ソースチェックに威力を発揮するＣプリプロセッサ
― 世界一の規格準拠性と豊富な診断メッセージ ―

松井　潔
Kiyoshi MATSUI
陽和病院（〒178-0062 東京都練馬区大泉町 2-17-1） E-mail: kmatsui@t3 .rim.or.jp

2004/03/31

1　背景

Ｃで書かれたソースプログラムには、プリプロセスのレベルでの問題を持っているものが少なくない。特定の処理系でコンパイルできることをもって良しとしてしまっているものの portability を欠いているもの、不必要にトリッキーな書き方をしているもの、C90 以前の特定の処理系の仕様をいまだにあてにしているもの、等々である。こうしたソースの書き方は portability と readability そしてメンテナンス性を損なうものであり、悪くすればバグの温床ともなりかねない。そうしたソースをより portable で明快な形で書き直すことは、多くの場合、簡単なことなのであるが、見過ごされている場合も多い。

そうしたソースが多く存在する背景となっているのは、一つには C90 以前のプリプロセス仕様がはなはだあいまいだったことである。これが、C99 が決まった今となっても尾を引いている。もう一つは、既存のプリプロセッサが寡黙すぎることである。プリプロセッサが怪しげなソースを黙って通すために、問題が見過ごされてしまうのである。

2　目的

私は久しく以前からＣプリプロセッサを開発してきた。その成果はすでに 1998/08 に cpp V.2.0 として、1998/11 に cpp V.2.2 として公開している。このプリプロセッサは V.2.3 への update の途中で、「平成１４年度未踏ソフトウェア創造事業」に新部プロジェクトマネージャによって採択された。そして、2003/02 に V.2.3 が公開された。さらに

「平成１５年度未踏ソフトウェア創造事業」にも伊知地プロジェクトマネージャによって継続して採択された。この cpp を他の cpp と区別するために MCPP と呼ぶ。Matsui CPP の意味である。

このプロジェクトの目的は、前年度の MCPP V.2.3 の成果を踏まえ、次のような V.2.4 を開発することである。

- 動作が正確で診断メッセージの豊富な、世界一優れたＣプリプロセッサを開発する。
- 対応する multi-byte character の encoding を増やし、移植可能なプラットフォームの範囲を拡大する。
- MCPP のソースコードから実行プログラムを自動生成するための configure スクリプトを開発する。
- 英語版のドキュメントも整備して、成果を世界に問う。

3　MCPP の概要

MCPP は次のような特徴を持っている。

1. きわめて正確である。C, C++ のプリプロセスの reference model となるものを目指して作ってある。C90 はもちろんのこと、C99, C++98 に対応する実行時オプションも持っている。
2. C, C++ プリプロセッサそのものの詳細かつ網羅的なテストをする検証セットが付属している。
3. 診断メッセージが豊富で親切である。診断メッセージは百数十種に及び、問題点を具体的に指摘する。それらは数種のクラスに分けられており、実行時オプションでコントロールすることができる。
4. デバッグ用の情報を出力する各種の #pragma ディレクティブを持っている。Tokenization をトレースしたり、マクロ展開をトレースしたり、マクロ定義の一覧を出力したりすることができる。
5. Multi-byte character の処理は日本の EUC-JP, shift-JIS、中国の GB-2312、台湾の Big-5、韓国の KSC-5601 (KSX 1001) 等に対応している。
6. 速度も遅いほうではないので、デバッグ時だけでなく日常的に使うことができる。16 ビットシステムでも使えるように作られているので、メモリが少なくても動作する。
7. Portable なソースである。MCPP をコンパイルする時に、ヘッダファイルにある設定を書き換えることで、UNIX 系、DOS/Windows 系のいくつかの処理系で、付属

のプリプロセッサに代替して使えるプリプロセッサが生成されるようになっている。C90, C99, C++98 のどれに準拠する処理系でもコンパイルでき、C90 以前のいわゆる $K\&R^{1st}$ の処理系でさえもコンパイルできる広い portability を持っている。

8. 標準モード（C90, C99, C++98 対応）のプリプロセッサのほか、$K\&R^{1st}$ の仕様やいわゆる Reiser モデルのもの等、各種仕様のプリプロセッサを生成することができる。規格そのものの問題点を私が整理した自称 post-Standard モードまである。
9. オープンソースである。
10. 詳細なドキュメントが付属している。次の各ドキュメントにそれぞれ日本語版と英語版が用意されている。
    （a）README: MCPP の実行プログラムを生成してインストールするための configure と make の方法を説明。
    （b）mcpp-summary.pdf: サマリ文書。
    （c）manual.txt: 実行プログラム用マニュアル。使い方、仕様、診断メッセージの意味。いくつかのソースコードをチェックした結果も報告している。
    （d）porting.txt: 実装用ドキュメント。任意の処理系に実装する方法。
    （e）cpp-test.txt: 検証セット解説。規格の解説を兼ねる。規格そのものの矛盾点も指摘し、代案を提示している。検証セットをいくつかの処理系に適用した結果を報告している。

