マイクロコンピュータ　60－1
とワークステーション
（１９９０．５．８）

C言語の標準化動向について

猪瀬 武久
日本電気株式会社

昨年12月にANSI　C規格案が最終承認され、規格として出版される運びとなった。一方、国際規格としても、ANSI　C規格と同一内容のものが　DIS　(Draft International Standard) として承認投票中であり、最終段階を向かえている。本稿では、ANSI、ISO、および日本における標準化作業の経緯を振り返ると共に、ANSI　Cの特徴について概観し、今後に残されているCに関する標準化作業の話題について言及している。

Overview of Programming Language C Standardization Activities

Takehisa Inose
NEC Corporation.
14-22, Shibaura 4-Chome, Minato-ku, Tokyo 108,Japan

The ANSI C draft standard was approved finally in December 1989, and will be published as the standard. The draft international standard which is the same as the ANSI C standard is being voted as DIS (Draft International Standard), and is now in the final stage. This paper looks back at the history of standardization activities by ANSI, ISO, and Japan, surveys the features of ANSI C, and describes the topics of future C standardization activities.

1. はじめに

C言語は、UNIXオペレーティングシステムを記述するためのツールとしてBell研究所で開発されたプログラム言語であるが、その記述性の良さ、コンパクト性、記述されたプログラムの移植性の良さ等の理由から、パソコンやワークステーションを中心に急速に利用者層が広がり、今やマイコンシステム等のシステム記述言語の主役の座を占めるに至っている。

このように、利用者層が広がり、多くの処理系が商用化されていたにもかかわらず、きちっとした言語の定義もなく、正確で曖昧性のない言語仕様の規格化の必要性が叫ばれてきたわけである。

C言語の標準化は、1983年にANSIのC言語の規格原案を作成・検討する専門委員会(X3J11)の設置に端を発している。同委員会で作成・検討されたANSIの規格案は、C言語の曖昧性の排除ならびにC言語の弱点をカバーするための拡張機能を中心に改版を重ね、1989年12月にANSI規格として承認され、出版の最終段階に至っている。

一方、ISOにおいてもANSIと同期する形で1986年に標準化作業を開始し、ANSI案にISOの意見を反映する形で両者同一の規格となるよう作業が進められている。現在DIS (Draft International Satndard) としての郵便投票期間中である。

本稿では、Cの標準化の経緯を振り返ると共に、ANSI Cの主な特徴および今後のCに関する標準化の話題について添れてみたい。

2. Cの標準化作業の経緯

Cの標準化作業はANSI(米国規格協会)を中心に展開され、ISOは、これと同一の内容を国際規格として制定すべく、意見を提出するという形で進んできた。

2.1 ANSIにおける作業の経緯

C言語は、Dennis M. Ritchieによって開発された言語であるが、言語自体の優秀性から、作成者自身の予想をはるかに越える規模と速度で広がっていった。

この結果、いくつもの異なる処理系やさまざまな拡張が行われるようになり、互換性のない処理系の存在が、プログラマにとって大きな問題となってきたわけである。このような問題の最大の原因は、Cの文法書とよばれるものがB.W.KernighamとD.M.Ritchieによる「The C Programming Language」の付録Aの「The C Reference Manual」のみしかなかったことによる。すなわち、本書は、同一のCの文法をもつ処理系を実現できる程正確なものではなく、曖昧性を含んだものであったのである。しかも、その後の言語の成長のフォローも十分でなく、Cプログラミング上重要な「ライブラリ」の定義も含まれていない。

そこでANSIでは、Cの設計思想を尊重し、既存のユーザ資産を継承するという前提で、C言語の曖昧性を払拭し、マシンやオペレーティングシステムに依存しないプログラミングを可能にする正確な定義を作成するため、専門委員会X3J11を1983年に設置した。

ANSIの下には、コンピュータと情報処理の標準化を扱うX3という委員会が存在し、さらにその下にいくつかの技術委員会が存在するが、X3Jで始まる委員会がプログラミング言語の標準化作業を行っている(図1 X3委員会の組織参照)。

