3V-2

V60リアルタイムOSにおけるタスク間通信 ー ランデブーの実現 ー

土屋 里枝、高橋 明子、北村 隆昭、今村 宣之、古城 隆
日本電気 (株) マイクロコンピュータ．ソフトウェア開発本部

1．はじめに

V60リアルタイムOSは、32ビットアーキテクチャーを持つマイコンV60のリアルタイム制御用OSである。[1]

本OSでは従来のセマフォ、メールボックスに加えランデブーによる通信を標準タスク間通信とし、Adaにおけるランデブー機能に近いシステムコールを提供している。本論文ではランデブーの特徴、本OSにおける実現、評価などについて述べる。

リアルタイムシステムでは、タスクは固有の走行環境で他タスクと独立に並行動作するとともに、複数の他タスクと同期、通信を行ないながら一連の処理を行なう。安全な走行環境は、タスク固有の走行環境を保護するだけでなく、他タスクとの通信をどれだけ安全に実現するかが重要な点である。本OSで提供するAdaランデブーによるタスク間通信方式は同期型タスク間通信であり、タスク間通信と同期の機能を同時に持ち、タスク間通信の往復が一組のシステムコールで実現できるため、高速でかつ安全なタスク間通信を供給することができる。

2 ランデブーの実現

2．1 基本機能

本OSにおいてランデブーの基本機能は、呼び側タスクのランデブー要求（エントリコール）システムコールrnd_cll と受け取りタスクのランデブ要求受け付け（アクセプト）システムコールrnd_acc,ランデブ終了システムコールrnd_end によって実現される。（図1 ）

呼び側タスクがrnd_cll を発行、あるいは受け取り側タスクがrnd_accを発行すると、先に実行したタスクが相手タスクのアクセプト、又はエントリーコールを待つという同期が行なわれる。両者が実行されるとランデブーが成立し受け取り側タスクは呼び側タスクのメッセージを受けとることができる。その後、受け取り側タスクのrnd_end をもって、メッセージがコール側に返されランデブーによるタスク間通信が終了する。

本OSでは、個々のタスクの管理データとして、それぞれタスク管理構造体（TCB）持つが、ランデブーを実現する為にそのTCB上にタスクの状態、状態要因、エントリ毎のエントリ待ちフラグ、アクセプト待ちキューフィールド、アクセプト待中エントリリスト、アクセプト中キューフィールドが用意されている。以下、メカニズムの実現方法を、それぞれのシステムコール毎に説明する。

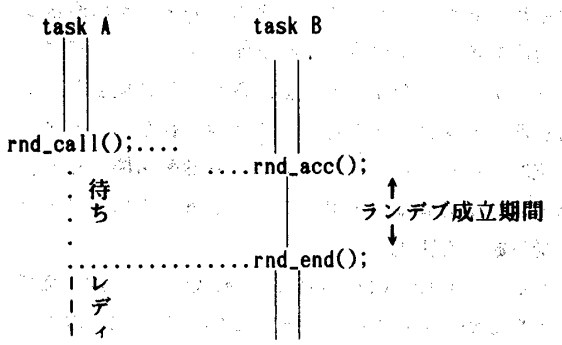

図 1 Adaランデブーによるタスク間通信

2．2 エントリコール（rnd_cll）

タスクAがタスクBにエントリコールする場合、パラメータとして相手タスクBのタスク名とその入り口（エントリ）を指定してrnd_cll が発行される。カーネル部はそのエントリに対してすでにアクセプト中かどうか判断し、アクセプト中であればB側にタスクを切り替え、アクセプト処理を実行する。逆に、アクセプト中でなければ、アクセプト待ちキューに自分を ENQしタスクAは中断する。

この時、rnd_cll のアーギュメント指定で永久待ち、時限、即時のいずれかでコールが可能である。永久待ちであればタスクAはタスクBのアクセプトが成立するまで中断し、時限であれば指定された時間だけ中断し、その時間内にランデブーが成立しなかった場合にはタイムアウトとする。又、即時ではエントリコールした時点で、相手タスクがアクセプトを実行していなかった場合には、アクセプトを待つことなくランデブ不成立とする。

ランデブー受け渡しパラメータはAdaではパラメータのならびであるのに対して本OSでは、パラメータブロックである点が異なる。受け渡したいパラメータを一つの領域に編集する作業はユーザの責任である。

Inter_Task Communication on V60 Real-Time OS
Rendezvous Implementation
Tuchiya, Rie Takahashi, Akiko Kitamura, Takaaki Imamura, Noriyuki Kojo, Takashi
NEC Corporation

