平成１４年度　未踏ソフトウェア創造事業「未踏ユース」

オープンソースＣＯＢＯＬコンパイラの開発

Development of Open-Source COBOL Compiler

西田　圭介
Keisuke NISHIDA

株式会社ネットワーク応用通信研究所（〒690-0826 島根県松江市学園南二丁目１２番５号
ＨＯＹＯパークサイドビル２Ｆ　E-mail: knishida@netlab.jp ）

ABSTRACT. The goal of this project is to develop a new COBOL compiler that is distributed as open-source software. COBOL is a programming language that has been used on large-scale computes such as mainframes for a long time. Nowadays, enterprise systems written in COBOL are running on Linux, and a good open-source COBOL compiler is desired. In this project, I have developed a COBOL compiler that works as a GCC frontend. This is a significant progress toward using Linux for enterprise system development.

１．背景

近年、Linux を始めとするオープンソース・ソフトウェアの発展と普及によって、多くの業務システムがオープンソース・ソフトウェアを用いて動作するようになってきた。従来、業務の中枢を担うようなシステムはメインフレーム等の大型コンピュータで動作してきたが、そうした高い信頼性が要求されるシステムを安定して稼働させられるまでにオープンソース・ソフトウェアが成長してきたのだと言えよう。

しかしながら、これまでメインフレーム上で最も使われてきた言語である COBOL となると、オープンソースでは十分な開発環境が整っているとは言い難い。現在、オープンソースとして開発されている COBOL コンパイラの中で、十分な完成度を持つものは皆無であるというのが現状である。

一方で、これまで高価なシステムで運用されてきた COBOL ソフトウェアを、PC や Linux 等の比較的安価なシステムに移行させたいというニーズは強く、それを助ける完成度の高い COBOL コンパイラが求められている。

Linux が高い信頼性を獲得し業務システムでも用いられるというなら、それに合わせたオープンな COBOL コンパイラが開発されることも自然な流れであろう。現に、ここ数年になってそうした流れは活発になりつつあり、いくつかの開発プロジェクトが進行している。

筆者もこれまで OpenCOBOL というオープンソース COBOL コンパイラの開発に携わってきた。これは既に COBOL の基本的な構文をサポートしているが、これを業務レベルで実用的なものとするためには、まだ多くの作業が残されている。

そこで今回のプロジェクトでは OpenCOBOL を拡張し、実用に耐える COBOL コンパイラの開発を進めることで、オープンソース・ソフトウェアによる業務システム開発のための基盤整備を進めることを目指した。

２．目的

本プロジェクトの目的は、オープンソースによる COBOL 言語のコンパイラを開発することである。具体的には、Linux における標準的なコンパイラである GCC（GNU Compiler Collection）のフロントエンドとして、COBOL85 規格をサポートするための拡張を行なう。

GCC のフロントエンドを開発することには、筆者がこれまで開発してきた OpenCOBOL と比べて、次に挙げるような利点がある。

（１） デバッグ機能の追加

OpenCOBOL では COBOL プログラムを C 言語に変換してから、C コンパイラによってコンパイルを行なっていた。これはコンパイラの開発を楽にするという利点があったが、C 言語に変換した時点で COBOL のデバッグ情報が失われてしまい、デバッガを用いることが困難となっていた。コンパイラを GCC 対応することで、デバッグ情報を含めたコード生成が行なえるようになり、オープンソースの代表的なデバッガである GDB によるデバッグが行なえるようになる。

（２） コンパイル速度の高速化

COBOL を C 言語に変換してから C コンパイラを通していたのでは、コンパイルするために余分な時間が必要となる。特に、一般的な COBOL プログラムは、合計で何十万ステップ、何百万ステップを超えるような巨大なソースコードとなるので、コンパイル速度は可能な限り速くしたい。

（３） スタンダードの確立

GCC はオープンソースの世界で標準的に用いられているので、それが COBOL をサポートすることとなれば、オープンソースによる標準の COBOL コンパイラとなる足がかりとなる。これはソフトウェアの有用性を高めるために有効である。

３．成果

今回のプロジェクトにより、以下のような開発成果が得られた。

（１） GCC による COBOL プログラムのコンパイル

OpenCOBOL の開発成果を流用し、COBOL の構文解析ルーチンを GCC に組み込んだ。これにより、GCC のフレームワークを利用して、COBOL で書かれたプログラムをネイティブの実行形式にコンパイルすることが出来るようになった。GCC のオプションを指定することで、プログラムをシェアードライブラリ化することや、他のプログラムとリンクすることも可能となった。