X3J11の作業は1983年の5月から開始されているが、委員会の作業は次の3つのセクションに分けて進められた。

① 環境(Environment)
② 言語(Language)
③ ライブラリ(Library)

②と③については、作業の基本となったドキュメントが存在し、②については、前述の「The C Reference Manual」であり、③については、/usr/groupによる「1984 /usr/group Standard」である。

X3J11では、これらのドキュメントをベースに、実在する多くの処理系も調査の上、ソフトウェアベンダ、コンパイラメーカ、技術者、学者等の意見を取り入れ、作業を進めてきた。X3J11は年4

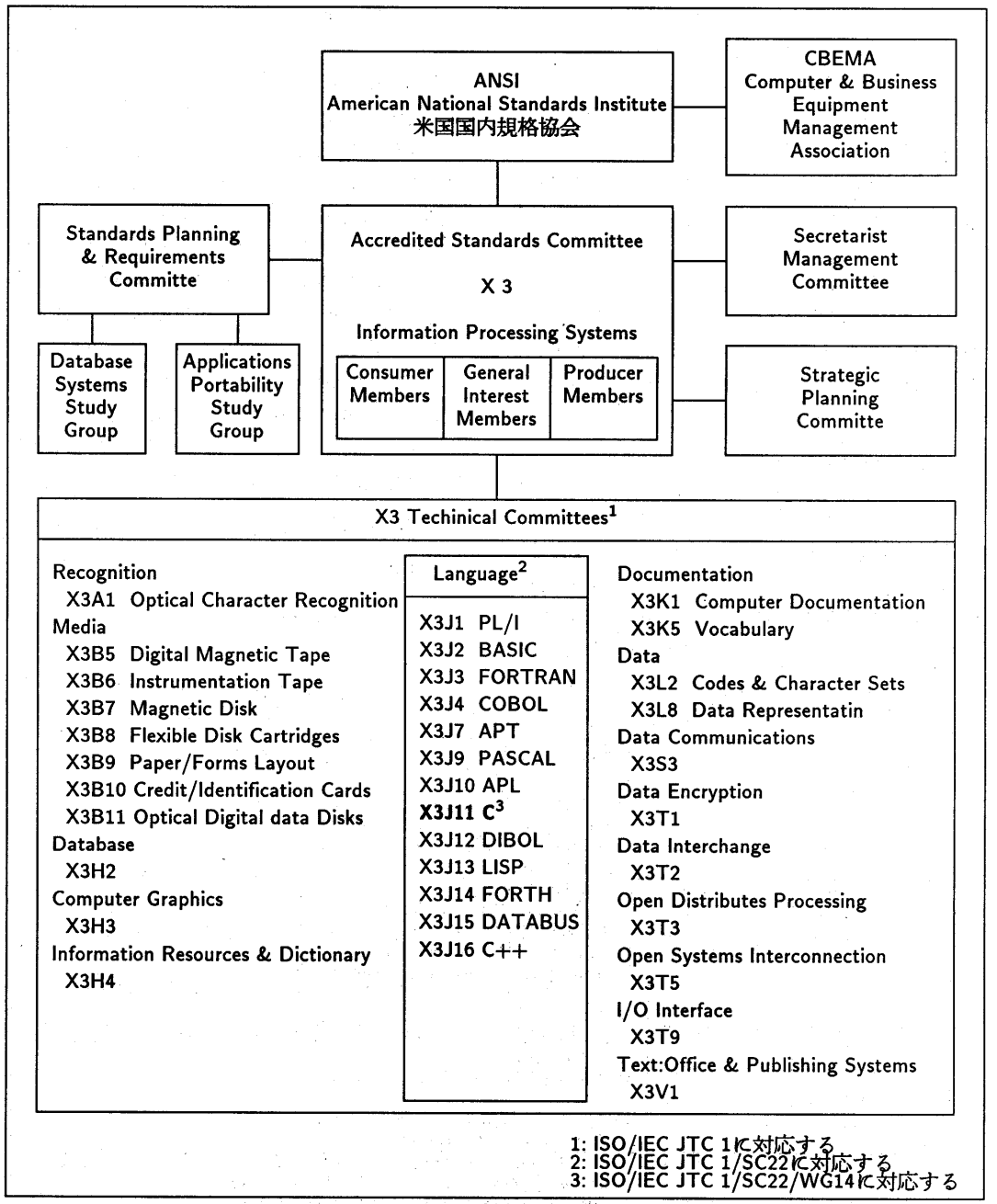
ANSI
American National Standards Institute
米国国内規格協会
CBEMA
Computer & Business Equipment Management Association
Standards Planning & Requirements Committe
Accredited Standards Committee
X 3
Information Processing Systems
Consumer Members
General Interest Members
Producer Members
Secretarist Management Committee
Strategic Planning Committe
Database Systems Study Group
Applications Portability Study Group
X3 Techinical Committees[1]
Recognition
X3A1 Optical Character Recognition
Media
X3B5 Digital Magnetic Tape
X3B6 Instrumentation Tape
X3B7 Magnetic Disk
X3B8 Flexible Disk Cartridges
X3B9 Paper/Forms Layout
X3B10 Credit/Identification Cards
X3B11 Optical Digital data Disks
Database
X3H2
Computer Graphics
X3H3
Information Resources & Dictionary
X3H4
Language[2]
X3J1 PL/I
X3J2 BASIC
X3J3 FORTRAN
X3J4 COBOL
X3J7 APT
X3J9 PASCAL
X3J10 APL
X3J11 C[3]
X3J12 DIBOL
X3J13 LISP
X3J14 FORTH
X3J15 DATABUS
X3J16 C++
Documentation
X3K1 Computer Documentation
X3K5 Vocabulary
Data
X3L2 Codes & Character Sets
X3L8 Data Representatin
Data Communications
X3S3
Data Encryption
X3T1
Data Interchange
X3T2
Open Distributes Processing
X3T3
Open Systems Interconnection
X3T5
I/O Interface
X3T9
Text:Office & Publishing Systems
X3V1
1: ISO/IEC JTC 1に対応する
2: ISO/IEC JTC 1/SC22に対応する
3: ISO/IEC JTC 1/SC22/WG14に対応する