2．3 アクセプト（rnd_acc）

タスクBがランデブ受け付けシステムコールrnd_accを発行すると、カーネル部は自タスクのアクセプト待ちキューにエントリコールが存在するか判断する。エントリコールが存在しない場合には、エントリコールされるまで一時実行を中断し、逆に存在した場合には、そのエントリに対してアクセプト処理を実行する。

アクセプト処理で問題点のひとつは、データの受け渡し方法である。本ＯＳの上のタスクは安全なタスク走行の為に、タスク毎の多重仮想空間によってプロテクション性を確保されているので、他タスクのデータは直接参照不可能である。本ＯＳでは呼び側タスクのランデブーパラメータエリアを受け側の組込みページに組込む。この組込みページはタスク毎に一つずつ存在し、共通アドレス空間に存在するので他タスクから参照可能となる。これにより不必要なデータのコピーをせずに高速にデータの受け渡しを実現している。

rnd_acc にもrnd_cllと同様に、永久待ち、時限 及び即時の指定ができrnd_cll のそれと同様な処理となる。さらに単一エントリに対するアクセプトと同時に複数のエントリに対して待つ選択アクセプトの組合せがある。これはrnd_acc のアーギュメントにエントリに対応するフラグの配列（エントリリスト）でアクセプトすべきエントリを指定する事により実現する。

2．4 ランデブー終了（rnd_end）

タスクBのアクセプト処理の後ランデブー終了システムコールrnd_end が発行されると、カーネル部では、ランデブーパラメータの組込みを解除し、以後タスクBからこのエリアが参照できないようにする。次にA側タスクを中断から起動し、A側の実行を再開させた後B側のタスクの処理にもどる。

以上簡単にランデブーの実現方法を処理概要を追って述べた。本ＯＳではこの他に割り込み処理のような中断を許されない時の通信をサポートするシステムコールとして、rnd_cll に替わってメッセージ送信システムコールrnd_snd が用意されている。rnd_cll,rnd_snd いずれのエントリコールも全く同様にアクセプト処理される。

3．リアルタイムＣ言語

アプリケーション言語の記述性がソフトウェア生産性の向上にとって重要な要因である事は言うまでもない。我々はそのような観点からランデブーの記述言語として、リアルタイムＣ言語を開発した。

リアルタイムＣは 通常のＣ言語にＡｄａのランデブー記述を可能にしたもので、リアルタイムＣコンパイラによって本ＯＳのランデブー用のシステムコールとその周辺のコーディングに展開される。ユーザは直接システムコールをコーディングしなくてもこのランデブ構文を用いることにより、高い記述性とコンパイル時のチェック機能による信頼性の高いコーディングが可能となる。また、リアルタイムＣ言語はランデブー記述を除けば従来のＣ言語仕様を満足するので、これまでＣ言語で記述されたソフトウェアは完全に流用する事が可能である。

リアルタイムＣ記述方法とその展開結果を以下に示す。

構文	展開イメージ

時限エントリコール
select {
call TASK.A(p);
}
or
delay (n) {
ﾀｲﾑｱｳﾄ処理；
}
}
	err=rnd_cll(&TASK,...,&p)
	switch(err) {
	case 0:
	break ;
	case ﾀｲﾑｱｳﾄ:
	ﾀｲﾑｱｳﾄ処理；
	break ;
	default:
	break ;
	}

時限アクセプト
select {
accept ｴﾝﾄﾘ(p1){
.....
}
or
delay(n) {
ﾀｲﾑｱｳﾄ処理；
}
}
	エントリリストなどの宣言
	err=rnd_acc(&entno,...)
	switch(err) {
	case 0: {
	switch(entno) {
	.....
	}
	rnd_end();
	break ;
	}
	case ﾀｲﾑｱｳﾄ:
	ﾀｲﾑｱｳﾄ処理；
	break ;
	default:
	break ;
	}

3．評価

本ＯＳの評価用プログラムとして、本ＯＳの上で動作する簡単な交換システムのアプリケーションを開発したが［2］、そのタスク間通信には、ランデブーによる通信を採用し、記述言語としてはリアルタイムＣを使用した。本システムの開発に要した工数は従来のリアルタイムシステムの開発工数に比べ２分の１以下であった。特にデバック工程の比率がいちじるしく減少したが、これはランデブーによるタスク間通信の安全性が一要因となっいるものと考えられる。このシステムでのタスク間通信の実行時間を評価したところ、Ｖ６０でのランデブ往復処理時間は２００μＳｅｃ 程度と予想された。これは従来のメールボックスなどによるタスク間通信の実行時間と比較してかなり良い結果と考えられる。

参考文献

［1］古城 他：V60ﾘｱﾙﾀｲﾑOSの設計
情報処理学会第３３回全国大会公演集1986 1C-4
［2］世良 他：通信処理汎用ﾘｱﾙﾀｲﾑOSの交換処理への適用 電子通信学会技術研究報告 SE-85-137