第26回TOPPERS開発者会議開催レポート

2025年10月26日（日）にオンラインにて、第26回TOPPERS開発者会議を開催しました。TOPPERS開発者会議は、TOPPERSプロジェクトの開発成果物の開発者・利用者が集まり、合宿形式にて集中的に議論、開発する会議で、非会員も参加可能です。数年前に合宿形式を見送ってからはオンライン開催を行っていましたが、今年もオンライン開催になりました。今年は準備不足もあり、例年のゲストトークを行うことは出来ませんでしたが、非会員も含め活動的な方々に参加いただき、活発な議論が展開されました。本レポートでは、その様子をお伝えします。

議論1：Open SDV API 技術解説

・Open SDV API についての技術面の解説
高田 広章 様（名古屋大学，TOPPERSプロジェクト会長）

このセッションでは、高田会長より「OpenSDV（Open Software Defined Vehicle）」の取り組みについて紹介がありました。
SDV（Software Defined Vehicle）とは、ソフトウェアによって自動車の機能や価値を定義するという考え方を指し、近年自動車業界で注目を集めています。
ソフトウェアはハードウェアに比べて更新が容易であり、購入後に機能追加や改善を行うことが可能です。これにより、販売後もユーザーのニーズに応じた新しい機能を提供できるようになり、自動車の価値を長期的に高めることが期待されます。すでに海外では、こうしたソフトウェア更新型のサービスが実際に提供され始めており、今後日本国内でも同様の動きが進むと見込まれます。
また、ソフトウェアをハードウェアと独立して開発することで、共通のソフトウェアを異なる車種へ展開することが可能となり、開発工数やコストの削減につながる点も重要な利点です。
「OpenSDV」としてオープン化を推進する狙いは、サードパーティによる追加機能提供を可能にすることにあります。スマートフォンにアプリを追加するように、車にも新たな機能を安全に追加できる世界を目指しています。
ただし、自動車は人命に関わる製品であるため、スマートフォンのようなビジネスモデルをそのまま適用することは容易ではありません。安全性や法的リスクへの十分な配慮が求められます。この課題に

対応するため、OpenSDV Initiativeでは車両機能を標準化するための「ビークルAPI」の策定が進められています。

高田会長からは、ウィンドウ制御を例としたAPIの説明がありました。たとえば、ウィンドウを閉める際には挟み込み防止機構の有無が車種によって異なります。API上でその機能を定義することで、外部アプリケーションが安全な操作判断を行えるようにすることが目的です。たとえば洗車アプリを考えた場合、洗車前にウィンドウを閉める際に、実際に自動で閉めるのか、ユーザーに促すのかをAPI情報から判断できるようになります。

APIの策定では、まず物理的なプログラミングインターフェースではなく、論理的な機能や数値の定義を行う「論理API」を整備する段階にあります。ここでは、挟み込みリスクのような安全面を含めた定義も可能とすることが重視されています。

質疑応答では、法令や地域条例への対応、自動車をリビング化した際の新たなリスク定義など、幅広い観点から意見交換が行われました。
特に、自動車では歩行者や他者への影響など、ユーザーの自己責任の範囲を超えるリスクが存在するため、オープン化にあたっては慎重な検討が必要である点が共有されました。

OpenSDV Initiativeは、自動車の機能を共通化・標準化するだけでなく、ソフトウェアを通じて自動車の価値創出のあり方を再定義する試みとして位置づけられています。今後の動向が注目されます。

議論2：TECSの最新研究

・TECS/Rust における排他制御の最適化と新しいRTOSへの統合

吉村 凪生 様（埼玉大学）

吉村氏からは、TECSとRustを組み合わせた新しい開発手法について発表がありました。
TECSのコンポーネントベース開発と、Rustの持つメモリ安全の仕組みを融合させることで、ソフトウェアの大規模化・複雑化に対応しつつ、メモリ関連のバグを防ぐという、堅実かつ挑戦的な取り組みです。

Rustでアプリケーションを記述しても、基盤となるRTOSがC言語で書かれている場合、システム全体としてのメモリ安全は完全には保証されません。そこで吉村氏は、Rustで開発されたRTOS「Awkernel」を採用することで、システム全体としてのメモリ安全性を確保する環境を整えました。この点が、既存のTOPPERSカーネルを用いた構成との大きな違いです。