図1 X3委員会の組織

回ほどのペースで開かれ、そこで検討されたドキュメントは「X3J11/年号−通番」という形式の文書番号を付けて管理されている。表1にANSI Cのドラフトの履歴をまとめている。

ANSIでのFormal Public Reviewにほぼ同期して、後述のJTC1/SC22においてもDP投票が実施され、 ANSI内部でのコメントはもちろん、各国からのコメントも反映されてきた。

1989年12月にANSIの BSR (Board of Standard Review) で正式に認可され、JTC1/SC22におけるDP 投票での日本、英国からのコメントおよびANSI内部のコメントを反映の上ANSI C規格として出版の運びとなった。

2.2 ISOにおける作業の経緯

プログラム言語の国際規格案造りは、ISO/IEC JTC1の下の分科会(Sub Committee)の一つであるSC22(事務局カナダ)で進められている。Cの国際規格造りの作業項目が1986年に承認され(作業項目JTC1.22.20)、SC22の下にWG14が発足した。

具体的作業の進め方としては、米国国内規格案と同期をとるために、X3J11が作成したANSI原案をそのまま活用し、国際規格案としてレビューおよび投票を行っており、その際の各国からのコメントはX3J11に送付され、回答を得るという形式で進められている。SC22内のDP投票(DP番号DP9899) は過去3回行われている。

1987年と1988年の投票は、ANSI規格案との非同期、品質的問題、マルチバイト機能の欠除等の理由で成立しなかった。1989年に3回目のDP投票があり (内容はX3J11/88-159と同じ)、4ケ国からのコメント付賛成があったものの、反対0で通過した。現在DIS投票中である(DIS9899)。

