

CLAUDE-LIMIT SPAREN: DAS LOOPSYSTEM MIT EXECUTOR-SUBAGENT

3 PROMPTS TOOL: CLAUDE CODE STAND 2026-07-29

Du baust dir in Claude Code ein Loopsystem: Dein stärkstes Modell schreibt nur noch den Plan und prüft das Ergebnis, ein **Executor-Subagent** auf Haiku oder Sonnet erledigt die Fleißarbeit dazwischen. Dadurch hält dein Nutzungslimit deutlich länger. Du bekommst drei fertige Prompts: den Executor-Prompt, den Loop-Prompt und das Ziel mit `/goal`, das dein Top-Modell den Executor so lange an die Arbeit schicken lässt, bis das Ergebnis wirklich steht.

3

FERTIGE PROMPTS

ca. 10 Min

SETUP

1x

PRO PROJEKT EINRICHTEN

WAS DU BRAUCHST

Claude Pro oder Max. Pro kostet rund 21 € im Monat (18 € plus Mehrwertsteuer, im Jahresabo etwas günstiger), Max startet bei rund 107 € im Monat. Im kostenlosen Plan ist Claude Code nicht enthalten.

Die Claude-Desktop-App für Mac oder Windows, kostenlos auf claude.com/download. Oben in der App gibt es die Bereiche Chat, Cowork und **Code**. Alles in diesem Guide passiert im Bereich Code.

Einen Ordner auf deinem Rechner, in dem Claude arbeiten darf. Für den ersten Test reicht ein leerer Übungsordner, zum Beispiel auf dem Schreibtisch.

Kurz erklärt: *Claude Code* ist der Teil von Claude, der selbstständig in einem Ordner auf deinem Rechner arbeitet: Dateien anlegen, lesen und überarbeiten, statt nur im Chat zu antworten. Ein *Subagent* ist ein kleiner Spezial-Assistent in Claude Code, mit eigenem Namen, eigener Anweisung und einem fest zugewiesenen Modell. Dein Hauptmodell gibt ihm Aufgaben wie einem Mitarbeiter und bekommt am Ende nur das Ergebnis zurück.

SO FUNKTIONIERT DAS LOOPSYSTEM

Wenn du einfach das stärkste Modell auswählst und drauflos arbeitest, macht dieses Modell alles selbst: Dateien lesen, Texte tippen, umformulieren, Fehler suchen. Das ist, als würdest du einen Architekten dafür bezahlen, Wände zu streichen. Je stärker das Modell, desto schneller ist dein Limit leer.

Das Loopsystem teilt die Arbeit auf drei Rollen auf:

- Der Planer:** dein Top-Modell. Es schreibt einen detaillierten Plan, baut aber selbst nichts.
 - Der Executor:** ein Subagent auf Haiku oder Sonnet. Er setzt den Plan Schritt für Schritt um, in einem eigenen Gesprächsverlauf.
 - Der Prüfer:** wieder dein Top-Modell. Es kontrolliert das Ergebnis gegen den Plan und schickt den Executor mit konkreten Korrekturen zurück, bis alles passt.
- Dazu kommt `/goal`. Damit gibst du eine Abschlussbedingung vor. Nach jeder Runde prüft ein kleines, schnelles Modell, ob sie erfüllt ist. Erst dann bekommst du die Kontrolle zurück. So entsteht die Schleife: planen, delegieren, prüfen, wiederholen.

01 CLAUDE CODE ÖFFNEN

Du gehst auf Claude und öffnest Claude Code: Starte die Claude-Desktop-App und klick oben auf **Code**.

Wähle dann den Ordner, in dem gearbeitet werden soll, und bestätige den Zugriff. Für den ersten Durchgang nimmst du deinen Übungsordner. Claude arbeitet nur in dem Ordner, den du hier freigibst.

01 DEIN STÄRKSTES MODELL AUSWÄHLEN

Jetzt wählst du dein stärkstes Modell aus. Tippe dafür `/model` ins Eingabefeld und drück Enter. Es öffnet sich eine Liste mit allen Modellen, die dein Plan anbietet. Nimm das stärkste davon, zum Beispiel **Opus**.

Dieses Modell ist ab jetzt dein Planer und dein Prüfer. Die eigentliche Arbeit gibt es gleich ab.

