目　录

ai工具推荐
AI 助手管理工具: Cherry Studio
Coding
Claude Code + CC Switch 使用指南
CC GUI - AI 编程助手插件
基础知识
MCP 和 CLI

ai工具推荐

AI 助手管理工具: Cherry Studio

Cherry Studio —— 多模型 AI 助手管理工具

> *一站式 AI 智能体管理平台，内置 300+ 预配置助手，支持多模型并行对话*

一、项目简介

Cherry Studio 是一款开源的 AI 助手管理工具，内置超过 300 个预配置 AI 智能体，涵盖学习辅导、编程开发、内容创作、数据分析等各个领域。用户还可根据需求轻松创建专属 AI 助手，实现个性化工作流。

特性	说明
多模型并行	同时打开多个大模型对话窗口，同界面高效协作
丰富智能体	300+ 预配置助手，覆盖全场景需求
自定义助手	支持创建专属 AI 助手，定制提示词与参数
多格式支持	导入 txt、png、docx、pdf 等多种文件
云端备份	数据安全同步，跨设备无缝衔接
实用工具	全局搜索、话题管理、AI 翻译、代码高亮
主题切换	明暗双主题，适配不同使用环境
跨平台	Windows、macOS、Linux 全平台支持

二、下载安装

2.1 官网下载（推荐）

https://cherry-ai.com/download

提供各平台开箱即用的安装包，无需复杂配置。

2.2 GitHub 源码

```
git clone https://github.com/kangfenmao/cherry-studio.git
```

2.3 Gitee 镜像（国内加速）

```
git clone https://gitee.com/mirrors/Cherry-Studio.git
```

三、核心功能

3.1 多模型并行对话

- 同时开启多个大模型对话窗口
- 在同一界面对比不同模型的回答
- 支持 GPT、Claude、Gemini、文心一言、通义千问等主流模型
- 灵活切换，高效辅助完成复杂任务

3.2 智能体市场

分类	示例助手
编程开发	代码审查、Bug 修复、算法优化
学习辅导	论文润色、知识点讲解、考试复习
内容创作	文案撰写、标题生成、SEO 优化
数据分析	数据清洗、可视化建议、统计解读
翻译助手	多语言翻译、术语校对、本地化
生活助手	旅行规划、菜谱推荐、健身计划

3.3 自定义 AI 助手

创建流程：

```
点击「新建助手」→ 设置名称与图标 → 编写系统提示词 → 调整模型参数 → 保存使用
```

支持自定义：

- 系统提示词（System Prompt）
- 温度参数（Temperature）
- 最大 token 数
- 对话上下文长度

3.4 文档处理

支持格式	功能
`.txt`	纯文本导入，快速分析

支持格式	功能
`.png` / `.jpg`	图片识别，OCR 提取文字
`.docx` / `.doc`	Word 文档解析，保留格式
`.pdf`	PDF 全文提取，支持扫描件

操作：直接拖拽文件到对话窗口，AI 自动分析内容。

3.5 云端备份

- 一键备份对话记录与助手配置
- 加密传输，保障数据安全
- 多设备登录，配置实时同步

四、实用工具

功能	说明
全局搜索	快速检索所有对话历史与助手
话题管理	对话分类归档，支持标签与收藏
AI 翻译	选中文字一键翻译，支持多语言互译
代码高亮	自动识别代码块，语法着色显示
主题切换	明亮/暗黑模式，保护视力
快捷键	支持自定义键盘快捷键，提升效率

五、快速上手

5.1 首次启动

1. 安装完成后打开 Cherry Studio
2. 选择主题（明/暗）
3. 配置 API 密钥（支持 OpenAI、Azure、Claude、国产大模型等）
4. 从智能体市场选择助手开始使用

5.2 添加自定义模型

```
设置 → 模型提供商 → 添加 → 填写 API 地址与密钥 → 测试连接 → 保存
```

5.3 创建专属助手

```
智能体 → 新建 → 输入名称 → 编写提示词模板 → 选择默认模型 → 保存
```

示例提示词：

```
你是一位资深 Python 开发工程师，擅长代码优化与性能调优。
请对用户的代码进行审查，指出潜在问题并给出改进建议。
```

六、平台支持

操作系统	安装包	说明
Windows	`.exe` / `.msi`	支持 Win10/Win11
macOS	`.dmg` / `.zip`	支持 Intel 与 Apple Silicon
Linux	`.AppImage` `.deb` `.rpm`	主流发行版兼容

