オープン化が進むC++の現状と展望

高橋 晶 (Akira Takahashi)
cpp_akira@Twitter
faithandbrave@GitHub

2021/08/25 (水) OSS X Users Meeting #31

自己紹介

- 高橋 晶 (Akira Takahashi)
- C++日本語リファレンスサイトcpprefjpのコアメンバ
- C++の勉強会C++ MIXの運営
- 著書
  - 『C++テンプレートテクニック』
  - 『C++ポケットリファレンス』
  - 『プログラミングの魔導書 Vol.1』

お仕事

- Preferred Networks (PFN)という会社で、深層学習用プロセッサMN-Coreを作っています
- 電力性能で何度か世界一位を獲得しました
- そのチームで私は、コンパイラ以下の低レイヤーなソフトウェアスタックを開発しています

本日のお話

- C++はオープンソースではなくISOで標準化されている言語です
- ですが、オープンソースの影響を受けて標準C++の策定作業が徐々にオープン化してきています
- この発表では、以下のようなことを話します
  - 標準C++のオープン化
  - オープンソースライブラリの作られ方・使われ方の変化
  - C++が活躍する場の変化

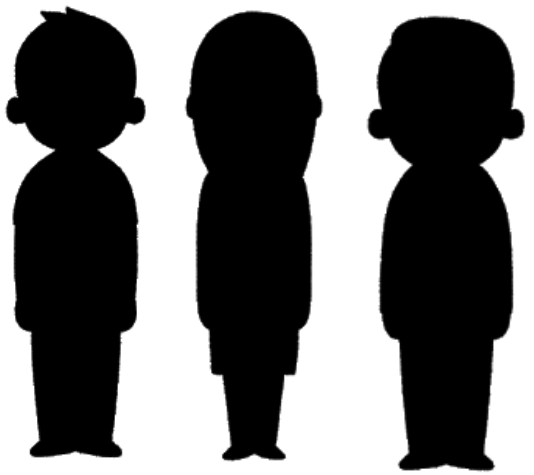
標準C++のオープン化

現在はC++20

- 1998年にC++98が策定されてから、次の規格は2011年のC++11と、13年の策定間隔があった
  - 2003年にC++03もあったが非常に小さなアップデート
- その後は3年間隔での定期アップデートが行われている
- C++11以降では、策定体制の変化、言語進化の高速化、それによる外部ライブラリの事情の変化などがある

C++11策定までのクローズドな標準化

- 一昔前までのC++は、標準化のメーリングリストがクローズドで、標準化委員会のメンバーしかアクセスできなかった
- 定期的に行われる国際標準化委員会も、もちろん委員会のメンバーしか参加できない
- 一般のC++ユーザーは議論に参加することはできず、決定したものを追いかけるしかなかった



謎のC++標準化委員会

C++14以降での変化

- C++11には、オープンソースライブラリであるBoostから多くの機能が導入された
- それを受け、標準化委員会はオープンソースの開発者に、標準化への参加を依頼し始めた
- しかし、オープンソースの開発者たちはクローズドなものに興味が薄く、参加に消極的だった
- それもあり、C++の標準化を議論する**メーリングリストがオープン**になり、だれでも議論に参加できるようになった

https://isocpp.org/

標準C++でオープンになったもの

- メーリングリスト
  - 新機能の提案 (std-proposals)
  - より広いC++の議論 (std-discussion)
  - 分野ごとの専門グループ (study group)
- 規格案
  - https://github.com/cplusplus/draft
  - 編集ミス程度なら、ここからPull Requestを送れる
- 各提案の進捗状況
  - https://github.com/cplusplus/papers
  - ここのissueで確認できる

標準C++でオープンになっていないもの

- 国際標準化委員会への参加
  - 標準化に参加している企業、および国の代表に限られている
- 新機能の提案には国際標準化委員会でのプレゼンテーションが必要
  - だが、オープンなメーリングリストで提案して、委員会メンバーが提案を引き継ぐ、という形式で広く提案が行われるようになった
- 国際標準化委員会での完全な議事録
  - 機能の経緯を完全に追いきれないことは、まだある

オープンになっていないものはまだまだあるが…

- 標準C++策定のオープン化によって、ハンドリングしきれないほど**多くの新しい提案が行われる**ようになった
  - 多すぎて日本語情報の発信もとてもたいへんです
- 元々、言語の議論グループとライブラリの議論グループくらいしかなかったのが、20を超える専門グループごとに議論が同時並行に行われている
- オープン化によって**言語の進化は大幅に加速**していると言える

C++オープンソースの変化

C++とオープンソース

- C++は標準ライブラリによって環境差を吸収しているが、すべての環境差を吸収できているわけではない
  - ある：I/O、スレッド、ファイルシステム、スタックトレース
  - ない：ネットワーク、オーディオ、グラフィクス、プロセス、GPUなど
- 標準ライブラリで扱いきれない範囲はオープンソースに頼ることになる
- 定番と言えるライブラリもあるが (Boost、ICU、OpenCVなど)、言語の進化にともなってライブラリの事情も変わってきている

Visual Studio標準ライブラリ実装のオープンソース化

- GCCやClangといったコンパイラはもともとオープンソース
- それらに加えて、Visual StudioでのC++標準ライブラリの実装もオープンソース化された
  - https://github.com/microsoft/STL

