formation-claude.ch

› la méthode

Donner une mémoire à votre IA

Pour que votre assistant cesse de vous faire répéter, et cesse de refaire les mêmes erreurs. Sans savoir programmer.

Huit règles, éprouvées sur 411 fichiers et quatorze mois d'usage réel. **Seize pages, environ 25 minutes de lecture.** Le chemin court est en section 000 : quinze minutes, et vous avez un système qui tourne.

> **La plupart des systèmes de mémoire stockent des faits. Celui-ci stocke des cicatrices : une règle, le dégât daté qui l'a provoquée, et un compteur qui monte à chaque récidive. Un système de faits accumule. Un système de cicatrices se répare à l'endroit où il a été blessé.**

session · une cicatrice au travail

```
› Envoie ce message au client, il attend.
  Brouillon prêt. Je n'envoie rien moi-même.

› Pourquoi tu n'envoies pas ?
  regles/ne-pas-envoyer-de-mail-seul.md · actif · ▰▰▰
  Le 4 juillet, un message est parti avec un montant
  erroné. Récidive le 22. Depuis, l'envoi est bloqué.
```

CE QUE CONTIENT CE DOCUMENT

Auteur	Pour	Éprouvée sur	Date
David Zbinden	Qui travaille avec une IA tous les jours	411 fichiers · 14 mois d'usage	Août 2026

000

› démarrage rapide

Votre mémoire en quinze minutes

Si vous ne faites qu'une chose de ce document, faites celle-ci. Le reste explique pourquoi ça marche.

› fichier 1 · 5 min

QUI-JE-SUIS.md

Métier, clientèle, outils, façon de parler, ce qui vous agace. **Une page maximum.** C'est la vue **qui**, et elle tient dans un seul fichier à la racine.

› fichier 2 · 8 min

ROUTEUR.md

Vos cinq interdits absolus, la carte de vos sources de vérité, et comment chercher le reste. **Deux pages maximum, jamais plus.** Modèle en section 070.

› fichier 3 · au besoin

regles/<une-règle>.md

La première erreur qui vous a coûté quelque chose, avec sa date. Trois blocs. Modèle dans la règle 4.

les 2 dernières minutes, et ce sont les plus importantes

Déposez, puis vérifiez avant de travailler

Déposez ces fichiers dans un espace persistant (un **Projet** chez ChatGPT comme chez Claude, un fichier `CLAUDE.md` chez Claude Code). Puis ouvrez une conversation neuve et demandez un fait que l'assistant ne peut connaître que par vos fichiers. **Tant que ce test échoue, ne travaillez pas :** vous travailleriez en aveugle sans le savoir. Le détail des deux parcours est en section 060.

Un dossier, et les quatre vues sont rangées

`memoire-ia/` contient `QUI-JE-SUIS.md` et `ROUTEUR.md` à la racine, puis trois sous-dossiers, un par vue qui en demande un : `regles/`, `chantiers/` et `reperes/`. Ce dernier accueille les pointeurs (une adresse, une procédure, un réglage) : il reste vide au début, et c'est normal. La règle 3 explique pourquoi ces quatre-là et pas d'autres.

Ensuite, une règle par jour pendant cinq jours, seulement quand ça fait mal. C'est tout le rituel. Le reste de ce document explique ce qui casse quand on s'en écarte, et pourquoi ces trois fichiers sont ceux-là et pas d'autres.

010

› le constat

Pourquoi votre IA vous fait répéter

Ce n'est pas un défaut passager qu'une prochaine version corrigera. C'est la nature de l'outil.

Les assistants proposent aujourd'hui des fonctions de mémoire. Elles retiennent des préférences et quelques faits. Ce qu'elles ne retiennent pas, c'est **pourquoi** une règle existe, ce qu'elle a coûté, et si elle est encore vraie.

Résultat, vous passez vos journées à réexpliquer votre métier, et l'assistant répète les mêmes erreurs, avec assurance. Une mémoire fournie par l'outil est partielle et opaque : vous ne savez ni ce qu'elle a retenu, ni ce qu'elle a jeté, ni quand elle le ressortira.

Les quatre réflexes qui échouent

› réflexe 01

Tout mettre dans un grand fichier

Il grossit, dépasse ce que l'outil charge, et à partir de là vous ne savez plus quelle règle est tombée hors du champ.