七、相关资源

资源	地址
官网下载	https://cherry-ai.com/download
GitHub	https://github.com/kangfenmao/cherry-studio
Gitee 镜像	https://gitee.com/mirrors/Cherry-Studio.git

> *提示：首次使用建议从智能体市场选择 2-3 个常用助手体验，熟悉后再创建自定义助手，充分发挥多模型并行优势。*

Coding

Coding 简介

什么是 Coding

Coding（编程）是将人类的思维和逻辑，通过计算机语言转化为可执行指令的过程。
它是软件开发、数据分析、人工智能和自动化等现代技术的基础。

Coding 的分类

1. 传统编程

使用经典的编程语言（如 Python、Java、C++）来开发软件和应用程序。
特点：

- 明确的语法和规则
- 高度可控
- 适用于大型项目和复杂系统

2. Vibe Coding

一种创意与直觉驱动的编程方式，通常用于快速原型、互动艺术、小游戏和视觉化项目。
特点：

- 强调快速迭代和“Flow”
- 更关注体验和创意
- 常用工具：Processing, p5.js, Unity

3. AI Coding

利用人工智能技术辅助编码，自动生成、优化或修复代码。
特点：

- 提高开发效率
- 自动生成重复性代码
- 常用工具：GitHub Copilot, ChatGPT, Tabnine

Coding 的作用

- 软件开发：创建桌面应用、Web 应用、移动应用等

- 数据处理：数据清洗、统计分析、可视化
- 自动化：自动执行任务，提高效率
- 创意表达：通过编程实现艺术、游戏和互动项目
- 学习与探索：理解计算机思维和逻辑结构

常用编程语言

- Python：易学，适合 AI、数据分析和 Web 开发
- Java：跨平台，适合大型企业系统
- JavaScript：网页开发必备
- C/C++：性能优化和系统级开发
- Go / Rust：现代高性能编程语言

总结

Coding 不仅是一项技能，更是一种思维方式。
它既可以用于解决实际问题，也可以成为创意表达的工具。
结合不同的方法和工具（如 Vibe Coding 或 AI Coding），可以让编程更高效、更有趣。

Claude Code + CC Switch 使用指南

Claude Code + CC Switch 使用指南

> *从零开始，在 Windows 上安装 Claude Code，通过 CC Switch 接入国产模型*

一、核心概念

1.1 Claude Code 是什么

Claude Code 是 Anthropic 推出的 AI 编程 Agent 框架。它可以：

- 读取项目文件
- 分析代码结构
- 生成和修改代码
- 执行命令
- 自主决定工作流程

关键理解：Claude Code 是"外壳"，需要接入"大脑"（大语言模型）才能工作。

1.2 CC Switch 是什么

CC Switch（Claude Code Switch）是给 Claude Code 换大脑的图形化工具：

- 添加任意兼容 OpenAI API 格式的模型
- 一键切换当前 Claude Code 使用的模型
- 管理多个 API Key 和模型配置

一句话：CC Switch = 给 Claude Code 换大脑的遥控器

1.3 为什么需要 CC Switch

原版 Claude Code 默认搭配官方 Claude 模型，国内用户很难直接使用。通过 CC Switch，可以给 Claude Code 换上国内能直接调用的模型（智谱 GLM、MiniMax、Kimi、DeepSeek 等），无需外国手机号、无需特殊网络、无需 Visa 卡。

二、安装 Claude Code

2.1 方式一：官方脚本安装（推荐）

```
# 在 PowerShell 中执行
irm https://claude.ai/install.ps1 | iex
```

2.2 方式二：winget 安装

```
winget install Anthropic.ClaudeCode
```

2.3 方式三：npm 安装

```
npm install -g @anthropic-ai/claude-code
```

2.4 验证安装

```
claude --version
```

看到版本号即成功。如果提示「claude 不是可识别的命令」，关闭终端重新打开再试。

三、安装 CC Switch

3.1 下载安装包

访问 GitHub Releases：

```
https://github.com/farion1231/cc-switch/releases
```

下载对应平台的安装包：

平台	文件
Windows	`CC-Switch-vX.X.X-Windows.msi` 或 `.zip`
macOS	`CC-Switch-vX.X.X-macOS.dmg`
Linux	`.AppImage` `.deb` `.rpm`