なお、1989年のSC22総会で、言語Cに関する作業項目を次の2つに分割することが決定された。

- JTC1.22.20.01:C言語(いわゆるANSI C規格と同一部分)
- JTC1.22.20.02:補遺

発行年月日	文書番号	備 考
1984年 8月21日	X3J11/84-131	ライブラリ仕様案なし、キーワードconst,enum, void,volatileあり、関数原型あり、long float, #elif,definedあり
1985年 2月11日	85-008	ライブラリ仕様案追加、キーワードsigned追加、#poragma,##追加
4月30日	85-044	インデックス追加
4月30日	85-045	45-004と同一内容(85-008との差分マークあり)
8月11日	85-102	memicmp(),stricmp(),strnicmp()追加
11月11日	85-138	bsearch(),qsort()追加、memicmp(),stricmp(),strnicmp()削除、Rationaleが付随される
1986年 2月14日	86-017	abs(),idiv(),ldiv()追加
5月 1日	86-074	idiv()をdiv()に変更、labs()追加
7月 9日	86-098	前処理の記述を大幅に修正,strtoul(),memmove(), mktime の追加
10月 1日	86-151	Localization仕様追加(locale.h,setlocale()), fgetpos(),fsetpos(),strcoll(),strftime()追加
10月 1日	86-157	1st.Formal Public Review用(86-151と同一内容,86-078との差分マークがない)
1987年 2月20日	87-019	非公式版(差分のある頁のみ)、strcoll()をstrxfrm()に変更、別機能のstrcoll()を追加
5月15日	87-116	非公式版(差分のある頁のみ)、errno.h追加
8月 3日	87-140	
11月 9日	87-221	型の記述が大幅修正、localeconv()追加、多バイト文字(wchar_t), 3関数追加 (mblen(),mbtowc(),wctomb())
1988年 1月11日	88-001	2nd. Formal Public Review用、多バイト文字列、 2関数追加 (mbstowcs(),wcstombs()),キーワードnoaliasあり
1月11日	88-002	88-001と同一内容(87-221との差分マークあり)
5月13日	88-090	3rd. Formal Public Review用、noalias削除
12月 7日	88-158	最終案、本質的な修正はなし
12月 7日	88-159	88-158と同一内容(88-090との差分マークあり)
1989年12月14日		ANSI BSRで正式に認可(公式版は88-158と若干編集上の差異がある)

表1 ANSIドラフトの履歴

後者は、イギリスからの未定義の動作(undefined behavior)等の明確化提案、オランダからの小さい文字セットでの表記に関する提案、日本からのマルチバイト機能の拡張の提案に対する作業であり、本年6月ロンドンで開催予定のWG14で討議されることになっている。

2.3 日本における作業の経緯

日本国内では、情報処理学会規格調査会(IPSJ/ITSCJ)の中に、JTC1の各SCに対応する専門委員会が設けられており、JTC1/SC22/WG14に対してはITSCJ/SC22/C WGが対応している。C WGは JTC1/SC22/WG14発足とほぼ同期して発足し、ANSI C規格案および国際規格案に対し審議を行い、コメントを提出すると共に、DP投票に際しては、日本のポジションに関する答申を行っている。

1988年までは、主として規格案に対する曖昧性の指摘/提案、マルチバイト機能に関する提案の作業を中心に行ってきたが、昨年は、現状の規格案におけるマルチバイト機能の不満足な点に対する拡張案作りに集中した。この拡張案を本年初めJTC1/SC22/WG14の方に、Cの規格本体の補遺として採用するよう提出したところである。

図2にANSI−ISO−日本の関係を示す。

3. ANSIC規格の主な特徴

紙面の都合上、ANSI C規格の解説する余裕もないし、又、本稿の主旨でもないので、簡単なANSI C の特徴のみを表2に示した。

4. 今後のCに関する標準化の話題

ANSI C規格も承認され、C言語本体の標準化は、国際規格の作業のみとなった。したがって、今後の作業は2.2で述べたように、補遺の作成と承認が主体となる。このうち、日本が提案している多バイト機能の拡張案について簡単に添れてみたい。

