システムソフトウェアと
オペレーティング・システム 73－19
（1996. 8. 27）

Tender のプロセス管理構造

谷口 秀夫† 田中 徳穂‡
†九州大学大学院システム情報科学研究科
‡九州大学工学部情報工学科

プログラム構造に力点を置いたオペレーティングシステム***Tender***（The ENduring operating system for Distributed EnviRonment）について、そのプロセス管理構造を述べる。***Tender***では、プロセスの各構成要素の関連を最小限として独立化させ、「資源」として管理する。また、「資源」を管理するプログラム部分の関係も独立化させる。これにより、プロセス生成の高速化や2種類の仮想メモリ空間の共存などの特徴を持つ。資源「プロセス」を紹介し、プロセスに関連する資源と処理の相互関係および処理の流れを説明する。

キーワード
プロセス管理、オペレーティングシステム、メモリ管理、資源

Process Management Architecture of *Tender*

Hideo TANIGUCHI and Noriho TANAKA
Kyushu University

This paper describes process management architecture of ***Tender*** that has been developed by paying attention to program construction. On ***Tender*** , the relation among elements of process is minimum, each elements are independently and are managed as "resource." Each resource management programs are independently too. As the result,this process management architecture has some features, e.g. high-speed process creation and two kinds of virtual memory spaces.This paper shows process resource and explains the resources enclosing process resource and the control flow.

key word
process management, operating system,
memory management, resource

1．はじめに

プロセスは、オペレーティングシステム（以降、ＯＳと略す）が制御の基本単位とするものである。そのため、ＯＳの処理は、多くの部分がプロセスに深く関係している。制御の上では、各処理はプロセスコンテキストを意識しており、どのプロセスのコンテキストの上で動いているかに留意しながら処理を進める。また、処理構造の上では、各処理はプロセス管理表を共通利用している。これらのことは、ＯＳの処理を行う上で有効な面があるものの、不都合な面もある。１つは、各処理はプロセスコンテキストを意識するため、ＯＳ内の処理でさえ、その処理動作を行いやすくするために、プロセス化することも少なくないことである。この結果、処理速度の低下を招いている。もう１つは、各処理はプロセス管理表を共通利用するため、情報が１つの管理表に集中する点である。この結果、プロセス管理表の変更が困難になり、プログラムの部品化を阻害している。

我々は、プログラム構造に力点を置いたオペレーティングシステム***Tender***（The ENduring operating system for Distributed EnviRonment）[1]を開発している。本稿では、上記の不都合に対処した***Tender***のプロセス管理構造述べる。

2．プロセスの構造と管理

2．1　プロセスの構成要素

プロセスは様々な要素から構成されている。その様子を図１に示し、以降に説明する。

プログラムは、テキスト部、データ部、およびスタック部からなる。スタック部は、ユーザスタックとカーネルスタックに分けれる。プロセス管理表は、実行を制御するため、あるいはプログラム処理が必要とするものを操作できるようにするために存在する。プロセス管理表に格納される情報は、大きく２つに分類できる。１つは、実行の制御に必要な情報である。これを内部情報と名付ける。内部情報には、プロセス識別子などがある。もう１つは、プログラム処理が必要とするものの操作に必要な情報である。これを外部情報と名付ける。外部情報には、ファイル記述子などがある。

プログラム実行のために必要なものとして、メモリ空間、プロセッサ、およびレジスタがある。これらを内部資源と名付ける。メモリ空間には、プログラムとプロセス管理表が置かれる。また、プログラム処理が必要とするものとして、ファイルやソケットなどがある。これらを外部資源と名付ける。

