

Paměť oprav. AI, která si pamatuje, co jste jí vytkli.

Byl jsem neustále nespokojený s výstupy umělé inteligence a každý text jsem několikrát ručně upravoval. Tak jsem si postavil systém, který při každé mé úpravě porovná původní AI výstup s finální verzí, a když se stejná oprava opakuje, odvodí z ní pravidlo. To si AI načte před každou další prací. Tady je celý postup, jak si ho postavíte u sebe, a soubor, který to udělá s vámi.

Co dostáváte.

BALÍČEK

Hotový systém, ne ukázka

Jeden soubor `pamet.py` obsahuje celý mechanismus. Bez závislostí, bez API klíče, bez sítě. Python 3, který na Macu a Linuxu je a na Windows se stáhne za dvě minuty.

INSTALACE

Postaví ho s vámi váš AI nástroj

Rozbalíte složku, otevřete v ní Claude Code nebo Codex a napíšete jednu větu. Nástroj podle `INSTALACE.md` zapojí hooky, ověří instalaci a předvede vám první opravu.

TENTO DOKUMENT

Přesně, co se kdy děje

Mechanismus, schéma, instalace krok za krokem, první použití, denní provoz, návrat změn, hranice a řešení potíží. Ukázky jsou skutečné výstupy z ověřovací instalace.

■ VĚTA, NA KTERÉ TO STOJÍ

Rozdíl mezi užitečnou a otravnou AI není ve větším modelu, ale v tom, zda se učí z vašich oprav.

Model, který si nic nepamatuje, dělá stejnou chybu podesáté. Tenhle systém ji udělá dvakrát a potřetí už ne, protože vy jste z ní udělali pravidlo.

MECHANISMUS

Pět kroků, které se opakují při každém textu.

Nic z toho nevyžaduje vaši pozornost kromě jediné věci: opravujete text tak, jak jste ho opravovali doted. Zbytek se děje v hoocích, tedy v malých skriptech, které AI nástroj spouští sám před zápisem souboru, po něm, při startu a při každé vaší zprávě.

- 01

AI napíše text do souboru.

Po zápisu hook uloží snímek této AI verze do `pamet / snimky /` . Pokud soubor existoval, hook před přepsáním uložil jeho dosavadní verzi do `pamet / verze /` , aby šla kdykoli vrátit.
- 02

Vy soubor upravíte

ve svém editoru. Škrtnete slovo, přepíšete oslovení, zkrátíte odstavec. Nic dalšího nehlásíte.
- 03

Při vaší další zprávě

(nebo při startu nového sezení) hook porovná snímek se souborem na disku. Když se liší, vznikne záznam v `pamet / zaznamy /` : AI verze, vaše verze, rozdíl řádek po řádku. Záznamy se nikdy nemažou.
- 04

Z rozdílů se odvodí signály:

vyškrtnuté slovo, nahrazený výraz, vyškrtnutá věta, dlouhá pomlčka, doplněná diakritika, tykání a vykání, velké a malé Vy, emoji, zkrácení, odrážky. Asistent navíc každou opravu pojmenuje jednou větou (štítek). Když se stejný signál nebo štítek objeví **ve dvou různých záznamech**, vznikne návrh pravidla.
- 05

Návrh schválíte vy.

Pravidlo se zapíše do `pamet / pravidla .md` i s odkazem na záznamy, ze kterých vzniklo. Tento soubor si AI načte před každou další prací, takže příští text už opravu obsahuje.

TŘI HRANICE, VYNUCENÉ KÓDEM

Pravidlo vzniká jen z vaší opravy.

Příkaz pro zápis pravidla odmítne text, který nemá zdrojový záznam. Nikdy z internetu, nikdy z odhadu.

Nic se nemaže.

Záznamy i zálohy verzí jsou trvalé, žádný příkaz je neumí odstranit. U každého pravidla dohledáte, z čeho vzniklo.

Nic se neodesílá.

Skript nemá síťový kód. Příkaz `kontrola` to ověří a vypíše.

CO TO NENÍ

Není to trénování modelu.

Model zůstává stejný. Mění se instrukce, které dostává před prací, a ty máte v čitelném souboru pod kontrolou.

Není to černá skříňka.

Každé pravidlo je jeden řádek s odkazem na dva soubory, které si můžete otevřít.

Není to služba.

Běží ve vaší složce, na vašem počítači. Když balíček smažete, nic po něm nezůstane.

