Ｔｅｎｄｅｒにおける資源「演算」の扱い

5Ｌ－8

村上 大介　青木 義則　谷口 秀夫　牛島 和夫
九州大学工学部

1　はじめに

負荷の大きなソフトウェア処理の実現、また多様なサービスの提供のために、オペレーティングシステム（以降ＯＳと略す）は様々な機能を提供し、また機能の高度化を進めている。既存のＯＳの多くは、機能の実現においてプログラム構造を軽視する傾向にある。その結果、機能の高度化とプログラム構造の複雑化はあまり関係しないはずであるにも関わらず、プログラム構造は複雑化している。そこで我々の研究室では、プログラム構造の明瞭な Tender (The ENduring operating system for Distributed EnviRonment) の開発を進めている[1]。

Tender では、資源を「１つの操作を行なう場合、その操作で処理が完結し、他の処理を必要としないもの」と定義する。本稿では、Tender における「演算」資源の扱いについて述べる。具体的には、プロセス構成要素の独立化による効果、また演算を資源として扱うことによる利点を述べる。そして、「演算」を管理するモジュールの機能やインタフェースを述べる。

2　実行の単位

ＯＳにおいて、プロセッサの実行を割り当てる単位はプロセスである。

2.1　プロセス要素の分解と独立化

プロセスの構成要素を、図１に示す。プロセスは、手続き（テキスト）の部分やデータ部分、それにスタック部分や制御表を持つ。それらにアクセスできるよう、仮想空間上に割り付ける。そしてプロセッサの実行権を確保することで、プロセスは実行される。

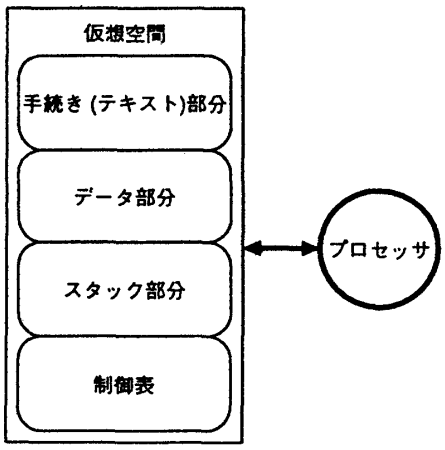

図１　プロセスの構成要素

Tender では、プロセスを大きく、「プロセス」資源と「演算」資源に分解する。さらに、「プロセス」は、「仮想ユーザ空間」資源、「仮想カーネル空間」資源、「プログラム」資源を利用する。

「プロセス」は、その内部で「仮想ユーザ空間」管理へ必要なメモリ空間を要求する。また、「プログラム」管理に要求してテキストとデータを得る。そして、獲得したテキスト、データ、及びプロセスのスタック領域をメモリ空間に割り付けるよう、「仮想ユーザ空間」管理に要求する。さらに、「仮想カーネル空間」管理に要求して、プロセスの制御表を割り付ける。

その上で、「演算」管理に要求し、プロセッサ実行を確保し、プロセスは走行する。

このように、プロセスを構成要素に分解し、要素を独立化すると、以下の利点を得る。

（１）プロセス生成の高速化
演算の独立化により、プロセスを生成した後、プロセッサ実行権を与えずにおくことが可能になる。頻繁に使用されるプロセスをこのような状態で準備しておけば、使用時にはプロセッサ実行権を与えるだけで実行でき、プロセスの生成を高速化できる。

（２）プロセスの処理途中での停止や再開が可能
演算の独立化により、プロセスの処理の途中で演算を放棄させ、プロセスを停止させることができる。また、その後演算を再確保することで、プロセスの処理の再開が可能になる。

（３）多重仮想空間と単一仮想空間の混在が可能
仮想空間とプログラムの分離により、１つの空間に１つのプログラムのみを置くことも、複数のプログラムを置くことも可能になる。従って、多重仮想空間と単一仮想空間が混在でき、両者の利点をともに得ることができる。

（４）プロセス間でのデータの共有が可能
仮想空間とプログラムの分離により、データのみを持っている仮想空間も存在できる。これにより異なるプロセス間でデータを共有できる。

2.2　「演算」資源

Tender では、「演算」を「ユーザモードで走行するプロセッサの演算」と定義する。これにより、ユーザの処理とＯＳ処理を分離する。その結果、ＯＳ処理の自由度が増加する。例えば、現在走行中のプロセスとは無関係に、別のプロセスが発行したシステムコールを処理す

