*Tender*における資源「時計」の実現

4M－8　野口 裕介　後藤 真孝　谷口 秀夫　牛島 和夫
九州大学工学部

1　はじめに

現在、我々はプログラム構造に重点をおいたオペレーティングシステム ***Tender***(The ENduring operating system for Distributed EnviRonment) の開発を行なっている[1]。

Tenderでは、時計を一つの資源として扱い、各時計毎に時刻や時間の進み具合 (時刻進度) を設定できる機能を持つ。これにより、それぞれのプロセスが異なった時間軸上で処理を行なったり、同一プロセスで速度の異なる時間を持つことが可能になる。

本稿では、***Tender***における資源「時計」の機能内容、実現方式について述べる。具体的には、複数の時計を提供する機能や時計の速度を変化させる機能について述べる。また、実現方式として「時計」資源管理における時計の管理と処理の方式を述べ、提供インタフェースを説明する。

2　資源「時計」の機能

Tenderでは、資源を「一つの処理を行なう場合、その操作で処理が完結し、他の処理を必要としないもの」と定義している。***Tender***では時計を一つの資源として扱う。具体的には、時計管理が複数の時計を管理し提供する (図 1)。時計は、以下の機能を持つ。

(1) 独立した時刻
(2) 可変な時刻進度
(3) 時刻逆行操作の禁止

それぞれの機能について以降に説明する。

2.1　独立した時刻

各々の時計は、時刻と時刻進度を持つ。各時計が持つ時刻や時刻進度は独立に設定できる。これにより、以下のような利用法が考えられる。

(1) **各プロセス毎での独立した時刻の保持**
各プロセスが個別に時計を保持することにより、あるプロセスによる時刻の変更が他のプロセスに予期せぬ影響を与えることを防ぐ。

(2) **カーネル時刻とプロセス時刻の分離**
カーネルはプロセスの影響を受けない絶対時刻を保持できる。

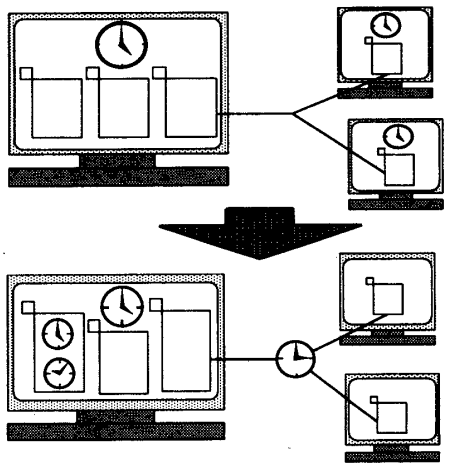

図 1　複数の独立した時計の提供

(3) **同一プロセスによる複数の時刻の保持**
いくつかの異なる時刻を利用するプロセスは、必要分の時計を保持することにより、一つの時計から時刻を変換する必要はない。

2.2　可変な時刻進度

各時計は、時間の進み具合 (時刻進度) を持つ。さらに、その時刻進度を変更できる機能を持つ。

Tenderでは、「プロセスに対して任意のプロセッサ性能を割り当てる」機能を持つ資源「演算」[2]を使用することで、プロセスの実行速度を変化させることができる。

従って、処理中にその時々の時刻を意識したサービスを行なうプロセスに対し、資源「演算」と可変な時刻進度の時計を使用することによって以下のようなプロセスの実行が可能になる。

(1) プロセッサ性能に合わせて時計の速度を変化させることで、プロセスに対しプロセッサ性能に左右されない一定条件での実行を保証する。
(2) プロセッサ性能を固定した上で時刻進度を変化させ、様々な速度条件でのシミュレーション実行を行なう。

2.3　時間逆行操作の禁止

時計の時刻が逆行する操作を禁止する。これにより、トランザクション処理のように複数の計算機によって一つのサービスを提供するような場合に、時間の逆行による処理上の矛盾が発生することを防止できる。複数の時計間で時刻の同期を必要とする場合は、それぞれの時計の速度を調節することで行なえる。

Implementation of Resource "Clock" on Tender.
Yusuke NOGUCHI, Masataka GOTO, Hideo TANIGUCHI and Kazuo USHIJIMA
Faculty of Engineering, Kyushu University.

