オペレーティング・システム 32－2
（１９８６．９．１２）

並列処理機能をもつプログラミング言語 ＣＨＩＬＬによるＯＳの記述

久保田　稔
ＮＴＴ　電気通信研究所

多重実時間処理の観点から交換プログラムの概要、交換プログラム用ＯＳ（ＣＣＯＳ：Concurrent CHILL Operating System）のアーキテクチャと実現法について述べる。またこれらの開発に用いられたＣＨＩＬＬの記述性について述べる。本稿では交換プログラムは一般の計算機用ＯＳに相当する処理を行い、ＣＣＯＳはそのカーネルと位置づけている。ＣＨＩＬＬは、多重処理の記述に適したプロセスの生成、同期通信、排他制御を記述するための並列処理機能をもつ高水準プログラミング言語である。ＣＣＯＳはこの並列処理機能の実行と、ＣＨＩＬＬだけでは記述しにくい周期処理や時間依存の処理のような実時間制御機能をサポートする。

OS IMPLEMENTATION USING CONCURRENT PROGRAMMING LANGUAGE CHILL

Minoru KUBOTA
NTT Electrical Communications Laboratories
3-9-11, Midori-cho, Musashino-shi, Tokyo, 180 Japan

From the multiple real-time processing point of view, the structure of switching programs and the architecture and implementation of Concurrent CHILL Operating System(CCOS) which supports switching program execution, are described. Then the suitability of CHILL for writing these programs is evaluated. In this paper, a switching program is considerd as an OS in the case of general-purpose computers and CCOS as its kernel. CHILL is a programming language that has concurrent facilities such as process creation and termination, inter-processs synchronization and communication, and mutual exclusion. CCOS provides the functions for both CHILL concurrent facilities, and real-time facilities, such as cyclic program invocation and time-dependent processing.

1．はじめに

通信システムの中核の一つである交換機は、有限のリソースを用いて複数の呼を同時に扱う一方、サービス基準を満たすため、接続要求に対して一定時間以内にこれを処理する必要がある。すなわち呼の発生を検出し接続を行う交換プログラムは、多重実時間制御により実行されなければならない。

一方、プログラムの生産性と保守性の向上を狙いとして、交換プログラムを高水準プログラミング言語で記述することが不可欠である。このため並列処理機能をサポートする高水準プログラミング言語ＣＨＩＬＬ[1]のコンパイラが新たに開発された[2],[3]。

ＣＨＩＬＬ並列処理機能を用いれば多重性をもつ交換プログラムを高水準言語で簡潔に記述できる。しかし装置の監視や装置駆動時のタイムアウト処理といった時間に依存する処理を記述するための機能は、ＣＨＩＬＬに含まれていない。

そこで、ＣＨＩＬＬ並列処理機能の実行と実時間制御機能をサポートする交換プログラム用ＯＳ（ＣＣＯＳ：Concurrent CHILL Operating System）を開発した[4]。サービス処理を除く交換プログラムは、装置の管理や駆動を行うため一般の計算機用ＯＳに相当し、ＣＣＯＳはそのカーネルである、とみなすことができる。本稿では多重実時間処理の観点から交換プログラムの概要、ＣＣＯＳのアーキテクチャと実現法について述べる。またこれらの開発に用いられたＣＨＩＬＬの記述性について述べる。

2．交換プログラムにおける多重実時間処理

交換機のハードウェア構成例を図１に示す。交換プログラムは、その処理の論理レベルにより次のようなプログラムに分けてモデル化できる。

(1) 呼状態管理・制御：　呼が行うサービスと呼の状態の管理と制御を行う。

(2) 呼接続：　呼に係わる回線や通話路の接続を行い、その状態を管理する。

(3) 装置管理：　呼を形成する各装置の状態の管理を行う。

(4) 装置監視・駆動：　回線や通話路等各装置の状態監視と駆動制御を行う。たとえば回線に信号が送られてきたかどうか監視し、通話路装置等を駆動し、他の交換機へ信号を送信する。