さらに、TECSがRustコードを生成する際に生じる「排他制御の過剰生成」という課題にも踏み込みました。複数タスクからアクセスされる変数には当然排他制御が必要ですが、単一タスクからしかアクセスされない変数にまで一律に排他処理を入れてしまうと、オーバーヘッドが増えてしまいます。吉村

氏は、TECSによるタスク定義からコールフローを解析し、排他制御が不要な箇所を自動的に判断する最適化手法を提案。これにより、不要な排他制御コードを生成しないよう改善を図りました。
　実行時間の計測では、大きなオーバーヘッドもなく、実行時間のばらつきも抑えられるという有望な結果が得られたとのことです。詳細な検証は現在進行中とのことでしたが、Rustの安全性とTECSの柔軟性を組み合わせたこのアプローチは、今後のRTOS開発に新しい可能性をもたらすものと感じました。

・Zig独自のビルドシステムを活用したASP3カーネルの実現

　越野 仁 様（名古屋大学）

　越野氏からは、Zig言語を用いたTOPPERS/ASP3の移植について、その最新の取り組みが紹介されました。
Zigは、近年注目を集めるシステムプログラミング言語で、Cと同等の低レベルな制御が可能でありながら、コンパイル時の安全性チェックや型システムの強化によって、堅牢なソフトウェア開発を支援します。発表ではまず、そのZig言語の特徴が丁寧に解説されました。
　Zigでは、コンパイル時に値を評価して実行時処理を削減できるほか、`null`値の有無を型として明示的に扱うことで、実行時の`null`参照エラーを防ぐことができます。また、構造体に関数を定義できるため、C++のクラスに似た表現も可能です。
さらに、ソースファイル間での外部参照の可否を定義できる点、変化する変数としない変数を明示して意図しない変更をコンパイル時に検出できる点、モジュールとしての分割管理が容易である点などが挙げられました。もちろん、C言語との相互運用性も確保されています。
　こうしたZigの強力な機能を活かして、TOPPERS/ASP3カーネルの移植作業が進められていますが、Zigのバージョンアップに伴って仕様が変更され、特にソースファイルの外部参照に関する制約が厳しくなったため、従来のMakefileベースのビルドから、Zigの独自ビルドツールへの移行が必要になったとのことです。
これに対応するため、上位ディレクトリを介したファイル依存関係の整理や、エイリアスを用いた参照方法の調整が進められています。また、C言語のヘッダーファイルのインクルードパス設定についても、Zigビルド環境に適合させる工夫がなされたとのことでした。
　Zigは今後も頻繁なバージョンアップが予定されており、それに追従しながらASP3カーネルを維持・発展させる作業は容易ではありません。それでも越野氏は、Zigの安全性と柔軟性を活かした新しいRTOS開発の可能性を感じさせる研究として、今後も継続的に改良を進めていく意欲を示されていました。

・TECS の非同期データ授受機構 DataPumpHolderPlugin

　大山 博司 様（オークマ株式会社）

　大山氏からは、TECSの既存の通信モデルを応用し、Pub/Sub通信のような非同期データ交換を柔軟に定義できる仕組みの設計について発表がありました。

現在のTECSでは、関数呼び出しの方向とデータ授受の方向が一致している場合に限り非同期化が実現されています。しかし、呼び出し方向とデータの流れが逆の場合には対応出来ていませんでした。そこで大山氏は、データの生成と消費の関係、同期／非同期、バッファの有無といった多様な条件を組み合わせ、最適なコード生成を行う仕組みの構築に取り組んでいます。

発表では、データ授受の形態を細かく分類し、それぞれに対応する実装パターンが紹介されました。まず、直接的なデータ授受の場合には、「データ生成セルが関数呼び出しで渡す」パターンと、「関数呼び出しで受け取る」パターンの2つがあります。
一方、中継セルを介した場合はさらに複雑で、生成側・消費側の双方に中継セルへ「関数を呼び出してデータを渡す」「関数が呼び出されてデータを渡す」の2種類があり、組み合わせた合計4パターンが想定されています。