3 実現方式

時計管理の実現方式について述べ、インタフェースについて説明する。

3.1 方針

時計管理の実現は、以下の方針で行なった。

(方針 1) プロセッサへの負荷を抑える

(方針 2) 他の資源の利用を避ける

3.2 処理方式

3.2.1 管理情報

時計管理は、複数の時計を管理するため、各時計毎に以下の情報を保有する。

(a) 識別子

(b) 時刻

(c) 時刻進度

識別子は複数の時計を区別する。時刻の初期値は、利用者から与えられる。時刻の更新方式については次項で述べる。時刻進度は、利用者から与えられる。

3.2.2 時刻の更新方式

現在時刻を更新する方法としては、周期更新と要求時更新の二つの方式が考えられる。

(1) **周期更新**

時刻の更新を、時刻進度に対応した適当な周期で、定期的に行なう。これは、常に針が動いている一般的な時計のイメージである。

(2) **要求時更新**

時刻の更新を、時刻の読み出しを要求された時に行なう。これは、通常は止まっているが、要求があると現在時刻を表示する時計のイメージである。

両方式の比較を表 1 に示す。(方針 1) より、要求時更新の方式を採用する。

表 1 時刻の更新方式の比較

方式	長所	短所
(1) 周期更新	それぞれの時計は常に現在の時刻を保持しているため、時刻の取得や変更が容易である。	更新の負荷が大きい。さらに、時計の数の増加とともに、時刻の更新の負荷が増加する。
(2) 要求時更新	時刻の参照がなされない時には負荷が生じない。	時刻の取得時に、時刻の計算を行なう必要がある。

3.2.3 経過時間の所得法

要求時更新の方式では、前操作からの経過時間の取得法が問題となる。この経過時間の取得法には、次の二通りの方式が考えられる。

(方式 1) 資源「時間カウンタ」を利用する

指定した期間の時間を計測できる資源「時間カウンタ」を利用し、前操作からの経過時間を取得する。

(方式 2) ハードウェア時計 (HC) を利用して計算する

前操作時の HC の時刻と、現在の HC の時刻から経過時間を計算する。

両方式の比較を表 2 に示す。(方針 2) より (方式 2) が好ましい。また、時刻の取得の頻度が少ない場合も、(方針 1) を満足する (方式 2) が好ましい。

表 2 経過時間の取得方式の比較

方式	長所	短所
(1)	資源「時計」管理として経過時間を計算する必要がない。	資源「時間カウンタ」での計測処理が常に必要になる。
(2)	他の資源を利用しない。	管理情報として、HC の時刻も保有する必要がある。

3.3 提供インタフェース

時計管理が提供するインタフェース関数の機能と形式を表 3に示す。

表 3 「時計」管理が提供するインタフェース

操作	形式	機能
open	create_clock()	時計を確保し、時刻の読み出しと書き込みを許可する。
close	break_clock()	指定された時計を解放する。
read	get_time()	指定された時計の現在時刻を読み出す。
write	set_time()	指定された時計の時刻を変更する。
control	set_clock_speed()	指定された時計の速度を設定する。

4 今後の課題

今後の課題として、時計管理を実現し ***Tender*** への実装を行なう。また、時計資源の機能を利用した新しい観点からの時計の利用法についての検討を行なう。

参考文献

[1] 谷口秀夫 : "分散指向永続オペレーティングシステム ***Tender***", 情報処理学会シンポジウム論文集 Vol.95, No.7, pp.47-54(1995).

[2] 村上大介, 青木義則, 谷口秀夫, 牛島和夫 : "Tender における資源「演算」の扱い", 情報処理学会第 51 回全国大会予稿集, 5L-8 (1995).