Claude Code笔记

Claude Code 使用教程

目录

Claude Code 是什么?

Claude Code 是 Anthropic 公司于2025年5月正式发布的AI编程助手

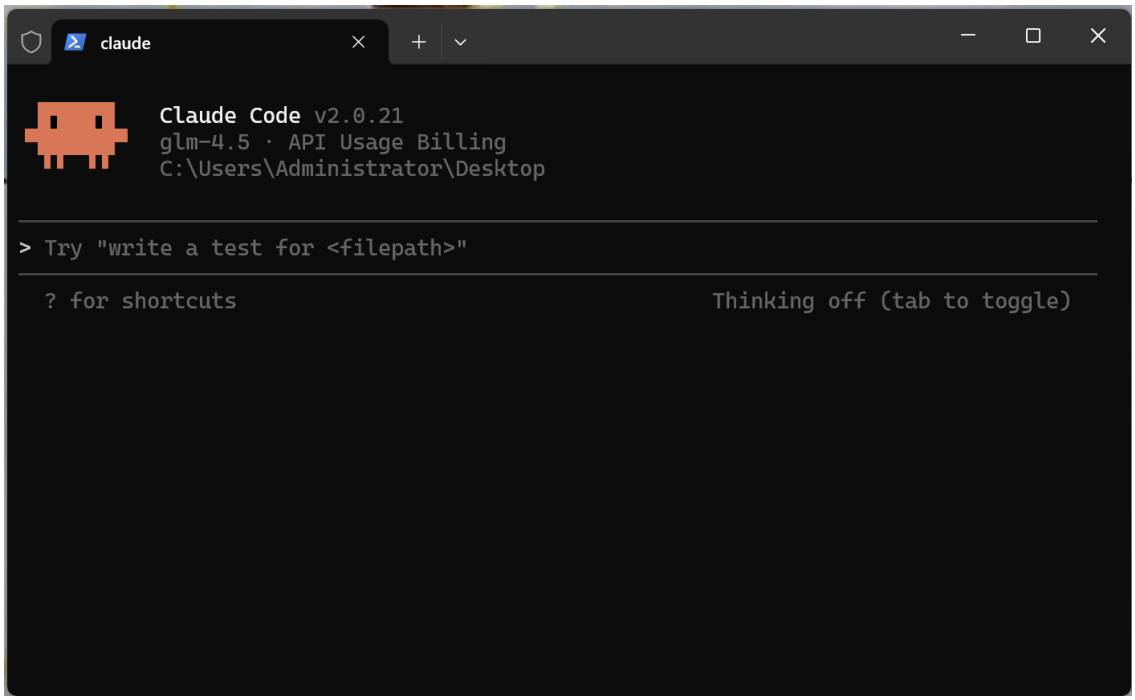
Claude Code 是一款基于终端原生的 AI编程助手。它通过 Multi-Agent 模式 协同工作，**具备 项目级理解 能力，能够直接理解开发者的高级意图，并自主规划、执行从代码修改、系统操作到任务管理的完整开发工作流，最终交付实际成果**。于是Claude Code不仅是一种AI工具，更代表了一种新的编程范式 **Vibe Coding** 。

关键词：终端　AI编程助手　Multi-Agent模式　项目级理解

Vibe Coding 核心理念包括：

- 去技术门槛：开发者无需深入编程知识，仅需用自然语言描述功能或设计意图；

- 人机协作：开发者与AI持续对话，通过迭代提示词（prompt）优化输出结果，而非手动编写代码；
- 效率优先：将传统“编写-调试-优化”的线性流程压缩为“需求-生成-测试”的闭环，大幅缩短开发周期。



Claude Code 强大的核心在于模型的强大，Anthropic 公司的Claude 4 Sonnet 和 Opus模型包括感知、规划、行动和记忆等的Agent 能力非常强大。