これらの多様なパターンに対応するために、大山氏は「DataPump」と「DataHolder」という2種類の型を用意し、それらを組み合わせて柔軟な非同期通信を実現する方式を提案しました。DataHolderでは同期型と非同期型の両方をサポートし、さまざまな動作検証を行うテストパターンも準備中とのことです。

さらに興味深い話題として、TECSの今後の方向性にも触れられました。これまでのTECSは静的コード生成を基本として発展してきましたが、大山氏は「動的生成にも良い面がある」とし、将来的には動的生成を取り入れる可能性にも言及しました。
静的設計の堅牢さと動的生成の柔軟さを両立させる試みは、TECSの新たな進化の兆しとも言えるでしょう。

議論3：RTOSの依存部の比較

・RTOSの依存部の設計に関しての情報共有

竹内 良輔 様（TOPPERS 教育WG主査）

竹内氏からは、TOPPERSカーネルの移植性をテーマに、歴代カーネル群とFreeRTOSを比較した興味深い報告がありました。JSP、APS、ASP3、そして参考としてFreeRTOSおよびμITRON4.0対応のFreeRTOS+JSPを並べ、ROM/RAM消費量やソースコードのファイル数といった観点から、構造的な違いを可視化した分析が紹介されました。

その中で特に話題となったのが、ASP3におけるターゲット依存部の多さと移植性の問題です。FreeRTOSに比べてファイルの種類が多く、構造が複雑に見える点が「移植しづらい」と感じられる一因ではないかとの指摘がありました。これに対して高田会長からは、ASP3のターゲット依存部にSTM32CubeMX由来のファイルが含まれているため一見大きく見えるだけであり、パッケージ固有の事情によるものだとの説明がありました。また、ファイルの種類の多さはTOPPERS特有のコンフィギュレーション機構によるものであり、それ自体がTOPPERSカーネルの強みでもあることが強調されました。過去には「コンフィギュレーションなし」の試みもあったものの、結果的に普及には至らなかったというエピソードも紹介されました。

さらに竹内氏からは、教育WGでの取り組みとして、TOPPERS公式サイトに掲載されているJSPカーネル最新版が現状ではビルドできない問題が挙げられました。教育現場での教材として活用するため、

ビルド可能な状態への修正や最新ボードへの対応作業を進めているとの報告があり、TOPPERSを学習用プラットフォームとして支える地道な努力が垣間見える発表となりました。

ハンズオン1：VSCode+Pico2でFreeRTOSを動かす

・Visual Studio Code の Raspberry Pi Pico 開発環境で FreeRTOS を動かすハンズオン
　本田 晋也 様（名古屋大学）

ハンズオンで使用したプロジェクトは下記のリポジトリを参照。
https://github.com/exshonda/Pico_FreeRTOS_sample

　本田氏によるセッションは、Visual Studio Codeを使ったRaspberry Pi Pico開発環境で、FreeRTOSの最新機能である「マルチコア拡張」を実際に試してみるハンズオン形式で行われました。FreeRTOSはVer.11からマルチコア対応を正式に取り入れており、今回のハンズオンではその仕組みを体験できるサンプルプロジェクトをビルドし、デバッグまで行う内容となっていました。
　ハンズオンの途中では、FreeRTOSにおけるマルチコア対応の実装方法についても説明がありました。特に排他制御の部分に注目すると、排他が必要な場面では他のコアの処理も一時停止させる設計となっており、その結果、オーバーヘッドが大きくなることが分かりました。実際に性能を評価したところ、TOPPERSカーネルのシングルコア版やマルチコア版と比較しても、FreeRTOSのマルチコア版は性能があまり良くないという結果が得られたそうです。むしろ、シングルコア版のFreeRTOSよりもTOPPERSカーネルの方が高い性能を示した点が印象的でした。
　また、FreeRTOSのマルチコア対応はシングルコア版と同じソースコード内で「#ifdef」によって切り替えているため、コード全体が複雑化しているとの指摘もありました。さらに、公式のターゲット依存部がすべて含まれている一方で、サードパーティー製の移植も別フォルダに収められており、同一ターゲットで公式版とサードパーティー版の両方が存在するケースもあるそうです。加えて、ツールチェインごとに複数のフォルダが存在する構成となっており、ソースコードの管理が煩雑化している点にも触れられました。本田氏ご自身も、FreeRTOS開発チームがこの複雑な構成をどのように管理しているのかに強い関心を寄せていました。
　参加者にとっては、単なるハンズオンにとどまらず、FreeRTOSとTOPPERSの設計思想や開発体制の違いを実感できる貴重な機会となりました。特に、リアルタイムOSのマルチコア対応という難題に対して、異なるアプローチがどのように性能や保守性に影響するのかを具体的に比較できた点は、非常に示唆に富む内容でした。