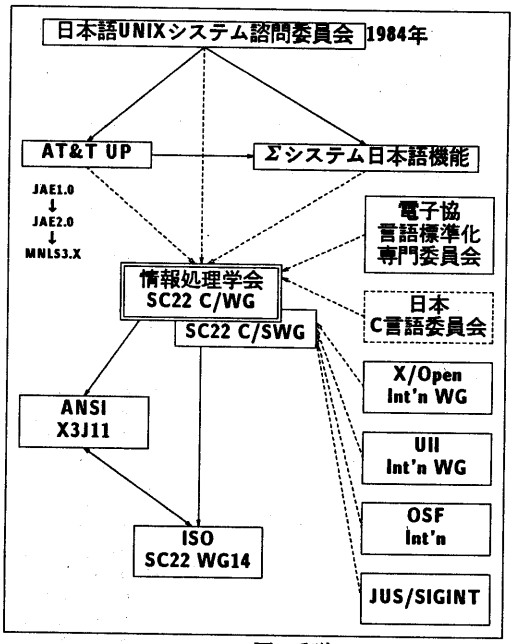

図3 系譜

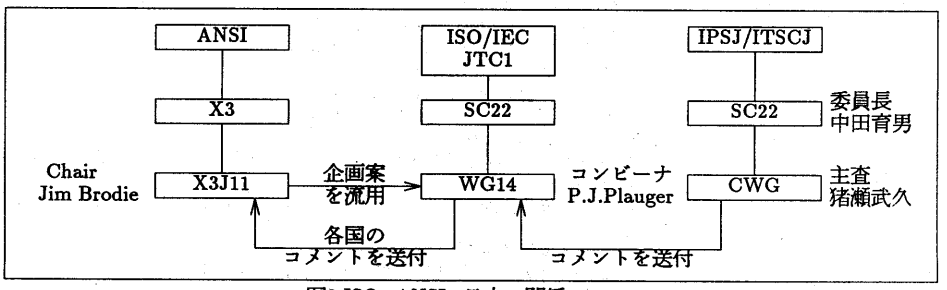

図2 ISO−ANSI−日本の関係

ANSI Cの特徴	説　明
1.環境	
(1) 2つの実行環境の定義	●フリースタンデング実行環境 ●ホスト実行環境
(2) 拡張文字の集合	多バイト文字も拡張可能
(3) 翻訳限界、数量的限界の導入	処理系が満たすべき各種入れ子レベルの最小数(翻訳限界)、各データ型の最小値、最大値等(数量的限界)を定義
2.言語	
(1) const修飾型、 volatile修飾型の導入	ex) `const int i;` `volatile int t;`
(2) 多バイト文字の記述 ワイド文字	ヘッダ名、文字列リテラル、コメント tpedef whar_t、ワイド文字定数、ワイド文字列リテラル、初期化
(3) 汎整数拡張の変更	値保存型を採用(注：K&Rでは符号なし保存型である)
(4) 算術型変換	float op floatはfloatで演算可(K&Rは常にdoubleで演算)
(5) 式の評価順序 かっこの再グループ分けの禁止	(3+f)+10をK&Rでは、伝統的にf+13のように再グループ分け可能であったが、結果が変わらないという保証がない限り禁止
(6) 関数原型の導入	ex) `int f(char,int,float);  f(a,b,c);`
(7) 構造体／共用体の代入	構造体/共用体同志の代入、引数は多くの処理系がサポートしている
(8) 可変個の引数を持つ 関数定義法の導入	ex) `int f(char format[],...);`
(9) 初期化の拡張	●共用体オブジェクトの初期化 ●autoな構造体／共用体の初期化他
(10) 列挙型、void型の導入	●`enum {first,second,third} order;` ●`void f();`

(11) 前処理指令の拡張	●#は必ずしも行の先頭でなくてよい
	●defined演算子 `#if defined (A)…`
	●#演算子 `#define x(y) #y`
	`x(abc) → "abc"`
	●##演算子 `#define x(a,b) S##a+S##b`
	`x(1,2) → S1+S2`
	等

