AI初心者がClaude Codeで一ヶ月間開発してみた

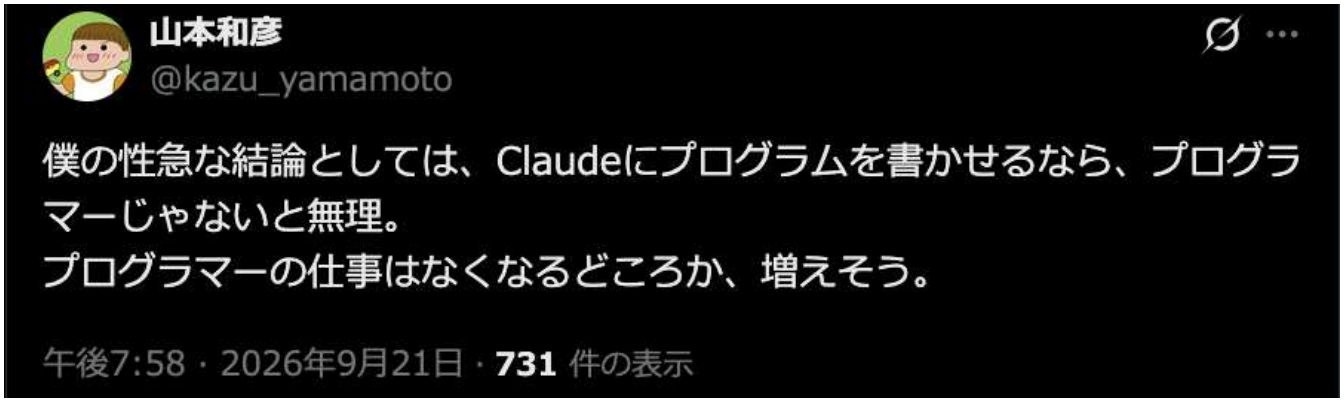
2026.10.15

技術研究所 技術開発室

山本和彦

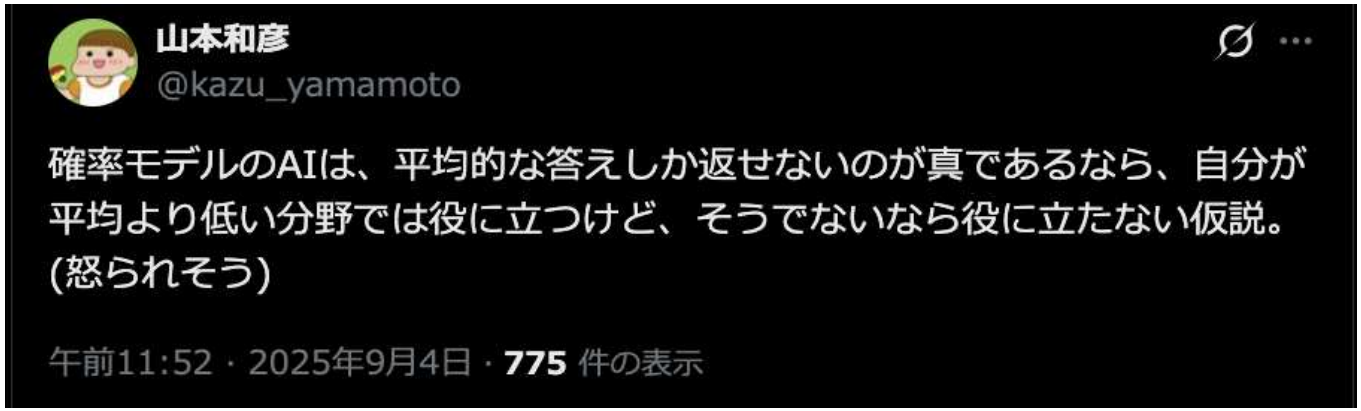
はじめに結論

- かなり使える
- 圧倒的に優秀な同僚が増えた感じ
- 日本語で仕事をお願いするだけなので、すぐ使える
- 今すぐ、使い始めた方がよい



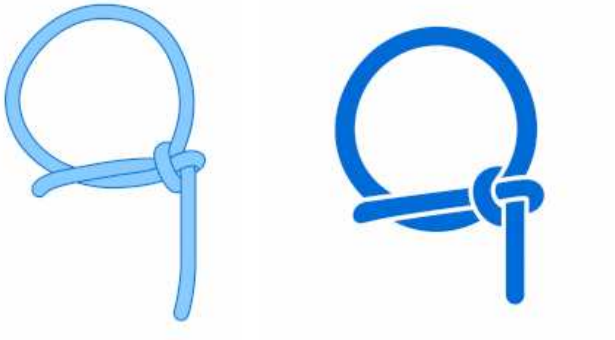
Claude Code を使う前

- 懐疑的



無料の ChatGPT

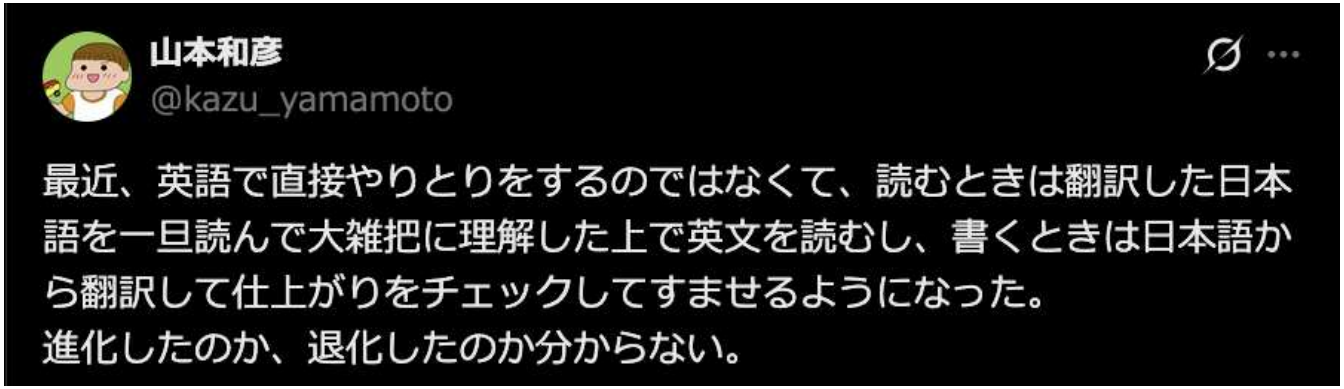
- もやい結び(bowline)のロゴを描かせる
  - 写真を与えた最初が一番ましで、ロゴ風に変換させると変になる
  - どう指示しても、結び目部分は思うように描けない
  - 今思うと：絵心と絵を描く技術がない人の指示では描けない



- Haskellのコードを書かせる
  - 大枠は合っているが、動かないコードができる
  - 今思うと：ローカルのコンパイラを使えないと精度が悪い

自動翻訳の活用

- これまで英語は直接読み書きしていた
  - RFC を読む
  - github issue に返事を書く
- 自動翻訳を使うようになった
  - 翻訳の品質は、それなりによいが、100点ではない

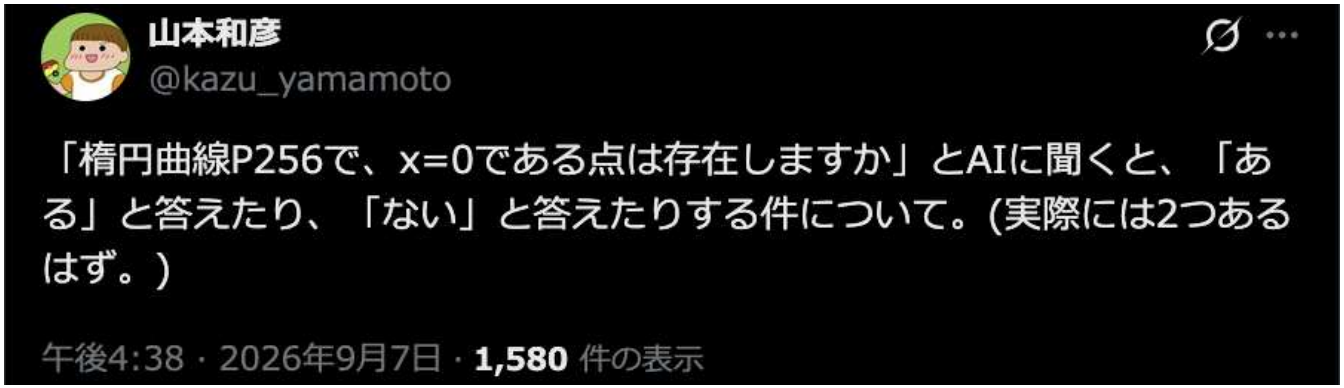


  - RFCは文章に改行が入るので、多くの自動翻訳サービスが苦手
  - ブラウザから使う Google Translate サイトだけが改行を気にしない