PŘÍKLAD Z JEDNOHO TÝDNE: TŘI TEXTY, JEDNO PRAVIDLO

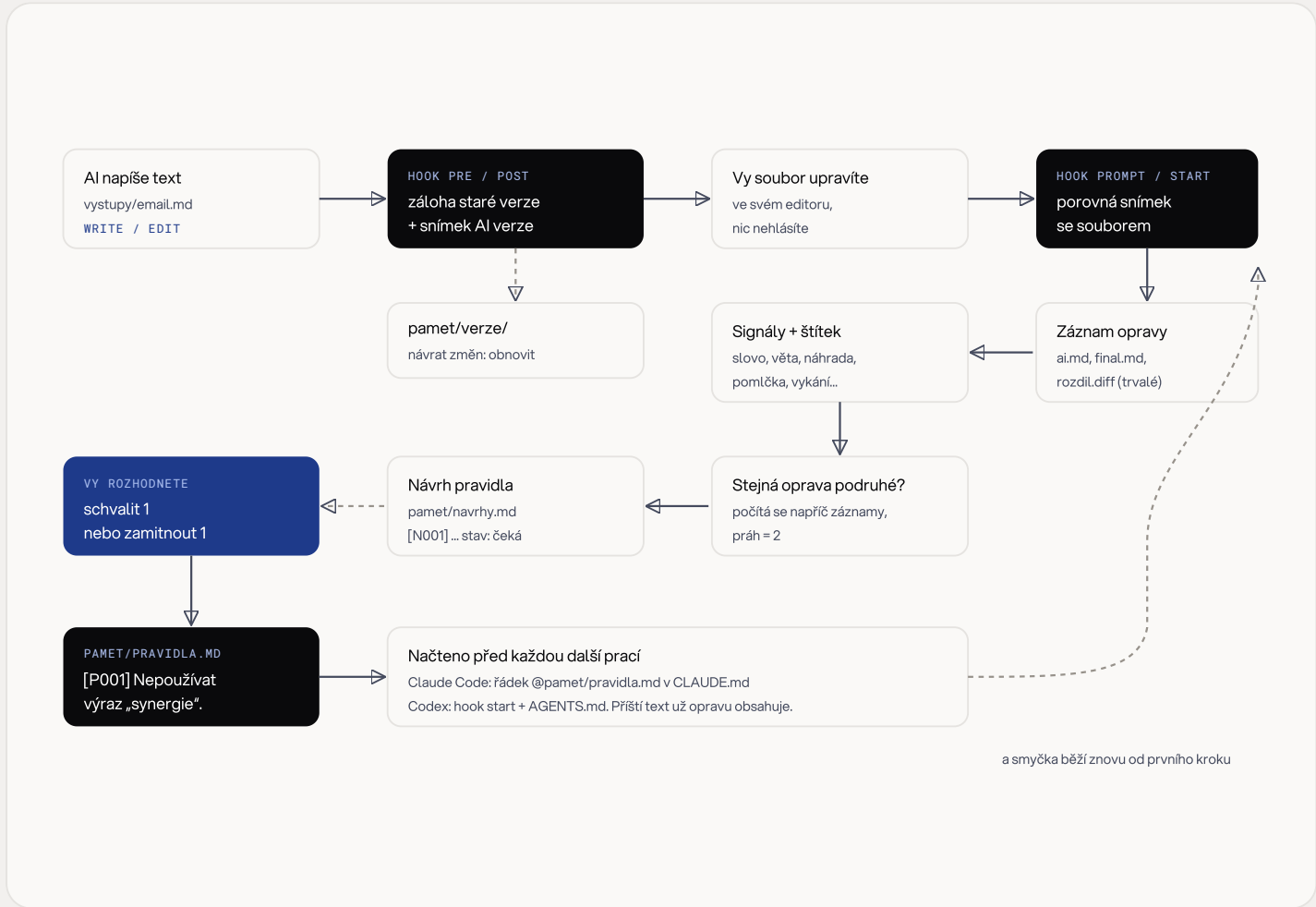
DEN	CO SE STALO	CO UDĚLAL SYSTÉM
pondělí	AI napsala e-mail dodavateli, škrtili jste slovo „synergie“ a přepsali „Váš“ na „váš“.	Záznam 1. Signály <code>slovo:synergie</code> a <code>typ:vy-male</code> , každý 1x. Nic víc.
středa	AI napsala nabídku, „synergie“ jste škrtili znovu, „Váš“ tam nebylo.	Záznam 2. <code>slovo:synergie</code> má 2 záznamy: vzniká návrh N001 „Nepoužívat výraz synergie“. Hook ho ohlásí při vaší další zprávě.
středa	Řeknete „schval“.	Pravidlo P001 v pravidla.md se zdroji záznam 1 a 2.
pátek	AI píše tiskovou zprávu.	Před prací načte P001. „Synergie“ v textu není. Vy nic neděláte.

Práh opakování je dva výskyty. Jedna oprava může být rozmar, dvě jsou vzor. Pokud chcete pravidlo hned po první opravě, jde to ručně (strana 8), ale i tehdy musí mít zdrojový záznam.

JAK TO DO SEBE ZAPADÁ

Čtyři hooky, jedna složka, žádný server.

Plné šipky jsou automatika. Přerušovaná šipka je jediné místo, kde rozhodujete vy: schválení návrhu. Vše, co je v tmavých polích, spouští AI nástroj sám.



KDE TO ŽIJE

Jedna složka v projektu

pamet/ vedle vašich textů. Přenosná: zkopírujete složku, přenesete i paměť.

CO ČTE AI

Jen pravidla.md a hlášení hooku

Záznamy čte asistent pouze, když je má pojmenovat. Do kontextu se nikdy nenačítají celé.

CO ČTETE VY

Dva čitelné soubory

pravidla.md a navrhy.md . Oba jsou obyčejný text, upravíte je v čemkoli.

ČTYŘI HOOKY PŘESNĚ (CLAUDE CODE, SOUBOR .CLAUDE/SETTINGS.JSON)

UDÁLOST	KDY JI NÁSTROJ SPUSTÍ	PŘÍKAZ	ZAPÍŠE
PreToolUse	před nástrojem Write, Edit, MultiEdit	pamet.py hook pre	verze/ (jen když soubor už existoval)
PostToolUse	po nástroji Write, Edit, MultiEdit	pamet.py hook post	snímky/
SessionStart	při spuštění nástroje ve složce	pamet.py hook start	zaznamy/, navrhy.md; do kontextu pravidla a hlášení
UserPromptSubmit	při každé vaší zprávě	pamet.py hook prompt	zaznamy/, navrhy.md; do kontextu jen novinky

PŘED INSTALACÍ

Tři věci, které musíte mít. Nic víc.