4　プリプロセス検証セットによる各種プリプロセッサの検証

プリプロセッサの開発と同時にもう一つ問題となるのは、プリプロセッサの動作や品質の検証である。私は MCPP 開発の一環として、プリプロセス検証セットを作製し、MCPP とともに公開している。これはきわめて多面的な評価項目を持ち、プリプロセッサのできるだけ客観的で網羅的なテストをするものである。

検証セット V.1.4 はテスト項目が 265 に及んでいる。うち動作テストが 230 項目、ドキュメントや品質の評価が 35 項目を占めている。各項目はウェイトを付けて配点されている。$K\&R^{1st}$ と C90 との共通仕様を正しく実装していれば 0 点、それさえも実装できていなければマイナス点、C90 以降の新しい仕様を正しく実装していればプラス点をつけるようになっている。「規格合致度」には診断メッセージとドキュメントの評価も含まれる。

検証セット V.1.4 をいくつかの処理系に適用した結果のサマリを表 1 に示す。処理系

表 1: 各種プリプロセッサの検証結果

ＯＳ	処理系	実行プログラム (版数)	規格合致度	総合評価	注
Linux		DECUS cpp	230	287	1
MS-DOS		JRCPPCHK (V.1.00B)	400	451	2
WIN32	Borland C++ V.4.02J	cpp32	397	444	3
DJGPP V.1.12 M4	GNU C 2.7.1	cpp	445	545	4
MS-DOS	LSI C-86 V.3.30c	cpp (改造版 beta13)	341	397	5
FreeBSD, WIN32, etc.	GNU C, Borland C, etc.	MCPP (V.2.0)	495	651	6
WIN32	Borland C++ V.5.5	cpp32	397	451	7
Linux, FreeBSD	GNU C 2.95.3	cpp0	470	570	8
Linux, FreeBSD	GNU C 3.2R	cpp0	530	646	9
Linux, etc		ucpp (V.1.3)	483	562	10
WIN32	Visual C++ .net 2003	cl	452	517	11
WIN32	LCC-Win32 V.3.2	lcc	396	476	12
Linux, FreeBSD, etc.	GNU C, LCC-Win32, etc.	MCPP (V.2.4)	583	750	13

は古い順に並べてある。

*1 Martin Minow による DECUS cpp のオリジナル版 (1985/01) を筆者が若干の修正を加えて Linux / GNU C 3.2 でコンパイルしたもの。

*2 J. Roskind による UNIX, OS/2, MS-DOS 用 shareware である JRCPP の MS-DOS - OS/2 用試用版 (1990/03)。具体的な処理系には対応しない stand-alone のプリプロセッサ。

*3 1993 年のものの日本語版 (1994/12)。

*4 GNU C 2.7.1 / cpp (1995/12) を DJ Delorie が DOS extender である GO32 に移植したもの。日本語版への移植で shift-JIS に対応。

*5 LSI C-86 / cpp のきだあきらによる改造版 (1996/02)。

*6 筆者による free software の V.2.0 (1998/ 08)。DECUS cpp をベースとして書き直したもの。FreeBSD / GNU C 2.7, DJGPP V.1.12, WIN32 / Borland C 4.0, MS-DOS / Turbo C 2.0, LSI C-86 3.3 等に対応。各種の動作モードのプリプロセッサを生成することができるが、このテストでは 32 ビットシステムでの標準版を使用。

*7 日本語版 (2000/08)。

*8 VineLinux 2.5, FreeBSD 4.4, CygWIN 1.3.10 で使われている GNU C 2.95.3 (2001/03)。

*9 GNU C 3.2R のソース (2002/08) を筆者が VineLinux 2.5, FreeBSD 4.7 上でコ

ンパイルしたもの。

*10 Thomas Pornin による portable な free software (2003/01)。Stand-alone のプリプロセッサ。

*11 Microsoft (2003/04)。

*12 Jacob Navia 等による shareware (2003/08)。ソース付き。プリプロセス部分のソースは Dennis Ritchie その人が C90 対応のプリプロセッサとして書いたもの。