交換プログラムは次のような基本要素となる処理の組み合わせを実行する。

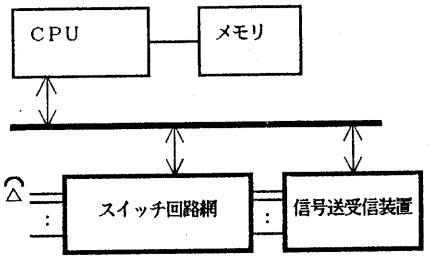

図1．交換機のハードウェア構成例

(a) 監視プログラムが回線を監視し、回線からの信号を検出すれば呼状態管理・制御プログラムを起動する。

(b) 呼状態管理・制御プログラムが信号を受信・分析して行うべき接続処理を決定し、呼接続プログラムと装置管理プログラムにより回線の接続を行うため装置の確保を行う。

(c) 装置駆動プログラムが通話路装置を駆動して回線及び信号送受信装置を接続して、信号の送受信を行う。

交換処理プログラムのモデルを図２に示す。上記の(1)、(2)は階層化の観点から分割したが、実行上は連続した一つのまとまった処理となるので、以後これらをまとめて呼制御処理あるいは単に呼処理と呼ぶ。

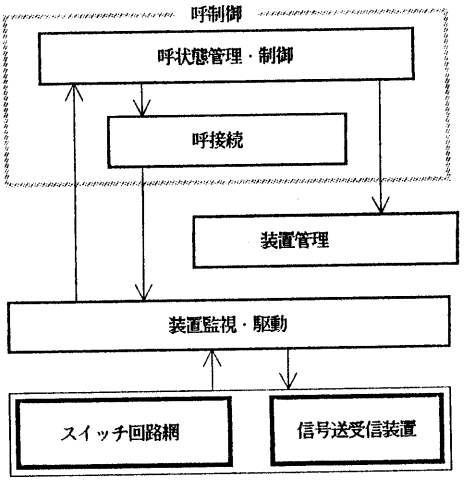

図2．交換プログラムのモデル例

３．ＣＣＯＳの多重実時間制御機能

３．１　多重処理制御

３．１．１　ＣＨＩＬＬ並列処理機能

ＣＨＩＬＬ並列処理機能は、複数のプロセスを並列に実行させる処理を簡潔に記述するための言語機能である。これを用いて記述されたプログラムの実行を制御すれば多重処理を実現できる。表１に本稿で対象としたコンパイラ[2],[3]でサポートしているＣＨＩＬＬの並列処理機能の概要を示す。

ＣＨＩＬＬ並列処理機能の文は、コンパイラにより実行時ルーチンの呼び出しに変換される。ＣＣＯＳでは、(a)プロセス実体のためのメモリ管理プログラム、(b)プロセスのスケジューラ、(c)プロセスの生成・消滅、同期・通信処理を行う実行時ルーチン、の三つによりＣＨＩＬＬの並列処理機能をサポートし、多重処理の制御を行う。プロセス実体は処理単位毎の作業用メモリをプログラムに割り付けたものである。

３．１．２　並列処理機能による交換プログラムの記述

ＣＨＩＬＬ並列処理機能の記述例を図３に示す。

（１）プロセスとする処理の単位

一つの呼は他の呼と独立にかつ並列に実行される。したがって一つの呼に対応させて一つの呼制御プロセスを割り付ければ良い。

一方、監視プログラムでは、回線の変化を端子毎に処理するとオーバヘッドが極めて大きくなるため、周期的に端子を監視し一つの周期に検出された信号をまとめて処理する方式をとっており[5]、複数の呼で共有される。この場合監視プログラムは呼制御プロセスと独立したプロセスとなる。同様に装置駆動プログラムも複数の呼制御プロセスが共有する独立したプロセスである。

