システムソフトウェアと
オペレーティング・システム 81－2
（１９９９．５．６）

Tender オペレーティングシステムにおける資源の永続化機構

谷口 秀夫　市川正也
九州大学　大学院システム情報科学研究科
E-mail: tani, ichikawa@csce.kyushu-u.ac.jp

要旨

Tender オペレーティングシステムにおいて、OS が制御し管理する対象の単位として実現している資源について、その永続化を行う機構を報告する。データに持続性を持たせるための永続化機能として、資源「プレート」について述べる。また、プレートを利用して ***Tender*** の資源を永続化する機構を述べる。さらに、実装と評価により、制御機構の性能を示す。

キーワード
オペレーティングシステム、永続化、資源、ソフトウェア、記憶

Resource Persistent Mechanism on *Tender*

Hideo TANIGUCHI and Masaya ICHIKAWA
Graduate School of Information Science and Electrical Engineering, Kyushu University
E-mail: tani,ichikawa@csce.kyushu-u.ac.jp

Abstract

We report persistent mechanism of resources that are controlled and managed by ***Tender*** operating system. Resource “plate” is persistent function to turn data into permanence. Resources of ***Tender*** are turned into permanence by “plate.” By implementation and evaluation, we show performance of mechanism furthermore.

Key-words
Operating System, Persistent, Resource, Software, Storage

１．はじめに

計算機を無停止で動かすことは難しいため、古くから、オペレーティングシステム（以降、OS と略す）は、応用プログラム（以降、AP と略す）が扱うデータを永続化（persistent）する機能を提供している。OS は、データを不揮発性の記憶媒体に保存することにより、計算機が再起動したときのデータの再利用を可能にしている。一方、最近では、計算機の動作停止が AP 処理に影響を与えないように、処理を継続動作

させるための永続化（enduring）機能をハードウェアで実現している。特に、ノート型のパソコンでは、この機能が提供されている。

データに持続性を持たせるための永続化（persistent）機能として、多くのOSでファイルを提供している。しかし、ファイルは外部記憶装置上に存在することを基本としているため、内容の操作はメモリ上にロードして行う必要がある。これを補う機能として、ファイルの内容を仮想記憶空間上にマップする機能がある。一方、***Tender*** オペレーティングシステム[1]では、データに持続性を持たせるための永続化（persistent）機能として、メモリ上に存在することを基本とするプレートを実現した。

ここでは、***Tender*** について、データに持続性を持たせるための永続化（persistent）機能を述べる。また、処理を継続動作させるための永続化（enduring）機能を支援するため、資源を永続化（persistent）する機構について述べる。具体的には、データに持続性を持たせるための永続化機能として、資源「プレート」について述べる。また、プレートを利用して ***Tender*** の資源を永続化する機構を述べる。さらに、実装と評価により、制御機構の性能を示す。ここで資源とは、OSが制御し管理する対象の単位である。

2．***Tender*** オペレーティングシステム

2．1　概要

Tender のプログラムは大きく三つの部分に分けられる。基盤部、表プログラム構造部、および拡張部である。基盤部は、ＯＳ動作の基盤となる処理を行う部分である。特に、資源インタフェース制御部は、***Tender*** 特有の部分であり、表プログラム構造と名づけたプログラム管理構造に基づき、資源を管理しているプログラム部分（以降、資源管理処理部と名づける）の呼出しを制御している。表プログラム構造部は、プロセス、プログラム、仮想ユーザ空間などの各資源管理処理部の集まりである。拡張部は、ＯＳ動作の拡張機能に位置づけられる処理を行う部分である。

Tender は、制御し管理する対象の単位を資源として管理する。資源には識別のために、文字列による名前（以降、資源名と呼ぶ）や数字による識別子（以降、資源識別子と呼ぶ）を付与する。また、資源の分離し独立化している。これにより、資源の事前用意や保留により、資源作成を伴う処理の高速化ができる。また、ＯＳの動作や内部状態の理解や把握が容易になり、ＯＳの理解を支援できる。更に、プログラムを部品化できるため、機能の追加や変更が容易になる。

