CTRONインタフェース仕様に準拠したシステム開発におけるオブジェクト指向プログラミングの適用

3F−8

田畑　政雄[1]、斎藤　賢爾[1]、林　朋典[2]、佐藤　由紀子[2]

[1]日立ソフトウェアエンジニアリング（株）　[2]（株）日立製作所ソフトウェア開発本部

1．はじめに

筆者らは、日立版のCTRONインタフェース仕様準拠OSの上で動作する通信サーバ・プログラムの開発プロジェクトにおいて、ソフトウェア製品の従来の再利用の際に生じる問題点の解決を目的としてオブジェクト指向プログラミング（OOP）[1][2]を導入し、リアルタイム処理分野のための、高度に組み合わせが可能なソフトウェア製品の部品の設計を試みている。

本稿では、我々がいかにCTRONインタフェースに基づくシステムにOOPを適用したかについて報告する。

2．背景

2．1　従来の再利用における障害

ソフトウェアの従来の再利用には、次に示すように2つの目立った障害がある。

（a）OSインタフェースの違い

高級言語インタフェースがOSインタフェース間の差を隠蔽できたとしても、いくつかの応用プログラムはOSインタフェースを直接使用する必要がある。

現存するOSインタフェースは多くの場合システムにより異なるため、目的のシステムのために開発されていないこのようなプログラムを再利用することはいささか困難となる。

（b）高級言語の抽象化単位におけるモジュラリティの欠如

構造化プログラミングをサポートする多くの高級言語において、主となる抽象化の単位は手続きである。PascalやCなどの言語では、データ抽象はデータ型の定義という形でサポートされ、手続きは特定の型に属するデータを扱うように設計されなければならない。これは、あらゆる手続きはその処理するデータ型に依存することを意味する。更に悪いことに、データの内部構造は常にあらゆる手続きから見えている。これは、これらの抽象化の単位が情報隠蔽の能力に欠け、組み合わせにくくなっていることを示している。

2．2　オブジェクト指向

カプセル化による適度な情報隠蔽は、プログラマが既存のオブジェクトを組み合わせて容易に新しいシステムを構築することを可能にする。また、継承はプログラマがオブジェクトを新しいシステムに適用させるために、既存の定義から少し異なる新しいオブジェクトを定義することを可能にする。ポリモルフィズムはオブジェクト間のインタフェースの数を減らし、インタフェースを標準化する。

これにより、オブジェクトは高度な組み合わせが可能となり、前節で示した問題（b）を解決する。

2．3　C++のCTRONインタフェースへの適用

（1）C++選択の理由

筆者らは、次の利点があることからOOP言語の内、C++[3]をプロジェクトに導入した。

（a）ポータビリティ

既存の多くのC++コンパイラはCのソースコードを出力できるため、C言語を流通言語としているCTRONプロジェクトでのC++で書かれたプログラムのポータビリティには大きな問題はないと考えられる。

（b）既存の環境の再利用性

C++はC言語のスーパーセットであるため、筆者らがC言語による日立版CTRONカーネルの開発のために用意した環境（インクルードファイル、ライブラリ関数、開発ツール等）を再利用することができる。

同じ理由により、カプセル化とポリモルフィズムをサポートし、CTRONプロジェクトで使用可能な言語であるにかかわらず、報告者らはAdaを導入することを考慮しなかった。

（c）OOP言語の中での効率の良さ

一般的に、OOP言語ではメッセージパッシングと動的なポリモルフィズムの効率の悪さが問題となる。

C++におけるメッセージパッシングは関数呼び出しとして実現され、Cのような従来の言語を用いるのと比較しても性能上劣らない。そして多くの場合、ポリモルフィズムは静的に解決される。動的なポリモルフィズムも、関数へのポインタを用いることにより、要求されるルーチンを検索する時間が削減されるため、比較的安価である。それに加え、C++では、関数呼び出しのような機構でも不経済と判断される場合に備え、inline関数がサポートされている（詳細は5．1　inline関数の使用で説明する）。

（2）問題点

（a）C言語と比較した場合の効率の悪さ

C++で書かれたプログラムは他のOOP言語で書かれたものと比較すると効率がよいかも知れないが、Cで書かれたものと比較すると依然性能が劣っている。具体的には、仮想関数は、単なる関数へのポインタよりもコスト高のように見える。

（b）ライブラリの欠如

オブジェクトの再利用を推進するため、OOP環境は基本クラスのソースとして、クラスライブラリと呼ばれるクラスの集合体を持つ必要がある。既存のC++のクラスライブラリはリアルタイム応用に向いていないか、少なくともリアルタイム・オペレーティングシステムの使用を前提として設計されていない。

