E2 Claude Code

ハーネスエンジニアリング入門

考え方と6つの部品（E1のあとに読む）。mdファイル（指示書・メモリ・スキル）の使い方と行数の限界、ルールを置く場所の決め方、採点表、事故から生まれたルール、10の確認

更新日：2026-10-03（2026年10月時点の情報）｜HUMAC 伊藤晴通

対象ツール：Claude Code

考え方は、どのAIエージェントにも共通です。ただし、**ファイル名・画面・設定・コマンドは、Claude Code**（Claudeのデスクトップアプリの「Code」タブ）の場合です。ほかのAIエージェントでは、読み替えが必要です。

対象の方：開発者ではない方（経営者・事務・現場の担当者）。プログラミングは不要です。**コードは書かず、AIエージェントに頼んで、自分の業務アプリをつくる**ところまでを、順に進めます（ターミナルの黒い画面は、使わない形で書いています）。

先に E1（フォルダの作り方）で、手を動かすと、読みやすい

このページは「考え方」（なぜ・何を）。**E1で、仕事のフォルダ・指示書・現在地を、実際に作ってから**読むと、各部品が、自分のフォルダのどこにあたるか、すぐ分かる。

ハーネスとは「馬具」のこと

AIエージェントは力の強い馬。放っておくと、頼んでいない相手にメールを送ったり、消してはいけないファイルを消したりする。**手綱と装具（ハーネス）を付けて、決めた道を安全に走らせる仕組みづくり**が「ハーネスエンジニアリング」。AIを賢くするより、**AIが間違えても事故にならない形**を作ることが仕事の中心になる。

1. まず1枚の絵：エンジンと車体

AIモデル＝エンジン

ChatGPT・Gemini・Claude の「頭脳」にあたる部分。とても高性能だが、エンジンだけでは走れない。

ハーネス＝車体まわり一式

ハンドル・ブレーキ・ナビ・シートベルト。エンジンを「目的地まで安全に走れる車」にする仕組み。私たちが普段さわっているのは、実はこちら。

NVIDIAのジェンスン・フアンCEOは2026年5月、エージェントは言語モデルの上で直接動くのではなく、その手前に挟まる**「安全に管理されたソフトウェア層」**の上で動くと説明した。この層が担うのは、考えて道具を使う手順の繰り返し、外部システムの呼び出し、モデルの使い分け、記憶と文脈の管理、そして暴走の停止や行動の記録。

「賢いAI」と「使えるAI」は別物

質問して答えをもらうだけなら、ハーネスは薄くてよい。しかし「AIが自分でメールを送る・データを更新する・複数のシステムをまたいで作業する」ようになると、**何を許すか・どこで止めるか・何を記録するか**の設計が要る。同じモデルでも、ハーネスの出来で「使い物になるか」が決まる。

2. なぜ今、大事なのか

- **モデルの差が縮まった**：どの会社のAIも、一定以上に賢い。差がつくのは「どう仕事につなぐか」の側になった。
- **人間の統制は、機械の速度に間に合わない**：人の担当者なら、権限・承認・監査で管理できた。AIは人間よりずっと速く複数のシステムを動かすので、別の設計が要る。
- **失敗の多くはモデルではなくハーネス側にある**：権限が広すぎた、資料が古かった、止める条件がなかった。「もっと賢いAIにすれば解決する」とは限らない。

3. ハーネスの6つの部品

部品	たとえ	中身	Claude での実体	これが無いと
① 指示書	新人に渡す「職場のルールブック」	この仕事の目的、置き場所、やってはいけないこと、言葉づかい	プロジェクトごとのCLAUDE.md	毎回、同じ説明をする。人によって結果が違う
② スキル	業務ごとの手順書	「この仕事はこの順で、この資料を見て、この形で仕上げる」	スキル（合言葉で呼び出す）	毎回ゼロから頼む。品質がばらつく
③ 権限	渡す鍵の範囲	読んでよい場所・書いてよい場所・使ってよい道具	許可設定（ファイル・コマンド・外部サービスごと）	触ってはいけないものに触れてしまう
④ 関所（承認）	判子が要る書類	送信・公開・削除・支払いの前は、必ず人が確認する	承認の確認画面、フック（自動チェック）	取り返しのつかない操作が、人の知らないうちに実行される
⑤ 記憶	引き継ぎノート	前回の失敗・決まったこと・好みを次回に持ち越す	メモリ、決定ログ、STATUS	同じ失敗と説明を、何度も繰り返す
⑥ 点検	定期監査	ルールと実態がずれていないか、定期的に確かめる	週次の自動点検（定期実行）	ルールが古くなり、いつの間にか守られなくなる