3.2 安装

双击安装包，一路 Next 完成安装。

四、获取 API Key

以 DeepSeek 为例（其他平台类似）：

1. 访问 platform.deepseek.co...
2. 注册账号并登录

3. 进入「API Keys」页面
4. 点击「创建 API Key」
5. 复制生成的 Key（形如 `sk-xxxxxxxxxxxxxxxx`）

其他可选平台：

- 智谱 AI（GLM）：open.bigmodel.cn
- MiniMax：platform.minimaxi.co...
- Kimi（Moonshot）：platform.moonshot.cn
- 硅基流动：cloud.siliconflow.cn

五、CC Switch 配置详解

5.1 打开 CC Switch

在开始菜单找到 CC Switch 并打开。

5.2 添加 Provider

```
点击「添加 Provider」→ 选择「OpenAI Compatible」
```

5.3 填写配置

字段	填写内容
名称	自定义，如 "DeepSeek"
Base URL	API 地址，如 `https://api.deepseek.com/v1`
API Key	从平台复制的 Key
模型	选择或填写模型名称，如 `deepseek-chat`

5.4 常用模型配置参考

DeepSeek：

```
Base URL: https://api.deepseek.com/v1
模型: deepseek-chat / deepseek-coder
```

智谱 GLM：

```
Base URL: https://open.bigmodel.cn/api/paas/v4
模型: glm-5.1 / glm-4-plus
```

硅基流动（多模型聚合）：

```
Base URL: https://api.siliconflow.cn/v1
模型: deepseek-ai/DeepSeek-V3 / Qwen/Qwen2.5-72B
```

5.5 激活配置

配置完成后，点击「激活」按钮，CC Switch 会自动修改 Claude Code 的配置文件，使其使用你设置的模型。

六、启动 Claude Code

6.1 在项目目录中启动

```
# 进入你的项目文件夹
cd D:\projects\my-project

# 启动 Claude Code
claude
```

⚠ 重要：不要在电脑根目录（如 `C:`）启动，信息量太大容易导致分析变慢、误操作。为每个项目建独立文件夹，cd 进去再启动。

6.2 首次启动配置

首次启动会提示：

1. 选择主题（Dark mode / Light mode）
2. 登录验证（如果接的是国产模型，CC Switch 已处理，直接回车）
3. 进入交互界面

七、CLAUDE.md 配置（推荐）

CLAUDE.md 是 Claude Code 的配置文件，用来设定协作规则。

7.1 全局配置

文件位置：

```
Windows: %USERPROFILE%\.claude\CLAUDE.md
macOS/Linux: ~/.claude/CLAUDE.md
```

7.2 项目级配置

在项目根目录创建 `CLAUDE.md` ：

```
my-project/
├── src/
├── CLAUDE.md      ← 项目级配置
└── ...
```

7.3 配置示例

```
# 协作规则

## 语言
- 所有回复使用中文
- 代码注释使用英文

## 安全
- 删除文件前必须确认
- 修改配置文件前必须确认
- 执行危险命令前必须确认

## 代码风格
- 使用 4 空格缩进
- 函数命名使用 camelCase
- 类命名使用 PascalCase

## 项目结构
- 源代码放在 src/ 目录
- 测试文件放在 tests/ 目录
- 文档放在 docs/ 目录
```

八、常用操作

8.1 基本命令

命令	作用
`claude`	启动 Claude Code
`/help`	查看所有命令
`/theme`	切换主题
`/cost`	查看本次会话费用
`/clear`	清空当前对话
`/exit`	退出

8.2 文件操作

```
@file src/main.py      # 引用特定文件
@directory src/        # 引用整个目录
@git diff              # 查看 Git 变更
```

8.3 常用提示词

```
"帮我分析一下这个项目的架构"
"给这个函数加上单元测试"
"把这段代码重构得更简洁"
"解释一下这个报错的原因"
"帮我生成一个 REST API 接口"
```