Q：什么是Agent -- 大模型的感官与四肢

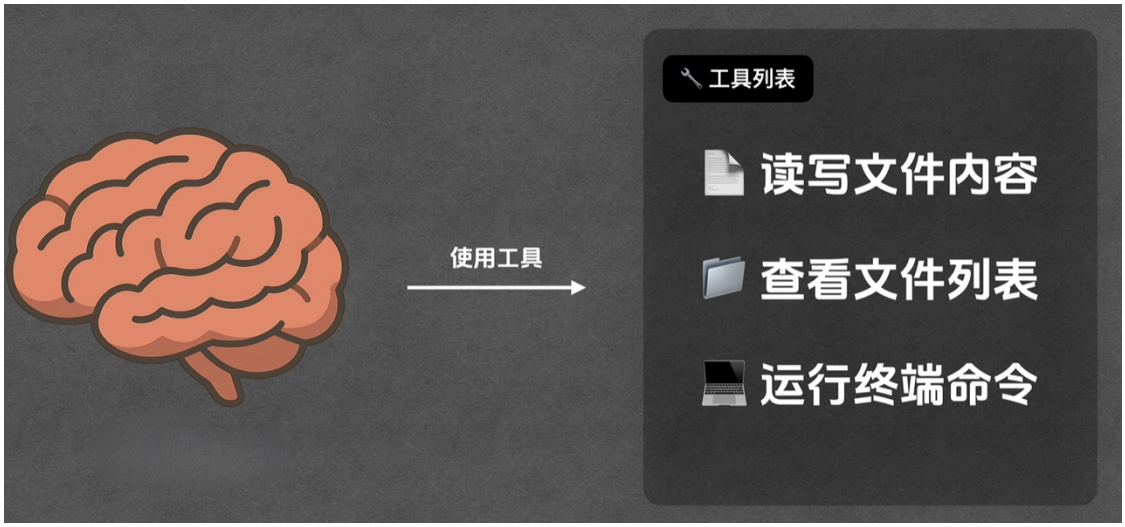
我们知道现在的大模型，比如GPT-5、Deep Seek等，有很强的逻辑能力，擅长与用户进行各种类型的交互问答，但是他无法感知或改变外部环境。

可视化AI工具（Deep Seek、Chat GPT网页版）

- 交互直观，适合探索性任务
- 需要复制粘贴代码

- 无法直接操作文件系统
- 对话式交互，需要手动执行建议

这个限制有没有办法解决？ 给大模型接入一些工具，如同给大模型配备感官和四肢，这样大模型可以自行查询、阅读、编辑已有文件，进行代码编写等全自动流程。把大模型和一系列工具组装起来，就变成了能感知和改变外部环境的智能体，这个智能体就叫Agent。



IDE AI助手

IDE（集成开发环境）是用于提供程序开发环境的应用程序，集成了代码编辑、编译、调试等功能的工具套

- 工作场所：固定在IDE（如VS Code）这个“办公室”里
- 核心任务：设计和建造——写代码、设计架构、调试、解释代码
- 使用工具：代码编辑器、编译器、调试器、语法分析器
- 权限：主要访问你打开的项目文件，在编辑器内操作

Claude Code 内置有 18 个工具，这些工具能帮助Claude Code完成检索代码，执行命令，TODO 读写，架构分析等多项工作。模型借助这些工具可以很好地模拟程序员解决问题的思路：制定计划、分析问题、检索代码库找到相应的代码位置、解决问题、测试验证。

Claude Code

CLI（命令行界面）是操作系统或应用程序提供的基于文本的交互式界面，用户通过键盘输入指令进行操作

- 工作场所：在整个操作系统的“工地”上跑动

- 核心任务：协调全局和执行系统及操作——统领全局、操作文件、自动重构代码、生成测试、修复Bug、部署服务等
- 使用工具：系统命令、脚本、各种命令行工具
- 权限：可以访问整个文件系统、运行任何系统命令、操作环境变量

