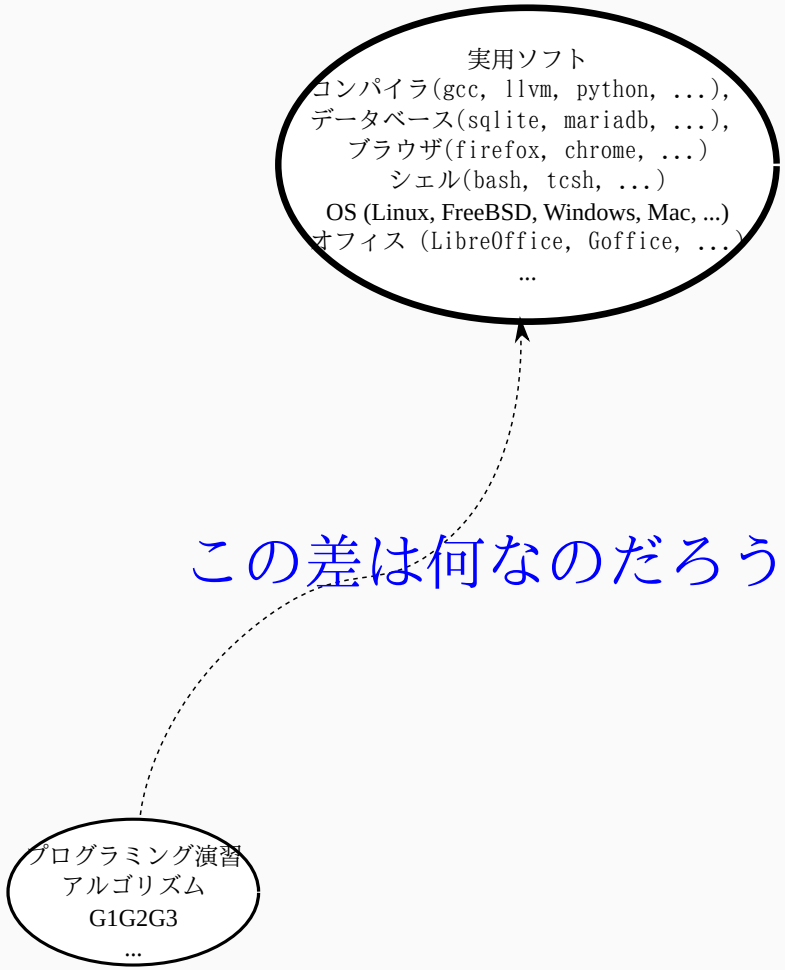
大規模ソフトウェアを手探る

田浦 健次朗
高橋淳一郎 (田浦研 M1), 瀧川雄理 (田浦研 B4)

2025/10/07

この課題の目標

- 「演習レベルの小さなプログラムが作れること」と,「実用規模のプログラムが作れること」のギャップを埋める(ための知識と経験を得る)
- ささやかでも, ソフトをどう改良するかについて, アイデアを出しあう(what と how)



ぜひマスターしてほしいスキル

- 全容がよくわからないソフトウェアの概要を把握し, 拡張・変更できるようになる
- ← そのために, ソースファイル中で修正が必要な場所を突き止める
- ← そのためのツール
  - 汎用的な文字列検索(grep など)
  - クロスリファレンスツール(globals など)
  - デバッガ ← 個人的にはこれが最重要と考えている
  - プロファイラ, トレーサ (uftrace)

代表的なプログラミング言語とビルドの流儀

- C/C++
  - configure; make; make install
  - cmake; make; make install
- Python
  - pip
- JavaScript (Node.js)
  - npm
- Rust, Go, …
- ソースコードのダウンロード → ビルド → デバッガなどで追跡 → 修正 → 反映 の流れをマスター

これを身につけることの意義

- ソフトウェアの研究では, 成果をソフトウェアとして発信することが重要
- しかし大きなソフトウェアを一から作ることは, 困難な場合が多い
- → 既存のソフトウェアの拡張として作ることが多い
- 一から作る場合でも, 成果を実用的なソフトウェアとして発信する際, 常識的なお作法を守っておくことは重要

課題期間全体のロードマップ (目安)

- 1 日目 : ソフトのインストール, デバッガで動作追跡手法を練習
- 2-3 日目 : チームを作る, 手探るソフト, 目標議論スタート
- 3-4 日目 : 試しになんでもいいから変更してみる
- 4-9 日目 : 目標を決め, 実行
- 10 日目 : 最終発表

日々行うこと

- 議論
- 作業
- その記録 (進捗記録)
  - 他の人, あるいは後の自分が一から再現できるような情報を記録
  - トラブルも進捗
    - 他の人が再現できる情報
    - なるべく小さな例で (Minimum Working Example)
- 進捗発表 (チーム結成以降, 各回終了時, 持ち回りで何班かずつ)
- 他の班の進捗発表を見て, 質問, コメント, 互助

本日の以降の作業

- 新テキスト を眺める
- そこにある「コードのダウンロード, ビルド, デバッグの練習」をやってみる
  - C/C++ 編
  - Python 編
  - JavaScript 編

AI 利用について

- プロ (またはその卵) として生産性向上のために使うことは重要
  - プログラムを見ていて文法的に謎な式を見たときに解説させる
  - 謎なエラーに遭遇したときに原因を推測させる
  - コードを断片ではなくどっさり与えて解説させることも可能
  - コード補完 AI (github copilot など), コーディング AI (Claude Code など) では作業の大部分をやらせることも可能であろう
- プログラムを書く技術をどのような気持ちで学べばいいのか?

精神論

- AI はどんな一人の人間が持ちうる知識よりも多くの知識を持っているので, 「自分にできないが AI がすぐにやってしまうこと」がたくさんあるのは当たり前
- やがて皆さんが取り組む問題には, AI にはできないことが必ず出てくる → 勉強はその時のため
- この演習で AI にできない (e.g., ハルシネーションの無限ループ) に遭遇するかもしれないし, しないかもしれないが, その時に備えて, 自分が何を学びたいのかを考え続けることが重要
- → 「何かができた」ことに大した意味はなく, (AI に教えたもらった場合でも) 知見を 自分の脳に刻む ことが重要