Google Gemini の活用

- 2026初頭からDNS権威サーバcloveの開発を開始
- DNSのRFCはカオス
  - 数が多過ぎ
  - 文章の質が低い
  - パッチを貼っていくスタイル
- DNS の仕様を Google 検索で問う
  - 「DNSのauthority sectionに入る可能性のあるRRは何ですか？」
  - 「○○○を定めているのは、どのRFCですか？」
- 無料GeminiはDNSに関しては正確に答える
- DNSSECに関しては間違うこともある
  - 同じ質問をしても真逆の答え

真逆の答え


山本和彦
@kazu_yamamoto
「楕円曲線P256で、x=0である点は存在しますか」とAIに聞くと、「ある」と答えたり、「ない」と答えたりする件について。(実際には2つあるはず。)
午後4:38 · 2026年9月7日 · 1,580 件の表示

AIが脆弱性を見つける時代(1)

- tlsライブラリの脆弱性
  - 脆弱性に関するメールを3つもらう
  - 重箱の隅をつついたようなバグ
  - そんな状況はないでしょう？
  - メールで返答しても届かない、はぁ
  - メールに付いていた直し方は間違っていた、はぁ
- CERT からも連絡が来た
  - 「VINCE を使って」
  - 1週間かけて直し、VINCE で CERT に連絡
  - CERT「単にリリースしてね」
  - kazu「はぁ？」

AIが脆弱性を見つける時代(2)

- tls や暗号ライブラリ crypton にも PR が多数
  - 片っ端からレビュー
  - 返答には自動翻訳を使う
  - 理解していることが分かるように、査読のように書く
  - なる早でマージしてリリース