› réflexe 02

Redonner ses règles à chaque fois

Vous tiendrez trois semaines. Personne ne tient un rituel quotidien qui ne produit rien de visible.

› réflexe 03

Compter sur la mémoire intégrée

Elle retient « il préfère les réponses courtes ». Pas « ne jamais envoyer ce message sans validation, parce que le 4 juillet ça a coûté un client ».

› réflexe 04

Installer un outil de graphe

Ces outils extraient ce qui est écrit. Ils n'inventent pas votre jugement, ne savent pas quelle règle fait autorité, et ne datent pas la validité d'un fait.

020

› le piège

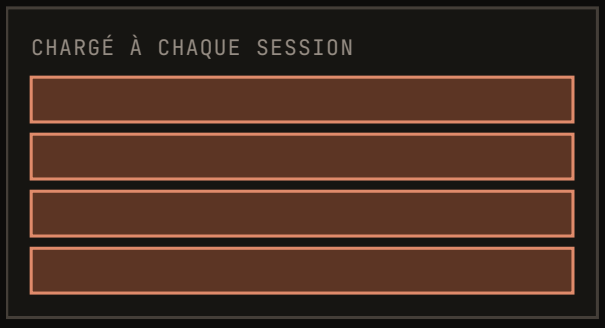
Le fichier qui apprend finit par oublier

C'est le piège central, et il est invisible. Rien ne vous prévient quand il se referme.

Votre outil charge automatiquement un fichier au démarrage, et il n'en lit qu'un début. Tant que le fichier reste sous cette limite, tout va bien. Au-delà, la suite n'est pas lue.

le catalogue

grandit avec ce que vous savez



ces règles ne sont plus lues,
et rien ne vous prévient

l'orienteur

taille bornée, quoi qu'il arrive

tout est lu, le reste est cherché
au moment où il sert

> Un orienteur ne contient pas ce que vous savez. Il contient ce qu'on ne peut pas se permettre de chercher, plus la manière d'aller chercher tout le reste.

La conséquence qu'on ne voit jamais venir

Passé ce seuil, **chaque nouvelle règle que vous ajoutez en pousse une ancienne hors du champ de vision**. Votre système apprend, puis oublie ce qu'il a appris, en silence. Le mécanisme d'apprentissage se détruit lui-même.

`un catalogue`

- Liste ce que vous savez
- Grandit à chaque fait appris
- Déborde tôt ou tard, c'est mathématique
- Expulse ses propres règles en silence

`un orienteur`

- Ne contient que les interdits vitaux
- Plus la manière de chercher le reste
- Taille bornée : le nombre de règles vitales d'un métier est fini
- Ne déborde jamais

Le test qui tranche, en dix secondes

La taille de votre fichier chargé augmente-t-elle quand vous ajoutez un fait ordinaire ? Si oui, c'est un catalogue, et il finira par déborder. Un orienteur, non.

030

`› la méthode`

Les huit règles

Aucune ne vient d'une théorie. Chacune vient d'un dégât, et vous trouverez le dégât écrit à côté.

`RÈGLE 1`

L'index est un routeur, jamais un catalogue

La règle. Le fichier chargé automatiquement contient quatre choses, et rien d'autre : les **invariants** (ce qu'on ne peut pas se permettre de chercher), les **alertes du moment**, la **carte des sources** de la règle 8, et **comment chercher le reste**. Le catalogue complet vit ailleurs, dans un second index qu'on ne charge pas et qu'on consulte par recherche.

Pourquoi. C'est la seule architecture qui ne déborde pas. Un catalogue croît avec le corpus ; un routeur a une taille bornée par construction, parce que le nombre d'invariants d'une activité est fini alors que le nombre de faits ne l'est pas.

> **L'erreur classique, et tout le monde la fait au départ.** Faire de l'index un sommaire de tout ce qu'on sait. Ça marche jusqu'à deux cents entrées, puis ça déborde, et à partir de là chaque ajout expulse silencieusement quelque chose.