（２）プロセスの同期と通信

呼制御プロセスと、装置の監視・駆動用プロセスは独立に動作可能であるが、一つの呼に関連するプロセスの間では同期をとり、通信を行う必要がある。このため回線に対応する論理端子はイベント変数あるいはバッファ変数で記述する。同期はdelay文とcontinue文、通信はsend文とreceive文で記述する。

（３）排他的アクセス

各プロセスは共通リソースであるスイッチや送受信の装置を用いて呼の接続を行う。このため制御プログラムでは、共通リソースに対する複数プロセスの間の競合を避けるために、regionモジュールを用

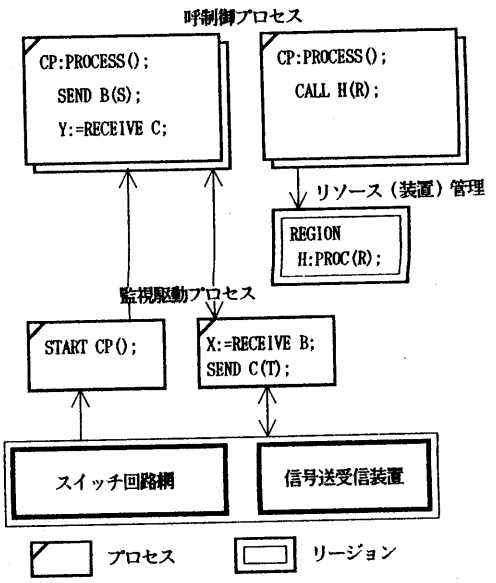

図３．ＣＨＩＬＬ並列処理機能を用いた交換プログラム例

表１．ＣＨＩＬＬコンパイラ並列処理機能

機　能		説　明
プロセスの定義	process文	プロシージャと同様の形式でプロセスの処理を記述する。
プロセスの生成と終了	start文	指定されたプロセスの定義に対して、プロセス実体を生成する。
	stop文	プロセス実体を消滅させる。
同期と通信	イベント変数	プロセス間の同期をとるための変数。
	delay 文	事象（イベント変数で表現される）が発生するまでプロセスの実行を中断させる。
	continue文	事象発生待ちで中断中のプロセスの処理を再開させる。
	バッファ変数	プロセス間の同期とデータ転送用の変数。
	send文	データをバッファに送る。
	receive文	バッファ内のデータを取り出す。
排他的アクセス	regionモジュール	複数のプロセスによって共有されているデータの排他的アクセスを行う。

いて記述する。リソースの確保ができない場合に確保要求の順序を制御するためにはイベント変数とdelay文とcontinue文を用いる。

３．２　実時間処理制御

交換プログラムでは、多数の装置に対する駆動要求を一定時間内に検出し処理しなければならない。またこのような実時間的な要求はその程度に応じて多くの種類があるため、ＣＣＯＳではこれらにあわせて制御する機構を提供している。

３．２．１　周期処理

実時間性の厳しいハードウェア事象に対する応答を保証するためＣＣＯＳは、周期的スケジューリング、すなわち実時間性の厳しい処理を行うプログラムを定められた時間間隔で優先的に起動する機構を提供している。これは優先度の高い処理を周期的にまとめて実行する（群処理）ことにより、プロセスの切り替えを少なくし、オーバヘッドの増加を防止するためである。

周期起動プログラムは回線の監視やスイッチの制御に用いられる。また通常は入出力割り込みをマスクしておき、周期的に割り込みの有無を調べて（ルックイン方式）、その周期の間にあった割り込みをまとめて処理するためにも用いている。すなわち割り込み処理プログラムは周期起動される。

周期起動プログラムは優先度を高くしてＯＳのレベルで実行されるべきであるが、その内容はアプリケーション毎に異なる。そこで図４に示すように、これらのプログラムをＣＣＯＳの実行制御表に登録することによって、ＯＳとアプリケーションの独立性を高めている。アプリケーション側はシステム初期設定時に必要なプログラムを登録する。