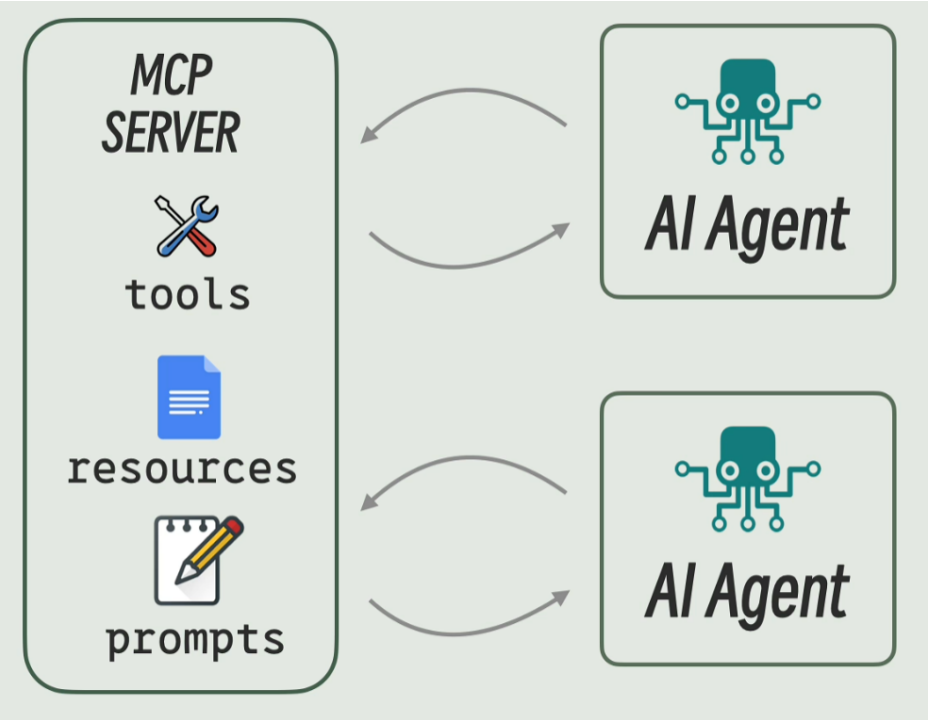
特性维度	可视化AI工具（高级顾问） *如: DeepSeek, ChatGPT网页版*	IDE AI助手（首席架构师） *如: Cursor, VS Code Copilot*	CLI Agent（现场执行总监） *如: Claude Code*
工作环境	网页聊天界面	集成开发环境内部	系统终端/命令行
核心焦点	提供信息、创意、解释和代码建议	代码本身：编写、补全、调试、重构	任务执行：操作文件、运行命令、管理流程
代码库理解	无，依赖用户粘贴上下文	**片段级**和**文件级**理解	**项目级**理解，能洞察全局架构
系统权限	无，无法直接操作	仅限于IDE沙盒内，对打开的文件进行操作	完全的系统权限，可访问整个文件系统、运行任何命令
交互模式	对话式，需用户手动执行建议	响应式，分析用户需求，在IDE内提供代码块或建议，用户审核、接收并执行后续操作	代理式，分析用户需求并规划任务，自动执行全系列任务，整合并展示最终结果
输出形式	文本、代码块建议	代码片段、代码修改建议、解释	直接的操作结果：修改后的文件、命令输出、提交记录等
最佳适用场景	探索性问答、学习概念、头脑风暴、获取初步代码思路	高效编码、代码调试、理解复杂代码、在文件内部进行重构	自动化复杂工作流：搭建项目、编写测试、修复bug、部署服务

模型自身不可能知晓和集成世界上所有的工具和数据，AI Agent也需要不断学习。 如果我想要根据自己的需求自定义大模型的工具列表，或者需要大模型提供最新的数据。我们可以将Tools写入Agent程序，通过函数调用来使用工具或数据，这是最直接的方法。如果想要更灵活快捷，那么我们就需要了解MCP。