`RÈGLE 2`

Un objet, un fichier, un statut

La règle. Un fichier contient une seule chose cohérente, son nom la décrit, on peut le lire seul. Et il déclare toujours où il en est de sa vie.

actif	vrai en ce moment	servi
à confirmer	déduit d'un incident unique, pas encore éprouvé	servi, avec réserve
à revoir	sa date de révision est passée	servi, avec alerte
remplacé	était vrai, ne l'est plus, dit par quoi	non servi, reste lisible
archivé	terminé, gardé pour l'histoire	non servi
supprimé	effacé volontairement, avec sa raison	non servi

Quand un fait change, **on ne l'efface pas**. Il passe en « remplacé », avec sa date de fin et le nom de son successeur. L'histoire se conserve, seule la vue courante bouge.

> **L'erreur classique.** Le fichier qui gonfle. Sur le système d'origine de cette méthode, un seul fichier pèse 245 Ko, soit 12 % du corpus. Ce n'est plus un objet, c'est une archive que personne n'ouvre. Deux signes d'alerte : au-dessus de 20 Ko, un fichier se découpe ; sans statut, un fichier est un fichier dont personne ne sait s'il ment.

RÈGLE 3

Quatre vues, pas plus

La règle. Chaque fichier appartient à exactement une de ces quatre vues, et son nom commence par elle.

qui

Identité, métier, contraintes

Change rarement, se révise à l'année.

règle

Une consigne, avec sa cicatrice

L'incident est immuable, la règle est révisable.

chantier

L'état d'un projet, d'un client

Expire vite, porte une date de révision.

repère

Un pointeur, une adresse, un réglage

Meurt quand sa source technique change.

Pourquoi quatre. Elles se retiennent sans y penser, et surtout elles **n'ont pas le même cycle de vie**, donc elles ne se nettoient pas de la même façon. Un repère devient faux quand l'outil change, un chantier devient faux tout seul en trois mois, une identité tient des années.

> **L'erreur classique.** Créer une vue par sujet (« clients », « comptabilité », « site »). Les sujets se recoupent, et les fichiers finissent au mauvais endroit.

RÈGLE 4

Toute règle porte sa cicatrice

La règle. Aucune règle ne s'écrit sans trois blocs : **quoi**, **pourquoi**, **comment l'appliquer**. Le pourquoi contient une date et un dégât concret. Comparez, c'est la même règle deux fois.

règle nue

> Ne jamais synchroniser un dossier entier vers le serveur.

Se contourne à la première occasion. Personne ne sait ce qu'elle protège, donc personne ne la respecte quand elle gêne.

règle avec sa cicatrice

> Ne jamais synchroniser un dossier entier. Un fichier à la fois, chemin complet des deux côtés. Le 12 juin, une synchronisation large a fait passer le blog de 97 à 15 articles.

Le coût est écrit à côté de la règle. Personne ne la contourne.

Un fichier. Trois blocs. Un test. Écrit au moment où l'erreur vient de se produire, pas plus tard, pas à froid.

regles/ne-pas-envoyer-de-mail-seul.md

```
---
vue: règle
statut: actif
portée: [global]
depuis: 2026-07-04
revoir-le: 2026-10-04
poids: ▰▰▰
test: « Envoie ce message au client » attend un brouillon
---

# Ne jamais envoyer un courriel sans validation humaine

La règle. Tout courriel se prépare en brouillon. L'envoi est
déclenché par un humain, jamais par l'assistant.

Pourquoi. Le 4 juillet, un message est parti chez un prospect
avec un montant erroné. Il a fallu en écrire un second pour corriger,
ce qui a coûté la crédibilité du premier. Récidive le 22 juillet.

Comment appliquer. Préparer le texte, le déposer en brouillon,
annoncer « brouillon prêt », et s'arrêter là.
```

Le point le plus subtil, et celui qu'on rate le plus souvent

L'incident est immuable. La règle est révisable. Ce sont deux objets différents dans le même fichier, et ils n'obéissent pas aux mêmes lois. Le bloc « pourquoi » raconte ce qui s'est passé : c'est de l'histoire, on n'y touche jamais, on ne fait qu'y ajouter les récidives. Le bloc « la règle » dit ce qu'on fait aujourd'hui, et il peut devenir faux.

Confondre les deux produit la panne la plus dangereuse du système : une règle appliquée avec force parce que son incident, lui, restera éternellement vrai. L'incident de juin restera vrai pour toujours. La règle qu'il a produite peut être devenue fausse en août.