メモリの管理機構を図１に示す。図１において、仮想領域は、外部記憶装置あるいは外部記憶装置と実メモリのデータ格納域を仮想化した資源である。仮想空間は、仮想アドレスの空間であり、仮想アドレスを実アドレスに変換する変換表（ページテーブルなど）に相当する。さらに、仮想領域を仮想空間に「貼り付ける」ことにより、仮想カーネル空間や仮想ユーザ空間を資源として作成できる。仮想カーネル空間や仮想ユーザ空間は、プロセッサが仮想アドレスでアクセスできる空間であり、各々、カーネルモードのみ、カーネルモードとユーザモードの両方、でアクセスできる。仮想カーネル空間に

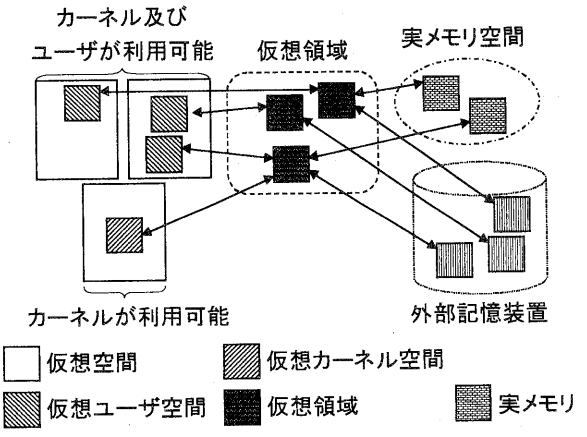


図1　メモリ管理機構

はＯＳがあり、仮想ユーザ空間にはプロセスのテキスト部やデータ部やユーザスタック部がある。

２．２　資源「プレート」

資源「プレート」は、データに持続性を持たせるための永続化（persistent）機能であり、既存OSのファイルに相当する。プレートの特徴を以下に示す。

（１）　仮想記憶空間上に存在する。
（２）　永続化（外部記憶装置への保存）は自動的に行われる。APが陽に永続化を要求することもできる。
（３）　仮想記憶空間上への読み込みは自動的に行われる。

特徴（１）が基本であり、特徴（２）と（３）は、APに対し、プレートが仮想記憶空間上に常に存在するように見せるためのものである。

既存OSのファイルは、外部記憶装置上に存在することを基本としている。このため、APは、ファイルの内容を操作するために、仮想記憶空間上にファイルの内容を読み込む必要がある。これに対し、プレートは仮想記憶空間上に存在することを基本としているため、APは直に内容を操作できる。つまり、APのデータ操作性が向上する。このような概念によるデータの永続化は、仮想記憶空間の拡大と実メモリ量の増加により可能になった。現在、***Tender*** は32ビットのプロセッサ上で走行しているため仮想記憶空間は広くない。しかし、64ビットのプロセッサを利用することにより仮想記憶空間は大きくなり、このプレートの概念が有効に利用できる。

プレートと関連する資源の様子を図２に示す。プレートは、プログラムやプロセスおよび演算のように、APが利用している内容が制御管理の上からも明白であり、利用形態を意識した資源といえる。これに対し、仮想ユーザ空間や仮想カーネル空間は、プログラムが操作できる仮想記憶空間を提供しており、APが利用している内容に関知しない。つまり、利用形態を意識しない資源といえる。プレートやプログラムやプロセスおよび演算は、仮想ユーザ空間や仮想カーネル空間を利用している。仮想ユーザ空間や仮想カーネル空間におけるページ例外処理は、二つに分類される。ページ例外発生アドレスがプレートとして利用されている部分であれば、***Tender*** 拡張部の機能の一つである永続化制御により例外時処理が行われ、外部記憶装置の永続化領域との入出力が行われる。この領域はUNIXファイル形式である。これにより、***Tender*** が制御管理するプレートとUNIXが制御管理するファイルの間でのデータ授受を可能にしている。これに対し、ページ例外発生アドレスがプレートとして利用されていない部分であれば、***Tender*** 拡張部の機能の一つである仮想化制御により例外時処理が行われ、外部記憶装置の仮想化領域との入出力が行われる。仮想化領域は永続化領域と異なり、永続化機能は必要ないものの高速な処理が要求される。このため、外部記憶装置において、二つの領域は分割して存在する。