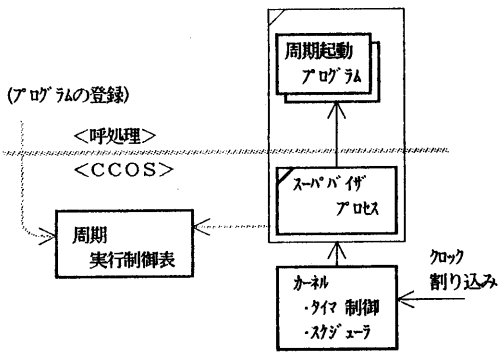

図4．表駆動プログラム制御

３．２．２　時間と時刻に依存した処理

ハードウェアの動作時間は一般にプログラムの実行時間よりも極めて長い。したがってプロセスは装置の処理と同期をとるため、一定時間の間処理を中断しなければならない。またこの場合、一定時間以内に信号が送られてこなければ、これを装置の異常として処理しなければならないため、プロセスの中断とあわせてタイミングをとる必要がある。さらに交換プログラムでは呼処理以外に運用管理の処理のため特定の時刻にプロセスを起動したり、一定時間間隔でプロセスを起動する機構が必要となる。ＣＣＯＳは以下に示す機能を提供する。

（１）処理の中断

一定時間、あるいは指定された時刻までプロセスの実行を中断させる。これは時間や時刻に依存する処理の記述に効果がある。

（２）時間制限付きプロセス間通信

プロセス間通信の受信待ちの際に、待ち時間の計測を同時に行う（タイムアウト処理）。この機能を用いると、プロセスは、指定された時間以内に信号データが送られてこなければ、時間切れの信号を受け取ることができる。ＣＣＯＳではこれを、(a)タイムアウトを通知するイベントあるいはバッファをタイムアウト値と共にＯＳの制御表に登録し、(b)プロセスをこのタイムアウト通知用イベント（バッファ）からの信号と本来受信すべき信号に対し同時に受信待ちとする、ことにより実現している。