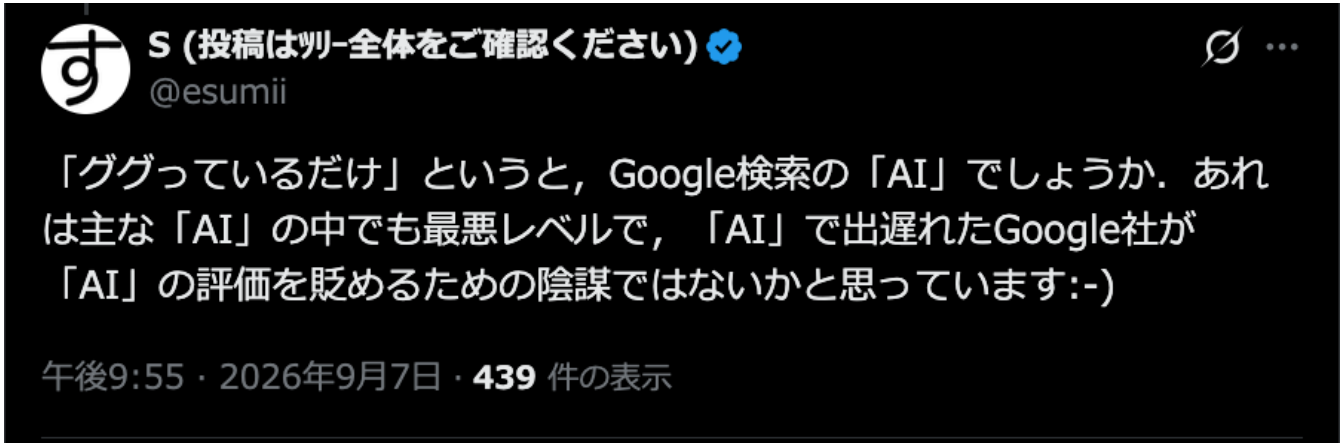
圧倒的な力量不足

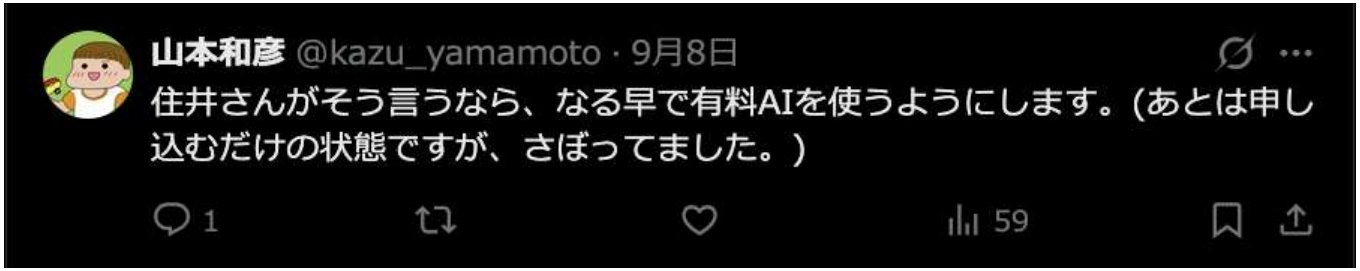
こちらもAIを使い
事前に直しておくしかない

AIの選定

- Claude Security という機能が良さそう
  - Claude はシャノンのファーストネームらしい
- Claude Enterprise
  - 従量課金
  - Claude Security あり
- Claude Team Premium
  - 固定料金：5時間、1週間の上限あり
  - 5人から、$125/人
  - Claude Security なし
- 一旦保留
  - 予算は確保した
  - Claude TeamでClaude Scurityが使えるようになるまで待つ

後押し





Claude Security を待たず 9/15から利用開始

ブラウザから Claude Team

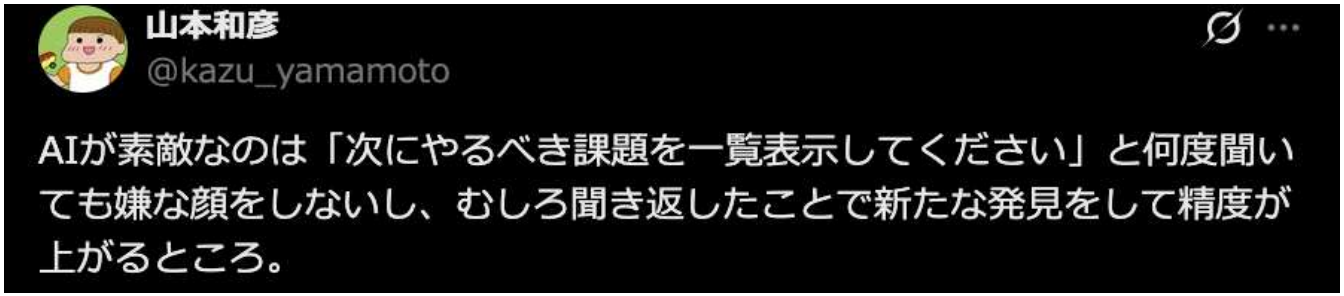
- Claude にはパスワード認証はない
  - 登録したメールアドレスに届くURLを読めるかで認証
- 何もしなければ、Opus 5 が AI のモデルとして使用される
- AI のモデル：Haiku < Sonnet < Opus < Fable
  - Opusの複数形はOpera
- Haskellの問題を解かせると、よいコードを書いた
- 使えそうだが、コピペするのは面倒