Q: 什么是MCP -- 扩展能力的引擎

首先我们需要了解 **MCP Server** 的概念：将实用的工具、数据等变成服务统一管理，让所有的Agent都可以调用。而用来规范Agent与Agent Tool之间交互的通讯协议，就是**MCP（模型上下文协议）**。用通俗的语言来理解，MCP 可以看作是一种针对 Agent 的能力扩展协议，帮助 Agent 管理工具、数据和提示词模板等。

运行或管理Tools的服务叫做 **MCP Server**，调用 MCP Server 的 Agent 叫做 **MCP Client**。



核心特点：

- **直接在终端工作**：不是聊天窗口或代码编辑器，而是在直接命令行里工作；
- **理解整个代码库**：不像Cursor只能看片段代码，它能理解你整个项目的架构；
- **真正能干活**：不是只给建议，而是直接修改文件、运行命令、提交代码；
- **可配置智能体**：Claude Code 可以看作是multi-agent架构，其中Claude Code充当唯一的主Agent角色，来跟用户进行输入输出的交互。
  - 当面临比较简单的任务时：主Agent就会走单Agent的模式，争取通过一次对话解决用户的问题。
  - 当面临较为复杂的任务时：主Agent会智能的指定一个或者多个子Agent来完成任务，这里的子Agent可以理解成Agent Tool。拿到子Agent的结果后，主Agent就会将结果进行整合，按照System Prompt规定的要求返回给用户。

Claude Code代表了一种新的编程范式。传统编程要求开发者精确地描述“如何做”，而 Vibe Coding 则允许你专注于定义 “做什么”和“要达到什么效果”。你只需提供高层的意图、方向和上下文（即所谓的 "Vibe"），AI 代理便会自主规划并执行具体的实现步骤。Claude Code 正是实现 Vibe Coding 的理想载体。

Claude Code的安装、订阅、API获取流程

1. 系统要求

支持的操作系统：

- Windows 10+　Claude Code 自 1.0.51 版本起已正式支持 Windows 原生安装，无需依赖WSL
- macOS 10.15+
- Linux (Ubuntu 18.04+, CentOS 7+)

环境要求：

- Node.js 18+
- Python 3.8+ (某些功能需要)
- Git

2. 安装流程

1. **下载git**

访问 https://git-scm.com/downloads/win，安装时全都下一步，不要修改路径

2. **下载nodejs**

访问 https://nodejs.org/zh-cn/download，安装时全都下一步，不要修改路径

验证安装

代码块

```
node --version
git --version
claude --version
```

3. **安装Claude**

访问https://claude.com/product/claude-code

验证安装

代码块

```
claude --version
```

3. API获取与环境配置

目前国内使用Claude Code较为困难，需要纯净境外ip魔法/境外支付，还要承担高概率封号的风险。主流解法有第三方拼车代订阅和镜像站两种方案。

https://www.aicodemirror.com/

配置环境变量

需要设置以下三个环境变量：

- 变量名：`ANTHROPIC_BASE_URL`，变量值：`https://api.aicodemirror.com/api/claudecode`
- 变量名：`ANTHROPIC_API_KEY`，变量值：`你的密钥`
- 变量名：`ANTHROPIC_AUTH_TOKEN`，变量值：`你的密钥`

使用其他模型接入Claude Code（更新最新的模型）

如果我们不想将选择限制在 Claude 上，有没有其他选择？推荐一些Claude Code 还能接入的其他模型，获取API KEY后相同的步骤进行环境配置，或直接修改claude的配置文件接入其他模型进行使用。

1. OpenAI模型

- **GPT-5 Pro**: 最先进的通用模型，代码能力强

- **GPT-5**: 业界最强程序涉及与自主任务模型

- **GPT-4o**

2. Google模型

- **Gemini 2.5 Pro**: Google最新大模型

- **Gemini 2.5 Flash**: Google首个混合推理模型

3. KIMI

- **kimi-k2-0905-preview**

- **kimi-k2-turbo-preview**

4. 智谱AI

- **GLM系列**: 包括GLM-4.6、GLM-4.5v等

5. DeepSeek

- **deepseek-chat**：对应 deepseek v3.2 非思考模式

- **deepseek-reasoner**：对应 deepseek v3.2 思考模式

6. 阿里

- **通义千问**: 通用对话模型Qwen3

- **通义万相**: 多模态模型

https://docs.bigmodel.cn/cn/guide/develop/claude#%E6%8E%A8%E8%8D%90%E5%AE%89%E8%A3%85%E6%96%B9%E5%BC%8F