01

AI nástroj, který pracuje se soubory

Claude Code nebo Codex (oba mají hooky, oba jsou ověřené). Cursor a podobné nástroje s hooky fungují také. ChatGPT a Claude.ai v prohlížeči zvládnou ruční verzi (strana 12).

02

Python 3

Na Macu a Linuxu už je. Na Windows stáhněte z python.org a při instalaci zaškrtněte **Add python.exe to PATH**. Nic dalšího se neinstaluje, skript nemá žádné knihovny navíc.

03

Složka, ve které píšete

Systém sleduje textové soubory v jedné složce projektu. Doporučujeme zavést zvyk: AI píše do `vystupy/`, vy tam opravujete. Citlivé soubory dáte do `ignorovat.txt`.

PODPORA NÁSTROJŮ (STAV ZÁŘÍ 2026, OVĚŘENO INSTALACÍ)

NÁSTROJ	ZACHYCENÍ OPRAV	NAČTENÍ PRAVIDEL	POZNÁMKA
Claude Code	automaticky (4 hooky)	automaticky (CLAUDE.md)	Plně ověřeno: hook nahlásil záznamy, asistent je pojmenoval, pravidlo nezapsal bez souhlasu.
Codex	automaticky po potvrzení důvěry hookům	hook start + AGENTS.md	Ověřeno. Codex se ptá, zda hookům z projektu důvěřovat; do té doby má asistent pojistku v AGENTS.md a spouští kontrolu sám.
Cursor, Gemini CLI a další s hooky	stejně čtyři příkazy	soubor s instrukcemi projektu	Neověřeno námi. Události "před zápisem", "po zápisu", "start", "zpráva" mapujete 1:1.
ChatGPT, Claude.ai (web)	ručně: vložíte opravenou verzi	soubor pravidla.md v projektu	Stejný princip, o dva kroky pracnější. Postup v NAVOD-CHATGPT.md.

OBSAH BALÍČKU

SOUBOR	K ČEMU JE
pamet/pamet.py	Celý mechanismus. Hooky, porovnání, záznamy, signály, návrhy, pravidla, návrat verzí, samotest, kontrola hranic.
pamet/INSTALACE.md	Soubor pro váš AI nástroj. Nahoře tři kroky pro vás, pod tím přesný postup pro asistenta.
pamet/hooks/	Hotové konfigurace: claude-settings.json, codex-hooks.json, bloky do CLAUDE.md a AGENTS.md.
pamet/NAVOD-CHATGPT.md	Ruční verze pro nástroje bez přístupu k souborům, včetně textu instrukcí projektu.
pamet/README.md	Jedna stránka: co je uvnitř a čím začít.
vznikají používáním	pravidla.md, navrhy.md, zaznamy/, snimky/, verze/, stav.json, denik.log, ignorovat.txt

CO INSTALACE ZMĚNÍ VE VAŠÍ SLOŽCE

Tři věci, nic jiného

Podsložka `pamet/`, soubor `.claude/settings.json` (nebo `.codex/hooks.json`) a blok v `CLAUDE.md` (nebo `AGENTS.md`). Existující obsah těchto souborů zůstává.

CO SE NEINSTALUJE

Žádné knihovny, žádná služba

Skript používá jen standardní knihovnu Pythonu. Nespouští se na pozadí, neběží jako služba, nemá plánovač. Žije jen v okamžicích, kdy ho hook zavolá.

KOLIK TO STOJÍ ČASU

10 minut jednou, pak nula

Instalace včetně ukázky. V provozu opravujete jako dřív a schválení návrhu je jedno slovo. První pravidla máte typicky po třech až pěti textech.

Vy uděláte tři kroky. Zbytek udělá asistent.

- Rozbalte balíček** do složky, ve které s AI přiseté texty. Vznikne v ní podsložka `pamet/`. Když takovou složku nemáte, založte novou, třeba `texty/`.
- Spusťte Claude Code v té složce.** V terminálu: `cd texty` a pak `claude`.
- Napište asistentovi jednu větu:**

ZPRÁVA PRO ASISTENTA

Nainstaluj Paměť oprav podle `pamet/INSTALACE.md` a ověř ji.

Co asistent udělá (a co uvidíte)

- Zjistí, že je Claude Code, a najde Python (`python3 --version`). Na Windows vás případně požádá o instalaci Pythonu a počká.
- Spustí samotest `python3 pamet/pamet.py test`. Ten v dočasné složce projede celou smyčku: snímek, oprava, záznam, druhá oprava, návrh, schválení, zamítnutí, odmítnutí pravidla bez zdroje, štítky, návrat verze, ignorovaný soubor. Musí skončit **17/17 v pořádku**.
- Zapojí čtyři hooky do `.claude/settings.json` v kořeni projektu. Když soubor existuje, doplní ho, nepřepíše.
- Do `CLAUDE.md` přidá blok Paměť oprav s řádkem `@pamet/pravidla.md`. Ten způsobí, že se pravidla načtou při každém startu.
- Spustí `python3 pamet/pamet.py kontrola` a ukáže vám výsledek. Očekávaný výstup:

PYTHON3 PAMET/PAMET.PY KONTROLA

```
Hranice 1 (pravidla jen ze záznamů): OK
Hranice 2 (nic se nemaže): OK
Hranice 3 (žádná síť): OK
Hooky zapojené v projektu: .claude/settings.json
Načítání pravidel před prací: CLAUDE.md
Deník: 3 událostí, poslední: 2026-09-21 16:27:37 snímek vystupy/email.md
Výsledek: vše v pořádku
```

JEDEN RESTART, KTERÝ NESMÍTE PŘESKOČIT

Hooky zapsané během sezení se načtou až při dalším spuštění. Asistent vás proto požádá, abyste sezení ukončili (`/exit`) a spustili `claude` ve stejné složce znovu. Pak mu napíšete **ukázka** a on vám předvede první opravu (strana 7).