プレートが提供するインタフェースを表１に示す。表１に示すインタフェースは、プレートが ***Tender*** 核内の他のプログラムに提供するも

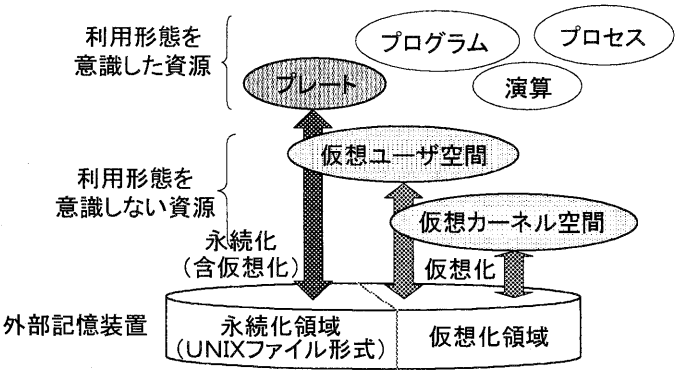


図2　資源「プレート」

表1　資源「プレート」の提供インタフェース

通番	形　式	機　能
1	create_plate(size,[vkmid/vumid],vrid,vmid,vaddr)	プレートを作成する。
2	delete_plate(plateid,type)	プレートを削除する。
3	get_plate(plateid,vmid,vaddr)	プレートを仮想空間に貼り付ける。
4	free_plate(plateid,vmid,vaddr)	プレートを仮想空間から剥がす。
5	control_plate(plateid,type,vaddr,size)	プレートの大きさを変更する。 プレートの内容を書き出す。

のであり、APに提供するものではない。プレートの作成と削除は、仮想記憶空間上で行われる。プレートを複数の仮想記憶空間の間で共有できるために、プレートの仮想空間貼り付けと剥がしの機能がある。また、プレートの大きさを変更でき、プレートの永続化を陽に要求できる。

先に述べた特徴（3）の処理は、OS立ち上げ処理の中で行われ、永続化制御の機能として実現している。

3．資源の永続化機構

処理を長続きさせるための永続化(enduring)機能を支援するため、資源を永続化（persistent）する基本機構について述べる。具体的には、プレートを利用して ***Tender*** の資源を永続化（persistent）する。資源を永続化するために必要なデータは全て仮想記憶空間上にあるため、仮想記憶空間上の存在を基本とするプレートを利用することにより、資源の永続化が簡単に行える。

永続化の基本機構を図3に示す。資源を管理処理する部分は、手続き部、データ部、管理表および資源の実体からなる。手続き部は、管理処理により変更されることはなく、かつOS核の実行プログラムとして外部記憶装置上に存在するため、永続化のための処理を必要としない。データ部や管理表および資源の実体は、管理処理により変更される。これらは、仮想カーネル空間や仮想ユーザ空間上に存在する。このため、仮想カーネル空間や仮想ユーザ空間がプレートを利用することで、データ部や管理表および資源の実体の永続化を行う。これらの対処は、プレートを含む全ての資源に対して行う。

