🏗️ Архитектура скиллов Claude: от простого к модульному

🏗️ Архитектура скиллов в Claude: от одного файла к мини-системе

Ключевые тезисы:

- 🔥 Разница между простым и профессиональным скиллом — не в коде, а в **архитектуре**.
- 🗃️ **Плохая архитектура**: один файл, где всё перемешано (логика, правила, источники). Любое изменение — риск сломать.
- 🏛️ **Хорошая архитектура**: модульная система с отдельными файлами и папками. Каждый файл отвечает за своё. Легко работать, модернизировать и читать.
- 🎯 Цель — превратить «банальный» скилл в собранный инструмент с помощью 4 последовательных апгрейдов.

🎯 Базовый скилл (Исходная точка)

Простой скилл — это **одна папка с одной инструкцией** (`.skill` файлом). В нём описано:

- Что делает скилл.
- Когда запускается.
- Пошаговый план действий.

Пример: Скилл для сбора новостей и формирования дайджеста. Работает, но **непрофессионально**.

🔄 Апгрейд 1: Вынос ресурсов

Проблема: Источники (например, список сайтов) «зашиты» в инструкцию. Чтобы их изменить, нужно лезть в код скилла и вручную искать нужное место.

Решение: Вынести **ресурсы** (списки, данные, конфигурации, источники) в **отдельный файл** (обычно в формате JSON).

Ключевой принцип: *Progressive Disclosure* (постепенное раскрытие). Скилл не держит всё в голове, а **сначала читает файл с источниками**, когда это нужно.

- ✅ Меняешь один файл — остальное не затрагивается.
- ✅ Удобнее и нагляднее для редактирования.

⚙️ Апгрейд 2: Вынос логики (Самый важный технический момент)

Проблема: Логика работы (например, обход сайтов и сбор информации) описана в инструкции. Это непредсказуемо и «съедает» много токенов.

Решение: Вынести **исполняемую логику** в отдельный **скрипт** (например, на Python).

Как это сделать без глубоких технических знаний? Метод *Метапромтинга*:

1. **Шаг 1:** Попросить Claude: «Помоги мне составить промпт для генерации исполняемого файла внутри моего скилла». Опиши задачу своими словами.
2. **Шаг 2:** Claude сгенерирует технически грамотный промпт с правильными терминами.
3. **Шаг 3:** Скопировать этот промпт в новый чат и выполнить. Claude создаст нужный скрипт.

Результат: Главная инструкция лишь **запускает скрипт одной командой**. Скрипт возвращает готовые данные (например, JSON), а Claude тратит токены только на их анализ, а не на выполнение рутинных операций.

📋 Апгрейд 3: Вынос правил

Проблема: Главная инструкция раздувается от триггеров, правил фильтрации, шаблонов оформления.

Решение: Вынести **правила** в отдельные файлы внутри своей папки.

- 📄 Файл с правилами фильтрации.
- 🎨 Файл-шаблон финального вида.
- *(В вашем случае может быть больше: правила перевода, обработки дат, ошибок и т.д.)*

Главная мысль: Главный файл скилла должен быть **картой**, а не **энциклопедией**. Он только ссылается на другие файлы.

⚠️ **Критически важно:** В промпте нужно явно указать: **«Не дублируй эти правила в главной инструкции. Они должны жить только в своих файлах»**. Иначе Claude скопирует всё обратно «на всякий случай» и архитектура развалится.

🌐 Апгрейд 4: Связь с внешним миром

Благодаря модульной архитектуре скилл легко интегрируется с внешними сервисами и другими скиллами.

Пример реализации:

1. **Notion:** После формирования дайджеста скилл автоматически создаёт страницу в указанной базе Notion через коннектор.
2. **Telegram:** Скилл спрашивает пользователя, нужно ли сделать пост. Если «да», то **вызывает другой, уже существующий скилл**, который умеет писать посты в Telegram в определённом стиле.

Итог: Простой скилл превращается в **звено автоматизированного рабочего процесса**.

💎 Выводы и готовое решение

1. **Собирайте скилл за один раз.** Модернизация готового скилла — сложная задача. Лучше пересобрать его полностью по новому промпту.
2. **Используйте готовый шаблон.** Автор предоставил шаблон растущего промпта, куда нужно лишь подставить свою задачу вместо квадратных скобок.
3. **Апгрейды — опциональны.** Не для всех задач нужны Python-скрипты или сложные правила. Ненужные апгрейды можно просто удалить из шаблона.
4. **Архитектура — это сила.** Она делает скиллы предсказуемыми, легко изменяемыми и способными к интеграции.
