3F-1

MISRA-C:1998 と MISRA-C:2004 の C90，C99 との検討

坪井泰樹、吉川直邦†、小川清、斉藤直希‡
大同工大†　　名市工研‡

１．まえがき

自動車、家電製品をはじめ人工衛星から IC カードまで、CPU が内蔵される組み込みシステムにおいて装置を制御するソフトウェアが、システムの重要な要素となっている。組込みシステムでは要求機能、性能を限定し、厳しい費用制限から利用可能な資源に個別の制約がある。

これらの制約を実現する場合に、C++,JAVA のような他の高級言語ではなく C 言語を利用することがある。しかし、C 言語は“C の精神”に基づいてプログラマを信頼する仕組みであり、書かれたプログラムの信頼性を確保する仕組みが必要になっている。また、組み込みシステムのプログラミングをアセンブラではなく、C 言語 を利用する場合に、移植性に対する期待もある。

MISRA(Motor Industry Software Reliability Association)は、欧州の自動車関連のソフトウェアの信頼性を検討する団体で、ソフトウェア開発作業標準と C コーディング標準を発行している。これは、作業と製品との両方において標準を作成することにより、信頼性の高いソフトウェア開発を実現しようとするものである。前者は 1998 年発行後、ISO の TR 15497:2000 として発行されている。後者は 1998 年に発行された”Guidelines For The Use Of C Language In Vehicle Based Software”で、MISRA-C:1998 と呼んでいる。

MISRA-C は欧州だけでなく、日本や北米の自動車業界においてプログラミングガイドラインとして、その内容に基づいて改訂されたのが MISRA-C:2004 である。

MISRA-C1998,2004 とも基にしている C 言語規格は ANSI-C89 または C90 と呼ばれる ISO/IEC9899:1990 である。ISO/IEC 9899 は 1999 年改訂され、C99 と呼ばれている。

２ C プログラミング言語教育

名古屋市工業研究所では企業の技術者への C プログラミング言語の教育の中で、C コンパイラのソースコードを使うとともに、C Puzzle Book[1]に記述されている誤解しやすいサンプルを教育に利用してきた。具体的に処理の過程、結果を出力することにより、C 言語の機能を確認するものである。この教育で、C 言語には可読性上の課題と、機種依存性の課題があることが確認できた。

C 言語は、OS のない状態でのプログラミング環境と、OS を前提としたプログラミング環境とに区分されている。C 言語が想定している実行環境には POSIX に対応する UNIX/LINUX のような OS がある。POSIX 準拠の OS では、OS が解決すべき課題もある。しかし、OS のない組込みシステムの開発や、C 言語が想定していなかった OS 上でのアプリケーション開発において、ソースコードを読み下し、検査を行うためには、C 言語の CPU とコンパイラに依存した部分を理解している必要がある。C 言語の機種依存性を理解するための方法としてオープンソースである GCC のソースコードを、辿っていく方法が考えられる。しかし、この作業は特定の OS を開発するためには必要であるが、開発対象をすぐに書きたい場合には、膨大である。そのため、単体試験プログラムを作成し、コーディングルールによる該当する規則との関係で理解する手掛かりを掴むきっかけとすることができる可能性がある。

３.コーディングサンプル

MISRA-C 関連書籍でのサンプルは断片的なコードであったため、コンパイルできるソースの状態への復元を図った。復元の過程としては、第一段階として Windows 上のコンパイラ(Visual Studio6.0 及び gcc(cygwin))で MISRA-C:1998 に対応するサンプルをコンパイル、実行を行った。そのためには、stdio.h を include し、printf 関数と main 関数を利用して処理結果と処理経過を表示できるようにした。利用した GCC は、C99 の処理系定義の文書がある。また、Visual Studio6.0 の C コンパイラは、C90 への準拠率が高いものと思われる。ここで、C90 と C99 との違いのうち、確認できる事項を調査した。

二段階として具体的な CPU として、ルネサステクノロジーの M16C/M32C 評価ボードで、TOPPERS/JSP カーネル上にて動作するサンプルを作成し処理を実行させ、処理経過を表示するようにした。具体的には、サンプルをカーネル上でのタスクとして実行させた。ここでの発見事項とし

Study for MISRA-C 1998 and 2004 with C90 and C99
Tsuboi , Yoshikawa Naokuni (daido-it)
Ogawa Kiyoshi, Saito Naoki(NMIRI)