（３）ＣＨＩＬＬによる実時間処理の記述

ＣＣＯＳでは時間遅延機能、タイムアウト機能をＣＨＩＬＬのプロシージャとして提供し、実時間機能の記述を可能としている。

図５に時間と時刻に依存した処理の記述例を示す。

４．　ＣＣＯＳのプログラム構造と実現法

４．１　ソフトウェア階層

ＣＣＯＳは図６に示すような３つの階層から構成されている。

（１）最も下位の階層のカーネルは、すべてのアプリケーションに共通なプログラムからなる。こ

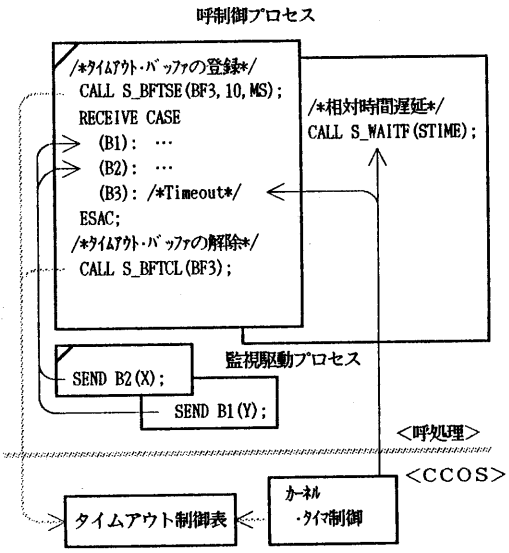

図5．CCOSにおけるタイマ処理

れらのプログラムは、クロック割り込み処理、メモリ管理、プロセスのスケジューリングを含むプロセス管理を行う。

（２）２番目の階層は最も高い優先度を持つスーパバイザプロセスであり、周期処理を含むタイマ処理、割り込みのルックイン処理等の実時間性の厳しい処理を行う。

（３）３番目の階層は制御プログラムとＣＨＩＬＬで書かれた呼処理プログラムとのインタフェースである。これは、(a)ＣＨＩＬＬ並列処理機能の実際の処理を記述した実行時ルーチン、(b)並列処理機能で記述できないＯＳ機能を呼び出すためのＯＳプリミティブ、のエントリからなる。ＯＳプリミティブはＣＨＩＬＬのプロシージャ呼び出しとして記述される。表２にＣＣＯＳが提供する主なＯＳプリミティブを示す。これらにより、実時間処理向きの記述ができるようにしている。

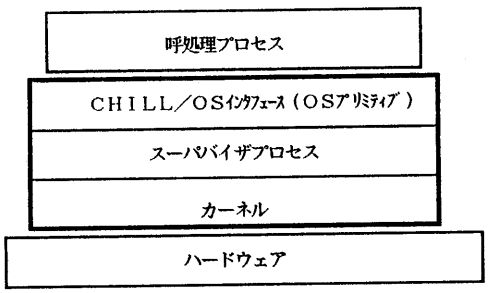

図6．CCOSの階層構造

上に述べたＣＣＯＳの上で、並列処理をもつＣＨＩＬＬで書かれた呼処理プログラムが実行される。

４．２　プロセス階層

ＣＣＯＳは図７に示すように、スーパバイザプロセス、周期起動プロセス、動的プロセス、という３つのタイプのプロセスの実行を制御する。動的プロセスのみが動的に生成・消滅し、それ以外は常駐である。またプロセスは仮想的（マルチプロセッサ・システムでは物理的）に並列に実行される。スーパバイザプロセスは２番目の階層に属し、周期起動プロセス及び動的プロセスは最も上位の階層に属する。

周期起動プロセス及び動的プロセスはＣＨＩＬＬ並列処理機能を用いて記述される。これらは優先度順に起動されるが、クロック割り込みの発生によりスーパバイザプロセスにより実行を横取りされる以外は、自ら中断するか終了するまで実行を続ける。つまり周期起動プロセス及び動的プロセスの間で実行を横取りされることはない。これはプロセスの切り替えによるオーバヘッドを防止

表2．CCOSプリミティブ一覧

名　前	機　能
S_CYPAS	周期起動プロシージャの登録（初期設定用）
S_CYEAS	周期起動イベントの登録（初期設定用）
S_CYPAC	周期起動プロシージャの起動開始
S_CYEAC	周期起動イベントの起動開始
S_CYPSP	周期起動プロシージャの起動停止
S_CYESP	周期起動イベントの起動停止
S_CYPRG	周期起動プロシージャの登録
S_CYERG	周期起動イベントの登録
S_CYPCL	周期起動プロシージャの登録解除
S_CYECL	周期起動イベントの登録解除
S_WAITF	一定時間のプロセスの実行停止
S_WAITU	指定時刻までのプロセスの実行停止
S_KILL	プロセスの強制終了
S_EXLCL	プロセス優先度の読みだし
S_EXLSE	プロセス優先度の設定
S_EVTRG	タイムアウトイベントの登録
S_BFTRG	タイムアウトバッファの登録
S_EVTCL	タイムアウトイベントの登録解除
S_BFTCL	タイムアウトバッファの登録解除