九、CC Switch 高级功能

9.1 多 Provider 管理

可以同时配置多个平台的 API，随时切换：

- Provider A：DeepSeek（便宜量大）
- Provider B：智谱 GLM（中文优化）
- Provider C：官方 Claude（需要时切换）

9.2 用量统计

CC Switch 提供使用量分析，帮助你监控 API 调用和费用。

9.3 会话管理

- 历史会话记录
- 会话收藏
- 消息导出

十、常见问题

问题	解决
`claude` 命令找不到	关闭终端重新打开，或检查 PATH
API 调用失败	检查 Base URL 和 API Key 是否正确
响应慢	切换更快的 Provider 或模型
中文回复乱码	终端编码设为 UTF-8
费用过高	切换到更便宜的模型（如 DeepSeek）

十一、相关资源

资源	地址
Claude Code 官网	code.claude.com
CC Switch GitHub	github.com/farion123...
CC GUI（IDEA 插件）	github.com/zhukunpen...

资源	地址
DeepSeek 开放平台	platform.deepseek.co...
智谱 AI 开放平台	open.bigmodel.cn
硅基流动	cloud.siliconflow.cn

> *提示：模型会不断迭代更新，但好的 Agent 框架是更持久的基础设施。掌握 Claude Code + CC Switch 的组合，无论底层模型如何变化，你都能快速适配。*

CC GUI - AI 编程助手插件

CC GUI —— IntelliJ IDEA 的 AI 编程助手插件

> *在 JetBrains IDE 内深度集成 Claude Code 与 Codex，支持与 CC Switch 无缝配合*

一、项目简介

CC GUI（原 Claude Code GUI）是一款 IntelliJ IDEA 插件，为 Claude Code 和 OpenAI Codex 提供可视化图形界面。它将 AI 编程能力直接嵌入 IDE，让开发者在不离开编码环境的情况下，享受 AI 辅助编程的便利。

- GitHub：github.com/zhukunpen...
- 原名：Claude Code GUI（为避免商标风险已更名）
- 协议：MIT 开源
- 支持 IDE：IntelliJ IDEA、WebStorm、PyCharm、GoLand、Rider 等 JetBrains 全系

二、核心功能

2.1 双 AI 引擎支持

引擎	提供商	说明
Claude Code	Anthropic	上下文感知编程助手，支持 Opus 系列模型
OpenAI Codex	OpenAI	强大的代码生成引擎

2.2 智能对话系统

- 上下文感知：自动识别当前项目代码上下文
- @file 引用：精准引用特定文件作为对话上下文
- 图片发送：支持上传图片描述视觉需求
- 对话回溯：灵活调整历史对话记录
- 增强提示词：自动优化用户输入，提升 AI 理解准确度

2.3 Agent 自动化

- 内置 Agent：自动化执行复杂多步骤任务

- Skills 命令：斜杠命令系统（ `/init` 、 `/review` 、 `/test` 等）
- MCP 扩展：支持 Model Context Protocol 服务器扩展 AI 能力

2.4 开发者体验优化

功能	说明
代码 DIFF 对比	直观展示 AI 修改前后的代码差异
文件导航	点击即可跳转到相关代码位置
代码跳转	支持跳转到定义、查找引用
主题切换	明暗双主题，适配 IDE 外观
字体同步	自动跟随 IDE 字体设置
国际化	中/英文自动切换