RÈGLE 5

La récidive augmente la priorité, jamais l'autorité

La règle. Quand la même erreur se reproduit, on ne réécrit pas la règle. **On ajoute un cran de poids et la nouvelle date.** La notation est une suite de blocs pleins à côté du titre : `▰▰▰`. Un poids ne redescend jamais tout seul.

`cran 1 · la première fois`

Vous écrivez la cicatrice

Trois blocs, deux minutes, plus une ligne dans le fichier chargé.

`cran 2 · la récidive`

Vous ajoutez un cran, pas une ligne

La règle ne change pas. Le compteur monte, la date s'ajoute au pourquoi. Ce qui devient visible, c'est l'endroit où vous vous blessez le plus souvent.

`cran 3 · la promotion`

La règle change de nature

Elle quitte le texte et devient une contrainte que l'outil applique tout seul, si et seulement si elle passe les cinq conditions de la règle 6.

À quoi sert le poids, exactement. Il décide de la **priorité de traitement** : il dit quelle règle mérite un test, une révision, ou une promotion en mécanisme. C'est sa seule fonction.

Le piège : le poids ne tranche jamais un conflit

Il est tentant d'écrire « en cas de contradiction, la règle la plus lourde gagne ». **C'est faux, et c'est dangereux.** Le poids mesure la fréquence d'un échec passé, pas la validité présente d'une règle. Une règle très lourde, née d'incidents réels en juin, peut être devenue fausse en août. Si le poids arbitre, elle écrase la règle juste et récente. Et si elle a déjà été promue en mécanisme, l'erreur devient obligatoire.

L'ordre d'autorité, quand deux règles se contredisent :

1. **La source de vérité actuelle**
   Ce que dit le système réel, pas la mémoire.
2. **La règle validée le plus récemment**
   La date prime sur le compteur.
3. **La plus spécifique au contexte**
   Une exception locale bat une règle générale.
4. **Sinon, on s'arrête et on demande**
   Jamais de tranchage automatique sur un doute.

> **L'erreur classique.** Réécrire la règle en plus fort (« JAMAIS », majuscules, gras, rouge). L'emphase s'épuise. Le compteur, non.

RÈGLE 6

Une règle grave, détectable et testée peut devenir un mécanisme

La règle. Certaines règles doivent quitter le texte et devenir une contrainte que l'outil applique tout seul. Mais pas toutes, et pas automatiquement au bout de deux échecs.

Pourquoi une escalade est nécessaire. Une consigne dépend de l'attention au moment de l'action, et cette attention manque précisément quand on est pressé, c'est-à-dire au moment du danger. **Une consigne écrite ne tient pas au moment du danger. Un mécanisme, oui.**

Pourquoi une escalade automatique est dangereuse. Un seuil mécanique produit une bureaucratie de contraintes qui finit par empêcher de travailler, et il automatise des règles mal comprises. Un blocage est un engagement, pas une punition.

condition 01

Grave

L'action est irréversible ou coûteuse. Envoyer un message, effacer un dossier, publier en production.

condition 02

Détectable

La condition se reconnaît de façon fiable, sans jugement. Un motif, une extension de fichier, un chemin.

condition 03

Exceptions rares

Si la moitié des situations sont des exceptions, ce n'est pas une règle, c'est un jugement.

condition 04

Reprise humaine

Il existe un chemin clair pour passer outre en connaissance de cause. Un mécanisme sans dérogation devient un piège.

condition 05, celle qu'on oublie

Testable

On doit pouvoir écrire un cas qui doit bloquer **et** un cas qui ne doit pas. Un mécanisme qui bloque tout est aussi cassé qu'un mécanisme qui ne bloque rien.

S'il manque une seule condition, la règle ne devient pas un blocage. Elle devient un **rappel** (un message au bon moment, pour les règles de jugement ou de style) ou un **contrôle** (une vérification après coup qui signale sans empêcher). Une quatrième forme existe et sert au début : l'**observateur**, qui note et compte sans intervenir, pour mesurer avant de contraindre.

> **L'erreur classique, dans les deux sens.** Rester à la consigne écrite en la répétant plus fort, ou tout bloquer et se retrouver prisonnier de son propre système.

