环境变量概述与配置

环境变量是什么？为什么要配置环境变量？

环境变量是操作系统中用于定义系统或应用程序运行时环境的一组动态变量。这些变量包含一些配置信息，比如路径、设置、用户名、主机名等，操作系统和程序可以根据环境变量的值来确定某些行为。

1. 环境变量的作用

- **配置系统和应用程序的运行环境**：环境变量可以传递一些配置信息，例如数据库连接、API 密钥、文件路径等，使得不同程序或服务在不同的环境中能够动态调整行为。
- **便于共享信息**：操作系统和不同的应用程序可以通过环境变量共享信息，无需在程序中硬编码路径或配置。
- **提高可移植性**：使用环境变量可以让程序在不同的环境中运行时，不需要修改源代码，只需修改环境变量。例如，一个应用程序可以在开发、测试和生产环境中使用不同的数据库，只需要通过配置不同的环境变量指向不同的数据库即可。
- **安全性**：一些敏感数据（例如密码、密钥）可以通过环境变量来管理，而不是硬编码在代码中，减少了敏感信息泄露的风险。

2. 为什么要配置环境变量

- **设置系统搜索路径**：通常，操作系统通过环境变量 `PATH` 来定义可执行程序的搜索路径，这样用户可以在任意目录下运行程序，而不需要提供完整路径。
- **简化配置管理**：通过环境变量，可以方便地管理系统的配置，而不需要硬编码在程序中。这样在不同环境下（如开发环境、测试环境、生产环境）可以通过修改环境变量来调整程序行为，而不需要修改代码。
- **跨平台支持**：环境变量允许程序更灵活地适应不同平台上的差异，例如 Windows 和 Linux 系统之间的文件路径结构不同，通过配置环境变量可以处理这些差异。
- **集中管理配置信息**：当一个系统中有多个程序或服务需要共享某些配置（如数据库连接信息、缓存配置等），通过环境变量可以集中管理这些配置，方便更新和维护。

常见环境变量

- `PATH`：指定系统查找可执行文件的路径。
- `HOME`：表示当前用户的主目录。
- `USER`：表示当前登录的用户名。
- `SHELL`：表示当前使用的 shell 类型。
- `LANG`：设置系统的语言和区域信息。

示例：配置环境变量

假设你需要配置一个 Python 项目的数据库连接信息，可以通过设置以下环境变量来避免把数据库的密码硬编码到代码中：

```bash
export DATABASE_URL="mysql://user:password@localhost/dbname"
```

在 Python 代码中，可以通过以下方式读取：

```python
import os
db_url = os.getenv("DATABASE_URL")
```

这样，当你需要更改数据库连接时，只需修改环境变量，而不必更改代码。