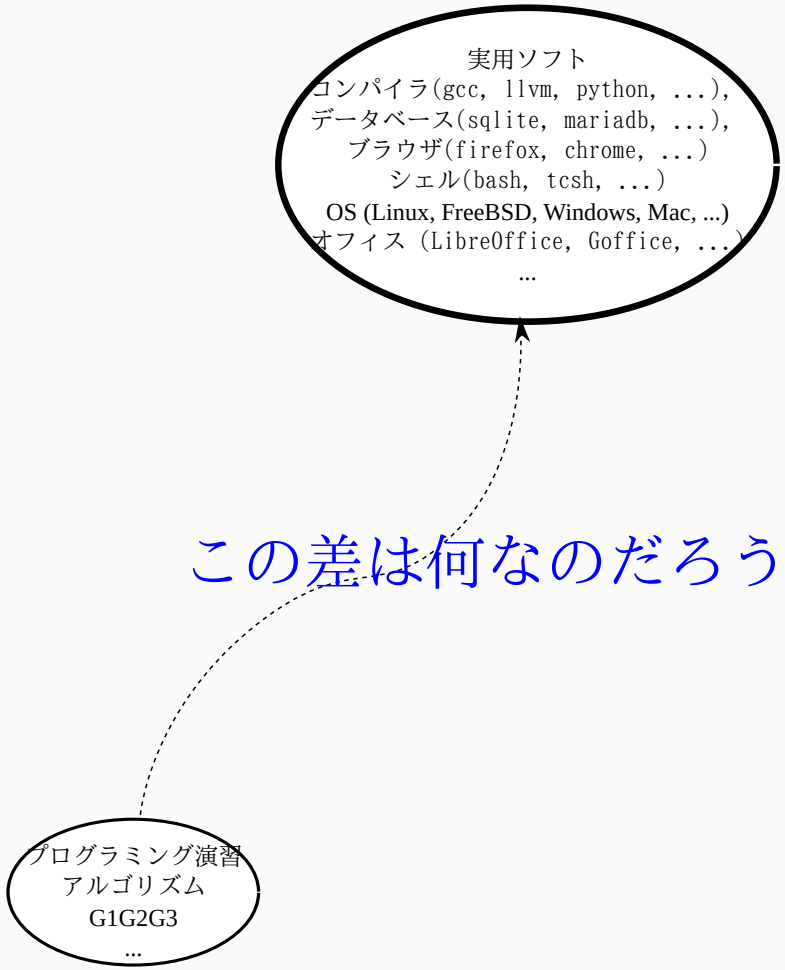
大規模ソフトウェアを手探る

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)



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(gloals など)
  - デバッガ (gdb, lldb)
  - 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

道中で学ぶ「基礎知識」

- 多数のファイルからなるソフトはどう書くのか (ファイルをまたがったシンボル参照の基本)
- 実用ソフトで常識的なソースの書きかたあれこれ
  - コマンドライン引数，設定ファイルなど，プログラムを汎用的にするための書き方
  - C/C++の `#ifdef` など，単一ソースを何通りにもコンパイル(条件付きコンパイル)するための処理
  - その他修正がしやすいように，大きなプログラムがよくやっている技法

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

- デバッガでのソフトウェアの動作追跡手法を練習(本日)
- チームを作る, 手探るソフト, 目標を議論
- 試しに変更してみる
- 目標を決め, 実行
- 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)