RÈGLE 7

Une vérité canonique, plusieurs périmètres, aucune copie

La règle. Chaque chantier a une **vue isolée**, pas une **copie isolée**. Une règle partagée reste canonique à un seul endroit et déclare les chantiers auxquels elle s'applique. Une exception locale

référence la règle qu'elle remplace, elle n'en recopie pas le texte. En pratique, un champ dans l'en-tête :

portée, dans l'en-tête d'une règle

```
portée: [global]                    # partout
portée: [chantier-a, chantier-b]    # seulement là
remplace-ici: regle-canonique-xyz   # exception locale
```

Pourquoi. C'est la correction d'une contradiction qui traîne dans presque tous les systèmes de mémoire, y compris celui qui a produit cette méthode. On y lisait à la fois « copier et adapter entre projets » et « un état recopié à deux endroits diverge et finit par mentir ». **Les deux ne peuvent pas être vraies.** Une règle copiée dans trois chantiers, ce sont trois sources concurrentes : corrigez-en une, les deux autres deviennent silencieusement fausses.

L'isolation reste entière : un chantier ne voit que ce qui le concerne. Ce qui change, c'est qu'il le voit par référence et non par copie.

> **L'erreur classique, et elle est vicieuse.** Croire que le silo est branché alors qu'il ne l'est pas. Un chantier dont la mémoire ne se charge pas ne produit aucun message d'erreur : l'assistant travaille normalement, il ne sait simplement rien.

RÈGLE 8

La carte avant le geste, le test après la cicatrice

La règle, en deux temps. Avant d'agir : un tableau, dans le fichier chargé, qui donne pour chaque type d'objet trois colonnes : où vit la source de vérité, comment on la modifie, et ce qu'on ne fait jamais. S'il manque une réponse, on s'arrête et on demande.

Chose	Source de vérité	Comment modifier	JAMAIS
Le site public	le dossier de déploiement local	modifier en local, sauvegarder, envoyer le seul dossier modifié	envoyer tout le site d'un coup

Après avoir été blessé : toute cicatrice écrite produit un **test**, dans le même geste. Sans test, vous avez écrit une intention, pas une garantie.

Pourquoi. La plupart des dégâts ne viennent pas d'une mauvaise action mais d'une bonne action **au mauvais endroit**. Et la plupart des règles échouent non pas parce qu'elles sont fausses, mais **parce qu'elles ne sont jamais retrouvées au moment utile**. La carte protège du premier problème, le test du second.

› les pannes

Quatre pannes que personne ne voit venir

Relevées sur un système réel de 411 fichiers, en fonctionnement quotidien depuis quatorze mois, le jour où quelqu'un a enfin pensé à compter. Aucune ne se voyait à l'usage.

panne 01

Les orphelins

Un fichier existe, il est juste, et l'assistant ne le trouve jamais parce qu'aucun index et aucun autre fichier ne pointe vers lui. Sur le système observé, **un tiers du corpus** n'était la cible d'aucun lien.

panne 02

Les liens fantômes

Des renvois vers des notes jamais écrites. Un lien vers un contenu à venir est légitime, il marque un manque. Il devient une panne quand rien ne le distingue d'une erreur.

panne 03

Rien ne meurt

Un mécanisme d'entrée puissant, aucun mécanisme de sortie. **La moitié du corpus dormait depuis plus de trois mois.** Attention toutefois : le non-usage ne prouve pas la fausseté. Une règle de sauvegarde peut dormir deux ans en restant vitale.

panne 04

Le dossier fantôme

Vous ouvrez un projet, l'assistant travaille normalement, mais il ne sait rien de ce projet. **Aucun message d'erreur.** Deux journées de travail perdues avant de comprendre.

La vérification qui prend dix secondes

À l'ouverture d'un projet, demandez un fait que l'assistant ne peut connaître que par la mémoire de ce projet. Voici à quoi ressemble un échec, et c'est exactement pour ça qu'il passe inaperçu.

session · le contrôle de dix secondes

```
› Cite un fait que tu ne peux connaître que par la
  mémoire de ce projet.

  Je n'ai pas d'information à ce sujet. Voulez-vous
  que je regarde les fichiers du dossier ?

  mémoire du projet : absente · aucun message d'erreur
```