Raději ručně? Pět příkazů

Když asistentovi nechcete svěřit zápis do nastavení, uděláte totéž sami. Vše z kořene projektu:

```
$ python3 pamet/pamet.py test # musí skončit 17/17 v pořádku
$ mkdir -p .claude && cp pamet/hooks/claude-settings.json .claude/settings.json
$ cat pamet/hooks/CLAUDE-blok.md >> CLAUDE.md # doplní blok na konec, nepřepisuje
$ python3 pamet/pamet.py kontrola # hooky i načítání musí být ANO
$ claude # nové spuštění, hooky se načtou
```

Máte už vlastní `.claude/settings.json`? Pak místo kopírování vložte čtyři události z balíčku do jeho klíče `hooks`. Struktura je stejná, jen pole doplňte.

INSTALACE JINDE NEŽ V CLAUDE CODE

Codex: stejné soubory, jeden krok navíc.

Codex používá pro hooky stejný formát jako Claude Code, jen v jiném souboru. Věta pro asistenta je totožná. Rozdíl je v tom, že Codex spouští hooky z projektu až poté, co jim v daném projektu vyslovíte důvěru.

CODEX · CO ASISTENT ZAPÍŠE

.codex/hooks.json a AGENTS.md

Hooky z `pamet/hooks/codex-hooks.json` (události Edit, Write, apply_patch, start, zpráva) a blok z `pamet/hooks/AGENTS-blok.md`. Ten blok je zároveň pojistka: asistent spustí `hook start` sám na začátku sezení, i kdyby hooky ještě neběžely.

CODEX · CO UDĚLÁTE VY

Potvrdíte důvěru hookům

Při dalším spuštění se Codex zeptá, zda spouštět hooky z tohoto projektu. Potvrdíte. Že hooky běží, poznáte v `pamet/denik.log`: objeví se řádky `hook start`, `hook pre`, `hook post`. V neinteraktivním režimu `codex exec` je potřeba přepínač `--dangerously-bypass-hook-trust`.

SKUTEČNÝ ZÁZNAM Z OVĚŘOVACÍ INSTALACE V CODEXU (PAMET/DENIK.LOG)

```
2026-09-21 16:35:37 hook start (cwd ../e2e-codex)
2026-09-21 16:36:13 hook pre (cwd ../e2e-codex)
2026-09-21 16:36:13 záloha vystupy/email.md -> verze/2026-09-21_163613__vystupy-email.md.md
2026-09-21 16:36:14 hook post (cwd ../e2e-codex)
2026-09-21 16:36:14 snímek vystupy/test2.md
2026-09-21 16:36:14 snímek vystupy/email.md
```

Jiný nástroj s hooky

Potřebujete čtyři události. Namapujte je na příkazy v tabulce; vstup je JSON na stdin s cestou k souboru v `tool_input.file_path` (nebo text patche s řádky `*** Update File:`), výstup hooku start a prompt jde do kontextu asistenta.

UDÁLOST V NÁSTROJI	PŘÍKAZ	CO UDĚLÁ
před zápisem nebo úpravou souboru	<code>python3 pamet/pamet.py hook pre</code>	uloží dosavadní verzi do verze/
po zápisu nebo úpravě souboru	<code>python3 pamet/pamet.py hook post</code>	uloží snímek AI verze
start sezení	<code>python3 pamet/pamet.py hook start</code>	porovná, zapíše záznamy, vypíše pravidla a návrhy
odeslání vaší zprávy	<code>python3 pamet/pamet.py hook prompt</code>	porovná, vypíše jen novinky

Nástroj bez hooků

Ruční režim: po každé své opravě spustíte `python3 pamet/pamet.py scan` (nebo o to požádáte asistenta na začátku práce) a v instrukcích projektu má asistent pokyn číst `pamet/pravidla.md` před každým textem. Zálohy a snímky pak dělá asistent příkazy `hook pre` a `hook post` se vstupem `{"tool_input":{"file_path":"cesta"}}`. Funguje to, jen to stojí o jeden příkaz víc.

WINDOWS: CO JE JINAK

Příkaz je `python`, ne `python3`; asistent to v hoocích přepíše. Cesty v konfiguraci píšete s lomítky (`pamet/pamet.py`), fungují i ve Windows. Konzole občas nezobrazí češtinu správně; skript si výstup vynucuje v UTF-8, takže soubory jsou v pořádku i tehdy.

VÍCE PROJEKTŮ, JEDEN ČLOVĚK

Každá složka má vlastní `pamet/` a vlastní pravidla; e-maily zákazníkům mají jiná pravidla než interní zápisy, a to je správně. Pravidla, která platí všude, zkopírujete mezi soubory `pravidla.md` ručně i se zdroji. Sloučení víc pamětí do jedné zatím neděláme schválně.

UKÁZKA, KTEROU VÁM ASISTENT PŘEDVEDE

První oprava, slovo od slova.

Toto je skutečný průběh z ověřovací instalace v Claude Code (21. 9. 2026), zkrácený jen o hlavičky. Asistent napsal e-mail zákazníkovi, my jsme v něm přepsali „Váš“ na „váš“, škrtnli „aktuálně“ a nahradili závěrečnou frázi. Pak jsme napsali „hotovo“.