Claude Code へ移行

- Claude Code をインストール
  - 端末アプリで使うツール
  - `% claude`
- 最低限覚えるべきコマンド
  - 対話の行頭を "/" で始めるとコマンド
  - 終了 `/quit`
  - 再開 `/resume`

clove

- よく分かっているDNS権威サーバcloveから始めた
  - 「dnsext-bowline/clove/ を読んで clove の問題点を指摘してください」
  - たくさんの指摘、ひー
  - 「main からブランチを切って、問題1から解決してください」
  - 「PRを出してください」
  - レビューしてマージ
  - 「もう一度、課題の一覧を表示してください」

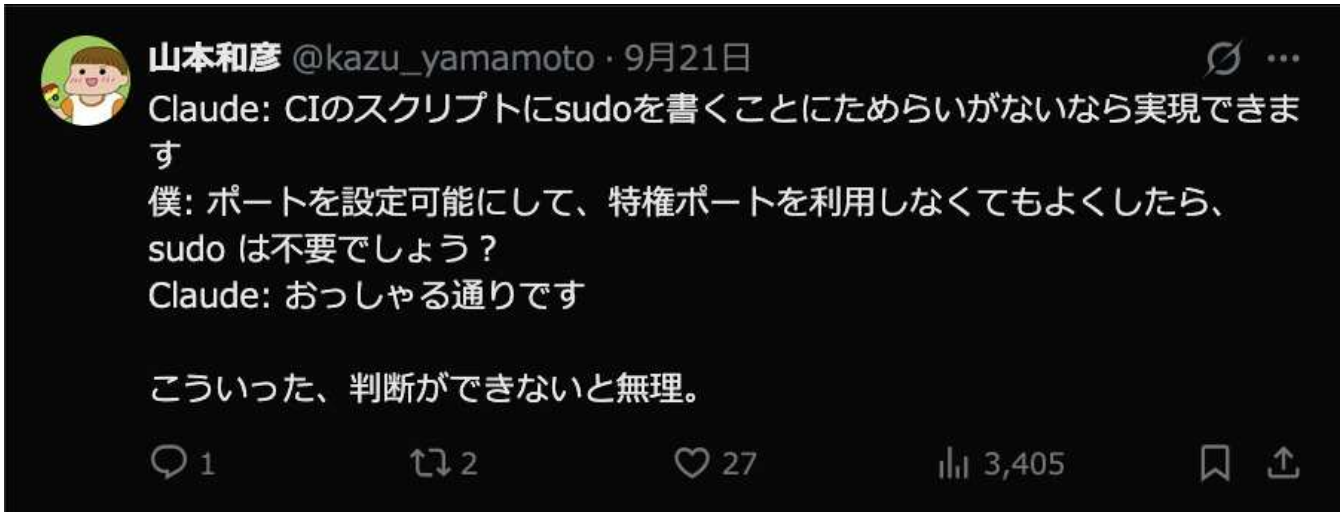


権限を与えろ

- PR を作らせると失敗する
  - Claude「自動化するにはトークンを作成して権限を与えてください」
- github トークン
  - Settings -> Developer Settings -> Fine-grained tokens
  - Repository access にリポジトリを追加
  - Claude Code に言われた権限
    - Actions: Read and write
    - Contents: Read-only
    - Issues: Read and write
    - Metadata: Read-only
    - Pull requests: Read and write
- 権限のない上流には？
  - タイトルと本文をURLパラメータに埋めたPR作成のURL
  - ファイルでもらって、コピペしてもいい
    - macOS： `% pbcopy < ファイル`

実装のコストは安い(1)

- 何を選択するかが大事



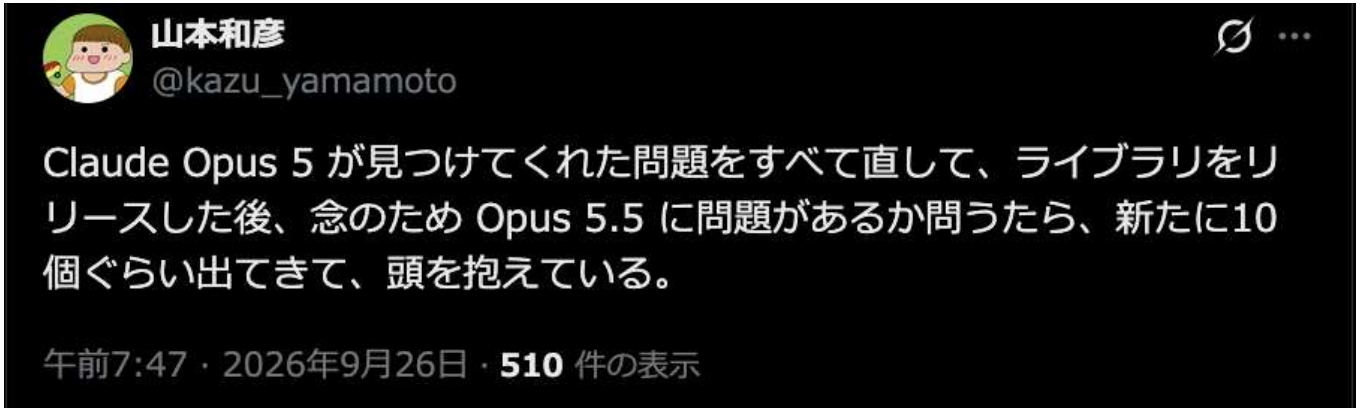
実装のコストは安い(2)