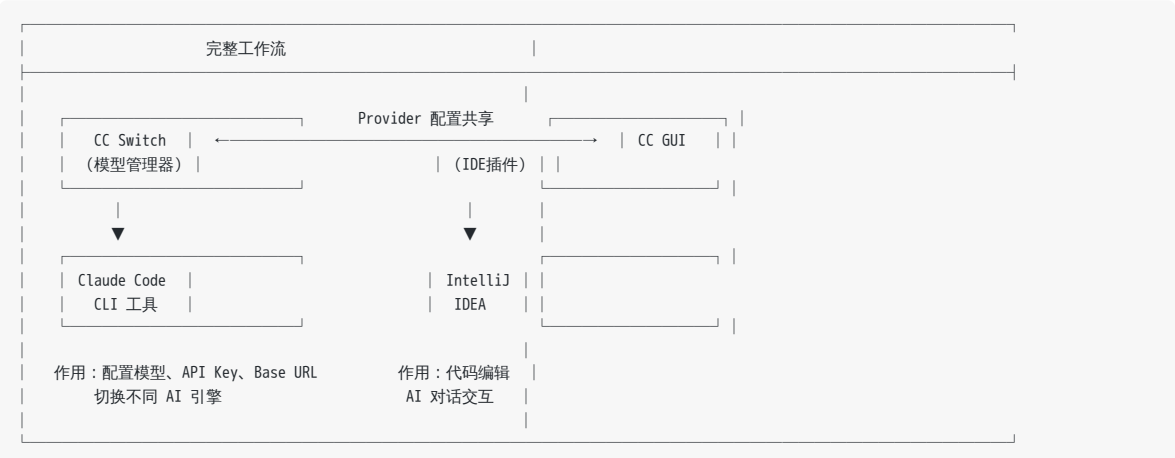
2.5 会话管理

- 历史会话记录与搜索
- 会话收藏（标记重要对话）
- 消息导出（保存为文件）
- 使用量统计分析

三、与 CC Switch 配合使用

3.1 为什么需要配合 CC Switch

CC GUI 是 IDE 内的操作界面，CC Switch 是模型配置管理器。两者配合：



3.2 配合流程

第一步：在 CC Switch 中配置模型

```
打开 CC Switch → 添加 Provider → 填写 API 信息 → 激活
```

支持的 Provider 类型：

- OpenAI Compatible（DeepSeek、智谱、硅基流动等）
- 官方 Claude（需特殊网络）
- 官方 Codex（需特殊网络）

第二步：在 CC GUI 中加载配置

CC GUI 会自动检测 CC Switch 的配置，或手动同步：

```
CC GUI 设置 → Provider → 选择「使用 CC Switch 配置」
```

第三步：在 IDE 中使用

```
打开项目 → 点击 CC GUI 面板 → 选择 AI 引擎 → 开始对话
```

3.3 配置共享说明

配置项	存储位置	共享方式
API Key	CC Switch 加密存储	CC GUI 读取调用
Base URL	CC Switch 配置文件	两者同步
模型选择	CC Switch 激活状态	CC GUI 自动跟随
会话历史	CC GUI 本地存储	独立管理

四、安装方法

4.1 方式一：插件市场安装（推荐）

```
IntelliJ IDEA → 设置 → 插件 → 市场 → 搜索 "CC GUI" → 安装 → 重启 IDE
```

4.2 方式二：离线安装

```
1. 从 GitHub Releases 下载插件包（.zip）
2. IDEA → 设置 → 插件 → 从磁盘安装 → 选择 .zip 文件
3. 重启 IDE
```

4.3 方式三：源码构建

```
git clone https://github.com/zhukunpenglinyutong/jetbrains-cc-gui.git
cd jetbrains-cc-gui
./gradlew clean buildPlugin
# 插件包位于 build/distributions/（约 40MB）
```

五、首次使用

5.1 打开 CC GUI 面板

```
IDEA 右侧工具栏 → 点击 "CCG" 图标 → 打开 AI 对话面板
```

5.2 连接 CC Switch

```
CC GUI 面板 → 设置齿轮 → Provider → 选择「CC Switch」→ 测试连接
```

5.3 开始对话

```
1. 在编辑器中选中代码
2. 右键 → "发送到 CC GUI" 或直接输入问题
3. AI 分析后给出建议
4. 点击「应用修改」或「复制代码」
```

六、典型使用场景

场景一：代码审查

```
选中代码 → 输入 "/review" → AI 自动分析代码问题
→ 查看 DIFF → 点击应用修改
```

场景二：生成测试

```
选中函数 → 输入 "给这个函数写单元测试" → AI 生成测试代码
→ 查看 DIFF → 应用到测试目录
```

场景三：重构建议

```
选中类文件 → 输入 "重构这个类，使用设计模式优化"
→ AI 给出重构方案 → 逐步应用
```

场景四：Bug 修复

```
选中报错代码 → 输入 "修复这个报错"
→ AI 分析错误原因 → 给出修复代码
```

七、安全审计

项目承诺定期安全审计：

- 每个小版本发布前进行 `/security-review` 审计
- 每 10 个小版本进行全面 `claude-code-security` 审计

八、常见问题

问题	解决
CC GUI 无法连接 CC Switch	确认 CC Switch 已启动并激活 Provider
模型响应慢	在 CC Switch 中切换到更快的 Provider
中文显示乱码	IDEA 设置 → 编辑器 → 文件编码 → UTF-8
代码应用失败	检查文件是否被其他程序锁定
插件崩溃	检查 IDEA 版本兼容性，更新到最新版