HLÁŠENÍ HOOKU, KTERÉ ASISTENT DOSTAL PŘI NAŠÍ ZPRÁVĚ (VÝSTUP PAMET.PY HOOK PROMPT)

```
## Paměť oprav
Nové záznamy oprav od minula: 1
- 2026-09-21_1646_vystupy-email-md (soubor vystupy/email.md): Vy/Váš přepsáno na vy/váš; vyškrtnuta věta: Děkujeme za trpělivost a přejeme hezký den.; vyškrtnuto: aktuálně
- bylo: děkujeme za Váš dotaz k termínu dodání. Vaši objednávku aktuálně připravujeme a předpokládané doručení je do tří pracovních dnů.
Jakmile bud
+ je: děkujeme za váš dotaz k termínu dodání. vaši objednávku připravujeme a předpokládané doručení je do tří pracovních dnů. Jakmile bude
zásilka
Postup pro asistenta (Paměť oprav):
1. Pravidla výše platí pro každý text, který teď píšete. Před psaním je projděte.
2. U každého nového záznamu si přečtete pamet/zaznamy/<id>/rozdil.diff a jednou větou pojmenujte, co uživatel opravil: python3 pamet/pamet.py stitek <id> "<věta>". Stejný stitek použijte, když jde o stejný typ opravy jako dřív (viz pamet/stav.json). Při druhém výskytu vznikne návrh pravidla sám.
3. Návrh pravidla nikdy nezapíšíte do pravidla.md sami. Ukažte ho uživateli a po jeho souhlasu spusťte: python3 pamet/pamet.py schvalit <číslo>.
4. Nic z pamet/ nemažte a neodesílejte. Soubory v zaznamy/ a verze/ jsou trvalé.
```

PAMET/ZAZNAMY/2026-09-21_1646_VYSTUPY-EMAIL-MD/ROZDIL.DIFF

```
--- ai
+++ final
@@ -2,2 +2,2 @@
-děkujeme za Váš dotaz k termínu dodání. Vaši objednávku aktuálně
připravujeme a předpokládané doručení je do tří pracovních dnů. [...]
Děkujeme za trpělivost a přejeme hezký den.
+děkujeme za váš dotaz k termínu dodání. vaši objednávku připravujeme
a předpokládané doručení je do tří pracovních dnů. [...] Kdybyste
potřebovali cokoliv dalšího, napište.
```

PYTHON3 PAMET/PAMET.PY STAV (PO PRVNÍ OPRAVĚ)

```
Záznamy oprav: 1 Pravidla: 0 Návrhy čekající: 0
Sledované soubory: 1 Zálohy verzí: 0
Opakující se opravy (signál: počet záznamů):
1x typ:vy-male -> Zdvořilostní zájmena vy, vás, váš psát s malým
písmenem.
1x veta:dekujeme za trpelivost a prejeme hezky den -> Nepoužívat
větu „Děkujeme za trpělivost a přejeme hezký den.“
1x slovo:aktuálně -> Nepoužívat výraz „aktuálně“.
```

CO NA TO ASISTENT UDĚLAL (CLAUDE CODE, MODEL SONNET)

```
> cat pamet/zaznamy/2026-09-21_1629_vystupy-email-md/rozdil.diff
Nyní oba záznamy pojmenuji podle diffu.
> python3 pamet/pamet.py stitek 2026-09-21_1629_vystupy-email-md "Nahradit obecnou frází
o poděkování a přání hezkého dne konkrétní nabídkou další pomoci"
Oba záznamy jsou nyní označeny (počet 1 u každého štítku, žádný zatím nedosáhl prahu 2,
takže žádný návrh pravidla nevznikl). Pokud se stejná oprava objeví podruhé, pamet.py
sám vygeneruje návrh pravidla a ukáží vám ho ke schválení. Nic jsem do pravidla.md nezapisoval.
```

Druhá oprava: vzniká návrh

O dva texty později škrtnete „hezký den“ i v nabídce. Signál veta:dekujeme za trpelivost a prejeme hezky den má dva záznamy, práh je splněn, v navrhy.md přibude řádek a hook ho při další zprávě ohlásí:

```
Návrhy pravidel čekající na rozhodnutí uživatele:
- [N001] Nepoužívat větu „Děkujeme za trpělivost a přejeme hezký den.“ (zdroj: 2026-09-21_1629_vystupy-email-md, 2026-09-23_0910_vystupy-nabidka-md)
Zeptejte se uživatele, zda návrh schválit: python3 pamet/pamet.py schvalit <číslo> / zamitnout <číslo>
```

Řeknete „schval“, asistent spustí python3 pamet/pamet.py schvalit 1 a v pravidla.md přibude

```
- [P001] Nepoužívat větu „Děkujeme za trpělivost a přejeme hezký den.“ (zdroj: ...) .Od příštího startu ji AI nenapíše.
```

JAK S TÍM ŽÍT

Opravujete jako dřív.

Rozhodujete jedním slovem.

PSANÍ

AI příše do souborů

Nechte asistenta ukládat texty do souborů (`vystupy/nazev.md`), ne jen do chatu. Jen soubor se dá porovnat. Když text vzniká v chatu, požádejte: „ulož to do vystupy/...“.

OPRAVA

Upravujete v editoru

Cokoli, co umí uložit prostý text: VS Code, TextEdit, Poznámkový blok, Obsidian. Word ne, ten změní formát; pokud opravujete ve Wordu, uložte výsledek jako .txt a použijte `zapsat` .