6つの部品は、フォルダの中のどこに置くか

① 指示書＝`CLAUDE.md`／② スキル＝`.claude/skills/`／③ 権限＝`.claude/settings.json`とフォルダの範囲／④ 関所＝承認の設定と、書き込み先（`outputs/`）・退避（`_archive/`）／⑤ 記憶＝`STATUS.md`と**決定ログ**`.md`／⑥ 点検＝`RAID.md`と週1回の点検。実際の作り方と進め方は E1 はじめてのAIエージェント：フォルダ構成と管理。

4. mdファイル：ハーネスの本体は「文章」

Claude Code のハーネスは、プログラムではなく、**人が読める普通の文章ファイル（Markdown＝「.md」）**で作る部分が大きい。AIは会話のたびに「白紙」から始まるので、**前回の学びと、守ってほしいルールを持ち越す手段は、mdファイルしかない**。ここを理解すると、プログラムが書けなくても、AIの振る舞いを設計できる。

4-1　3種類のmdファイルと、それぞれの役割

ファイル	誰が書く	いつ読まれる	一言でいうと
CLAUDE.md（指示書）	人	セッションの開始時に、毎回	「このルールで働け」
メモリ（MEMORY.md ＋ 個別のメモ）	Claude が自分で書く	目次（MEMORY.md）は毎回。個別のメモは必要になったとき	「前に教わったこと」
SKILL.md（スキル）	人（Claude と一緒に作ってもよい）	呼んだとき、または必要と判断されたとき	「この仕事のやり方」

mdは「お願い」であって、「強制」ではない

CLAUDE.md は、Claude に渡される**文脈**で、設定のように強制されるものではない。曖昧な指示や、食い違う指示は、守られないことがある。**必ず守らせたいこと（送信・削除・公開の前の確認など）は、md に書くだけでなく、権限の設定やフック（決まった場面で自動で動くチェック）で、仕組みとして止める**（→ ④関所）。

4-2　置き場所＝効く範囲

置き場所	効く範囲	向く内容
組織の管理者が置く場所（OSごとに決まった場所）	その組織のすべての利用者	会社の規程、セキュリティの方針
`~/.claude/CLAUDE.md`	自分の全セッション	言語、秘密情報のルール、共通の作法
案件のフォルダ`/CLAUDE.md`（または `.claude/CLAUDE.md`）	その案件のセッション。共有できる	案件の目的、置き場所、ルール、依頼例
`CLAUDE.local.md`	その案件・自分だけ（共有しない）	個人の作業メモ

置き場所	効く範囲	向く内容
サブフォルダの CLAUDE.md	そのフォルダのファイルを読んだときだけ	部分的なルール
`.claude/rules/` のmd	パスを指定すると、該当するファイルのときだけ。指定なしは毎回	テーマごとに分けたルール

- **上書きではなく、連結される**：見つかったすべての CLAUDE.md が、文脈に並べて読み込まれる。広い範囲のものが先、作業フォルダに近いものが後。食い違うときは、近いほうが優先されやすい。
- **どこで開いたかで、読まれるものが決まる**：案件のフォルダで開けば、その案件の CLAUDE.md とメモリが載る。全案件に効かせたいことは、全体用に昇格させる。

4-3　行数・サイズの限界（2026年10月時点・公式の数字）

対象	限界・目安	超えるとどうなるか
CLAUDE.md（1ファイル）	**200行以内が目安**。それより長くしない	文脈を多く使い、**守られにくくなる**。長いと、起動時などに警告が出る。4MiBを超えると、そもそも読み込まれない
メモリの目次（MEMORY.md）	起動時に読まれるのは、**先頭200行、または25KBの、先に来たほう**まで	**それ以降は読み込まれない**（書き込み自体はできる）。1記憶1行、詳細は個別のメモに分ける
メモリの個別ファイル	起動時には読まれない	必要になったとき、Claude が読みに行く
SKILL.md（本体）	**500行以内が目安**。詳細は別ファイルへ	呼ぶたびに全部が文脈に載るので、長いほど毎回のコストが増える
スキルの説明文（description）	説明文＋使う場面の合計で **1,536文字まで**（一覧に載る分）	それ以降は一覧で切れる。「いつ使うか」を先頭に書く
@で読み込む別ファイル（import）	入れ子は**最大4段**まで	整理には役立つが、**読み込む量は減らない**（起動時にすべて読む）
.claude/rules/（パス指定なし）	起動時に毎回読まれる（CLAUDE.md と同じ扱い）	長くしたくなったら、パス指定で「該当するファイルのときだけ」にする