*13 MCPP V.2.4 (2004/02)。V.2.0 以降、Linux / GNU C (2.95.3, 3.2), FreeBSD / GNU C (2.95.4, 3.2), CygWin 1.3.10, LCC-Win32 3.2, Borland C 5.5, Visual C++ .net 2003, Plan 9 ed.4 / pcc 等への対応が追加されている。

このように、MCPP はずば抜けた成績である。動作の正確さ、診断メッセージの豊富さと的確さ、ドキュメントの詳細さ、portability、どれをとっても抜群である。５年前のバージョンである V.2.0 でさえも、まだそれを超えるものが見当たらない。V.2.4 ではさらに update され改良されている。自分で作って自分でテストしているのであるから当然とも言えるが、これだけ多角的なテストであればかなりの客観性があろう。

5 MCPP によるソースチェック

プリプロセスは実行時環境からほぼ独立したフェーズであるので、処理系の他の部分と異なり、portable なプリプロセッサというものが成立しやすい。各処理系が高品質で portable な同一のプリプロセッサを使うという形態さえ、ありえないことではない。

MCPP を処理系付属のプリプロセッサと置き換えて使うことで、ソースプログラムのプリプロセス上の問題点を、潜在的なバグや規格違反から portability の問題まで、ほぼすべて洗い出すことができる。

MCPP を各種のソースに適用した結果をドキュメントで報告してきている。FreeBSD 2.2.2R (1997/05) の kernel および libc、Linux の glibc 2.1.3 (2000/09)、GNU C V.3.2 (2002/08) 等のソースである。FreeBSD / libc, GNU C V.3.2 にはほとんど問題は見当たらなかったが、FreeBSD 2.2.2 / kernel, Glibc 2.1.3 にはかなりの問題が発見された。たとえば次のようなものである。

- 行をまたぐ文字列リテラル
- Ｃのプリプロセスを要する ∗.S ファイル
- ‘defined’ に展開されるマクロ
- 関数型マクロとして展開されるオブジェクト型マクロ

- Undocumented な環境変数を使うもの

GNU C / cpp がこれらを、少なくともデフォルトの設定では黙って通してしまうことが、こうした感心しない書法のソースを温存させ、そればかりか新たに生み出す結果になっていると考えられる。

6　MCPP の実装方法

私が MCPP の実装にあたって目標としたのは「MCPP の概要」で述べた諸点である。これは完全主義的な目標であるため、かなりの時間がかかってしまったが、ほぼ達成できてきている。

中でも網羅的な検証セットの作製と並行して開発してきたことは、バグの少ないプリプロセッサを作る結果につながってきている。

さらに MCPP の内部的な実装方法に立ち入ると、私が重視したのは次の原則である。

- トークンベースの原則
- 関数型マクロの関数的展開

これらの原則を明確にしないまま、古い文字ベースのマクロプロセッサのソースなどをベースとして、つぎはぎをしながらバージョンアップを繰り返してきたのではないかと見られるプリプロセッサもあるが、そうした方法では優れたプリプロセッサは決してできないのである。

7　V.2.3: 平成１４年度の成果

１４年度のプロジェクトでは、主として次のような成果が得られた。

1. GNU C 3.2 への対応
2. 検証セットの GNU C / testsuite への対応
3. 英語版ドキュメントの作成

8　V.2.4: 平成１５年度の成果

平成１５年度の終わりに公開された MCPP V.2.4 では次のような成果が得られた。

1. 市販の処理系で最もユーザの多い Visual C++ をとりあげ、これに検証セットを適用するとともに、MCPP を移植した。
2. UNIX 系のシステムでは、MCPP のコンパイルを configure スクリプトによって自動的にできるようにした。
3. ISO-2022-JP, UTF-8 等、Multi-byte character の多様な encoding に対応した。
4. Plan 9 / pcc に移植した。
5. これらに伴って、ドキュメントを日本語版・英語版ともに更新した。
6. 英語版ドキュメントとともにパッケージングして、gnu.org, freebsd.org に提起した。

9　協力企業

平成１４年度も平成１５年度も、MCPP の日本語版ドキュメントの英語への翻訳を有限会社・ハイウェル（東京）に委託した。MCPP の英語版ドキュメントは、それに筆者が技術的な内容についての修正を加えて作成したものである。

10　おわりに

今後は、MCPP V.2.5 の開発と並行して、英語の newsgroup である comp.std.c に投稿したり、雑誌で紹介したりして、MCPP の普及を図ってゆきたい。

MCPP は V.2.0 以来、vector と @nifty で公開してきた。

平成１４年度未踏ソフトウェアのプロジェクトでは、新部 PM によって m17n.org に CVS repository, ftp site, web page が用意された。開発中の revision を含めて最新版はここに置かれている。

```
http://www.m17n.org/mcpp/
```