- アイディアが大事



Claude Opus

- Claude Codeは、起動時の最新のエンジンを使う
  - `claude` コマンドを起動し直すと Opusは 5 から 5.5 へ



bowlineの謎のバグ

- DNSフルリゾルバ bowline
- 長い間解決できていない稀に起こる2つのバグ
  - ソケットの `close` 後にワーカー・スレッドが無反応になる
  - CPUの消費が激しくなりモニターも反応しなくなる
- 日比野さんとの議論
  - GHC (Glasgow Haskell Compiler) のランタイムの問題では？
  - IO マネージャにバグがあるのでは？
  - IO マネージャに不適切なコールバックを登録すると `close` で永遠にブロックするのでは？
    - PoC は簡単に実装できた
  - しかし、不適切なコールバックを登録しているコードはない

bowline の謎を解明すべく
我々は AI と共に
GHC の奥地へと向かった

GHC

- IOマネージャ
  - 軽量スレッドの入出力を束ねる軽量スレッド
  - イベント駆動のカーネルの上で、
    スレッド・プログラミングを可能にする
  - バックエンドは、epoll や kqueue
  - kqueue 部分は僕が実装した
  - macOS で OneShot 機能が無効化されている積み残しあり
- 尻込み
  - GHC のビルドが長く、エラー＆トライが面倒そう
  - gitlab の細かい仕組みが分からない
    - MR(merge request) って、PR(pull request) と何が違うの？
  - GHC 開発者の慣習が分からない
    - どこでも bisect できるように、失敗するテストは disable しておく
- Claude Codeによる必死の探索
  - ノー・ヒントで調べさせる、PoCを見せて調べさせる
  - ビルドの長さを気にせずエラー＆トライ
  - 見つけた複数のバグを慣習を守って MR

ネットワーク関連のライブラリ

- network-run、http2、tls、quic、http3
- 1巡目：深刻な問題を直してリリース
- 2巡目：軽微な問題を直してリリース
- 3巡目：← イマココ
- AIが脆弱性を見付ける時代にCVDは必要か？
  - Coordinated Vulnerability Disclosure
  - すごく深刻でなければ、気にせず github に PR を出して直した

並列実行

- Claude Opus は、待ち時間が長い
- 3つのプロジェクトを3並列で回す
  - 仮想デスクトップ1：普段の生活用
  - 仮想デスクトップ2：プロジェクト1
  - 仮想デスクトップ3：プロジェクト2
  - 仮想デスクトップ4：プロジェクト3
- github CI のチューニング
  - 複数のプロジェクトが、CIのマシン・インスタンスを取り合う
  - ビルドのキャッシュが正しく効くようにする
  - 不必要なOSとGHCのバージョンの組みを落とす
- それでも待たされるときは雑用をする
- 僕の使い方でのトークンの上限
  - 5時間制限にはかからない
  - 1週間制限にはかかる

Claude Code インスタンス間の通信

- 適切なインスタンスに状況を説明して任せる
  - tlsとcryptonを同時進行
  - kazu「そのディレクトリは別のインスタンスが触ってます」
  - Claude「なら、そのインスタンスに作業を依頼しましょうか？」
- インスタンス名を変えて指定しやすく
  - `/rename` 名前
  - kazu「tlsインスタンスに仕事を依頼してください」

フレークを許すな

- フレーク
  - コードやテストを変えてないのに、確率的にテストが失敗すること
- Claude Code に追跡させる
  - 「テストにログを追加しました」
  - 封筒の裏：「N回に1回の確率で起き、1回のテストはM秒なので T分で再現すると思われます」
- 僕の場合は、だいたいはテストの誤りだった
  - テスト・サーバが何度もポートを再利用するため
- 驚くべき発見
  - quic では、race 関数が遅過ぎ
    - race の後では CONNECTION_CLOSE の送信が間に合わない
  - quic では、稀に輻輳制御のウインドウが開かない
- h3spec
  - QUIC、HTTP/3 のエラー・ケースをテストするツール
  - 確率的に失敗していたのが安定した
    - NAT のせいじゃなかった

tlsfuzzer

- fuzzing とは言えない TLS のテストツール
  - tls ライブラリの回帰テストとして利用
- 両方に専用ブランチを作っていた
  - tlsfuzzer 側
    - テストが期待するエラー・コードを変更したコミット
  - tls 側
    - tlsfuzzer が期待する弱い暗号を復活させたコミット
- 問題
  - main が育ったら専用ブランチをrebaseしないといけない
  - 回帰テストの際に専用ブランチに移動しないといけない
- Claude Code を使い専用ブランチをなくした
  - 要望により復活さていた CBC モジュールを利用
  - tls-server に --use-weak-ciphers オプションを定義
  - tlsfuzzer の期待するエラー・コードを返すように tls を改良
- カバー率の向上
  - 利用してなかったテスト・スクリプトを再調査 (135/168)
  - テスト・スクリプトに与えていた例外事項を再調査