行数の限界は「機械的に切られる線」ではなく、**「長いほど守られにくい」という目安**（CLAUDE.md）と、**「それ以降は読まれない」という線**（メモリの目次）の2種類がある。長い文章の中ほどが見落とされやすいことは、研究でも知られている（→ ①）。

4-4　使い方の原則

- **書くタイミング**：「同じ間違いを2度された」「前回と同じ訂正を、また入力した」「新しく来た人にも同じ説明が要る」。**毎回説明し直していることを、文章にする**。
- **書くこと**：毎回必要な事実（ルール、置き場所、言葉づかい、「必ずこうする」）。**書かないこと**：複数の手順（→ スキルへ）、一部の場所でしか要らないこと（→ サブフォルダやパス指定へ）、ファイルを見れば分かるこ

と（古くなって嘘になる）。

- **確かめられる具体さで書く**：「ちゃんと書いて」は守りようがない。次のように、守れたかが分かる書き方にする。

曖昧な書き方	確かめられる書き方
丁寧な文章で	「です・ます調」で、1文は60字以内
資料は整理して置く	見積書は「営業/見積書」に、ファイル名は「会社名_日付_v番号」で保存する
メールは気をつけて	メールは下書きまで。送信はしない

- **見出しと箇条書きで整理する**：長い文章より、構造のあるほうが、AIも読みやすい。
- **食い違う指示を残さない**：矛盾があると、AIはどちらかを適当に選ぶ。追記を重ねたら、定期的に読み直して、古い指示や矛盾を消す。点検は、Claudeに頼めばよい（「CLAUDE.md を点検して。古い指示・存在しないファイルへの言及・食い違いを、一覧で報告して。直す前に、私に確認して」）。Claude Code には、同じ点検をする機能（`/doctor prompt-audit`）もある（新しいバージョンのみ）。
- **人間向けのメモは、コメントで書ける**：行頭のHTMLコメント（`<!-- ... -->`）は、AIに読み込まれず、文脈を使わない。
- **会話の中だけで伝えた指示は、消える**：長い会話を要約（コンパクト）したとき、案件の直下の CLAUDE.md は読み直されて残るが、会話の中で言っただけの指示は失われる。残したいことは、ファイルに書く。
- **他のAIツールとの共用**：他のツールが使う指示書（AGENTS.md）がある場合、CLAUDE.md が無いときだけ読まれる。両方使うなら、CLAUDE.md の中から取り込む。

4-5　長くなったときの直し方（優先順位）

1

1 自明なことを消す
ファイルを見れば分かること、古くなった情報

2

2 手順はスキルへ
複数のステップは、呼んだときだけ読まれるスキルに

3

3 場所で分ける
その案件だけ、そのフォルダだけのルールは、近くのファイルへ

4

4 詳細は別ファイル＋案内だけ
本体には「詳しくはここ」だけ書く（※@で取り込むと、量は減らない）

コピーして使う

（行数の点検：ターミナルは使わず、Claudeに頼む）
このフォルダの CLAUDE.md と、全体用の CLAUDE.md の行数を数えて、200行を超えていないか教えてください。メモリの目次（MEMORY.md）も、行数とサイズ（KB）を教えてください。

4-6　確認のしかた：コマンドより、Claudeへの頼み方

ターミナルが苦手でも大丈夫。**やりたいことを、そのままClaudeに頼めばよい**。コマンド（右の列）は、慣れた人向けの参考で、デスクトップアプリでは、使えないものもある。

やりたいこと	Claudeへの頼み方	（参考）コマンド
読み込まれている指示書・メモリの確認。「書いたのに効かない」ときは、まずこれ	「いま読み込んでいる指示書とメモリを、ファイル名と行数で一覧にして」	`/context`、`/memory`
指示書のひな形を作る	「この仕事のCLAUDE.mdを作って。書く前に、私に質問して」（コードの無い案件は、先に議論してから作るほうがよい）	`/init`
指示書の点検	「CLAUDE.mdを点検して。長すぎる所、古い所、食い違いを報告して。直す前に確認して」	`/doctor`
覚えさせたいことを、ファイルに残す	「これをCLAUDE.mdに追記して」（自動メモリに残したいときは「覚えておいて」）	`/memory`