Claude Code基本使用技巧

1. 基础命令

启动Claude Code：

`claude`

基本交互：

Shift+Tab

`Shift+Tab`：切换模式

/init

打开一个项目第一件事：使用 `/init`
`/init` 会通读整个项目目录下的所有文件，会在项目目录下生成一个 `CLAUDE.md` 文件

你的 `CLAUDE.md` 文件成为 Claude 提示词的一部分，因此应该像任何经常使用的提示词一样进行优化，而不是添加大量内容而不迭代其有效性。

可以手动向 `CLAUDE.md` 添加内容，或按 # 键给 Claude 一个指令或记录命令、文件和风格指南，它会自动将其纳入相关的 `CLAUDE.md`。

/clear和/compact

`/clear`：清除上下文

在长时间的会话中，Claude 的上下文窗口可能会充满不相关的对话、文件内容和命令。这可能会降低性能，有时会分散 Claude 的注意力。在任务之间频繁使用 `/clear` 命令来重置上下文窗口。

`/compact`：压缩上下文

在上下文tokens要达到上限的时候，使用该命令会提要上下文进行压缩，可以在后面加上对压缩的要求，如：/compact 主要保留前端相关的对话

/resume

`/resume`：找到所有的历史话题

在此基础上双击 `Escape`，可以选择历史对话进行继续对话

/ide

`/ide`：将Claudecode与IDE集成，需要在对应IDE工具中添加Claudecode插件。

- 在VSCode里面选中的代码，在Claudecode中可以感知到
- 当Claudecode修改代码时，会在VSCode弹出一个页面对比修改前后的差异

/permission

`/permission`：权限控制

比如在Allow下添加一行：

`Bash（npm install *）` #代表所有的npm install命令不需要经过同意直接执行

#的作用

输入 `#` 时，进入记忆模式，随后输入的指令都会被Claude code作为长期记忆，在~/.claude/CLAUDE.md（用户级别）配置文件中添加这条语句作为规则；项目级别保存在./claude.md中。

在输入框输入#添加记忆，如 `#Always reply in Chinese.`

选择User memory，会在~/.claude/CLAUDE.md文件中添加这条语句作为规则

！的作用

使用 `!` 可以把对话窗口切换成普通命令行模式，方便执行一些临时的命令而不需要另开终端，且该过程会记录到上下文，避免后续AI在工作时重复执行一些指令。

MCP Server

模型上下文协议，AI与外部工具的中间层，可访问、操作外部工具

退出Claudecode，使用命令行，查看MCP Server配置文件进行参数填写，例如：

`claude mcp add context7 -- npx @upstaash/context7-mcp`

添加--scope user参数可以将MCP配置到全局

自定义命令

打开项目目录中的.claude文件夹，新建文件夹commands。（同样可以在Claudecode配置文件中添加，变为用户级别命令）

在该文件夹下新建文件就可以自定义命令，文件名为命令名称，可以使用自然语言描述这个自定义命令的作用。

使用 `$ARGUMENTS` 作为占位符，可在使用该自定义命令时添加参数

代码块

```
1  "description": "分析项目结构和代码质量",
2  "prompt": "请分析当前项目的结构，检查代码质量，识别潜在问题，并给出改进建议。重点关注：代码组织、依赖管理、性能问题、安全风险。"
```