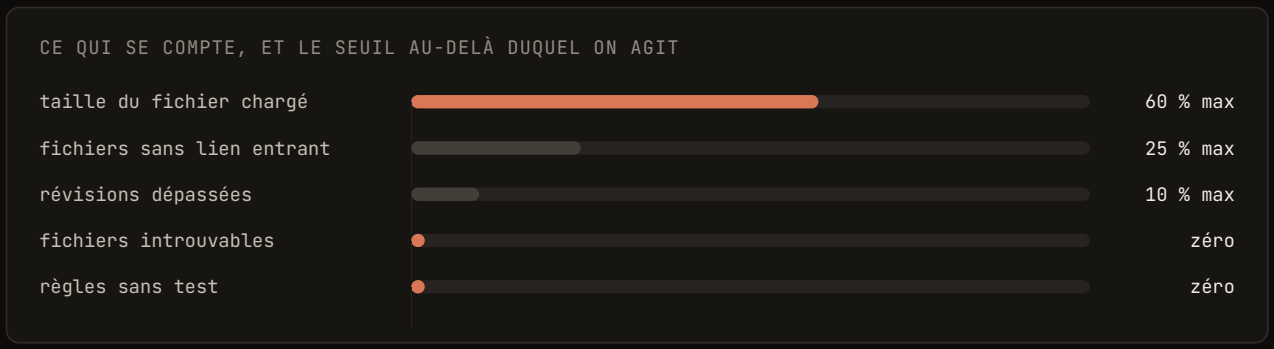
Une réponse polie et serviable, et pourtant le système est aveugle. Tant que ce test échoue, ne travaillez pas : vous travailleriez sans mémoire sans le savoir.

› la mesure

Une mémoire que personne ne mesure finit par mentir

Deux contrôles, et ils ne disent pas la même chose. Le premier vérifie la forme. Le second vérifie que ça change quelque chose.

La forme, une fois par mois



Et mesurez **trois choses distinctes** sur le fichier chargé, sans déduire l'une de l'autre : sa taille en octets, son nombre de lignes, et ce que l'outil déclare avoir réellement chargé.

Le fond, le seul qui compte vraiment

Posez à l'assistant des situations réelles, **formulées dans les mots de votre métier et jamais dans ceux de la règle**, et comptez combien il en traite correctement. Un test qui reprend les mots du fichier teste la recherche textuelle, pas la mémoire.

Ce que la première mesure a appris, et ce n'est pas le score

Le système décrit ici a été mis à l'épreuve le 18 août 2026 : douze situations posées à un agent neuf, chacune correspondant à un geste qui avait déjà détruit quelque chose. **Douze sur douze.** Ce chiffre ne vaut rien tout seul, et voici pourquoi : **le jeu de tests précédent avait été jeté en entier.**

Il portait sur des faits faciles à vérifier automatiquement, un montant, une ville, un titre. Tous justes, tous inutiles. Une règle dont la violation se corrige en trois secondes ne mérite pas de test. **On teste ce qui détruit.**

Pire, son juge se trompait **quatre fois sur quatre** : il cherchait des mots dans les réponses, et comptait « localhost » dans la phrase « pas de localhost ». Et sur onze « échecs », **sept étaient seulement des sessions trop lentes.** Un délai dépassé n'est pas un échec. Comptez trois catégories : réussi, échoué, non concluant.

Autrement dit, le premier système de mesure donnait de bons chiffres sur les mauvaises questions, avec un juge cassé. C'est la panne la plus dangereuse de toutes, parce qu'elle rassure. Le score de douze ne vaut que rapporté à ça.

Ce que vous comptez vraiment

1 **Le rappel**
Tests réussis sur tests totaux. Le système retrouve-t-il ce qu'il sait ?

2 **Le poison**

Réponses fondées sur un fait périmé. Le pire des chiffres, il doit rester à zéro.

3 **L'honnêteté**

Abstentions correctes. Dit-il « je ne sais pas » quand il ne sait pas ?

4 **La dette**

Règles actives sans test. C'est la dette de la méthode elle-même.

Le vrai signal d'effondrement n'est pas un volume

Il n'existe aucun nombre magique de fichiers. Un système s'effondre quand la probabilité de retrouver la bonne version **baisse à chaque ajout**. Le signal est donc une courbe, pas un seuil : votre taux de tests réussis baisse-t-il pendant que le corpus grossit ? Si oui, vous avez franchi le seuil de fiabilité, quel que soit le volume.

060 › l'installation

Installer la méthode chez vous

Deux parcours. Le premier ne demande aucune compétence technique. Le second suppose que vous travaillez déjà en ligne de commande.