Boostの衰退

- 準標準ライブラリという立ち位置だったBoostだが、徐々に衰退している
- 現在のバージョンは1.77.0
  - Boost 2.0の議論が進まないまま、バージョン1系で77までずるずるいっている
- Boost設立メンバーの離脱、コンサルティング会社BoostProの解散
  - リーダー不在で、重要な決定ができない状態
- Boost内のライブラリを個別に扱える仕組みの頓挫
  - Boostは大きくなり続け、ユーザーは巨大ライブラリのダウンロードを避けるようになった
- ただ、分野によってBoostのライブラリがまだまだ最新なことはある
  - 古いC++バージョンで最新の標準ライブラリの機能を使う、という用途でも使える

個人オープンソースへの移行 1/3

- ボランティア組織の大きなライブラリから、
  個人の軽量なオープンソースライブラリへ、ユーザーは移行してきている
  - Boostが衰退していることも影響

個人オープンソースへの移行 **2/3**

- 文字列フォーマットライブラリ{fmt}も**個人のオープンソースだが、広くユーザーを獲得**し、C++20で標準化されている

```
cout << format("{} {}", "Hello", 314) << endl; // "Hello 314"
```

  - ちなみに、このライブラリが作れるようになった背景に、可変引数テンプレート、汎用定数式constexprなど言語の新機能が入ったおかげ、というのがある
  - フォーマット文字列のコンパイル時検証もある

個人オープンソースへの移行 3/3

- 広く使われているテストライブラリCatch2、
  ロギングライブラリspdlogなども個人リポジトリ

```
TEST_CASE("test name") {
  REQUIRE(a == b);
}
```

```
spdlog::info("information log");
spdlog::error("error log");
```

- どのライブラリを使えばいいか悩ましい時期だが、
  Awesome C++やStackoverflowを見て自分の用途にあるものを
  選ぶことになるだろう

言語更新によるライブラリ設計への影響

- 標準C++が3年スパンで更新されることもあり、**最新のライブラリ設計というものが変わりやすい**
- 例として、C++11でラムダ式 (無名関数) が導入されたことにより、その後のライブラリ設計が大きく変わった

```
setTimer(10s, [] { cout << "10秒経過" << endl; });
```

- C++20でもコルーチンが導入されたことにより、今後のライブラリ設計に大きな影響があるだろう

パッケージマネージャは？

- Conanというのがあるが、それほど流行っていない
- CMakeの機能で依存ライブラリをダウンロード・インストールする人が、ちらほらいるくらい
- パッケージマネージャがあると、ダウンロード数などで人気ライブラリの可視化がされやすいが、いまは人気ライブラリがわかりにくい
- まだまだ発展途上

オープンソース関係の今後の課題

- 変わりやすい最新のライブラリとその設計の知見をいかに共有していくか
- 母国語情報をどう増やすか
- ボランティアの開発者に継続開発してもらうためのインセンティブ
  - オープンソースでは開発者の失踪もよくある
  - GitHub Sponsorsがもっと流行るといいですね

C++が活躍する場の変化

C++の仕事がなくなった

- という話をよく聞くようになった
- これは、C++が活躍する場が変化したことによる
  - GUIやゲームはC#へ
  - Web需要増によるJavaScriptや、サーバー処理が得意な言語へ
- **領域特化した言語がより使われるのは正当進歩だと思う**
  - C++が真に必要になる領域とはなにか

C++が活躍する場はどうなったか 1/3

- 高速化のためのバックエンド
  - フロントエンドには使いやすい (スクリプト) 言語を使い、バックエンドにC++がいる、という開発が増えている
- C#のバックエンドにC++
  - ゲーム開発では、Unity 3DもIL2CPPでC#からC++に変換してコンパイルしている
  - Unreal Engineでのゲーム開発も、基本はBluePrintというスクリプトを使用し、必要なところでC++を使用する
- Pythonのバックエンドに C++
  - 機械学習が盛んなPythonでも、高速化が必要なライブラリはC++で実装され、Pythonインタフェースを介して使用している

C++が活躍する場はどうなったか 2/3

- 大規模計算・シミュレーション
  - ムーアの法則は実はまだ細々と続いていて、高速化するコンピュータを極限まで使い切りたい要求はまだまだある
  - 天気予報が1週間先までだったのが、新世代のスパコンによって2週間先まで予測できるようになったニュースは、わかりやすいインパクトがあった
  - コンピュータ性能はどれだけあっても足りない

C++が活躍する場はどうなったか 3/3

- エッジデバイス上のアプリケーション
  - 車など電力要求の厳しい環境
  - 自動運転、スマートデバイス、ロボットなどエッジデバイスアプリケーションの需要は高まっている

C++が活躍する場にどう向き合うか

- 道具によって仕事を選ぶよりも、仕事によって道具を選んだほうが、よいのではないか
- C++を使う必要がなくなった仕事は、C++を使うよりも簡単に作れるようになったのだろう
- **特定の言語の固執せず**、いろいろな言語を学び、状況に応じて適切な言語を選択することを私は推奨する
- ただし、高速化や省メモリを要求される環境で、**C++はこれからも使われ続け**、その要求はコンピュータが果てしなく速くならない限り、なくならない

まとめ

- 標準C++はオープン化によって議論が加速した
- C++界隈のオープンソースライブラリは移ろいやすい状況になっており、個人ライブラリが使われることが増えた
- C++が求められる領域はたしかに減ったが、これからも必要とされる領域はある
- C++の進化が高速化することにともない、教育が追いつかなくなってきている。情報共有はさらに必要とされるだろう