TIPP

In der Desktop-App findest du die Modell-Auswahl auch direkt am Eingabefeld. Ein Klick auf den Modellnamen öffnet dieselbe Liste.

01 EXECUTOR-SUBAGENT ERSTELLEN

Jetzt sagst du deinem Top-Modell: Erstell mir einen Executor-Subagent, der auf Haiku oder Sonnet läuft. Kopier dafür den Prompt unten in das Eingabefeld und schick ihn ab. Claude legt den Subagent daraufhin selbst an.

Fragt Claude, ob der Subagent nur für dieses Projekt oder für dich persönlich gelten soll: Wähle **Projekt**, wenn du das System erst testen willst. Wähle **persönlich**, wenn der Executor dir in allen Projekten zur Verfügung stehen soll.

HAIKU ODER SONNET?

Haiku ist am günstigsten und schnellsten, perfekt für klar beschriebene Routinearbeit. Sonnet ist spürbar schlauer und kostet mehr als Haiku, bleibt aber deutlich sparsamer als dein Top-Modell. Starte mit Haiku. Scheitert der Executor am Plan, ersetze im Prompt einfach Haiku durch Sonnet.

PROMPT 01: EXECUTOR-SUBAGENT ERSTELLEN

Erstelle mir einen Subagent namens "executor", der auf dem Modell Haiku läuft.
Seine Aufgabe: Er bekommt einen fertigen, detaillierten Plan und setzt ihn exakt um, Schritt für Schritt. Er trifft keine eigenen Entscheidungen, die nicht im Plan stehen. Ist etwas im Plan unklar, notiert er die offene Frage, statt zu raten. Am Ende fasst er kurz zusammen, was er umgesetzt hat und wo er vom Plan abweichen musste.

WENN DAS NICHT KLAPPT

Alternativer Weg zum selben Ergebnis: Lass Claude die Subagent-Datei direkt mit festem Inhalt anlegen. Schick diesen Text ab:

Lege im Projektordner die Datei `.claude/agents/executor.md` mit genau diesem Inhalt an:

```
---  
name: executor  
description: Setzt einen fertigen, detaillierten Plan exakt Schritt für Schritt um. Trifft keine eigenen Entscheidungen außerhalb des Plans.  
model: haiku  
---
```

Du bekommst einen fertigen Plan und setzt ihn exakt um, Schritt für Schritt. Triff keine Entscheidungen, die nicht im Plan stehen. Ist etwas unklar, notiere die offene Frage, statt zu raten.
Danach Claude Code einmal schließen und wieder öffnen, damit der neue Subagent geladen wird.

01 KURZ GEGENCHECKEN

Schick zur Kontrolle: *Zeig mir den Inhalt von `.claude/agents/executor.md`.*

In der Antwort muss die Zeile `model: haiku` (oder `model: sonnet`) stehen. Dann ist das Setup fertig. Das machst du pro Projekt genau einmal. Hast du den Executor persönlich angelegt, liegt die Datei im Ordner `.claude/agents` in deinem Benutzerordner und gilt automatisch überall.

01 DEN LOOP-PROMPT STARTEN

Jetzt startest du das Ganze mit dem Loop-Prompt. Er teilt die Arbeit sauber auf: erst planen (teuer, aber kurz), dann delegieren (günstig), dann prüfen (teuer, aber kurz). Ersetze **[DEINE AUFGABE]** durch das, was erledigt werden soll, zum Beispiel „20 Hook-Varianten für mein nächstes Video-Skript zum Thema KI im Alltag“.

Was dann passiert: Dein Top-Modell zerlegt die Aufgabe in einen nummerierten Plan, übergibt ihn an den Executor, nimmt jedes Teilergebnis ab und schickt bei Bedarf konkrete Korrekturen zurück. Weil der Executor in einem eigenen Verlauf arbeitet, bleibt dein Hauptgespräch schlank und übersichtlich.

PROMPT 02: DER LOOP-PROMPT

Schreib zuerst einen detaillierten Plan für folgende Aufgabe: [DEINE AUFGABE].
Baue in dieser Phase noch nichts. Wenn der Plan steht, übergib ihn Schritt für Schritt an den executor-Subagent. Prüfe nach jedem Schritt selbst das Ergebnis gegen den Plan. Wenn etwas fehlt oder nicht passt, schick den executor mit konkreten Korrekturanweisungen noch einmal los. Melde dich erst zurück, wenn alles dem Plan entspricht.