講師の実例

- 全体用のCLAUDE.mdは約70行、案件用は約40行。毎回読まれるものは、短く保つ。
- 詳しいルールは別ファイル（真実源）に置き、指示書には「どこを見るか」だけを書く。
- スキルの一覧は指示書に書かない。各スキルの説明文が自動で読まれるので、二重管理を避ける。
- メモリには、教訓だけを1行ずつ。長くなったら整理する（上の「限界」の表）。

5. 部品を、もう一段くわしく

① 指示書：短く、判断を書く

- **毎回読み込まれる**ので、長いほど損をする（目安は200行以内。くわしくは上の「4. mdファイル」）。コストがかかるうえ、肝心のルールが埋もれる（長い文章の真ん中は、見落とされやすいことが研究で知られている）。
- **書くこと**：目的／置き場所／やってはいけないこと／言葉づかい／「ここを見れば分かる」という場所。**書かないこと**：フォルダ一覧の丸写しなど、ファイルを見れば分かること（すぐ古くなって嘘になる）。
- **置き場所＝効く範囲**：全員共通のルールは全体用に、案件ごとのルールは案件のフォルダに。詳しい規則は別ファイルに置き、指示書にはそこへの案内だけ書く（真実源を1か所にする）。

- **矛盾する指示を同居させない**：「記録を必ず残す」と「メモは書かない」が両方あると、AIは迷う。
- **一律の禁止より、判断の軸**：Anthropicは2026年7月、最新のモデルは「周囲に合わせて」と判断を渡すほうが、細かい一律ルールで縛るより良く働く、と整理している。
- **秘密（パスワード・鍵）は絶対に書かない**。指示書はコピーされ、チャットにも読み込まれる。

コピーして使う

【この仕事の目的】お客様への見積案内メールの下書きを作る
【置き場所】見積書の雛形は「営業/見積書の雛形」。過去の案内文は「営業/案内文」
【言葉づかい】丁寧語。「させていただく」は使いすぎない
【やってはいけないこと】
・送信しない（下書きまで）
・お客様の個人情報を、AIに入力する文章に含めない
・金額は、見積書の数字を写す（計算し直さない）
【迷ったら】作らずに、私に質問する

② スキル：必要なときだけ呼ぶ手順書

- 指示書は「常に読まれる」、スキルは「呼んだときだけ読まれる」。**手順は、指示書ではなくスキルに入れる**と、軽く保てる（段階的に開く、という考え方）。
- スキルの中身は、F1の手順書の5項目（出来上がり・材料・手順・人が確認するところ・例外と目安）。
- 置き場所で効く範囲が決まる。どのフォルダからでも使うなら全体用、その案件だけなら案件のフォルダ。

③ 権限：必要な分だけ、渡す

鍵は**仕事に必要な最小限**だけ渡す。「念のため全部」は事故のもと。

区分	考え方	例
🟢 **自由にしてよい**	読むだけ、元に戻せる、社内で完結	資料を読む、下書きファイルを作る
🟡 **確認してから**	元に戻しにくい、または外に影響する	既存ファイルの上書き、決まった手順以外のコマンド
🔴 **させない（できなくする）**	事故になる、取り返しがつかない	メール送信・公開・削除・支払い、本番データの書き換え

④ 関所：ルールを「どこに置くか」はリスクで決める

ここが一番大事。**指示書に「やらないでね」と書くだけでは足りない**。AIは確率で動くので、ルールを「ほぼ」守るが、「必ず」ではない。長い文章の中で忘れる、外から紛れ込んだ文章に上書きされる、解釈を間違える、という理由がある。

破られたときの影響	ルールを置く場所	例
致命的でない	**指示書**（お願い）	文体、言葉づかい

破られたときの影響	ルールを置く場所	例
業務が回らなくなる	**スキル**（手順として組み込む）	請求書チェックの手順
事故になる	**関所・権限** （AIの外側で、できなくする）	個人情報の外部送信、メールの自動送信、ファイルの削除

原則：「信頼して検証する」ではなく、「できないようにする」

ウェブの世界で、入力チェックを画面側だけでなく、サーバー側でも行うのと同じ考え方。AIが何を考えていても、**外側の仕組みが止める**。事故になる種類は、「お願い」ではなく仕組みで止める。