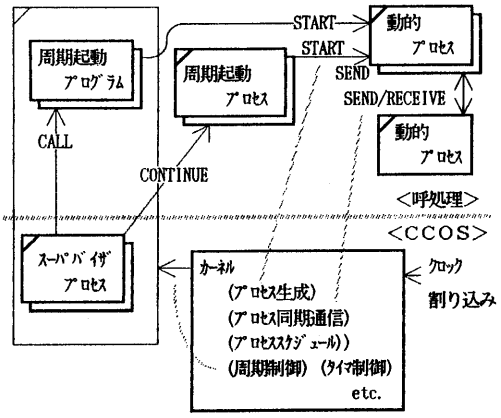

図7．CCOSにおけるプロセス

するためである。

周期プロセスは一度作られた後、スーパバイザによって周期的に起動される。周期的プロセスは無限ループとdelay文からなっており、delay文が実行されたイベント変数に対して、スーパバイザプロセスが周期的にcontinue文を実行する。周期プロセスを実現する手法としては、周期毎にプロセスを生成する方法もあるが、プロセス生成の失敗時の後処理、同一プロセスの二重生成の防止処理が煩雑であるため、ＣＣＯＳでは採用していない。

動的プロセスは必要になった時点で動的に生成される。これは交換プログラムで主に呼処理に用いられる。

５．ＯＳの記述に適した言語機能

ＣＣＯＳ及びこの上で走行する交換プログラムを記述しているＣＨＩＬＬの記述性について述べる。なおここで対象とするＣＨＩＬＬの言語仕様は、ＣＣＩＴＴが勧告した仕様[6]から必要性の小さい機能を除き、交換プログラムに特に必要な機能を追加したものである。

５．１　カーネルの記述

ＣＣＯＳのカーネル部を記述する上で有効だった言語機能について述べる。

（１）アセンブラコード定義[2]

ＣＨＩＬＬは図８に示すように、アセンブラの記述をプロシージャと同様の形式で定義し、その呼び出しをインライン展開する機能を持つ。ＯＳ

```
SAVER7: CODE( A BIN LOC );
  ST R7 / A ;   /*(1)*/
END SAVER7;

DCL R7SAVEA   BIN;

CALL SAVER7( R7SAVEA);  /* (1)のｺｰﾄﾞがｲﾝﾗｲﾝで展開される */
                        /* R7SAVEAにﾚｼﾞｽﾀ7の内容が格納される */
```

図8．アセンブラコード記述

はハードウェアを直接制御するため、アセンブラによる記述は必須であるが、これを用いれば、高水準言語記述部とアセンブラ記述部を別個に管理しなくてもよいのでソフトウェアの管理が容易となる。また頻繁に実行される部分をこれで記述すれば処理能力向上にも役立つ。なお本機能はＣＣＩＴＴ勧告の仕様にはない。

（２）待ち行列処理[2]

ＣＨＩＬＬは各種のデータ構造に対応して、待ち行列を定義し操作する機能をもつ。プロセスの実行順序の制御や、各種リソース制御における待ち行列処理が、従来のポインタ操作に比べて、簡潔かつ汎用的に記述できる。図９に例を示す。なお本機能はＣＣＩＴＴ勧告の仕様にはない。

```
SYNMODE ELEM = 1, 2 NEXT  REF ELEM, /* ﾘﾝｸ用 */
/* 型定義 */      2 DATA  BIN;

DCL QUE     FIFOUE(ELEM.NEXT); /* first in first out ｷｭｰ */
DCL P1, P2  REF ELEM;          /* ELEM型へのﾎﾟｲﾝﾀ */

  INSERT P1->ELEM IN QUE;      /* P1が指す要素の登録 */
  REMOVE ELEM FROM QUE SET P2; /* 要素の取り出し */
```

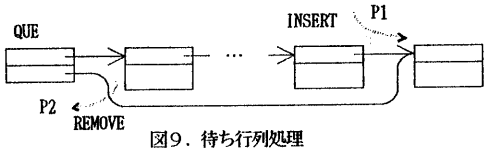

図9．待ち行列処理

（３）プロシージャ型

ＣＨＩＬＬではプロシージャが型として定義できる。周期起動制御において各要因に対応した処理のエントリを表として管理しているが、この表がＣＨＩＬＬのレベルで容易に記述できる。