これらの各構成要素が関連して、プロセスの走行を実現している。

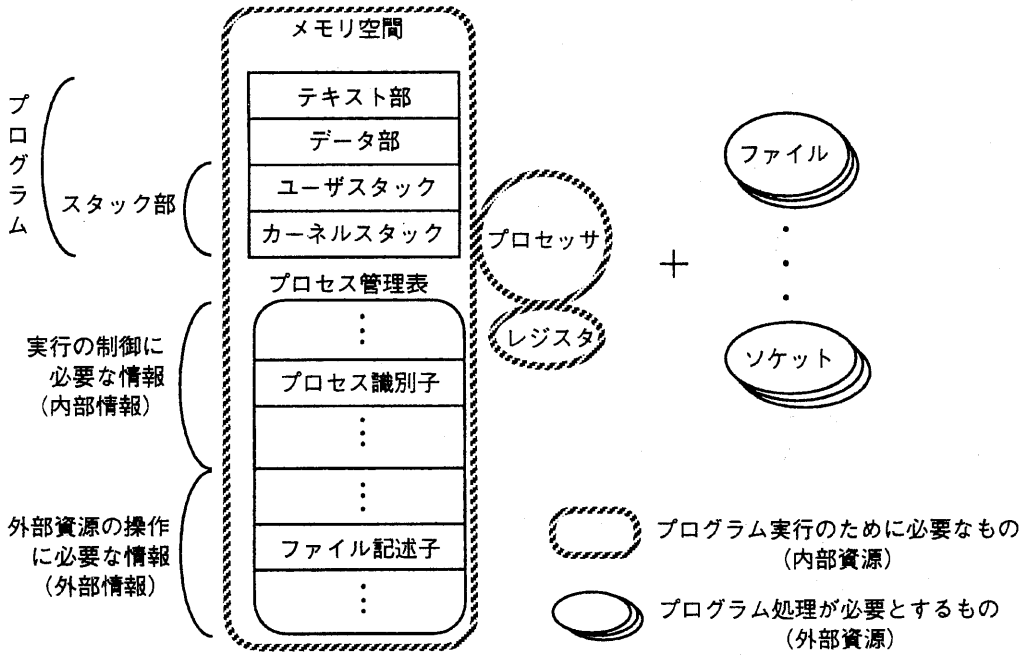


図１　プロセスの構成要素

2．2　問題点

既存のＯＳにおけるプロセス管理構造は、大きく２つの特徴を持っている。

１つは、プロセスの各構成要素は単独で存在することはなくプロセスの存在を前提としている点である。プロセスとしての実行を前提として、プログラムがメモリ空間上に置かれる。また、プログラムが置かれていないメモリ空間は存在しないのである。つまり、内部資源や外部資源の大半は、プロセスの存在を前提として存在する。このため、プロセスの生成や終了に合わせて、資源の生成や削除あるいは操作のための情報が処理される。その結果、同じプログラムを利用するプロセスの生成や終了では、処理に無駄が発生している。

もう１つは内部情報と外部情報が１つのプロセス管理表に存在している点である。このため、ＯＳ処理のほとんどは、プロセス管理表の参照や更新なくしては行えない状況にある。このことは、ＯＳプログラム構造の把握を困難にし、ＯＳの効果的な利用を妨げている。つまり、ＯＳへの機能の追加や削除および変更を困難なものにしている。

3．*Tender* のプロセス構造と管理

3．1　資源の独立化

Tender では、プロセスの各構成要素の関連を最小限として独立化させ、「資源」として管理する。また、「資源」を管理するプログラム部分（管理部と名付ける）の間の関係も独立化させる。

各資源は、その種類毎に、資源を管理する管理表（資源識別子など）と資源を操作するための制御表（操作位置など）を持ち、それらを利用して資源管理部が、資源を管理し操作を制御する。資源管理部間でのプログラムや管理表・制御表の共有はない。

