Preccs：実用的な通信プロトコルコンパイラ

—インターネットを支える基盤技術の構築—

１．背景

近年，インターネットでは様々な種類のサービスが提供されており，その基盤となる通信プロトコルは複雑かつ多様なものとなっている．一般的に通信プロトコルの実装は，プログラマが仕様（RFC など）を理解・解釈し，それに従って C 言語によってコーディングを行うという比較的退屈な作業である．通信プロトコルは厳密で正しい実装が要求されるが，その仕様は自然言語で記述されており，その曖昧さから時として実装系による差異が生じる．また，コーディングは C 言語のような命令型言語を用いることがほとんどであり，出来上がったプログラムは複雑でわかり難く，バグを埋め込みやすい．これらの問題は，システムの信頼性や開発における生産性の向上といったことを考える上で大きな障害となりうる．

２．目的

前節で述べた背景から，われわれは通信プロトコルのための直感的でわかりやすい仕様記述言語 Preccs を提案し，Preccs による仕様記述から C 言語のプログラムを自動的に生成する通信プロトコルコンパイラを開発している．図 1 に Preccs を用いた通信プロトコル開発の流れを示す．

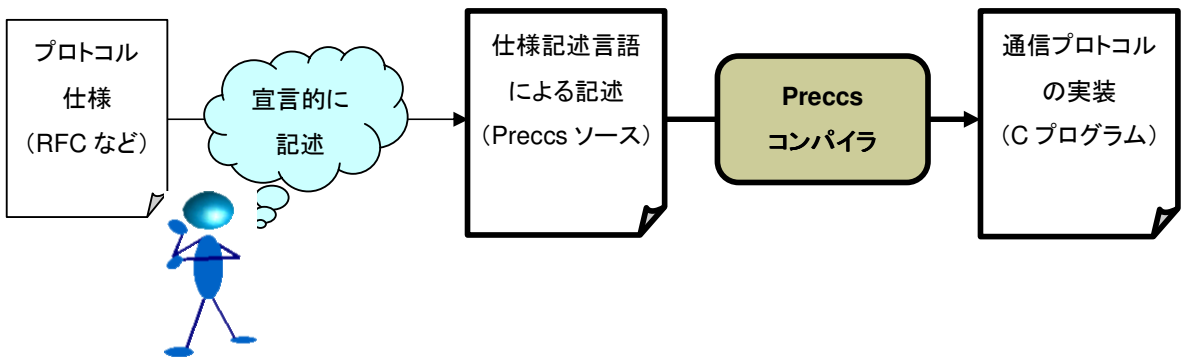

図 1　Preccs による通信プロトコルの開発

最初の Preccs は 2004 年度の未踏ソフトウェア創造事業の支援によって開発されたが，記述性や実行時性能など，改善の余地が数多く存在した．そこで，Preccs をより実用的なものとし，広く利用可能なものとするために，機能追加性能の改善を行った．

３．開発の内容

図 2 に Preccs を用いた通信プロトコルの実装の流れを示す．プログラマは RFC（Requst for Comment）など自然言語で記述された通信プロトコル仕様にもとづいて，メッセージ形式や送受信手順を Preccs のソースとして記述する．Preccs コンパイラは Preccs ソースから，パタンマッチに用いる状態遷移表やプロセス動作を実現するための処理を C 言語のコードとして出力する．ここで出力されたコードは Preccs 実行時ライブラリと協調しながら，プロトコルの中核的な処理をインタプリティブに