ROZHODNUTÍ

Schválit, zamítnout, nebo hned

Návrh vám asistent ukáže. „Schval“ nebo „zamítní“. Když chcete pravidlo bez čekání na opakování: „udělej z toho pravidlo hned“, asistent použije `pravidlo ... --zdroj <id>` .

CO SYSTÉM POZNÁ SÁM, BEZ ASISTENTA (DETERMINISTICKÉ SIGNÁLY)

SIGNÁL	KDY VZNIKNE	ZNĚNÍ NAVRŽENÉHO PRAVIDLA
slovo:<x>	slovo z AI verze v celé vaší verzi chybí	Nepoužívat výraz „x“.
veta:<...>	celá věta (3 až 14 slov) zmizela a nemá přepsanou podobu	Nepoužívat větu „...“.
nahrada:x->y	na tomtéž řádku jste výraz x nahradili výrazem y	Místo „x“ psát „y“.
typ:pomlcka	dlouhé pomlčky nahrazeny spojovníkem	Používat jen spojovník '-‘.
typ:diakritika	stejný řádek, doplněné háčky a čárky	Psát česky s plnou diakritikou.
typ:vykani / tykani	oslovení přepsáno jedním nebo druhým směrem	Oslovovat čtenáře vykáním / tykáním.
typ:vy-male / vy-velke	Vy, Váš, Vám přepsáno na malé písmeno nebo naopak	Zdvořilostní jména psát s malým / velkým písmenem.
typ:emoji	emoji odstraněna	Nepoužívat emoji.
typ:zkraceni	text zkrácen pod dvě třetiny	Psát stručněji.
typ:odrazky / nadpisy	odrážky nebo nadpisy přepsány do vět	Místo odrážek psát souvislý text. / Méně nadpisů.
stitek:<věta>	asistent (nebo vy) pojmenuje opravu; stejný štítek podruhé	Znění štítku.

PROČ ŠTÍTKY

Signály poznají mechanické opravy. Významové („úvod je moc formální“, „chybí konkrétní termín“) pozná jen někdo, kdo text čte. Proto asistent po každém záznamu přečte rozdíl a pojmenuje ho jednou větou. Stejný štítek podruhé = návrh. Štítky nepojmenované do týdne hook připomíná při každé zprávě, dokud je asistent nedoplní.

KDYŽ NÁVRH NEDÁVÁ SMYSL

Zamítněte ho. Zůstane v `navrhy.md` označený jako zamítnutý a stejný návrh už podruhé nevznikne. Signál „slovo“ bývá občas hloupý (škrtli jste slovo náhodou dvakrát), to je v pořádku, na to je zamítnutí. Pravidlo, které přestalo platit, prostě smažte z `pravidla.md` ; záznamy zůstávají.

HRANICE V PRAXI

Každou verzi vrátíte. Nic neodejde z počítače.

Návrat změn

Před každým přepsáním existujícího souboru asistentem se dosavadní verze uloží do `pamet/verze/`. Seznam a návrat:

```
$ python3 pamet/pamet.py obnovit vystupy/email.md --vypis
1 2026-09-21 16:36:13 verze/2026-09-21_163613__vystupy-email-md.md
2 2026-09-22 09:04:51 verze/2026-09-22_090451__vystupy-email-md.md
$ python3 pamet/pamet.py obnovit vystupy/email.md --verze 1
Obnoveno: vystupy/email.md <- verze/2026-09-21_163613__vystupy-email-md.md (předchozí stav uložen do verze/)
```

I návrat je vratný: než se soubor přepíše starou verzí, jeho současný stav se uloží také. Bez `--verze` se vrátí poslední záloha.

CO DO ZÁZNAMŮ NESMÍ

Záznamy obsahují celé texty. Smlouvy, mzdy, osobní údaje zákazníků, zdravotní informace do nich nepatří. Řešení je jeden soubor: `pamet/ignorovat.txt`, jeden vzor na řádek:

```
smlouvy/
mzdy/
*.csv
klienti/osobni-udaje.md
```

Soubory odpovídající vzoru se nesnímkují ani nezalohují. Samotest to ověřuje.

CO OVĚŘIT U AI NÁSTROJE SAMOTNÉHO

Paměť oprav nic neodesílá. Ale texty, které AI nástroj píše a čte, odchází k jeho poskytovateli tak jako tak. To je vlastnost nástroje, ne tohoto systému. Než do složky dáte firemní texty, ověřte ve smlouvě nástroje, zda se vaše data používají k trénování a kde se ukládají. U placených firemních plánů bývá odpověď „ne“ a „lze nastavit region“, u bezplatných často „ano“.

Pravidla v `pravidla.md` jsou krátké věty bez citlivého obsahu. Sdílet je v týmu je bezpečné.

KONTROLA HRANIC KDYKOLI

```
$ python3 pamet/pamet.py kontrola
Hranice 1 (pravidla jen ze záznamů): OK
Hranice 2 (nic se nemaže): OK
Hranice 3 (žádná síť): OK
```

každé pravidlo odkazuje na existující záznam
ve zdrojovém kódu není žádné mazací volání
žádný síťový modul není načten

VÍC LIDÍ VE FIRMĚ

Každý svou paměť, pravidla společná

Každý má vlastní `pamet/` ve své složce. Společná pravidla sdílejte jen jako `pravidla.md` (git, sdílený disk) a načítejte ho z `CLAUDE.md`. Záznamy zůstávají u toho, kdo opravoval.

ZÁLOHA A PŘENOS

Zkopírujte složku

Celá paměť je složka `pamet/`. Přenesete ji na jiný počítač i do jiného nástroje beze změny. Do gitu patří `pravidla.md` a `navrhy.md`; `zaznamy/` a `verze/` jen tehdy, když v nich nejsou citlivé texty.