５．２　並列処理機能による記述

並列処理機能を用いることにより、プロセスの生成や同期・通信が簡潔に記述できるので、ＣＣＯＳを含めた交換プログラムの記述に有効であっ

た。ここではそれらの記述性について述べる。

（１）プロセスの定義と生成

ＣＨＩＬＬではプロセスはすべて明示的かつ動的に生成する。これに対し宣言により静的にプロセスを生成する方法もある[7]。動的生成があれば静的生成は不要であるが、後者ではプロセス生成の記述が不要となり、実現を工夫すれば動的プロセスよりオーバヘッドを少なくして実行できる。ＣＣＯＳのスーパバイザプロセスのような実時間処理組み込み型システムでの常駐プロセスの記述には後者が適している。

ＣＨＩＬＬではプロセスを親子関係によりグループ化する機能を持たない。しかしプロセスを識別するためのデータ型があり、各プロセスで関連するプロセスを管理することができる。

（２）同期と通信

実時間処理要求に応えるため、交換プログラムでは群処理（ルックイン）を多用し、プロセスの切り替えによるオーバヘッドを少なくしている。このため図１０に示すように監視を行うプロセスと呼制御プロセスは非同期で動作し、これらの間での通信が必要となる。本稿で対象としたＣＨＩＬＬでサポートする並列処理機能も、プロセス切り替えを少なくし効率化することに重点をおいている。たとえばイベント変数の応用では、待ち状態のプロセスがつながれていないイベント変数にcontinue文を実行しても、この事象は破棄される。この事象の非保持性は効率上有効に活用できる。

これに対し、発生した事象を破棄せずに保持しておかなければならない処理も多い。これに対処するためには、continue文実行時にはイベント変数が空きか否かをチェックするか、バッファ変数を用いて事象を保持する方法がある。

Ａｄａのランデブ方式[7]は同期型のためバッファは不要であるが、そのつど同期のためのプロセス切り替えによるオーバヘッドが生じる。しかし入出力を行うようなサーバ的プロセスへの通信には処理の要求と結果の受信で両方向の同期的な通信が必要な場合もある（図１０）。ただしこれは片方向の非同期通信を２回行えばよく、非同期通信は同期通信を模擬できる。非同期通信のほうが融通性は高いが、記述性は劣る場合もある。

ＣＨＩＬＬにはガード付き受信機能[8]がないので、各状態にそれぞれ通信のための文を記述しなければならず煩雑となる。ガードに状態を反映させた式を記述すれば、各状態毎の同期・通信が不要となり記述量を減らせる可能性がある。

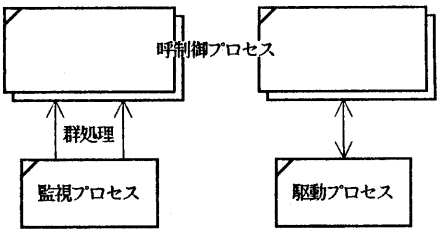

図１０．交換プログラムにおけるプロセス間同期・通信

（３）排他制御

リージョンとイベント変数の機構は、待ち行列のための処理が簡潔に記述できるため、有効である。

５．３　今後のＯＳ記述向き言語機能

（１）実時間処理向き機能

時間と時刻に依存した処理を記述する機能は、装置の駆動ばかりでなく、時刻に依存したサービスの処理にも有効である。信頼性の観点から送信側のタイムアウト機能も重要である。これらは(a)ＣＣＯＳの機能を用いて、(b)あるいは複雑な記述となるが周期処理を利用して、実現することができるが、将来的には言語機能として反映させる[9]ことが望ましい、と考える。

（２）実時間向きオブジェクト指向言語

信頼性の観点からは、即時に受け取られない事象を破棄するよりも保持する通信のほうが記述しやすい。これには発生した事象を直接相手プロセスに送信可能で、かつ相手がその事象を処理するまで保持される機構が向いている。これにデータ転送を付加すれば、これは非同期型プロセス間メッセージ交信機能となる。