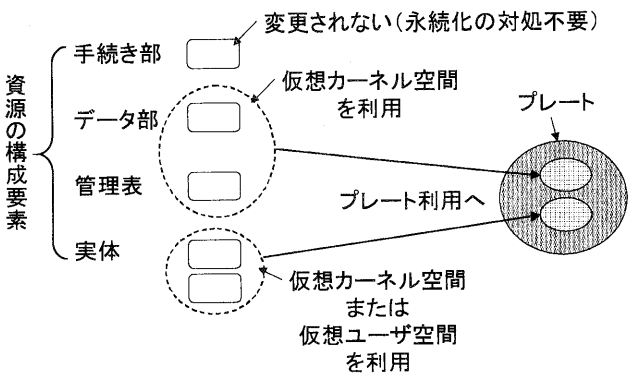


図3　永続化の基本機構

4．実装と評価

資源の永続化機構を ***Tender*** に実装し評価した。***Tender*** をPentiumプロセッサ（100Mhz）の計算機で走行させ、プレートに関係する処理の時間を測定した。処理時間は、同じ処理を20回繰り返し、1回あたりの時間を求めた。時間の計測は、Pentiumプロセッサの時間計測機能を利用した。以降の各図において、処理時間は実入出力に要する時間を除いた時間である。

プレートの作成処理時間として、新規に作成する場合、既存の仮想カーネル空間を利用する場合、および既存の仮想ユーザ空間を利用する場合について結果を図4に示す。図4から以下

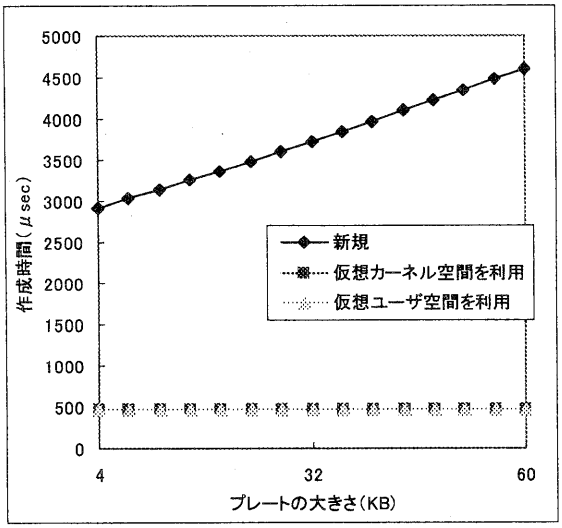


図4 プレートの作成処理時間(入出力時間を除く)

のことがわかる

(1) 新規に作成する場合、既存の仮想カーネル空間や仮想ユーザ空間を利用する場合に比べ、処理時間が大きい。また、作成するプレートの大きさに比例して処理時間も大きくなる。これは、新規に作成する場合、仮想空間を連続に確保するための確認処理、およびプレートの大きさ変更時の効率を向上させるために一つのプレートを複数の仮想領域を利用して構成する処理、に起因する。

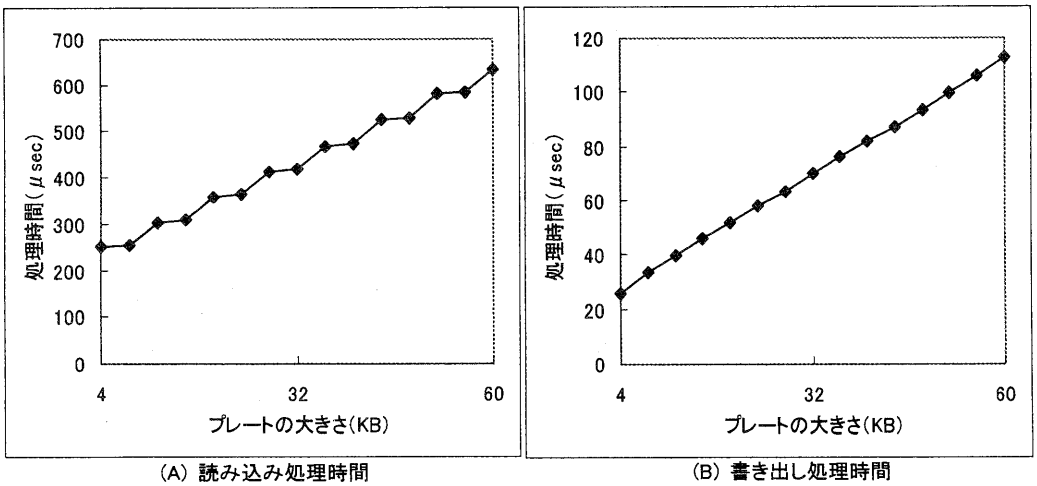
(2) 既存の仮想カーネル空間や仮想ユーザ空間を利用する場合、プレートとして制御管理するための登録処理が大半であるため、処理時間は、短く、かつ作成するプレートの大きさに影響されない。