3.ライブラリ	
(1) 標準ヘッダ	●<assert.h> assertマクロ ●<ctype.h> isalnum他の13個の関数 ●<error.h> ●<float.h> ●<limits.h> ●<locale.h> setlocale, localeconv ●<math.h> acos他 22個の関数 ●<setjmp.h> setjmpマクロ、longjmp関数 ●<signal.h> signal関数、raise関数 ●<stdarg.h>va_startマクロ他 3個のマクロ ●<stdio.h> fopen他の40個の関数 ●<stdlib.h> atof他の25個の関数 ●<string.h> memcopy他の22個の関数 ●<time.h> clock他の9個の関数
(2) ANSI Cで導入された関数	/usr/group Standardになく、ANSI Cで導入された関数 • 環境との情報交換 − atexit, system • 時間操作関数 − difftime, mktime, strftime • 算術演算関数 − div, labs, ldiv, • ファイル位置付け関数 − fgetpos, fsetpos • 地域制御関数 − setlocale, localeconv • 対数関数 − log10 • 多バイト文字/文字列関数 − mblen, mbtowc, wctomb, mbstowcs, wcstombs • 文字列操作関数 − memchr, memcmp, memcpy, memmove, memset, strcoll, strstr, strxfm • ファイル操作関数 − remove, rename • 書式付き入出力関数 − vfprintf, vprintf, vsprintf • 文字列変換関数 − strtod, strtol, strtoul • 可変個の実引数アクセスマクロ − va_start, va_arg, va_end

表2 ANSI Cの特徴

4.1 Cの多バイト処理機能の標準化の経緯

C言語の多バイト処理機能のルーツは、日本語UNIX仕様作成のためにAT&Tから委託を受けて1985 年に答申された日本語UNIXシステム諮問委員会 (委員長石田靖久)の報告書にみることができる (図3)。本委員会の諮問案は、AT&TのJAEやΣシステムの日本語機能の原型として採用され、ANSIおよびISOへの日本からの提案も本案をベースに提案を行ってきた(表3)。

しかし現状の規格は「多バイト文字を "1文字" として扱いたい」という日本の意図に対しては、不十分なものとなっている。そこで改めて昨年から、現規格案をベースに、多バイト処理ライブラリの拡張案作成にとりかかり、図3に示すように多くの関連団体の協力も得てその拡張案を今年初めJTC1/SC22/WG14に提案したところである。

なお、われわれの拡張案作りに当たっての基本方針としたものは、次の通りである。

(1) 現在のANSI Cの多バイト処理機能の仕様は尊重する。

(2) コード系独立のANSI Cの設計方針は尊重する(ISO2022符号拡張系のようなシフトエンコーディングの世界もカバーする)。

(3) その上で「1文字」を「1単位」として扱えるよう、 wchar_tのライブラリを拡張する。

(4) 仕様の決定、対策の検討等に当たっては、その理由を示す理由書(rationale)を残し、本体と対で公開する。

4.2 ANSI Cの多バイト処理機能の現状と日本からの拡張案

現状のANSI C規格には、(typedef名)wchar_tが導入され、1文字が何バイトから構成されているか等を知ることなしに文字を同一長のオブジェクトとして扱うことができ、我々の基本的要求は達成されている。しかしながら、ワイド文字から/への入出力ライブラリやワイド文字/文字列処理ライブラリは導入されていないため、ASCIIのような1文字が1バイトからなる世界に比べると、まだまだ不十分な仕様となっている(図4参照。一重枠が現状、二重枠が拡張が必要な部分)。

例えばstrstr関数の例を挙げてみよう。これは、第1アーギュメントの文字列の中に第2アーギュメントの文字列を構成するバイトの集合(サブセットでもよい)からなる、部分列を探す関数である。

例えば図5に示したような文字列があったとして、 S2で「日」という文字列を指定したとしよ

1983年	• ANSI X3J11設置
1985年	• 日本語UNIXシステム諮問委員会報告書
	• Σプロジェクト発足
1986年	• ISO SC22/WG14発足
	• 情報処理学会 SC22 C WG発足
	• X3J11国際化機能Adhoc委員会に日本語UNIXシステム諮問委員会案を提案
1987年	• ISO C言語のDP登録投票(日本反対)
	• 日本C言語委員会、X3J11に日本語機能の提案
	• X3J11 多バイト機能の追加開始
	• SC22総会プログラム言語の多バイト機能の標準への盛り込み勧告決議
	• C WG 2度に渡って、日本案をSC22/WG14及びX3J11に提案
1988年	• ISO SC22 CのDP投票(日本反対)
	• X3J11 多バイト機能を含んだ案(X3J11/88-001)のpublic review
	• 同上に対して、日本のコメント提出
	• コメント反映版(X3J11/88-090)のpublic review
	• 同上に対して、日本のコメント提出
	• X3J11/88-159(ANSI最終案)
1989年	• ISO SC22 CのDP投票(条件付き賛成)
	• 同上の結果、反対0で現在DIS投票中(6月21日締切)
	• C WGで、多バイト機能の拡張案作成
1990年	• 日本の多バイト機能拡張案をSC22/WG14に提案