自定义Agent （Subagent）

使用 `/agent` 自定义agent帮助我们处理特殊任务。

其他指令

`Claude -r`：查询历史会话

`Claude -c`：回到上一次会话

`Claude -p` "问题”：非交互模式（临时命令），使Claudecode变成命令行里的智能助手。

2. 优秀的使用习惯

探索、规划、编码、提交工作流

这种多功能工作流适合许多问题：

1. 要求 Claude 阅读相关文件、图像或 URL，提供一般指针（"阅读处理日志的文件"）或特定文件名（"阅读 `logging.py`"），但明确告诉它暂时不要编写任何代码。
    - 这是工作流中你应该考虑大量使用子代理的部分，特别是对于复杂问题。告诉 Claude 使用子代理来验证细节或调查它可能有的特定问题，特别是在对话或任务的早期，往往能保留上下文可用性，而在效率损失方面没有太多缺点。
2. 要求 Claude 制定如何处理特定问题的计划。我们建议使用"`think`"这个词来触发扩展思考模式，这给了 Claude 额外的计算时间来更彻底地评估替代方案。这些特定短语直接映射到系统中不断增加的思考预算级别：`"think"` < `"think hard"` < `"think harder"` < `"ultrathink"`。每个级别都为 Claude 分配逐步增加的思考预算。
    - 如果这一步的结果看起来合理，你可以让 Claude 创建一个文档或 GitHub issue 包含其计划，这样如果实现（第3步）不是你想要的，你可以重置到这个点。
3. 要求 Claude 在代码中实现其解决方案。这也是一个要求它在实现解决方案的各个部分时明确验证其解决方案合理性的好地方。
4. 要求 Claude 提交结果并创建 pull request。如果相关，这也是让 Claude 更新任何 `README` 或更改日志的好时机，解释它刚刚做了什么。

步骤 #1-#2 至关重要——没有它们，Claude 倾向于直接跳到编码解决方案。虽然有时这就是你想要的，但要求 Claude 先研究和规划可以显著提高需要前期深入思考的问题的性能。

修正监督Claude Code

这四个工具有助于路线修正：

1. 在编码前要求 Claude 制定计划。明确告诉它在你确认其计划看起来不错之前不要编码。
2. 在任何阶段（思考、工具调用、文件编辑）按 `Escape` 中断 Claude，保留上下文以便你可以重定向或扩展指令。
3. 双击 `Escape` 跳回历史记录，编辑之前的提示，并探索不同的方向。你可以编辑提示并重复，直到获得你想要的结果。
4. 要求 Claude 撤销更改，通常与选项 #2 结合使用以采取不同的方法。

多个 Claude协同工作

除了独立使用外，一些最强大的应用涉及并行运行多个 Claude 实例。一个简单但有效的方法是让一个 Claude 编写代码，而另一个审查或测试它。类似于与多个工程师合作，有时拥有单独的上下文是有益的：

1. 使用 Claude 编写代码
2. 运行 `/clear` 或在另一个终端中启动第二个 Claude
3. 让第二个 Claude 审查第一个 Claude 的工作

4. 启动另一个 Claude（或再次 `/clear`）来阅读代码和审查反馈
5. 让这个 Claude 根据反馈编辑代码

这种分离通常比让单个 Claude 处理所有事情产生更好的结果。

实际演示案例

案例1：网页版个人管理仪表板

案例2：安卓小游戏：俄罗斯方块

案例3：吃豆豆网页小游戏

案例4：工作中的实践

> 快速获取项目概要，交接项目框架；
> 针对某个代码或函数要求推荐更优算法或逻辑；
> 针对某个代码或函数检查逻辑漏洞。

总结

Claude Code作为一个强大的AI编程助手，通过其独特的命令行交互方式和强大的文件操作能力，为开发者提供了前所未有的编程体验。相比传统的IDE AI和可视化AI工具，Claude Code能够更深入地理解项目结构，提供更精准的代码建议，并能够直接执行复杂的编程任务。

通过本次学习，您应该已经掌握了Claude Code的基本使用方法和一些高阶技巧。当然，最好的学习方式是实践，建议您在自己的项目中尝试使用Claude Code，更深入地探索其强大功能。