ハンズオン2：VSCode+Pico2でTOPPERS/FMP3を動かす

・Visual Studio Code の Raspberry Pi Pico 開発環境で TOPPERS/FMP3 を動かすハンズオン
　長島 宏明 様（コアーズ株式会社）

ハンズオンで使用したプロジェクトは下記のリポジトリを参照。

https://github.com/exshonda/fmp3_pico_sdk_sample
ハンズオンの資料は下記のQiitaの記事を参照。
https://qiita.com/h7ga40/items/733f0f434757250bb654

本ハンズオンでは、「ユーザーが手軽にTOPPERS/FMP3の開発からデバッグまで体験できる環境を整える」という目的のもと、Visual Studio Codeを用いたRaspberry Pi Pico開発環境を紹介しました。

特に今回は、Raspberry Pi 5を利用したビルド環境の提案を行った石岡氏にもご参加いただき、より実用的な開発環境の構築方法について活発な議論が行われました。WSL、Docker、仮想マシン（VM）など、さまざまな環境構築手法について、それぞれの利点と課題を比較検討しました。

たとえば、Raspberry PiやVMを利用する場合は、SDカードイメージや仮想ディスクをそのまま共有できるため、ハンズオンなど短期間での体験には適しています。しかし、個々の開発者が自宅で学習を続けるには、環境の再現性や保守性に課題が残ります。
一方、WSLでは環境の構築が容易な反面、システム上で一つの環境しか作成できない制約があり、既存の開発環境との共存に不安を抱く利用者も少なくありません。
Dockerイメージを用いる場合は、複数人が同時にダウンロードを行う際にネットワーク帯域を圧迫し、初回のセットアップに時間がかかる点が問題として挙げられました。

最終的に、どの手法にも一長一短があり、ビルド以外にもそれぞれのツール特有の操作や知識が必要であることが確認されました。その中で長島氏は、今回使用したVSCodeベースの開発環境は、比較的に実践的で有用な選択肢であるという提案をしました。

石岡様のRaspberry Pi 5を利用したビルド環境に関する記事は下記になります。
https://qiita.com/Yukiya_Ishioka/items/652f295cd43b159857d8

2025年度開発者会議を振り返って

今年の開発者会議はゲストトークなしでの開催で、例年より開催時間も短いものとなりました。また、合宿形式を検討する期間的な余裕が無かったので、5回目のオンライン開催となりました。企画時には、TOPPERSを実際に使ってもらうのによい、ハンズオンをベースにしたプログラムで組み立てましたが、実際の開催ではTECS WGや教育WGから議論の提案をいただき、十分な議論の時間が取れたのではないかと思います。

オンライン開催が5回続きましたが、参加者から合宿形式の要望も出ています。今回は開催告知の期間も短く、少人数の参加者となりましたが、例年のオンライン開催では非会員も含め幅広い参加者に集まってもらえています。来年度の開催形態については、両方のいいところを活かせるハイブリッド開催などを含めて、これからの開発者会議のあるべき姿を含めて検討したいと考えています。

開発者会議2025実行委員会

堀武司（個人会員、実行委員長）伊与田健敏（創価大学）
小南靖雄（個人会員）高田光隆（名古屋大学）長島宏明（コアーズ(株)）
山根ゆりえ（特別会員）

NPO法人 TOPPERSプロジェクト
〒104-0042　東京都中央区入船1丁目5-11　弘報ビル5F
（一社）組込みシステム技術協会内
TEL & FAX: 03-6275-2981　Email: devconf@toppers.jp
Web：https://www.toppers.jp/index.html
GitHub：https://github.com/Toppers
Facebook：https://www.facebook.com/toppersproject