現時点では、COBOL85 規格のうち、以下の機能を実現している。

1. COBOL のコアとなる基本的な構文
2. COPY ファイルの読み込みと前処理
3. ファイル処理（順序・相対・索引・整列ファイル）
4. サブプログラムとの連結（CALL 文）

（２） GDB による COBOL プログラムのデバッグ

GDB を利用することで、COBOL のソースコードにブレイクポイントをセットして実行を一時停止することや、一行ずつのステップ実行を行なえるようになった。

ただし、デバッグ中に COBOL 特有の変数の値を参照するには、GDB 側にも COBOL のサポートを追加する必要があるが、残念ながら今回はそこまで開発を進めることは出来なかった。今後の課題としたい。

（３） エラーチェックの強化

以前の OpenCOBOL と比較して、コンパイル時、及び実行時のエラーチェックを強化し、ユーザにとってわかりやすいエラーメッセージを出力するようになった。

例えば、プログラム中で未定義の変数が使われていたとき、「変数が未定義である」という当たり前のエラーを出せるようになった。以前は、構文解析上の都合でそれが出来ず、エラーがわかりづらいものとなっていた。

今回、構文解析の処理を大きく見直すことで、より厳密でユーザフレンドリーなエラー処理を行なえるようになった。

（４） コンパイル速度・生成コードの改善

OpenCOBOL と比較して、コンパイル速度で 30%程度の向上がみられた。C 言語に変換してからコンパイルし直すという方法と比べて、構文解析の二度手間が省かれた影響が大きいようだ。

また、生成されるオブジェクトコードのサイズも 20%程度、小型化した。OpenCOBOL が生成していた C 言語のコードと比べて、より無駄のないコードを生成するようにしたためである。

もっとも、現時点ではこれらの性能を最適化するための作業は行なっておらず、性能改善の余地は多分に残されている。

４．今後の課題

これからの課題を以下に挙げる。

（１） COBOL 機能の実装充実

COBOL85 規格への完全準拠や、商用コンパイラとの互換性を得るため、今後、以下のような機能を実装する必要がある。

1. 組み込み関数機能
2. 埋め込み SQL
3. SCREEN 機能（ターミナル I/O）
4. REPORT 機能（帳票印刷）
5. COMMUNICATION 機能（オンライン通信）
6. 商用コンパイラとの互換性機能

（２） エラーチェックの強化

現在のところ、正しく書かれたプログラムを処理することには問題なくとも、誤りのあるプログラムをコンパイルした場合に、誤りをまったく指摘出来なかったり、あるいは意味不明のエラーが表示されることがある。

今後も、COBOL の仕様に従い、より厳密なエラーチェックを行い、ユーザにとってわかりやすいエラーを表示するよう改善していく必要がある。

（３） 充実したテストスイートの整備

COBOL の標準機能をチェックするテストスイートは公式に用意されているが、コンパイラのエラーチェック機能を確かめるテストスイートや、COBOL の拡張機能のためのテストスイートは十分に整っていない。より充実したテストスイートを整備し、コンパイラの品質を確かなものとする必要がある。

（４） コンパイル速度・生成コードの最適化

現時点ではまだ十分な最適化を行なえていない。例えば、コンパイル速度については、現在の実装ではまず構文解析の段階で COBOL の構文木を生成してから、それを GCC の内部構造に変換するという二段階のステップを踏んでいる。これを最初から GCC の内部構造を生成するよう修正することで、大きな速度向上が期待される。

また、生成されるコードについても、実行速度を高めるための余地が多く残されている。例えば、コンパイル時に変数の型や大きさをチェックし、それに合わせた最適なコードを生成することで、今より一層の速度向上が可能である。

（５） GCC への公式採用

現時点では、開発したコンパイラは GCC の標準配布には取り込まれていない。関係者への連絡は行なったが、これが実際に取り込まれるためには、今よりもソースコードのクオリティを上げることや、著作権上の書類を準備するなど、やるべきことが残されている。

開発したコンパイラが GCC の一部として日の目を見るのは、まだ先の話となりそうだ。

５．まとめ

オープンソースによる COBOL コンパイラとして、GCC で COBOL 言語をサポートするという課題は達成された。これが GCC 本体に取り込まれ、広く使われるようになるまでには多くの作業が残されているが、オープンソースによる業務システム開発に向けて着実な前進を遂げたといえる。

今回の成果物は OpenCOBOL のホームページ[1]にて、オープンソース・ソフトウェアとして公開している。今後も引き続き開発を継続し、成果はオープンソースとして公開する予定である。

６．参加企業及び機関

なし

[1] http://www.open-cobol.org/

7．参考文献

なし