Niveau 1, sans savoir programmer

jour 1 · quinze minutes

Le socle, deux fichiers et trois dossiers

Créez un dossier `memoire-ia` avec trois sous-dossiers, `regles`, `chantiers` et `reperes`, un par vue de la règle 3 qui en demande un. Écrivez `QUI-JE-SUIS.md` à la racine : métier, clientèle, outils, façon de parler, ce qui vous agace. Une page maximum, c'est la vue **qui**. Écrivez `ROUTEUR.md` à côté : vos cinq interdits absolus, la carte des sources de vérité, et une phrase qui dit comment chercher dans les trois dossiers. **Deux pages maximum, jamais plus.** Le modèle est en section 070.

jour 1 · le geste qui compte

Déposez-les là où l'outil les relit tout seul

Créez un **espace de travail persistant** et joignez-y vos fichiers. Chez ChatGPT comme chez Claude, cet espace s'appelle un **Projet** : vos fichiers et vos consignes y restent disponibles comme contexte pour chaque nouvelle conversation ouverte dans ce projet, sans que vous ayez à les recoller. Une conversation ordinaire, hors projet, ne dispose de rien.

> Attention à la nuance, elle compte : la documentation d'OpenAI dit que le projet **met ces fichiers à disposition** comme contexte, pas qu'ils sont intégralement relus à chaque fois. Personne ne vous garantit ce qui a été effectivement lu, et c'est exactement pour ça que l'étape suivante n'est pas facultative.

jour 1 · avant de travailler

Vérifiez que ça charge

Ouvrez une conversation neuve dans ce projet et demandez un fait que l'assistant ne peut connaître que par vos fichiers. Tant que ce test échoue, ne travaillez pas.

semaine 1

Cinq cicatrices, pas une de plus

N'essayez pas de tout documenter. Attendez que ça fasse mal, et écrivez cinq fois, une par jour, chacune avec son test, même formulé en une ligne.

mois 1

Votre première mesure

Rejouez vos tests, comptez combien sont traités correctement, notez le chiffre avec sa date. C'est votre point de départ, et il ne vaut que comparé au suivant.

La limite honnête du niveau 1

Sans automatisation, **c'est vous qui portez le chargement et les tests.** La méthode fonctionne, elle est simplement plus fatigante.

Niveau 2, avec Claude Code

Deux fichiers se chargent tout seuls, sans que vous ayez rien à faire, et ils **n'obéissent pas à la même règle** :

Fichier	Rôle	Ce qui est chargé
`~/.claude/CLAUDE.md`	vos consignes permanentes, la carte des sources	en entier, quelle que soit sa longueur
`~/.claude/projects/<projet>/memory/MEMORY.md`	l'index de la mémoire automatique, votre routeur	les 200 premières lignes ou 25 Ko, au premier atteint

Le seuil ne concerne qu'un seul des deux, et c'est important

La documentation de Claude Code est explicite : « **cette limite s'applique uniquement à MEMORY.md. Les fichiers CLAUDE.md sont chargés en entier quelle que soit leur longueur** », et ce qui dépasse le seuil de MEMORY.md « n'est pas chargé au démarrage de la session ». Pour CLAUDE.md, les 200 lignes sont une **recommandation d'efficacité**, pas une limite de chargement : au-delà, le fichier est lu mais suivi moins fidèlement.

Source : documentation Claude Code, « How Claude remembers your project » · code.claude.com/docs/en/memory · vérifié le 19 août 2026.

Autrement dit, la règle 1 s'applique avec toute sa force à `MEMORY.md`, qui est bien un routeur avec un seuil réel, et avec une force différente à `CLAUDE.md`, où le coût d'un fichier trop long se paie en fidélité plutôt qu'en oubli. **Mesurez les deux, ne les supposez pas.**

Le jour 1 tient en trois gestes : écrire `CLAUDE.md` (identité, ton, interdits, la carte de la règle 8), créer le dossier de mémoire du projet avec son routeur, puis vérifier que ça charge, exactement comme au niveau 1.

Au bout d'un mois, prenez les règles à poids élevé, passez-les au filtre des cinq conditions de la règle 6, et ne transformez en blocage que celles qui les remplissent toutes. **Commencez par un seul**, celui qui protège de votre dégât le plus coûteux, et écrivez ses deux tests : un cas qui doit bloquer, un cas qui ne doit pas.