プロセスに深く関連する資源として、「プログラム」「実メモリ」「仮想ユーザ空間」「仮想カーネル空間」「演算」[2]および「プロセス」がある。

3．2　特徴

資源の独立化による ***Tender*** のプロセス構造と管理の特徴を以下に説明する。

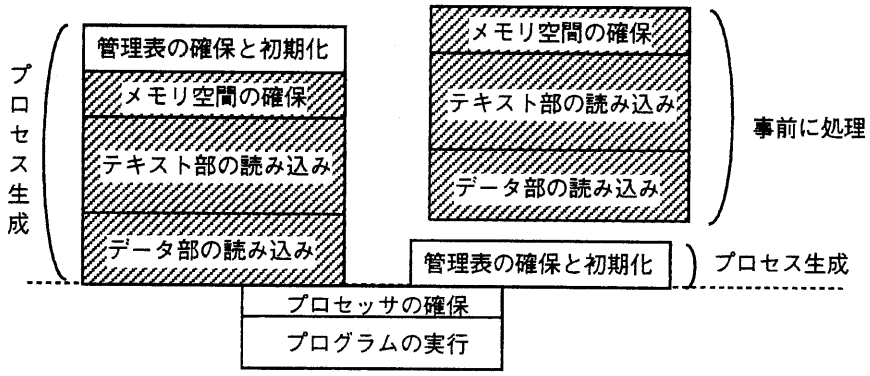
（１）資源の事前作成と再利用

Tender では、プロセスの存在がなくても、メモリ空間などプロセスの動作に必要なものの存在が可能である。このため、資源を事前作成することができる。将来、必要になる資源を事前に作成しておくことで、必要時の処理を軽減できる。また、資源の再利用が可能になる。資源が不要になったとき資源を消去せず、後に必要になった時点で再利用すれば、必要時の処理を軽減できる。

資源の事前作成の例として、プロセス生成の高速化を図２に示し、説明する。プロセス生成の処理は、

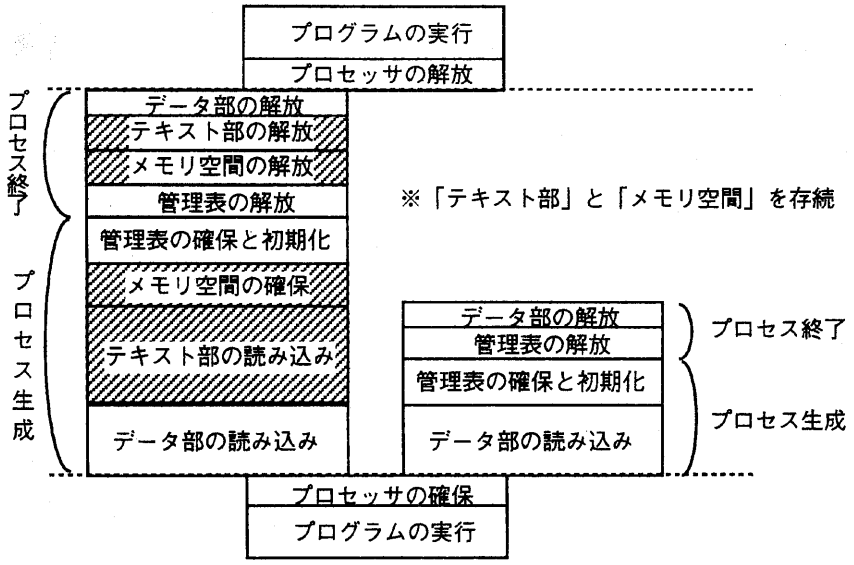
①管理表の確保と初期化
②メモリ空間の確保
③テキスト部の読み込み
④データ部の読み込み

を行い、プロセッサの確保によりプログラムが実行される。上記の処理の中で、処理②③④は、事前に、資源「プログラム」「実メモリ」「仮想ユーザ空間」を作成し、行う。これにより、プロセス生成要求時には、処理①を行うだけで、プロセスが生成できる。