⑤ 記憶：4種類を使い分ける

プリンストン大学の研究者が提案した整理（CoALA）では、AIエージェントの記憶は4種類。

種類	役割	Claude での実体
作業中の記憶	今この瞬間の作業（会話・直近のデータ）。セッションが終わると消える	会話の中身
知っていること	事実・規約・制約。常に読み込まれる	指示書（CLAUDE.md）
やり方	実行手順。必要なときだけ呼ばれる	スキル
経験	過去のやりとり・決定・結果から得た教訓	メモリ、決定ログ

- **経験は「蒸留」して残す**：生のやりとり全部ではなく、教訓だけを残す（例：45分の会話ではなく、「作業を始める前に、必ず対象の環境を確認する」の1行）。
- **忘れる仕組みも要る**：状況が変われば、古い教訓は捨てる。何を捨てるかが、使い勝手の質を決める。
- **記憶は、置き場所で効く範囲が決まる**：案件ごとの記憶は、その案件を開いたときだけ載る。全部の案件に効かせたいことは、全体用の指示書へ昇格させる。

⑥ 点検：1回で終わらせず、回し続ける

型	起動	止まり方	向く仕事
対話型	人が頼むたび	1往復ごとに人が判断	日常の相談・下書き
ゴール型	完了条件を決めて頼む	条件を満たしたら止まる	1つの作業を仕上げる
時間型	毎週・毎日など、決めた時間	実行が終わったら止まる	週次の点検、定期レポート
自律型	きっかけがあれば人の関与なしに起動	AIの判断	最も自律的。関所の設計が最重要

- **点検は、未解決を翌週に持ち越す**：指摘して終わりにせず、台帳（簡単なメモでよい）に残して、「先週の指摘は直ったか」を次の点検で確かめる。

- **AIが直して、AIが確認して、人が見ない**、という並びは避ける。自動で直した箇所は、次の点検で再検証する。
- 点検は、「新しく長持ちするものを作る」仕事ではなく、「調べて報告する」仕事に向く。

6. 「採点表」を持つ（評価）

動かして終わりではなく、**「良い仕事」の基準を数字にして、測る**。MicrosoftのサティアーナデラCEOは2026年6月、企業の「自分たちのAI」を、基盤モデル＋ハーネス＋**自社専用の採点表（private eval）**と説明した。一般的なAIの賢さの比較ではなく、**自社の仕事で、何を価値とし、何を許されない失敗とするか**を決めるもの。

例：問い合わせ返信の下書きAI	目標の例
直さずにそのまま使えた割合	7割以上
事実と違うことを書いた件数	ゼロ（1件でもルールを見直す）
人の確認が必要な案件を、見逃さなかった割合	ほぼ100%
自社の言葉づかいに合っているか（人が5段階で評価）	平均4以上

モデルを乗り換えても、この採点表で良い結果が出続けるなら、**主導権は自社にある**。特定のAIに依存しないと成果が出ないなら、主導権は提供会社側にある。事故になる種類は、測って減点するより、④の関所で**止める**。

7. 事故から生まれたルール（講師の実例）

ハーネスは机上で考えるより、**小さな事故を1つずつルールに変えていく**と強くなる。

起きたこと	ルールにしたこと	部品
AIを使うための鍵（APIキー）が公開の場所に出て、約6万8千円の不正請求	鍵はコードにもチャットにも書かない。鍵は専用の金庫に集め、プログラムは金庫から読む（→ E6）	①③
鍵や秘密を確認しようとして、中身をそのまま表示させてしまい、会話の記録に残った	秘密の中身は表示しない。確認は「存在するか」だけ。試験用の使い捨てを使う。漏れたら、すぐ入れ替える	③④
AIが「不要」と判断した経理書類を消してしまった	AIの判断で消さない。疑わしいものは「退避フォルダ」へ移し、最後は人が消す	③④
クライアントのブログを、確認前に公開しかけた	公開ボタンはクライアント本人が押す。AIは下書きまで	④
AIが古い知識で製品名を「訂正」して間違えた	製品名・料金は、直す前に必ずWebで確認する	①⑤
同じ説明を何度もさせられた	一度決めたことは記憶に残し、蒸し返さない	⑤