このため、筆者ら自身の手でリアルタイム応用向けのクラスライブラリを設計する必要性があった。

3．基本クラスライブラリ

3．1　基本クラスライブラリの概要

クラスライブラリはそこから派生クラスを定義する基本クラスの集合であることから、筆者らのプロジェクトでは基本クラスライブラリと呼んでいる。

筆者らはその設計において次の方針を設定した。

（1）様々な応用のために必要となるクラスを揃える。

（2）記号処理やテキスト処理のためのクラスを定義することにより、応用プログラムのためのコマンドインタプリタとエディタの機能を標準化する。

（3）ANSI−C仕様で定義されるライブラリ関数により仮想化されるものを除くすべてのCTRONインタフェースを仮想化することにより、基本クラスライブラリを使用するプログラムを他のインタフェース・サブセットやインタフェース標準に基づくシステムに移植し易くする。また、システム資源の派生クラスを定義することによりシステムの能力を拡張することを可能にする。

必要なクラスの種類は応用によって異なるため、筆者らは特定の応用分野をサポートするいくつかのライブラリを作成している。表1にライブラリの一覧を示す。

表1　基本クラスライブラリの一覧

No.	ライブラリ名	適用分野
1	基本クラスライブラリ1	すべての応用分野に共通
2	基本クラスライブラリ2	記号処理
3	基本クラスライブラリ3	テキスト処理
4	基本クラスライブラリ4	リアルタイム応用
5	基本クラスライブラリ5	ネットワーク通信

3．2　基本クラスライブラリ4

ここでは2．1で述べた問題点の解決に関連する基本クラスライブラリ4について解説する。

（1）目的

基本クラスライブラリ4は、典型的なリアルタイムカーネルで用意されているシステム資源を提供する。その目的は次を包括している。

（a）システム資源をクラスとして定義し、プログラマがそれらの派生クラスを定義してクラスライブラリを拡張することを可能にすることにより、リアルタイムシステムの生産性を向上する。

Applying Object Oriented Programming to Developing of System based on The CTRON interfaces

Masao TABATA[1], Kenji SAITO[1], Tomonori HAYASHI[2], Yukiko SATO[2]

[1]Hitachi Software Engineering Co., Ltd.　[2]Software Development Center, Hitachi, Ltd.

（ｂ）ＣＴＲＯＮカーネル・インタフェースで提供されるシステムコールをすべて仮想化し、本ライブラリを使用するプログラムを移植する際にプログラマがシステム資源のクラスを再定義することを可能にすることにより、リアルタイムシステムのポータビリタイを向上する。
（ｃ）プログラマがシステム資源をデータ型としてアクセスすることを奨励することによりリアルタイムシステムの信頼性を向上する。
（ｄ）高級言語インタフェースとして新しい操作を追加することにより、ＣＴＲＯＮカーネルの機能を拡張／制限する。

（２）内容
本ライブラリはタスク、イベントフラグ、セマフォ、メッセージボックス、メモリプール、メモリブロックなどのシステム資源を仮想化するクラスを含んでいる。これらのクラスはＣＴＲＯＮカーネルインタフェースのＣによる言語バインディングを用いて実装されている。これらの資源を操作するシステムコールは各々のクラスのメンバ関数（大多数がｉｎｌｉｎｅ関数）として表現されている。しかし、筆者らはすべてのメンバ関数が完全にＣＴＲＯＮカーネルのシステムコールに従うようには設計しなかったし、すべてのクラスがＣＴＲＯＮカーネルの資源を代表するようにも設計しなかった。その理由とは、

（ａ）インタフェースを仮想化することは、必ずしもそれに忠実であることではない。ポータビリティの向上のためには、ＣＴＲＯＮ仕様に特有な部分は隠蔽した方が得策である。
（ｂ）資源の分類がＯＯＰと馴染まない部分がある。図１に示すように、現在のＣＴＲＯＮカーネルのメッセージボックスには、２つのモード、ロケートモードとムーブモードがある。これら２つのモードに対して、ＳＥＮＤのシステムコールは共通であるが、ＲＥＣＥＩＶＥは異なるシステムコールとして提供されている。即ち、同じメッセージ（この場合はシステムコール）に対して２つのモードでは動作が異なっているのである。
ＯＯＰではこのような場合、これら２つのモードは別々のクラスに属するオブジェクトと見なすのが自然である。