表3 Cの多バイト機能標準化の経緯

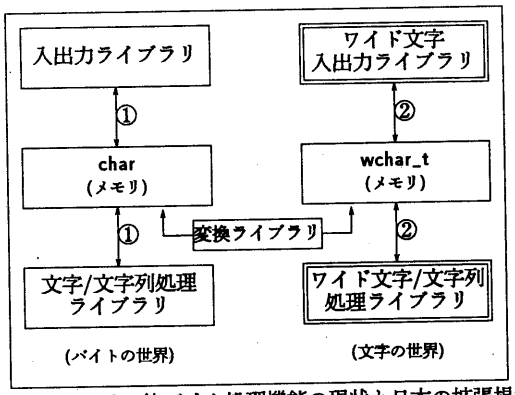

図4 ANSI Cの他バイト処理機能の現状と日本の拡張提案

処理内容	関数名
ワイド文字テスト関数	iswxxx
ワイド文字変換関数	towxxx
ワイド文字クラス関数	set_wctype, is_wctype
ワイド文字入出力関数	getwc/putwc群
ワイド文字フォーマット付入出力関数	printf/scanf群
ワイド文字コピー関数	wcscpy/wcscat群
ワイド文字比較関数	wcscmp群
ワイド文字検索関数	wcschr群
ワイド文字日時関数	wcsftime

表4 日本からの提案

う。これは分解すると、2バイト系ならば、b1とb2というバイトからなっている。ところが、文字列全体の構成を見ること、ここに示したb1b1b1という部分列も「日」を構成する最初のバイトに一致することになる。従って、この b1 のみからなる部分文字列も検索対象になってしまうのである。

つまり、「日」という単位とは認識されず、b1と b2からなる列としてしかみなされていないのである。そこで文字を文字としてきちんと認識する strstrに対応の関数wcswcsを導入する。

この関数だと、図5の列の場合はwcharストリングとみなしてくれる。例えば、ここでの例だと2バイトで1単位を構成しているとして、「日」と指定したら、今度はちゃんとした単位の1ユニットだと認識する。

このような考え方のもとに、日本からの提案に含まれる関数群を表4にまとめている。

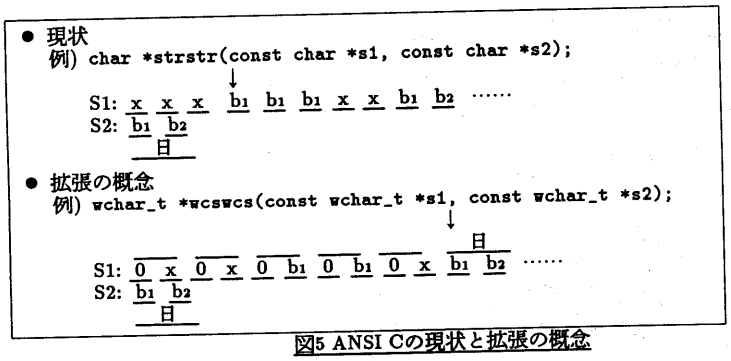

図5 ANSI Cの現状と拡張の概念

5. おわりに

以上、C言語の標準化動向について述べた。ANSI C規格も成立し、国際規格もほぼ同一内容のものが承認されるものと思われる。今後はJTC1/SC22/WG14において、補遺についての審議に中心は移る。

ANSI C準拠の処理系も出まわり始め、X/Openや UII、OSFといった業界団体もANSI Cの採用を表明している。従って、今後ANSI Cが急速に定着、普及していくものと思われる。

最後に、日頃熱心に検討していただいているITSCJ/SC22/C WGおよびC SWGの皆様に改めて感謝したい。