Management of Resource "Processing" on Tender
Daisuke MURAKAMI, Yoshinori AOKI, Hideo TANIGUCHI, and Kazuo USHIJIMA
Faculty of Engineering, Kyushu University

ることが可能になる。一方、ユーザプロセスの実行優先度とＯＳ処理が矛盾する可能性が生じる。具体的には、優先度が低いユーザプロセスがシステムコール処理を多発すれば、優先度が高い他のプロセスより多く実行される。しかし、演算を資源とすることで、特に確保や解放の要求を明示化できる。これにより、プロセッサ性能の指定が可能になる。プロセッサ性能の指定を用いることで、ユーザプロセスの実行優先度とＯＳ処理の矛盾に対処することが可能になる。また、確保や解放の要求の明示化により、将来のプロセッサの利用状況の把握が可能になり、負荷分散等の面で大きな効果をあげうる。

3 「演算」管理

「演算」の資源管理モジュールについて、以下に示す。

3.1 プログラム構造

「演算」管理は、表プログラム構造[2]の一部として実現する。

3.2 機能内容

「演算」管理は、「ユーザモードで走行するプロセッサの演算」を資源として管理する。「演算」管理の機能を以下に示し、以降に説明する。

（1）演算をプロセス対応に割り当てる。

（2）割当に従い、プロセスに演算を与える。

3.2.1 演算の割当

演算をプロセスに割り当てる機能とは、プロセスの実行予定を管理する機能である。本機能は、割当の契機（スケジュール契機）と割当の方式（スケジュール法）の観点から次のようになる。

割当の契機は、他モジュールからの要求を契機とするのみである。この場合、要求内容の実現が可能か否かを判断し、可能であれば要求に応じた演算の割当を行う。要求内容の実現が不可能であれば、要求内容は却下し、その旨を要求したモジュールに通知する。

割当の方式として、２つある。１つは、単位時間当たりのプロセッサ割当を要求プロセッサ性能に合わせて行う方式である。もう１つは、指定された優先度に基づき、演算を残余なく利用する方式である。

前者の方式では、プロセッサの実行を細かな時間片に分割し（分割した時間片を、以降タイムスロットと呼ぶ）、プロセスの要求性能に応じてタイムスロットを割り当てる。ここでは、連続する複数個のタイムスロット内におけるタイムスロットの割当個数を、要求性能に合わせて調整する方式を採用する。

後者の方式では、性能を要求しているプロセスが走行していない時間に、性能を要求していないプロセスの中から、プロセス優先度に基づいて走行させるプロセスを選ぶ。

２つの方式を共存させることにより、プロセッサを有効利用しつつ、可変なプロセッサ性能を提供できる[3]。

3.2.2 演算の授与

プロセスに演算を与える機能は、割当の方式（スケジュール法）により決定されたプロセスを実行させる機能である。プロセスに演算を与える契機（ディスパッチ契機）には、２つある。１つは他モジュールからの要求であり、もう１つは「時計」資源や「時間カウンタ」資源を利用した割当の変更である。いずれの場合も、割当の方式に従い、演算をプロセスに与える。「時計」や「時間カウンタ」を利用した割当の変更は、実際にはタイマ割込みが契機となる。そこで、カーネルスタックの内容（割込みを受けたプロセスの情報）を適切な領域へ退避し、これからプロセッサが割当てられるプロセスの情報をカーネルスタックに積めば、割込み処理を終えて元のプロセスへ戻るところで、これから実行されるプロセスへ処理が移る。

3.3 提供インタフェース

「演算」管理が提供するインタフェース関数の機能と形式を表１に示す。

表１ 「演算」管理の提供するインタフェース

形式	機能
get_execution	指定されたプロセスを、要求のプロセッサ性能で指定された番地から実行する。
free_execution	指定されたプロセスに割り当てられた演算を解放する。
wait_execution	指定されたプロセスへの演算割当を停止させ、休眠識別子を設定する。
wakeup_execution	指定されたプロセス、または指定された休眠識別子に対応するプロセスを覚醒させ、演算割当を再開する。
dispatch	指定されたプロセスへ、演算を授与する。

4 おわりに

Tenderにおける演算の扱いについて述べた。また、「演算」資源の管理モジュールについて、機能内容と提供インタフェースを述べた。

今後は「演算」管理を実現する予定である。

参考文献

[1] 谷口,“オペレーティングシステム Tender の開発”, 本論文集 5L-6

[2] 後藤, 谷口, 牛島,“Tender における資源管理方式”, 本論文集 5L-7

[3] 村上, 谷口, 牛島,“異なるスケジュール法の共存制御法”, 1995 年電子通信学会総合大会 D-161