01 /GOAL ANS ENDE SETZEN

Damit es nicht aufhört, bevor es wirklich gut ist, setzt du noch `/goal` ans Ende. Tippe `/goal`, dahinter deine Abschlussbedingung, und schick es ab. Jetzt schickt dein Top-Modell den Executor so lange an die Arbeit, bis das Ziel erreicht ist.

Nach jeder Runde liest ein kleines Prüf-Modell den Verlauf und entscheidet nur eine Frage: Ziel erreicht, ja oder nein? Bei Nein startet die nächste Runde von allein. Solange das Ziel aktiv ist, siehst du eine Anzeige, wie lange es schon läuft.

AUTO-MODUS

Damit die Schleife nicht bei jedem Arbeitsschritt auf deine Bestätigung wartet, stell neben dem Senden-Button den Modus auf **Auto**. Ein Sicherheitscheck im Hintergrund blockiert trotzdem riskante Aktionen.

PROMPT 03: ZIEL SETZEN

`/goal` Alle Schritte aus dem Plan sind umgesetzt und vom Hauptmodell geprüft. Es gibt keine offenen Korrekturanweisungen mehr, und es wurden nur die im Plan genannten Dateien verändert. Stoppe spätestens nach 20 Turns.

WENN DAS NICHT KLAPPT

Je nach Version läuft es auch so: Pack Loop-Prompt und Ziel in eine einzige Nachricht. Das Ziel startet die Arbeit dann sofort selbst, ein extra Prompt ist nicht nötig.

/goal Die Aufgabe [DEINE AUFGABE] ist vollständig erledigt. Ablauf: Schreib zuerst einen detaillierten Plan und baue dabei noch nichts. Übergib dann jeden Schritt einzeln an den Subagenten.

01 KONTROLLE BEHALTEN

/goal ohne Text zeigt dir den Status: das Ziel, die bisherigen Runden, den Verbrauch und die letzte Begründung des Prüf-Modells.

/goal clear stoppt die Schleife sofort.

Ist das Ziel erreicht, verschwindet es von selbst. Ein neues /goal ersetzt ein altes, es läuft immer nur eins.

Selbst prüfen, ob alles steht: Starte einen Testlauf mit einer kleinen Aufgabe. Siehst du im Verlauf, dass dein Top-Modell an den Executor übergibt, statt selbst zu tippen? Dann läuft dein Loopsystem.

SO SCHREIBST DU EINE GUTE ABSCHLUSSBEDINGUNG

Ob der Loop am richtigen Punkt stoppt, entscheidet fast nur die Bedingung hinter /goal. Das Prüf-Modell öffnet keine Dateien und führt nichts aus. Es liest nur, was Claude im Verlauf zeigt.

Ein messbarer Endzustand: „alle 20 Hooks sind geprüft“, „die Liste offener Punkte ist leer“. Nicht „mach es richtig gut“.

Ein Nachweis: etwas, das Claude selbst zeigen kann, zum Beispiel eine Abnahme-Liste oder ein Testergebnis.

Grenzen: was nicht passieren darf, zum Beispiel „keine Datei außerhalb des Projektordners verändern“.

Ein Turn-Limit als Notbremse: „Stoppe spätestens nach 20 Turns“. Ein Turn ist eine komplette Arbeitsrunde von Claude.

Die Bedingung darf bis zu 4.000 Zeichen lang sein.

WOFÜR SICH DAS SYSTEM EIGNET

Content: Skripte und Hooks in Serie. Das Top-Modell legt Kriterien und Struktur fest, der Executor schreibt die Rohfassungen, das Top-Modell siebt streng aus und lässt schwache Varianten überarbeiten.

Recherche: viele Quellen sichten und zusammenfassen lassen, das Top-Modell zieht das Fazit.

Aufräumen: Dateien umbenennen, Strukturen vereinheitlichen, Listen abarbeiten, bis die Liste leer ist.

Code: Fehler beheben, bis alle Tests fehlerfrei durchlaufen.

Mit diesem System schreibe ich auch teilweise meine Skripte. Damit habe ich über 40.000 Follower in unter 50 Tagen bekommen. Das Loopsystem ist einer der Bausteine dahinter.