8. 作り方の手順

1 仕事を1つ選ぶ
毎回同じ手順で、失敗しても取り返しがつくもの

2 指示書を書く
目的・置き場所・禁止事項を1枚に。長くしない

3 関所を決める
「外に出る操作」と「消す操作」は必ず人が押す

4 スキルにする
手順を言葉にし、合言葉で呼べるようにする

5 点検を回す
週1回、ルールと実態のずれを確かめ、事故はルールに変える

小さく始める1週間

日	やること
1日目	仕事を1つ選ぶ。何が出来上がればよいか、1行で書く
2日目	指示書を1枚書く（上の例を参考に）。禁止事項は3つまで
3日目	関所を決める。送信・公開・削除・支払いのうち、この仕事に関わるものを書き出す
4〜5日目	実際に頼んでみる。うまくいかなかった点を、指示書かスキルに1行ずつ足す
6日目	手順をスキルにする（F1のステップ4・5）
7日目	点検：事故やヒヤリとした点はなかったか。あればルールにする。なければ、翌週も同じ

9. AIに仕事を任せる前の10の確認

□	確認すること
□	目的と、出来上がりの形が、1行で書けている
□	使う資料の置き場所と、最新かどうかの見分け方が決まっている（B2）
□	入れてよい情報・いけない情報が決まっている（D1）
□	渡す権限が、仕事に必要な最小限になっている
□	送信・公開・削除・支払いは、人が押す関所になっている
□	鍵・パスワードが、指示書にもチャットにも書かれていない
□	間違えたとき、元に戻せる（退避フォルダ・履歴）
□	何をしたかの記録が残る

□	確認すること
□	最終的な責任者が、1人決まっている
□	点検の頻度と担当が決まっている

10. よくある失敗

指示書が長大になる

全部書こうとして、肝心のルールが埋もれる。短く、案内だけにする。

ルールが矛盾している

追記を重ねて、前の指示と食い違う。点検のときに読み直す。

権限が広すぎる

「念のため全部」を渡す。便利さと事故は同じ入口から来る。

お願いで済ませる

事故になる種類を、「やらないでね」と書いて終わりにする。仕組みで止める。

記憶が増えるだけ

古い教訓を捨てない。状況が変われば、捨てる・直す。

点検しない

作って満足し、半年後にルールと実態がずれている。週1回、短くてよい。

任せるほど、関所が大事になる

AIエージェントが自分で動くようになるほど、「権限のはみ出し」が起きやすくなる。便利さと事故は同じ入口から来る。**最初に決めるのは「何をさせるか」ではなく「どこで止めるか」**。

関連資料

- F1 業務の棚卸しと仕組み化 … 手順書の書き方（スキルの材料）
- E1 はじめてのAIエージェント：フォルダ構成と管理 … **セットで使う**。置き場所と進め方、最初の30分の手順
- E4 Gitの導入と設定 … 変更履歴という「保存の金庫」。ターミナルなしで進められる
- E5 SupabaseとCloudflare … 作ったものの置き場所。RLSと無料枠
- E6 APIキー（秘密）の扱い … .env に頼らず、Vault に置く
- E7 VPSサーバー … 自分のサーバーに、Webアプリを設置するとき
- E3 ローカルLLMと「情報を外に出さない」設計
- C3 事例：この資料ライブラリの自動化・スキル化 … ハーネスの実物
- D1 信号機ルール

出典（2026年10月確認）

- GPU数から「トークン経済性」へ　Dellが示したAIインフラ競争の新基準（AI新聞・2026年5月20日） (https://community.exawizards.com/aishinbun/news-05-20-2026/)：NVIDIAのハーネスの定義
- チャットボットを「自律型AI」に変える鍵、『4つの記憶とガードレール』（ITmedia・2026年5月） (https://blogs.itmedia.co.jp/osonoi/2026/05/ai4.html)：記憶4種（CoALA）、ガードレール、ルールを置く場所
- Claude 5世代のコンテキストエンジニアリング（@IT・Anthropic原文の解説） (https://atmarkit.itmedia.co.jp/ait/articles/2609/03/news040.html)：指示書は軽く、判断を渡す、手順は段階的に
- 「最強モデル」はもう無意味　ナデラCEOが語る「学習ループ」（@IT・2026年6月） (https://atmarkit.itmedia.co.jp/ait/spv/2606/25/news016.html)
- 全ての企業が「自分たちのAI」を構築する時代＝Microsoftのサティア・ナデラ氏（AI新聞・2026年6月5日） (https://aishinbun.com/)：モデル＋ハーネス＋自社専用の採点表
- Claude Code 公式ドキュメント：メモリ（CLAUDE.md・自動メモリ） (https://code.claude.com/docs/en/memory)：200行、25KB、4MiB、@importの4段、置き場所と読み込み順（2026年10月3日確認）
- Claude Code 公式ドキュメント：スキル（SKILL.md） (https://code.claude.com/docs/en/skills)：500行、説明文1,536文字