(a) 従来のプロセス生成　　(b) 事前作成を利用したプロセス生成

図２　資源の事前作成によるプロセス生成の高速化



(a)従来のプロセス終了・生成　　(b)再利用によるプロセス終了・生成

図3　資源の再利用によるプロセス終了・生成の高速化

資源の未消去による再利用の例として、プロセス終了・生成の高速化を図3に示し、説明する。プロセス終了の処理は、

①データ部の解放
②テキスト部の解放
③メモリ空間の解放
④管理表の解放

を行い、プロセス生成の処理は、先に述べた、

⑤管理表の確保と初期化
⑥メモリ空間の確保
⑦テキスト部の読み込み
⑧データ部の読み込み

を行い、プロセッサの確保によりプログラムが実行される。ここで、資源「プログラム」「実メモリ」「仮想ユーザ空間」を再利用することにより、処理②③⑥⑦を割愛できる。

（2）プロセス途中での停止や再開

プロセスを途中状態で停止させ、そのまま保存する。***Tender***では、資源の永続化を可能にしており、プロセス走行途中の状態をそのまま永続的に保存でき、必要な時に再開できる。再開時の走行開始アドレスやデータは、変更が可能である。

（3）データ空間の存在

プログラムが存在せず、データしか存在しないメモリ空間を実現する。これにより、高速な操作速度を要求するデータや操作頻度の高いデータは、利用するプロセスの存在の有無に関係なくメモリ空間上における。したがって、必要時に、即座に、メモリ上のデータとして利用できる。

（4）資源の状況把握

Tenderでは、資源の種類毎に管理表と制御表を分離しているため、各資源の状況を個別に把握することができる。このことは、ＯＳの理解を促進し、試験や改版あるいは学習など多くの点で有効である。

ただし、このままでは、資源単独の情報把握は容易なものの、処理の流れとの関係や資源間の関係は明確にできない。これについては、資源インタフェース制御部[3]において、資源管理部間の処理流れを把握するようにしている。

（5）異環境の構築

多重仮想記憶（ＭＶＳ）では、各プロセスが固有の仮想メモリ空間を持つ。プロセス毎に仮想メモリ空間が別々なため、プログラムのアドレスはプロセス毎に自由に決定でき、また、プロセス間の保護が強い。反面、プロセス間のデータ共有やプロセスディスパッチの処理オーバヘッドが大きい欠点がある。一方、単一仮想記憶（ＳＶＳ）では、全てのプロセスは１つの仮想メモリ空間を共有する。その特徴は、ＭＶＳの逆である。

Tenderでは、プロセスとメモリが資源とし

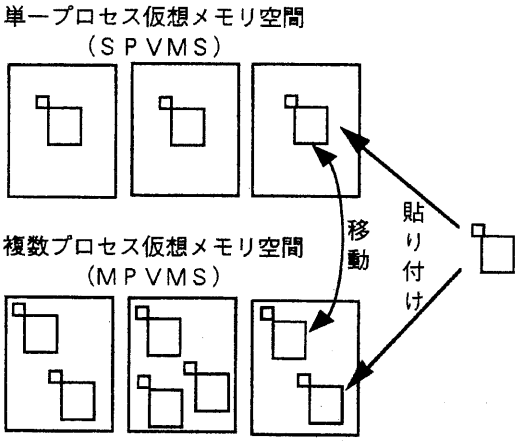


図４　ヘテロ仮想記憶（ＨＶＳ）

て分離独立化されている。この特徴により、１つの仮想メモリ空間に１つのプロセスを割り付ける単一プロセス仮想メモリ空間（ＳＰＶＭＳ：Single Process Virtual Memory Space）と、１つの仮想メモリ空間に複数のプロセスを割り付ける複数プロセス仮想メモリ空間（ＭＰＶＭＳ：Multi Process Virtual Memory Space）を共存させる。これを、ヘテロ仮想記憶（ＨＶＳ：Heterogeneous Virtual Storage）[4]と名付ける。ＨＶＳの様子を図４に示す。プロセス生成時に、ＳＰＶＭＳまたはＭＰＶＭＳのどちらか一方に、プロセスを貼り付ける。プロセスは、処理が進むにつれて、処理内容の特徴に合わせＳＰＶＭＳとＭＰＶＭＳの間を移動できる。