暗号ライブラリcrypton

- 不安と不満
  - 脆弱性があるかもしれない
  - OpenSSL に比べて(たぶん)遅い
- Claude Code を使って改良
  - あらゆるアルゴリズムを定数時間に
  - あらゆるハードウェア・アクセラレータを活用
  - arm64 と x86_64 はアセンブリ実装へ
  - それ以外は、定数時間でなるべく高速なC実装へ
- ライセンス問題
  - crypton は BSD
  - ISC、MIT はOK
  - Apache は、GPL 2 を切り捨てるので NG
- 数え切れないほどのエラー＆トライ
  - 星の数ほどのアイディア
  - 嫌になるほどのアセンブリ実装の探索
  - haskell-cafe ML での協力依頼

環境を与えろ

- 普段は macOS / arm64 で開発
- SSHでログインできるOS
  - Ubuntu / UTM / arm64
  - Windows 11 / UTM / arm64
    - UTM だと Windows 11 Home もインストールできる
    - 設定は Claude Code に言われるがまま
  - Rocky Linux / Vmware / x86_64
- コマンド
  - GHC
  - gsed
    - Claude Code 独自の編集機能を使えるようすれば不要かも
  - valgrind

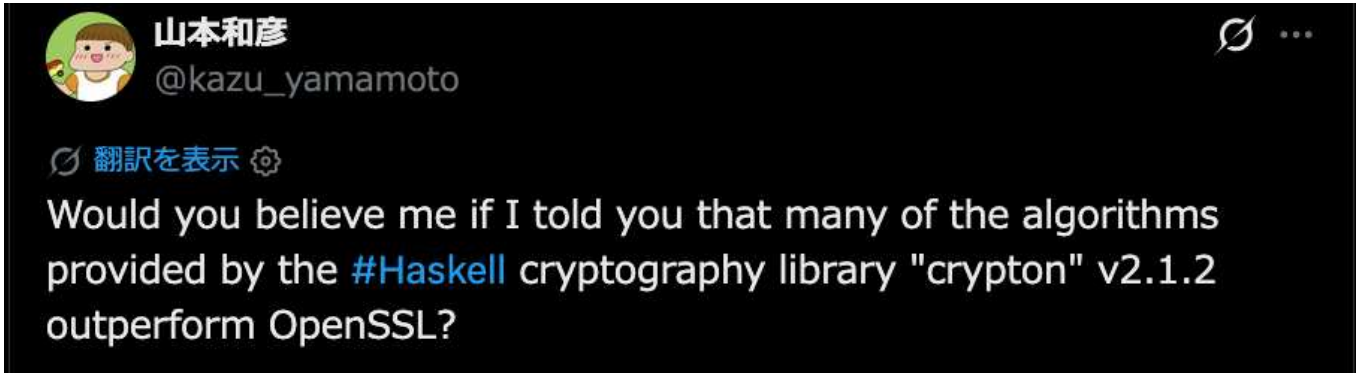
ラウンドテストを回せ

理想を言えば、指摘される最も重大な問題が
「ドキュメントの誤字」程度になるまで
複数回に渡ってレビューすべきです

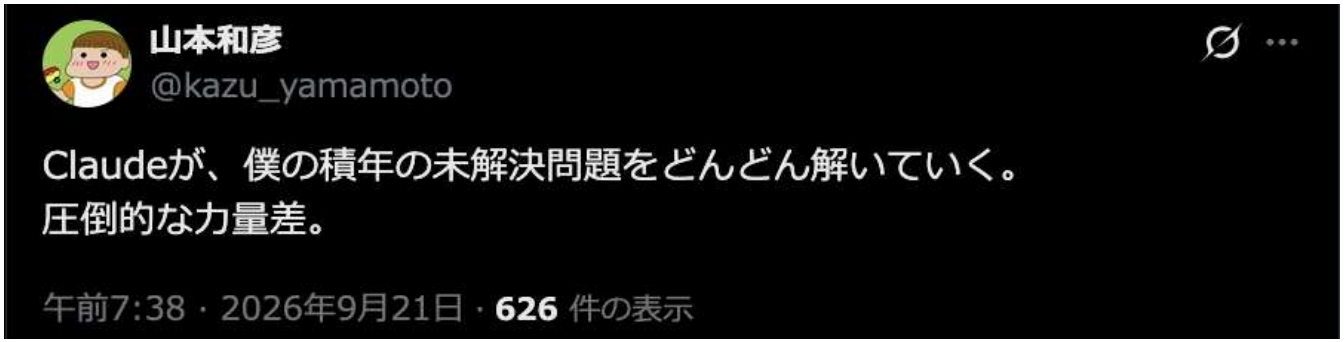
- 10回のラウンドテストを実施
  1: HaskellとCでの Int の取り扱いの違いで上位ビットを落としてないか
  2: C言語で配列を壊していないか
  3: アドレス sanitizer が何か見つけないか
  4: (警告は出ない)不要な import はないか
  5: マルチスレッドでうまく動くか
  6: cabalファイルのすべてフラグに間違いがないか
  7: 32bitで動くか
  8: 実行時間から秘密を読み取られないか
  9: ビッグエンディアンで動くか
  10: 秘密が実行後のメモリーから読み取られないか

計測

- AIに測らせろ、もう自分で測るな
  - マシンが静かになるまで待つ
  - ウォーミングアップ
  - 外れ値の排除
  - 前回の結果との差の追求
  - どれも人間が不得意で、AI が得意