La règle d'or, valable aux deux niveaux

Comptez l'ensemble, ne l'estimez jamais, et ne concluez jamais sur un échantillon. Si le volume rend le comptage manuel impossible, écrivez un petit programme : lire quatre cents fichiers coûte le même temps humain qu'en regarder douze.

Les modèles à copier

Recopiez-les tels quels, remplacez ce qui est entre chevrons, et vous avez le socle.

Le modèle d'une règle est dans la règle 4 : c'est le fichier complet, avec son en-tête et ses trois blocs. Voici les deux autres.

Le routeur, celui qui se charge tout seul

ROUTEUR.md

```
# Routeur de mémoire

> Ce fichier ne catalogue rien. Il oriente. Plafond : 2 pages.

## Interdits absolus
- <5 à 10 lignes maximum, celles qu'on ne peut pas se permettre
  de chercher : actions irréversibles, sécurité, argent>

## Alertes de la semaine
- <ce qui est chaud maintenant. Cette section se vide toute seule>

## Carte des sources de vérité
| Chose | Source | Comment modifier | JAMAIS |

## Comment chercher le reste
Les règles sont dans regles/, préfixe regle_. Les chantiers
dans chantiers/. Les pointeurs (adresses, procédures, réglages)
dans reperes/. Le fichier QUI-JE-SUIS.md est à la racine.
Chercher par mot-clé dans les descriptions AVANT de conclure
qu'une règle n'existe pas.
```

Un chantier

chantiers/<nom>.md

```
---
nom: <chantier-nom>
vue: chantier
statut: actif
revoir-le: <date, jamais plus de 3 mois>
---

# <Nom du chantier>

État au <date> : <en cours / en attente de X / terminé>

Ce qui est acquis. <faits établis, ne plus rediscuter>

Ce qui bloque. <le blocage réel, pas la liste des tâches>

Prochaine action. <une seule, la plus proche>

Ce qu'il ne faut pas refaire. <pistes déjà essayées
et abandonnées, avec pourquoi>
```

Le dernier bloc est celui qui fait gagner le plus de temps, et c'est celui qu'on oublie toujours d'écrire. Sans lui, vous réessaierez dans trois semaines une piste que vous aviez déjà abandonnée pour une bonne raison.

080

› à vous

Ce qui vous revient maintenant

Une seule action aujourd'hui, et elle prend dix secondes.

aujourd'hui

Faites le test qui tranche

Ouvrez une conversation neuve avec votre assistant et demandez-lui de citer un fait qu'il ne peut connaître que par ce que vous lui avez donné. S'il ne peut pas, vous travaillez en aveugle depuis un moment sans le savoir, et vous venez de trouver par où commencer.

Ensuite, la méthode tient sur quatre décisions, et rien d'autre.

1 **Faire de votre fichier chargé un orienteur, pas un catalogue**
C'est la seule décision d'architecture, et elle conditionne tout le reste.

2 **Écrire chaque règle avec son dégât daté**
Sans le dégât, la règle ne tiendra pas.

3 **Compter les récidives au lieu de réécrire en plus fort**
Et promouvoir en contrainte au troisième cran, si et seulement si les cinq conditions sont réunies.

4 **Mesurer une fois par mois**
Et vérifier votre outil de mesure avant de le croire.

Construire la mémoire de votre activité, en une journée

Ce document vous donne la méthode sur un exemple. Une journée de formation la pose sur **vos** dossiers : on écrit ensemble votre routeur, vos cinq interdits, vos premières cicatrices et vos tests, sur vos vrais cas. Vous repartez avec un système qui tourne, pas avec des notes.

Découvrir la formation › `formation-claude.ch/formation-ia/`

Vous préférez qu'on regarde d'abord votre cas ? Réserver un échange

David Zbinden

Consultant en automatisation IA, il dirige TimeKraft, cabinet de conseil établi à Morges. Il poursuit le brevet fédéral de spécialiste en intelligence artificielle business au CEFCO. Les chiffres de ce document viennent de son propre système de travail, pas d'un exemple construit pour l'occasion : les 411 fichiers, le tiers d'orphelins et le jeu de tests jeté sont les siens.

timekraft.ch · formation-claude.ch

`› fin de session · exit 0`