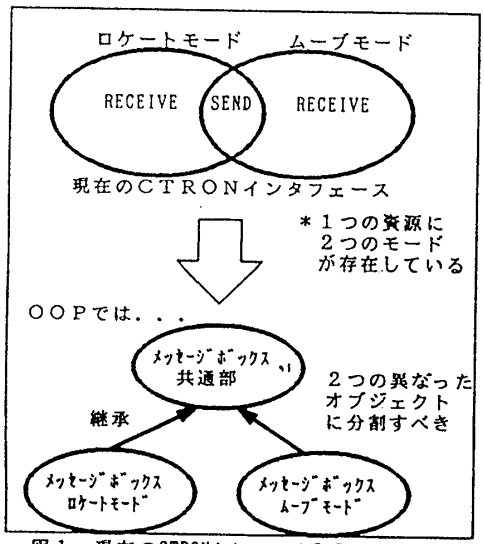

図１　現在のCTRONｲﾝﾀﾌｪｰｽでＯＯＰと馴染まない部分

（ｄ）オペレーティングシステム・インタフェース間の差の隠蔽
これまで述べてきたようなシステム資源の仮想化は、プログラムを他のリアルタイム・オペレーティングシステムに基づくシステムに移植することを容易にする可能性がある。
１つの例は、異なるＴＲＯＮ仕様間でのポータビリティである。本ライブラリを実際にＩＴＲＯＮ２カーネル上で動作するように書き換えるのは難しくないと考えられる。ＩＴＲＯＮ２仕様とＣＴＲＯＮ仕様の際だった違いの１つは、ＩＴＲＯＮ２ではオブジェクトはＩＤ番号により識別されるということである。しかし、ＩＤ番号はＣＴＲＯＮ仕様の資源名によって代用が可能である。
これを実現し、かつプログラマが資源の名称を指定することを奨励するために、資源名の指定は筆者らのライブラリでは必須となっている。勿論、Ｃによる従来の言語バインディングが要求するように、ＮＵＬＬＩを指定することにより名称を省略することが可能である。
他の例として、ＣＴＲＯＮ仕様に基づく既存のオペレーティングシステムの１つは[4]、ＣＴＲＯＮカーネルのμＣサブセットをより豊富な機能を持つオペレーティングシステムの上に実装することができることを示している。従って、本ライブラリと等価なライブラリをそのようなオペレーティングシステム上に構築することは可能なはずである。このアプローチは、５．１ｉｎｌｉｎｅ関数の使用で提示する理由により、ＣＴＲＯＮカーネルを実装するよりもよい性能を示す可能性がある。

５．性能の向上

Ｃ＋＋で書かれたプログラムの性能を向上する方法として次の性能向上策を行った。

（１）小さな関数に対してインライン展開を適用する。
（２）仮想関数をなるべく使用しない。

本章ではＣ＋＋で書かれたプログラムの性能を向上するための上記の方法を解説する。

５．１　ｉｎｌｉｎｅ関数の使用

ｉｎｌｉｎｅ関数はＣのプリプロセッサコマンドとしてサポートされているマクロ展開と似ているが、次の点でそれよりも優れている。

（１）ｉｎｌｉｎｅ関数には副作用がない。
（２）ｉｎｌｉｎｅ関数は関数プロトタイプ宣言を持つ。
（３）ｉｎｌｉｎｅ関数は局所変数を持つことができる。
（４）ｉｎｌｉｎｅ関数はメンバ関数となることができる。

しかし、関数のｉｎｌｉｎｅ宣言は、Ｃ＋＋ではコンパイラに対するヒントにしか過ぎない。これは、コンパイラによっては、ある条件でコードをインラインに展開しない場合があることを意味している。これを避けるために、筆者らはライブラリ中のｐｒｉｖａｔｅなデータにアクセスするメンバ関数のいくつかを削除し、該当するデータをｐｕｂｌｉｃあるいはｐｒｏｔｅｃｔｅｄなデータとすることによってｉｎｌｉｎｅ関数のネストのレベルを下げた。これは勿論、ＯＯＰの指針に反する。しかし、該当するデータに対する参照と更新の両方をｐｕｂｌｉｃなメンバ関数が行う場合に限ってこれを適用するようにしたことにより、筆者らはこの違反をライブラリの信頼性や保守性に影響がないレベルに抑えることができたと考えている。
ｉｎｌｉｎｅ関数を積極的に用いることにより、抽象化によるオーバヘッドをかなり削減することができる。

５．２　動的束縛の回避

仮想関数の使用はオブジェクトの組み合わせ易さを向上するが、その機構はやや経済性に欠けるといえる。筆者らはライブラリの設計にあたって次の方針を設けることにより、仮想関数の使用を極力避けるようにした。

（１）取り替え可能なオブジェクトが１つのシステムに混在する必要がない場合、通常のメンバ関数を用いる。
（２）関数が”ｔｈｉｓ”（Ｃ＋＋で暗黙的に宣言された対象となるオブジェクト）を必要としない場合、関数へのポインタを用いる。
（３）上記の条件が成立しない場合、仮想関数を用いる。

仮想関数の代わりに関数へのポインタを用いることは、構築子に余分なコーディングを要するが、組み合わせ易さには影響しない。

６．おわりに

現在、本稿で述べた基本クラスライブラリを使用して通信サーバ・プログラムを開発中である。
該プログラムを完成させた後、各基本クラスライブラリの再利用率および性能を測定する予定である。

参考文献
(1)Bertrand Meyer：Object-Oriented Software Construction：Prentice Hall Intl.(1988年)
(2)Anthony I. Wasserman, et al：Object-Oriented Structured Design and C++：
Computer Language, January 1991, pp. 41-52 (1991年)
(3)Bjarne Stroustrup：The C++ Programming Language：Addison-Wesley Publishing Company (1986年)
(4)Alfred Chao：Realization of the Micro-CTRON Kernel under pSOS+：TRON Project 1990, Springer-Verlag, pp. 427-431 (1990年)