３．３　資源「プロセス」

資源「プロセス」を管理するプロセス管理が提供するインタフェースの一覧を表１に示す。プロセス管理のインタフェースとして、作成・消去・情報の設定取得・環境設定がある。

３．４　処理の相関と流れ

プロセスに関連する資源と処理の相互関係および処理の流れを図５に示す。図５（ａ）は、処理の相互関係を示しており、実線は制御が戻る呼出、点線は制御が戻らない（つまり、制御の切替が行われる）呼出である。太めの実線は、制御は直ぐには戻らない（つまり、制御の切替が行われ、後に再び制御が戻る）呼出である。図５（ｂ）は、プロセス生成とプロセス終了およびデータ出力の処理について、処理の流れを示したものである。

プロセス生成処理は、大きく作成処理と起動処理に分かれている。資源の事前作成が可能である。プロセス終了処理は、大きく停止処理と消去処理に分かれている。資源の再利用が可能である。また、停止処理と起動処理を組み合わせることにより、プロセス途中での停止や再開が可能である。データ出力処理は、ＤＫ管理部で実Ｉ／Ｏ要求を行い、演算管理部への停止の呼出で実Ｉ／Ｏ終了を待つ。次に、割り込みからの通知を受けて、ＤＫ管理部から演算管理部への再開の呼出を行う。その結果、演算管理部への停止の呼出を行った直後から処理が再開され、実Ｉ／Ｏ終了の処理を行う。

上記以外の処理として、図５（ａ）では、割り込みを利用した周期タイマ管理部から演算管理部への周期通知の処理や、割込・例外処理から演算管理部への切替の処理も記述している。

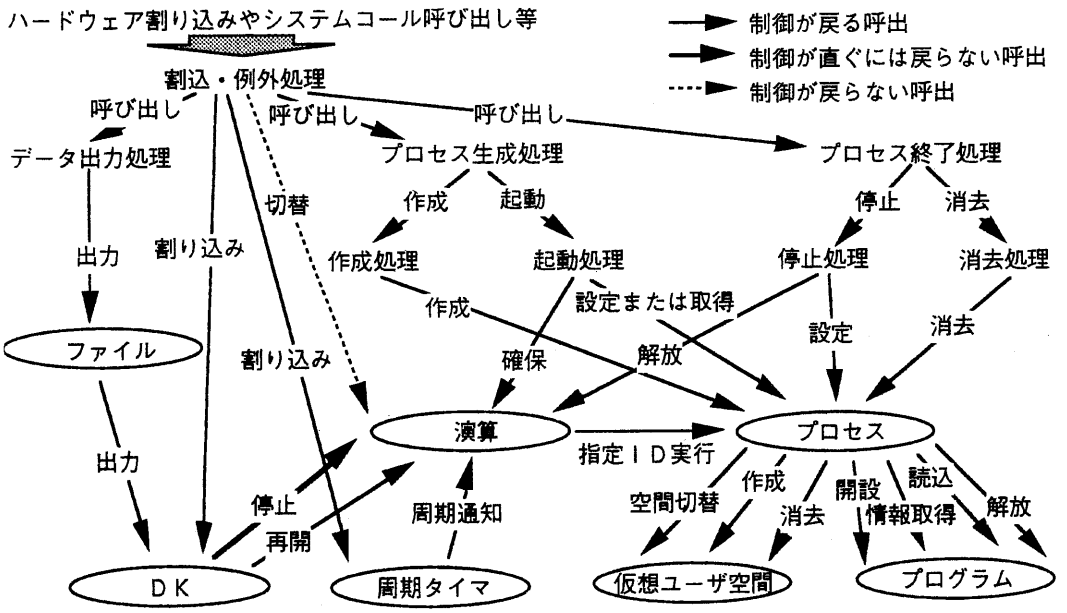
演算管理部は、資源「演算」の確保や解放などのインタフェースとして、スケジュールやディスパッチで扱う識別子を授受する。この識別子が、スケジュールやディスパッチの単位となる。この識別子は、プロセス識別子と別の識別子として管理している。