九、相关资源

资源	地址
CC GUI GitHub	github.com/zhukunpen...
CC Switch GitHub	github.com/farion123...
Claude Code 官网	code.claude.com
JetBrains 插件市场	IDEA 内搜索 "CC GUI"

十、与 CC Switch 的关系总结

维度	CC Switch	CC GUI
定位	模型配置管理器	IDE 内 AI 操作界面
运行环境	桌面系统	JetBrains IDE 内
核心功能	管理 API、切换模型	代码交互、AI 对话
是否必须	是（提供模型接入）	否（可选增强体验）

维度	CC Switch	CC GUI
配合效果	配置一次，多处使用	编码不离 IDE，效率最高

> *建议：先安装配置好 CC Switch，再安装 CC GUI，两者配合使用效果最佳。*

基础知识

AI 基础知识概览

本文整理了大语言模型（LLM）及其核心概念、架构和工作原理，适合初学者快速理解 AI 系统。

一、底层引擎：大语言模型（LLM）

核心定义与架构

- LM 全称：Large Language Model（大语言模型），简称大模型
- 底层架构：基于 Transformer（2017 年 Google 提出的 Attention 机制）
- 工作原理：通过预测下一个最可能的词生成文本

发展里程碑

时间	事件	意义
2017 年	Transformer 架构提出	奠定大模型技术基础
2022 年底	GPT-3.5 发布	首个达到可用级别的大模型
2023 年3月	GPT-4 发布	大幅提升 AI 能力上限
2023 年后	Claude、Gemini 等模型涌现	AI 赛道竞争加剧，OpenAI AI 技术多方竞争

二、数据处理单元：Token

核心特性

- 定义：文本的最小处理单元，通过 Tokenizer 分词
- 编码过程：文本 → Token → Token ID（数字表示）
- 解码过程：Token → 文本

Token 与语言单位关系

语言单位	与 Token 的关系	示例
中文词语	可能拆分	“工作坊” → “工作” + “坊”

语言单位	与 Token 的关系	示例
英文单词	常见单词对应 1 个 Token	"hello" → 1 Token
复杂单词	可能拆分	"helpful" → "help" + "ful"
特殊符号	可能多个 Token 表示	→ 3 Token

Token 参考量

- 1 Token ≈ 0.75 个英文单词
- 1 Token ≈ 1.5-2 个中文字符
- 40 万 Token ≈ 30 万英文单词或 60-80 万汉字

OpenAI Tokenizer 工具

- 网址：https://platform.openai.com/tokenizer
- 功能：
  - 输入文本可直观查看对应 Token
  - 支持不同模型 Token 计算（如 GPT-3、GPT-4、GPT-5）
  - 帮助理解文本长度和 Context Window 消耗
- 用途：
  - 预测 Token 消耗，控制上下文容量
  - 优化 Prompt，避免超过模型限制
  - 调试模型输入输出

三、记忆单元：Context

核心概念

- 定义：模型每次处理任务时能访问的信息总和，类似“临时记忆”
- 组成部分：用户问题、对话历史、工具调用、System Prompt 等
- 容量限制：由 Context Window（上下文窗口）定义

主流模型 Context Window 对比

模型	Context Window（Token）	约合汉字数量
GPT-5.4	105 万	~157.5 万
Gemini 3.1 Pro	100 万	~150 万
Claude Opus 4.6	100 万	~150 万

突破 Context Window 限制

- RAG 技术（检索增强生成）：通过检索相关文档片段，降低 Token 消耗

四、指令交互：Prompt

定义与分类

- Prompt：给 AI 的任务指令，决定输出内容
- 分类：
    - User Prompt：用户具体任务，如“写一首诗”
    - System Prompt：系统设定规则，如“保持幽默风格”