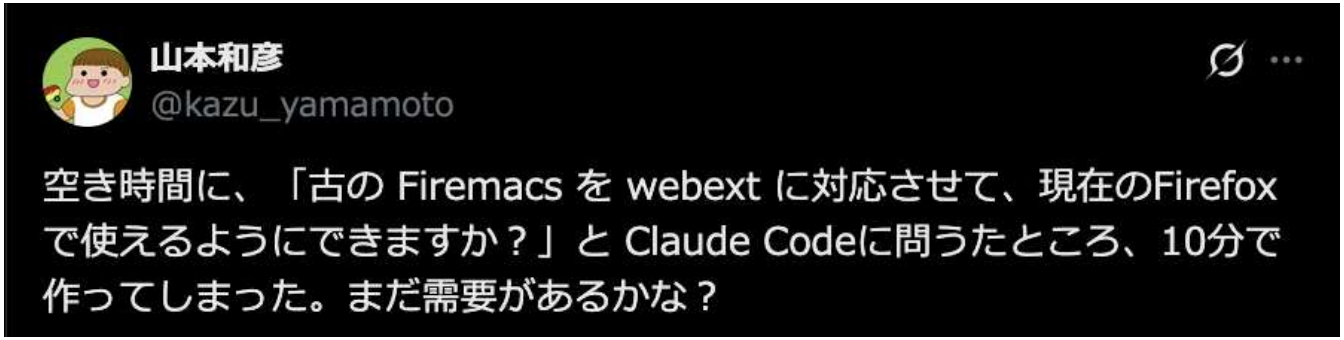
積み残し



宿題やアイディアがあるなら
今すぐやれ

Firemacs

- Firefox 用の Emacs キーバインディング
- Firefox 2.0 で Firemacs が登場
  - 拡張は XUL ベース
- Firefox 57 で Firemacs が退場
  - 拡張は WebExtensions API へ完全移行
    - Firefox を再起動しなくても、拡張をリロードできる
  - Firemacs の WebExtensions API 対応は
    よく分からなかったので諦めた



- 登録の方法も全部教えてもらった

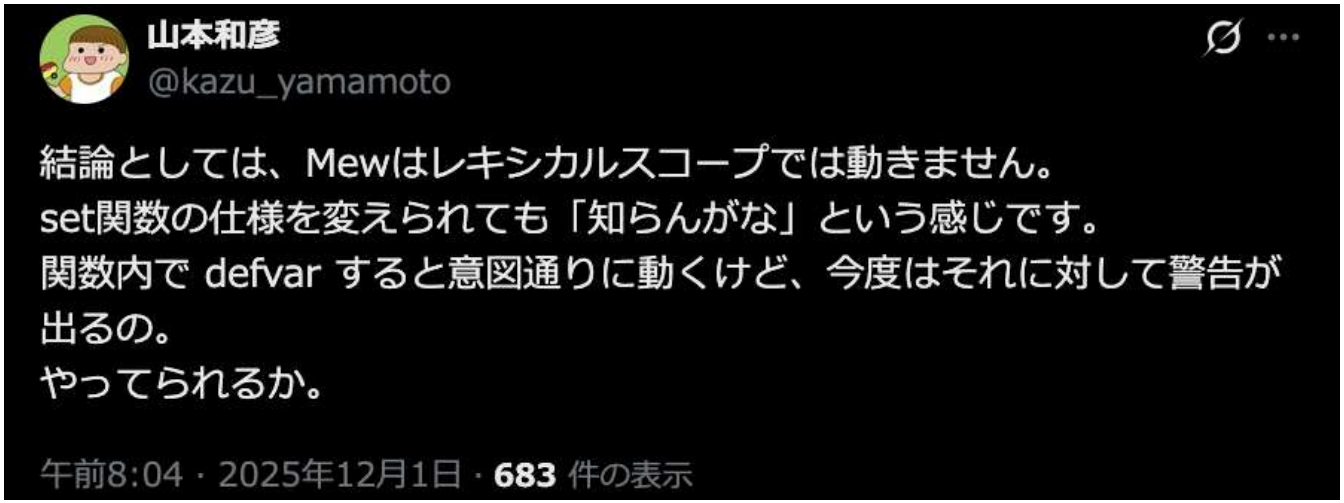
Mew の課題

- Emacs は元々動的スコープ
- Emacs 24からレキシカル・スコープが導入された
  - デフォルトは動的スコープ
  - 開発者達は、レキシカル・スコープに移行したい
- Emacs 31から警告
  - 動的スコープを警告なしで使うには、ファイルの先頭に以下が必要

```
;;; -*- lexical-binding: nil; -*-
```

- Mewは `set` 関数を用いて動的に変数を作る
  - 関数内で変数宣言 `defvar` するとレキシカル・スコープにできそう
  - しかし、「関数内で `defvar` するな」という警告が出る