交換プログラムは各プロセスが接続状態に対応した状態を遷移しながら、処理を進める。これらのことからメッセージ通信を更に進めて、処理をオブジェクト間のメッセージ交信としてモデル化した実時間向きオブジェクト指向言語が交換プログラムでは有効と考えている。

６．むすび

ＣＨＩＬＬ並列処理機能と実時間処理をサポー

トするＯＳであるＣＣＯＳの目的とアーキテクチャとＣＨＩＬＬ並列処理機能を用いた交換プログラムの記述について述べた。ＣＣＯＳはＯＳとして必要最小限の機能に限定して、まず広帯域交換システム[10]で実用に供した。その後、遠隔保守支援システム[11]や交換機用高機能コンソール装置のＯＳ[12]にも適用し、通信サービスの実現や保守に必要である入出力制御、ファイルシステム等その上に構築した。ＣＣＯＳはカーネルのハードウェアに依存した部分を除いてＣＨＩＬＬで記述されている。

呼制御プロセスを発呼側と着呼側にもそれぞれ一つずつ設けた既存交換プログラムのモデル[13]で、ＣＣＯＳと従来の処理効率を重視した実行制御方式を比較しオーバヘッドの増加を見積もったところ、呼処理総ステップ数の約20%となった。これは実行制御インタフェースの汎用化と、現在のカーネルがほとんどＣＨＩＬＬで記述されている事に起因すると考えられる。これは、(a)プロセス間でのデータ転送処理、(b)カーネル全体、をアセンブラ化することにより、オーバヘッドをそれぞれ呼処理総ステップ数の９％、５％まで減少させることも可能である。

並列処理機能とＣＣＯＳを適用することにより、交換プログラムをＣＨＩＬＬのレベルで簡明に記述できるが、十分でないところもある。並列処理機能を言語仕様にとりいれるにあたっては、(1)記述性は劣っても実現性や性能を重視する、(2)実現性や性能に問題があっても記述性を重視する、立場がある。両者の間のトレードオフをどこにおくかは難しい問題でありさらに研究が必要である。

参考文献

[1] 丸山，吉田：”交換プログラム用仕様記述法と高水準言語”，情報処理，Vol.21，No.3，pp.233-245(1980).

[2] 丸山，佐藤，鈴木，石森，小川：”交換プログラム作製用高水準言語ＣＨＩＬＬの実用化“，研実報，Vol.31，No.3，pp. 631-647(1982)

[3] 丸山，佐藤：”交換プログラム用並列処理言語“，情報処理，Vol.26，No.3，pp.407-413(1985).

[4] Kubota,M.and Sato,N. : "Concurrent CHILL Operating System", proceedings of ICC, pp.531-534(1984)

[5] 石野，毛利，望月：”ＤＥＸ２００実行制御プログラム”，研実報，Vol.18，No.10，pp. 2717-2726(1969).

[6] CCITT : "CHILL Language Definition",Recommendation Z.200(Apr. 1984)

[7] Dod : "Military Standard Ada Programming Language", Report MIL-STD-815(1980)

[8] Dijkstra, E.W. : "Guarded Commands, Nondeterminacy, and Formal Derivation of Programs", CACM, Vol.18, No.8, pp.453-457(1975)

[9] 久保田：”高水準言語ＣＨＩＬＬにおける実時間処理機能の検討“，昭59信学通信全大，338.

[10] 近藤，田村，村上，糸賀：”広帯域交換機のソフトウェア構成“，研実報，Vol.33，No.11，pp. 2599-2611(1984)

[11] 清水，麻生：”コンカレントＣＨＩＬＬ用ＯＳ・ＡＰＬインタフェースの構成法“，昭60信学情報全大，458.

[12] 久保田，宮山：”交換・通信処理システムの試験・デバッグ用コンソールの多機能化“，信学技報，SE85-15(1985).

[13] 渡辺，河辺，斉藤：”ディジタル加入者線交換機のプログラム構成“，研実報，Vol.31，No.11，pp.2003-2016(1982).