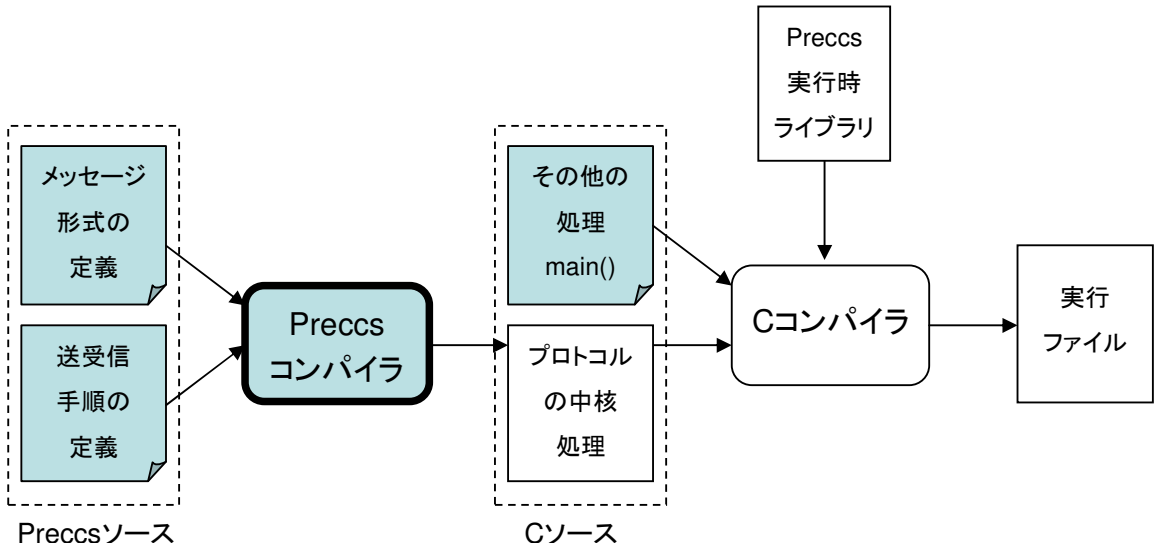

図 2　Preccs を用いた実装の流れ

実行するものである．Preccs 実行時ライブラリは，パタンマッチやチャネル通信，プロセス管理などを実現するためのルーチンから構成される．

その他，Preccs で記述するのが適当でない雑多な処理，たとえば，コンフィグレーションの取得やログ出力，OS に関する情報の取得などはプログラマが直接 C でプログラムを記述する．最終的には，これらのコードと実行時ライブラリから，C コンパイラによって実行ファイルが生成される．C のコードを生成することによって可搬性が高くなり，通信アプリケーションのほかに，組み込み系の通信プログラム，さらには OS のカーネルレベルで実装されているプロトコルスタックなど，様々なプラットフォームに柔軟に対応することができる．

【動作環境】

基本的に Windows NT 系（Windows 2000, Windows XP 等）で動作する．ただし，ソースコードのビルドには，MinGW と OCaml が必要である．

４．従来技術との相違

通信プロトコルの形式的仕様記述の方法としては LOTOS や Estelle などが知られている．これらは ISO において OSI のプロトコルを規定するために開発されたものである．LOTOS や Estelle の第一義の目的は新しく考案されたプロトコルの仕様検証であり，検証済みの既存プロトコルを対象としている本プロジェクトが目指す方向とは異なっている．LOTOS や Estelle で書かれた仕様からコードを自動生成するコンパイラも開発されているが，TCP/IP など実際に動作する処理系をこれによって記述するのは困難である．さらに，これらの言語は幅広い階層の通信プロトコルを対象として規定されたものであるため，その言語仕様は複雑である．さらにプロトコルの記述には形式的仕様記述に関する専門的な知識が必要となる．このため，一般的なプログラマがこれらの言語を用いて通信プロトコルを記述したり，その記述から通信プロトコルの仕様を理解したりするのは困難である．実際，これらの仕様記述言語が**インターネットの通信プロトコルの実装に用いられている例は皆無**といってよい．

Erlang は電話会社であるエリクソンの研究所で開発された並行関数型言語であり，Preccs と同様に並行プロセス間で通信を行いながらプロトコルの処理を行う．ただし，Erlang では，チャネルではなく個々のプロセスが備えているポートを用いて，プロセス間通信を行う．ソケットなどもポートとして抽象化されており，外部プロセスとの通信も同じように可能である．パタンマッチ機能もサポートしているが，Preccs のように正規表現によるパタンマッチを行うことはできない．また，Erlang VM 上で動作するものであり，Preccs コンパイラのように C 言語のコードを出力したり，インラインで C 言語のコードを記述したりすることはできない．

５．期待される効果

本ソフトウェアを利用することで，以下に挙げるような様々な効果が期待でき，通信プロトコルの実装に大きなブレークスルーをもたらすであろう．

⑴ 開発工数の大幅な削減

通信プロトコルの中核となるコードを自動生成することによって，実装における開発工数を大幅に削減することが可能となる．

⑵ 信頼性の向上

プロトコルの仕様さえ正しく記述すれば，実装系は Preccs コンパイラによって自動生成される．よって，プログラマの不注意による実装時のバグを無くすことができ，信頼性が大幅に向上する．

(3) プロトタイピングツールとしての利用

通信プロトコルのプロトタイピングとしての利用が考えられる．例えば，実験的なプロトコルを提案した場合，その有効性を検証するために C で実装を行うのは大変な労力と時間を要する．本ツールを用いれば，簡単に実装系を得ることができ，プロトコルの有効性をすぐに検証することができる．

(4) プロトコルのテストプログラムの実装

プロトコルを実装した通信機器やソフトウェアをテストするためには，そのプロトコルに従ったパケットを送信したり，受信したパケットを解析したりするためのテストプログラムを用意することが多い．そのようなテストプログラムも，Preccs によって簡単に実装することができる．

(5) 保守性・拡張性の向上

Preccs で記述したソースは直感的でわかりやすいため，ソフトウェアの保守性が向上する．また，制約指向スタイルと呼ばれる記述によって，新たな機能追加も容易になる．

６．普及の見通し

本ソフトウェアは，通信プロトコルの実装に携わっている世界中のプログラマに幅広く利用してもらうために，下記のサイトからオープンソースとして公開を行っている．

http://preccs.isp.jp

また，Preccs ユーザー向けのメーリングリストも開設し，コミュニティの形成を目指している．今後は，本ソフトウェアを実際の業務に適用し，効果を検証していく予定である．また，開発を通じて得られた知見や成果は，今後とも学会や研究会などで積極的に発表し，本ソフトウェアの知名度を高めていく．これによって，より多くの開発者に利用されることが期待できる．

７．開発者名

服部健太（株式会社システム計画研究所　技術本部）