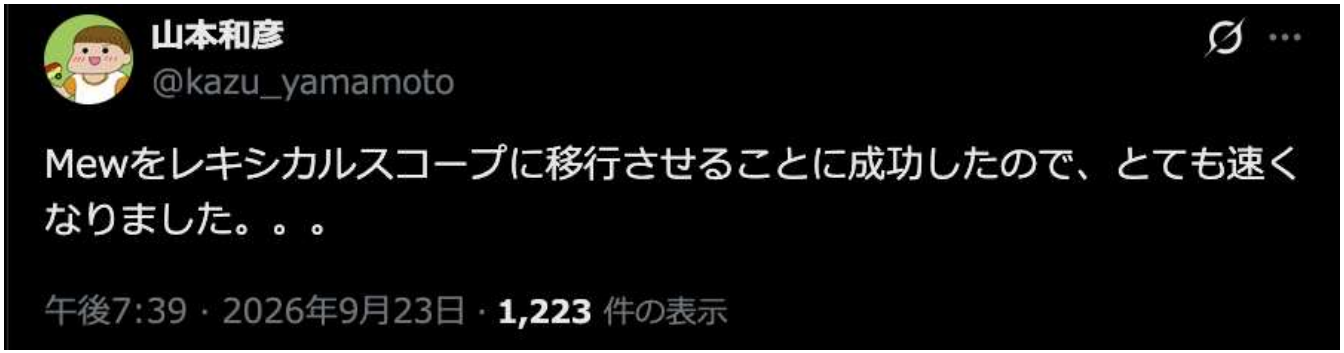
Claude Code からの共感



つまり 72 個の局所変数（file2 msg errmsg key case beg end …）を全部 mew- 付きの
グローバル特殊変数に改名しろ、という要求です。
それは確かに「やってられるか」です。

Mewの成果

ただ、その道を通らなくて済みます
行き止まりは「変数を special にする」
方向から来ています。
逆向き、「mew-set の側をマクロにする」と、
変数は普通のレキシカル局所変数の
ままで済みます。



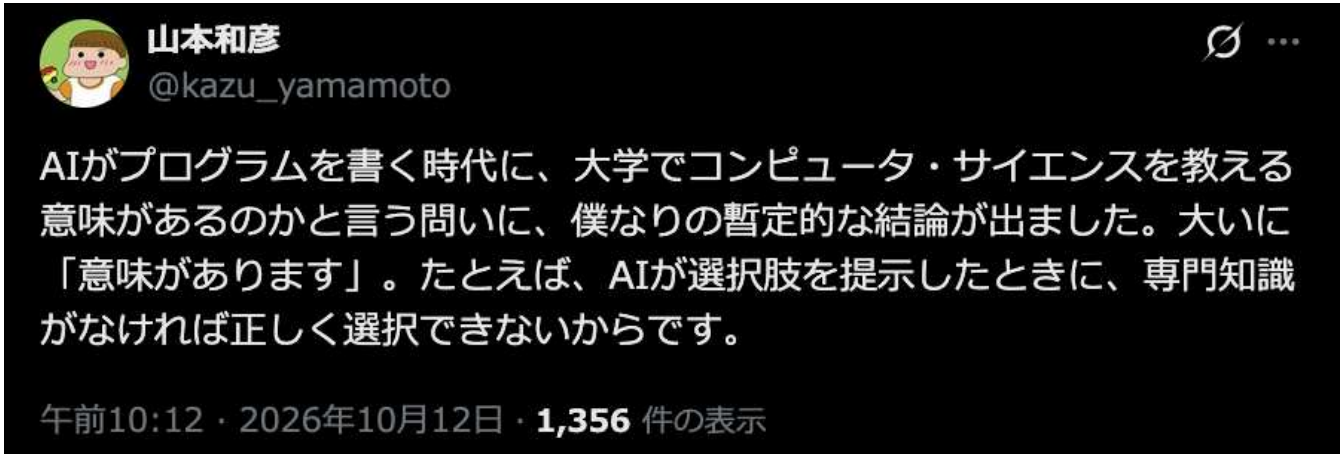
pgpdump

- PGPのデータ構造のビジュアライザ
  - gnupg の可視化機能が不満だったので実装
- 実装済みだった OpenPGP
  - RFC 2440 (1998年)
  - RFC 4880 (2007年)
- RFC 9580 (2024年)
  - gnupg は追従してない？
  - Sequoia-PGP の sq という代替品がある
  - 少しサポートしてissueを登録した人に見せたが無反応なので放置
- Claude Code に頼むと1時間で対応
  - 溜まっていた issue もほぼ解決
- RFC 9980 (2026年)
  - OpenPGP用の耐量子暗号

急いては事を仕損じる

- 初期にやった bowline (dnsext)
  - 「issue / PR は日本語でやりましょう」
    → Claude Code で全部日本語へ
  - 「説明は簡潔に」「再現条件を明確に」
    → Claude Code で説明の全部を書き直し
  - 「コード内のコメントは3行以内に」
    → Claude Code でコミット全部を作り直し
- 初期にやったGHC
  - 「MRの説明がLLMちっくで冗長」
    → 自分の言葉で説明を書き直し
  - 「コミットの説明が冗長。issueの説明を繰り返す必要はない」
    → 全部作り成す予定 ← イマココ

暫定的な結論


山本和彦
@kazu_yamamoto
AIがプログラムを書く時代に、大学でコンピュータ・サイエンスを教える意味があるのかと言う問いに、僕なりの暫定的な結論が出ました。大いに「意味があります」。たとえば、AIが選択肢を提示したときに、専門知識がなければ正しく選択できないからです。
午前10:12 · 2026年10月12日 · 1,356 件の表示

AI 時代に何を身につけるべきか？

- 1に国語、2に国語、3に国語
  - AI に的確に指示する能力
  - AI が何を言っているのかを理解する能力
  - 英語も大事だけど、自動翻訳でなんとかなりそう
- 論理的な思考
- 専門領域の知識