要的。

参考资料：

- 测试驱动开发, Yin Wang

- 测试的道理, Yin Wang