Prompt 工程（Prompt Engineering）

- 核心原则：清晰、具体、明确
- 提示设计技巧：
    - 避免模糊指令
    - 拆分复杂任务
    - 结合上下文提供信息

五、外部能力扩展：Tool

核心作用

- 定义：模型调用外部工具或接口，实现能力扩展
- 工作流程：
    i. 用户输入或平台转发

ii. 模型生成工具调用指令

iii. 工具执行并返回结果

iv. 模型生成最终输出

六、工具标准化：MCP

- 全称：Model Context Protocol（模型上下文协议）
- 作用：统一工具接口，解决多平台调用不一致问题
- 典型示例：OpenAI、Anthropic、Google 各自有接入规范，通过 MCP 统一

七、命令行工具（CLI）

核心概念

- CLI（Command-Line Interface） 是通过命令行与 AI 或工具交互的方式
- 常用于：
  - 快速调用模型功能
  - 自动化任务
  - 与脚本或系统集成
- 优势：
  - 高效轻量，不依赖图形界面
  - 可组合管道命令完成复杂任务
  - 适合开发者、数据科学家和运维场景

典型 CLI 示例

- OpenAI CLI：调用模型生成、管理 API keys
- Gemini CLI / Claude CLI：调度 Agent 执行任务
- Codex CLI：编程辅助和代码生成

八、自主执行系统：Agent

- 定义：能够自主决策、调用工具、完成任务的系统
- 核心能力：多步骤推理、工具选择、流程控制
- 代表产品：Claude Code、Codex、Gemini CLI 等
- 典型模式：React、Plan and Execute

九、任务定制：Agent Skill

- 定义：给 Agent 的能力模块，包含任务规则和执行步骤
- 核心功能：
  - 名称与描述
  - 任务目标
  - 执行步骤、判断规则、输出格式
- 技术实现：
  - Markdown 文档管理
  - 存放于专用目录，方便调用
  - 可附加用户问题映射规则，实现智能指令

十、概念体系关系

LLM（大脑） → Token（数据单元） → Context（记忆空间） → Prompt（交互指令） → Tool（外部能力）→ MCP（工具标准化） → CLI（命令行工具） → Agent（决策系统） → Agent Skill（任务定制）

补充说明

- Transformer 架构奠定基础
- Token 化处理文本，是模型理解的核心
- CLI 是现实场景中操作 AI 的重要接口
- Agent 与 Skill 扩展了 LLM 的实际应用能力
- RAG 技术可突破 Context Window 限制，提高信息调用效率

> *本文整理了 AI 核心概念、工作流程、扩展模块、CLI 工具及历史发展，便于快速入门和实践。*

MCP 和 CLI

本文分析 MCP 和 CLI 在 AI 流程中的优缺点、使用场景和未来趋势。

一、核心问题

项目	说明
MCP	AI 工具标准，告诉模型一次性“我有这些工具”
CLI	命令行工具，类似 `ffmpeg`、`grep`，直接执行任务

二、CLI 胜在两点

1⃣ 省钱（Token 消耗少）

- MCP 模式：需要把所有工具说明发送给 AI
    - 示例：44 个工具说明，占用 >14,000 Token
- CLI 模式：仅用几行命令即可，Token 消耗大幅降低
- 结论：CLI 更经济，尤其在大模型调用频繁时

2⃣ 高效（执行效率高）

- MCP 流程：
    i. AI 思考 → 调用工具 → 等结果
    ii. 再思考 → 再调用 → 多次往返
- CLI 流程：
    i. AI 生成一条命令 → 本地执行所有步骤 → 输出最终结果
- 类比：CLI 像搭积木，工具自由组合，一条命令完成复杂流程

三、MCP 还有用吗？

优势	说明
更可控	参数明确，文件名包含特殊符号也不会出错

优势	说明
更安全	受限制，企业和云端使用更放心；CLI 可能误执行删除命令

> *MCP 并非过时，仍适用于企业和云端场景*

四、未来走向

方式	适用场景
CLI	个人开发者，需要快速、便宜、轻量化的场景
MCP	企业、云端，对安全要求高的场景

- 两者都会存在，各自占据不同领域

五、总结

- CLI 优势：省钱、执行效率高、灵活组合
- MCP 优势：可控、安全，适合企业级场景
- 策略：个人快速开发用 CLI，企业云端使用 MCP，两者结合可形成完整 AI 流程