ては、コンパイラの処理系定義の確認作業の重要さと、試験プログラムの重要さが確認できた。サンプルプログラムは、実際にコンパイルして動作させると、C 標準に対する誤解、または処理系定義の意味の誤解により、想定したものと実行結果とが異なる場合があった。また、OS がある場合とない場合の記述上の制約について確認できた。第三段階として、MISRA-C:2004 に対応するサンプルを統合し、コンパイルして実行した。

プログラムの書式は次の通りである。

```
header: author, Create date, Update date
Rule: Rule #, rule(Japanese and English)
Body:
#inclue <misrac.h>
…
Result: Visual Studio(MicroSoft), GCC,
N308(Renesas Technology)
Footer: update log
```

１２７のルールに対して、原則１プログラム１ルールとした。ファイルに関連するルールでは、複数ファイルで１ルールとなる。プログラムファイルの内部構成は５部構成にした。ヘッダ部には著者、作成日を記述した。ルール部には、日本語と英語のルールを記載した。本体はプログラムそのもので、コンパイル時のスイッチによって、DOS モード、TOPPERS/JSP モードが切り替わるようにした。結果部は、各コンパイラでコンパイルした際のエラーと実行するための修正、実行した結果をつけ、比較をしやすくした。最後にフッタ部に改訂の履歴をつけた。

Gcc では C90 準拠を確認する必要がある場合は、-ansi でコンパイルした。それぞれの OS 上での MISRA－C の対応のガイドは存在していないが、OS ごとに逸脱の手続きを定義してあるとよい事項が確認できた。

４ 逸脱の手続きとまとめ

MISRA では、規則の一貫として、規則に適合しないコードは、逸脱とし、その逸脱の理由を明確にし、文書化することを規定している。この逸脱の手続きを取ることは信頼性の確保にとって重要である。あるコーディング規則はどんな種類の処理にも有効とは限らないという仮定を取れば、個々の目的によってどのように適用するのがよいかの検討が必要である。費用を最小にして規則を守ることを重視すると、処理速度の低下、コードの移植性の低下を招くかもしれない。例えば、goto 文を使わない方法でプログラムを書くには、リカーシブコールを使うか、関数のネストを深くすることによって実現することは容易である。しかし、関数のネストが深くなることは、スタックオーバフローを招いたり、呼び出しのオーバヘッドが多くなり、メモリが少なかったり、高速化が必要な場合には、安易に goto 文を使わないようにするだけでは信頼性が低くなることがある。また、CPU 固有の機能により、異なる CPU で振る舞いが異ならないようにするために、同一の動作をさせようとする場合に、機械語で直接書くより C 言語で goto 文を利用した方が明確に書くことができる場合もある。

C 言語で書かれた特定の OS では、その OS 自身の標準的な逸脱の手続きと、その OS の上に書くアプリケーションについて逸脱の手続きがあると便利である。名古屋市工業研究所では、ITRON 準拠のオープンソースである TOPPERS/JSP カーネルの M16C/M32C への移植を行い、教育に利用しているが、そのソースを評価するため、MISRA-C チェッカを利用し、OS における MISRA－C の適用について検討している。規則の逸脱の手続きを踏んでいるものが多ければ多いほど、信頼性が高い可能性もあり、形式的な規則の適用の危険性を確認することができた。

５ まとめと今後の課題

コーディング規則の解説におけるサンプルを複数のコンパイラでコンパイルし、評価した。また、組込み ITRON のオープンソースの OS である TOPPERS/JSP カーネル上でのタスクとして実行させた。規則を守らないときだけ文書化させるのでは、非対称であるためうまく使われない可能性があり、守らない方が品質が高いような事例に共通性がある事項は、運用する側が、その品質について記述するようにすることが重要であることがわかった。OS ごとに、MISRA-C の対応のガイドがあると、OS 上のアプリケーション開発者に役立つと思われた。プログラムは複数の規則を１つの実行ファイルにし、TOPPERS/JSP の Monitor 上でタスクとして選択できるように改良中である。

文 献

[1] C puzzle Book,Alan R. Feuer, Addison-Wesley

[2] 組み込み開発者におくる MISRA-C 組み込みプログラミングの高信頼化ガイド、MISRA-C 研究会、日本規格協会, 2004

[3] www.misra.org.uk, Moter Industry Software Reliability Association

[4] www.toppers.jp, www.sessame.jp