KDYŽ CHCETE SKONČIT

Smažete složku, nic nezůstane

Systém nemá žádnou stopu mimo projekt: žádný účet, žádný proces, žádný zápis do systému. Odstraníte `pamet/`, čtyři řádky hooků a blok v `CLAUDE.md`. Hotovo.

Všechny příkazy na jedné straně.

Všechno spouštíte z kořene projektu. Kde píšeme `python3` , na Windows bývá `python` . Většinu příkazů za vás spouští asistent; potřebujete znát hlavně **stav**, **schválit** a **obnovit**.

PŘÍKAZ	CO DĚLÁ
<code>pamet.py stav</code>	Přehled: počet záznamů, pravidel, čekajících návrhů, sledovaných souborů, záloh; opakující se opravy s počty.
<code>pamet.py scan</code>	Porovná sledované soubory s AI verzí, zapíše záznamy, spočítá signály, vytvoří návrhy. Totéž dělají hooky start a prompt.
<code>pamet.py zaznamy</code>	Seznam záznamů s datem, souborem a štítky.
<code>pamet.py stitek <id> "věta"</code>	Pojmenuje opravu. Stejný štítek ve dvou záznamech vytvoří návrh.
<code>pamet.py navrhly</code>	Vypíše návrhy se stavem: čeká, schváleno, zamítnuto.
<code>pamet.py schvalit <n></code>	Z návrhu udělá pravidlo v pravidla.md se zdrojovými záznamy.
<code>pamet.py zamitnout <n></code>	Označí návrh jako zamítnutý. Nic se nemaže, stejný návrh znovu nevznikne.
<code>pamet.py pravidlo "text" --zdroj <id></code>	Pravidlo ručně, hned. Bez existujícího záznamu příkaz odmítne (hranice 1).
<code>pamet.py zapsat ai.md moje.md</code>	Ručně zapíše dvojici verzí, třeba text z ChatGPT nebo z Wordu.
<code>pamet.py obnovit soubor [--vypis] [--verze n]</code>	Vrátí soubor na dřívější verzi; současný stav se před tím uloží.
<code>pamet.py kontrola</code>	Ověří tři hranice, zapojení hooků, načítání pravidel; ukáže poslední událost deníku.
<code>pamet.py test</code>	Samotest celé smyčky v dočasné složce, 17 kontrol. Projekt nemění.
<code>pamet.py hook pre post start prompt</code>	Vstupní body pro hooky. Ručně je nespouštíte.

PYTHON3 PAMET/PAMET.PY STAV (ILUSTRAČNÍ UKÁZKA PO DVOU TÝDNECH POUŽÍVÁNÍ)

Paměť oprav 1.0 | složka /Users/jana/texty/pamet

Záznamy oprav: 14 Pravidla: 5 Návrhy čekající: 1 Sledované soubory: 11 Zálohy verzí: 23

Opakující se opravy (signál: počet záznamů):

4x typ:vy-male -> Zdvořilostní zájmena vy, vás, váš psát s malým písmenem.

3x slovo:synergie -> Nepoužívat výraz „synergie“.

2x stitek:uvod je moc formalni, zacit rovnou veci -> Úvod je moc formální, začít rovnou věcí.

SLOŽKA PAMET/

pamet.py

mechanismus

pravidla.md

pravidla, čte je AI

navrhy.md

návrhy k rozhodnutí

zaznamy/

oprava = 4 soubory, trvalé

<id>/ai.md, final.md, rozdil.diff, info.json

snimky/

poslední AI verze souborů

verze/

zálohy před přepsáním, index.tsv

stav.json

počty opakování

denik.log

co kdy hooky udělaly

ignorovat.txt

co se nikdy nesnímkuje

hooks/

konfigurace pro nástroje

INSTALACE.md, NAVOD-CHATGPT.md, README.md

FORMÁT PRAVIDLA

- [P003] Misto „řešení“ psát „postup“.

(zdroj: 2026-09-22_0910_vystupy-nabidka-md, 2026-09-24_1402_vystupy-email-md)

Jeden řádek, jedno pravidlo, zdroje v závorce. Řádek můžete přeformulovat nebo smazat; číslo a zdroje nechte, ať jde dohledat původ. Čtení souboru je hloupé a odolné: hledá jen řádky začínající - [P .

© 2026 CIAD · Český institut pro AI a data · ciad.cz

10 / 12

KDYŽ NĚCO NEFUNGUJE

Deset situací, které nastanou nejčastěji.