DIE EXECUTOR-DATEI ZUM MITNEHMEN

Die fertige Executor-Subagent-Datei legst du in den Ordner .claude/agents deines Projekts. Dann kannst du Schritt 3 überspringen.

SPART DAS WIRKLICH LIMIT? EHRlich GERECHNET

Das System spart Limit im Vergleich dazu, alles mit dem Top-Modell zu erledigen. Die Fleißarbeit läuft auf Haiku und kostet nur einen Bruchteil.

Aber: Ein Loop ist ein Gaspedal, keine Bremse. Jede Runde des Executors verbraucht zusätzlich, und ein /goal mit schwammiger Bedingung kann viele Runden drehen. Wer die Schleife ohne messbares Ziel laufen lässt, ist sein Limit dadurch schneller los. Drei Regeln halten es sparsam:

1. Bedingung messbar formulieren und immer ein Turn-Limit einbauen.
2. Zwischendurch /goal tippen und Runden plus Verbrauch ansehen. Läuft es aus dem Ruder: /goal clear.
3. Das Top-Modell nie selbst tippen lassen. Merkst du, dass es Routinearbeit erledigt, zurück zum Loop-Prompt.

Chat und Claude Code teilen sich übrigens dasselbe Nutzungskontingent.

WENN DAS NICHT KLAPPT

/goal erscheint nicht in der Vorschlagsliste: Aktualisiere die Claude-Desktop-App und bestätige den Ordner-Zugriff. Danach Claude Code neu öffnen.

Das Top-Modell macht alles selbst: Schreib dazu: „Nutze für die Umsetzung ausschließlich den Subagent executor.“ Die Beschreibung des Subagents entscheidet, wann Claude ihm Arbeit gibt. Je klarer sie ist, desto zuverlässiger klappt die Übergabe.

Der Executor wird nicht gefunden: Lass dir mit „Zeig mir alle Dateien in .claude/agents“ anzeigen, ob die Datei wirklich dort liegt, und starte Claude Code einmal neu.

Der Executor liefert schwache Ergebnisse: Stell ihn auf Sonnet um: „Ändere in .claude/agents/executor.md die Zeile model: haiku in model: sonnet.“

Die Schleife wartet ständig auf dich: Modus neben dem Senden-Button auf Auto stellen.

Ziel wird zu früh als erreicht gemeldet: Bedingung schärfer formulieren und einen sichtbaren Nachweis verlangen.

HÄUFIGE FRAGEN

GEHT DAS AUCH IM NORMALEN CLAUDE-CHAT?

Nein, Subagents und `/goal` gehören zu Claude Code. Du brauchst also den Bereich Code in der Desktop-App.

MUSS ICH DAS SETUP JEDES MAL NEU MACHEN?

Nein. Einmal pro Projekt, oder gar nur einmal, wenn du den Executor persönlich anlegst. Danach besteht jede neue Aufgabe nur noch aus Loop-Prompt plus Ziel.

KANN ICH DEN EXECUTOR AUCH FÜR ANDERE AUFGABEN NUTZEN?

Ja. Immer wenn ein klarer Plan vorliegt, kannst du schreiben: „Gib die Umsetzung an den executor.“

QUICK REFERENCE

- Top-Modell plant und prüft:** Routinearbeit gehört nie ins teuerste Modell.
- Executor auf Haiku:** Erledigt die Fleißarbeit exakt nach Plan. Bei komplexer Arbeit Sonnet.
- Loop-Prompt:** Plan schreiben, an den Executor übergeben, Ergebnis abnehmen.
- /goal [Bedingung]:** Claude arbeitet weiter, bis die Bedingung erfüllt ist.
- /goal und /goal clear:** Status ansehen und Schleife sofort stoppen.

HINWEIS

Hinweis: Dieser Guide dient zu Informations- und Bildungszwecken. Alle Inhalte spiegeln den Stand bei Erstellung wider, Oberflächen, Funktionen und Preise der genannten Tools können sich ändern. Prüfe Ergebnisse von KI-Tools immer selbst nach. @GPTMarlon übernimmt keine Haftung für Entscheidungen auf Grundlage dieser Inhalte. Genannte Ergebnisse wie Follower-Zahlen sind persönliche Erfahrungswerte und keine Erfolgsgarantie.