４．おわりに

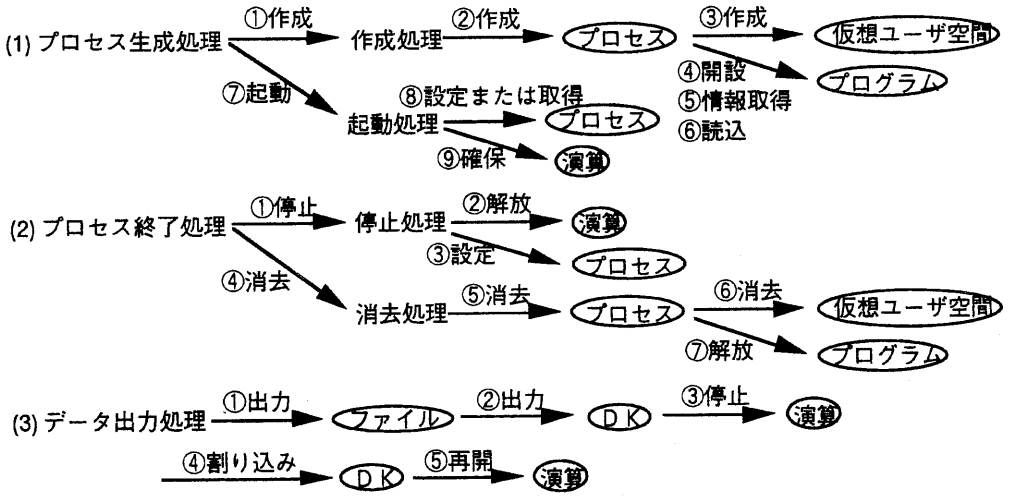
プロセスを構成する要素を示した。***Tender***では、プロセスの各構成要素の関連を最小限として独立化させ、「資源」として管理している。また、「資源」を管理するプログラム部分（管理部と名付ける）の関係も独立化させている。これにより、①資源の事前作成と再利用、②プ

表１　プロセス管理が提供するインタフェース一覧

通番	形　　式	機　　能
1	create_proc(path,argv)	pathで指定されたファイルを利用してプロセスを作成する。
2	delete_proc(pid)	pidで指定したプロセスを消去する。
3	control_proc(pid,type,buff)	typeの指定に従い、pidで指定したプロセスへの情報設定またはプロセスからの情報取得を行う。
4	prep_proc(pid)	pidで指定したプロセスを実行可能な環境にする。



（ａ）処理の相関



（ｂ）処理の流れ

図５　プロセスに関連する処理

ロセス途中での停止や再開、③データ空間の存在、④資源の状況把握、⑤異環境の構築、の特徴を持つ。さらに、「プロセス」の資源について説明し、プロセスに関連する資源と処理の相互関係および処理の流れを述べた。

今後は、評価を進める予定である。

最後に、***Tender*** 開発担当者に感謝します。

<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***”，情報処理学会コンピュータシステムシンポジウム，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，5L-8 (1995).

［３］後藤真孝，谷口秀夫，牛島和夫：“***Tender*** における資源管理方式”，情報処理学会第５１回全国大会，5L-7 (1995).

［４］長嶋直希，谷口秀夫，牛島和夫：“***Tender*** のメモリ管理機能－ヘテロ仮想記憶（ＨＶＳ）－”，情報処理学会第５３回全国大会，1B-3 (1996).