- 01 **Hook nic nehlásí, i když jsem soubor opravil.** Nejčastěji chybí restart po instalaci. Otevřete `pamet/denik.log` : pokud tam po vaší zprávě nepříběhl řádek `hook prompt` , hooky neběží. Claude Code: ukončit a spustit znovu. Codex: potvrdit důvěru hookům. Pojistka: napište asistentovi „spuště python3 pamet/pamet.py scan“.
- 02 **„python3: command not found“.** Na Windows je příkaz `python` . Asistent to při instalaci nahradí v hoocích sám; když ne, otevřete `.claude/settings.json` a přepište `python3` na `python` .
- 03 **Záznam vznikl, ale já jsem jen přeformátoval.** To je v pořádku. Záznam bez opakování nic nezpůsobí. Asistent ho pojmenuje, štítek se podruhé neobjeví, návrh nevznikne.
- 04 **Návrh je nesmysl.** „Zamítní“. Zůstane zapsaný jako zamítnutý a znovu nevznikne. Signál „slovo“ občas zachytí náhodu (dvakrát jste škrtnli slovo „také“). Zamítnutí je součástí systému, ne selhání.
- 05 **AI pravidlo ignoruje.** Spusťte `kontrola` . Řádek „Načítání pravidel před prací“ musí uvádět `CLAUDE.md` nebo `AGENTS.md`. Když ano a pravidlo se přesto nedodrží, je moc vágní („piš lépe“). Přeformulujte ho na konkrétní instrukci („místo x psát y“).
- 06 **Opravuji ve Wordu nebo v Google Docs.** Uložte výsledek jako prostý text vedle původního souboru a zapište dvojici: `python3 pamet/pamet.py zapsat vystupy/email.md vystupy/email-moje.txt` . Nebo požádejte asistenta, ať to udělá.
- 07 **Soubor, který jsem upravil, není v hlášení.** Sledují se jen soubory, které AI zapsala nástrojem pro zápis (ne příkazem v terminálu) a které mají textovou příponu (`.md`, `.txt`, `.html`, `.csv`...). A soubory z `ignorovat.txt` nikdy. Ověřte v `pamet/snímky/` , zda pro něj existuje snímek.
- 08 **Chci začít znovu.** Přejmenujte složku `pamet/` (třeba na `pamet-stara/`) a rozbalte balíček znovu. Nic nemažte, stará paměť se může hodit. Pravidla, která chcete zachovat, zkopírujte do nového `pravidla.md` i se zdroji.
- 09 **Asistent sám upravil svůj text a vznikl záznam.** Nevznikl. Snímek se po každém zápisu asistenta obnoví, takže se porovnává vždy jeho poslední verze s vaší. Záznam vzniká jen z rozdílu mezi tím, co napsal on, a tím, co je v souboru při vaší další zprávě.
- 10 **Pravidla si píšu raději sám.** Můžete. Otevřete `pravidla.md` a přidejte řádek ve formátu `- [P009] text (zdroj: id)` . Uvedte zdrojový záznam, ať platí hranice 1i pro vás; příkaz `kontrola` to hlídá a pravidlo bez zdroje vypíše jako chybu.

KDY JE TO HOTOVÉ

Až vás potřetí nenapadne, že jste to dřív opravovali.

Prvních pár dní budete návrhy schvalovat často. Pak přestanou chodit, protože AI dělá texty tak, jak byste je psali vy. To je stav, ve kterém má systém zůstat: tichý.

CHATGPT A CLAUDE.AI V PROHLÍŽEČI

Ruční verze: stejný princip, dva kroky navíc.

Bez přístupu k souborům nemá kdo porovnávat. Porovnání tedy spustíte vy tím, že asistentovi pošlete svou opravenou verzi. Pravidla žijí v souboru `pravidla.md`, který máte nahraný v projektu. Přesný text instrukcí projektu je v `NAVOD-CHATGPT.md`.

01 · JEDNOU

Projekt s instrukcemi

Založte projekt, nahrajte `pravidla.md`, vložte instrukce z návodu. Ty asistentovi říkají, co dělat se zprávou OPRAVA a kdy smí navrhnout pravidlo.

02 · PO KAŽDÉ OPRAVĚ

Zpráva OPRAVA:

Pod slovo OPRAVA: vložíte svou verzi. Asistent vypíše rozdíly, pojmenuje opravu a podívá se do deníku na konci pravidla.md, jestli se už objevila.

03 · PŘI OPAKOVÁNÍ

SCHVALUJI a nový soubor

Po slově SCHVALUJI vypíše celý nový `pravidla.md`. Uložíte ho a nahradíte v projektu. Od té chvíle platí.

POCTIVÁ POZNÁMKA

Ruční verze drží pravidla, ale nedrží záznamy, a při přetížené konverzaci si asistent deník nemusí přečíst celý. Když s AI píšete každý den, vyplatí se přejít na Claude Code nebo Codex; dvojice textů z ChatGPT si mezitím ukládáte příkazem `zapsat`, takže o nic nepřijdete.

PRVNÍCH ČTRNÁCT DNÍ PO INSTALACI

KDY	CO UDĚLÁTE VY	CO UVIDÍTE
den 1	Instalace a ukázka podle strany 5. Deset minut.	kontrola hlásí tři hranice OK, v <code>zaznamy/</code> je první záznam se štítkem.
týden 1	Píšete a opravujete jako dřív. Asistent ukládá texty do <code>vystupy/</code> .	Hook hlásí nové záznamy, asistent je pojmenovává. Pravidla zatím skoro žádná.
týden 2	Na návrhy odpovídáte „schval“ nebo „zamítní“.	stav ukazuje opakující se opravy; v <code>pravidla.md</code> přibývají první řádky se zdroji.
pak	Jednou za měsíc projdete <code>pravidla.md</code> a smažete, co přestalo platit.	Návrhů ubývá. Záznamy zůstávají, takže u každého pravidla dohledáte, proč vzniklo.

■ NA ZÁVĚR

Tohle je kostra, na které stojí náš systém. Postavte si ji sami.

ciad.cz · Český institut pro AI a data

Český institut pro AI a data, z. ú. · **Materiál má orientační a vzdělávací charakter.** Software je poskytován bez záruky; před nasazením na firemní texty ověřte interní pravidla a smluvní podmínky používaného AI nástroje, zejména nakládání s daty. Ukázky výstupů pocházejí z ověřovací instalace CIAD z 21. 9. 2026 v Claude Code a Codexu; chování nástrojů se může s jejich novými verzemi změnit. CIAD je nezávislý na dodavatelích AI nástrojů.