プレートの入出力処理時間として、プレートへの読み込み処理時間とプレートの書き出し処理時間を図5に示す。図5から以下のことがわかる。

(1) 処理時間は、いずれの場合もプレートの大きさに比例して大きくなっている。これは、プレートの入出力をページの単位(4KB)で行っているためである。

(2) 読み込み処理時間は、プレートの名前から内容が格納されている外部記憶装置の位置までの変換を UNIX ファイル形式に基づいて行っているために、大きな値になっている。この処理は、OS 立ち上げのときのみに行うものであるため、高速化の配慮は行わなかった。

(3) 書き出し処理は、プレート識別子対応に内容を格納する外部記憶装置の位置を管理しているめ、処理時間は数μ秒と小



(A) 読み込み処理時間　　(B) 書き出し処理時間

図5 プレートの入出力処理時間(入出力時間を除く)

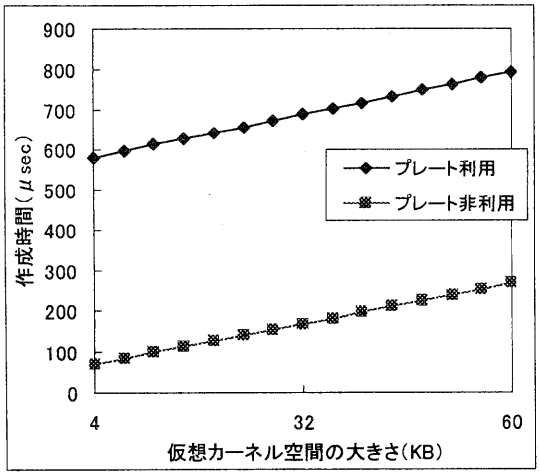


図6 仮想カーネル空間の作成処理時間(入出力時間を除く)

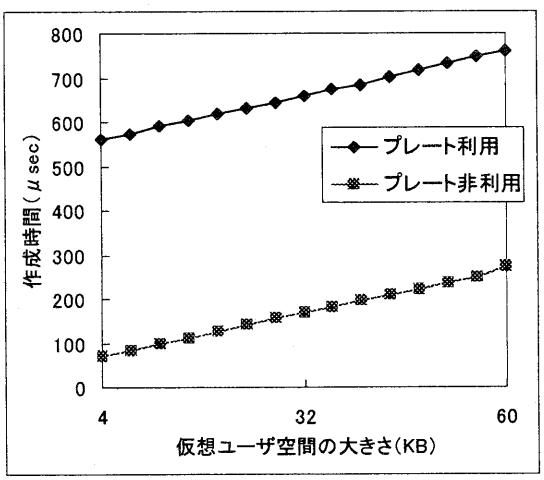


図7 仮想ユーザ空間の作成処理時間(入出力時間を除く)

さく、高速化な処理になっている。これは、この処理がページ例外により起動されることもあるためである。

仮想カーネル空間の作成処理時間として、プレートを利用する場合と利用しない場合について結果を図6に示す。図6より以下のことがわかる。

(1) 作成処理時間は、作成する仮想カーネル空間の大きさに比例し、次第に大きくなる。

(2) プレートを利用する場合と利用しない場合の差は約500μ秒である。

したがって、プレートを利用した仮想カーネル空間を作成する処理時間は、利用しない場合に比べ数倍の時間になるが、1ミリ秒以下の時間であるといえる。

仮想ユーザ空間の作成処理時間として、プレートを利用する場合と利用しない場合について結果を図7に示す。その傾向は、図6と同様である。したがって、プレートを利用した仮想ユーザ空間を作成する処理時間についても、利用しない場合に比べ数倍の時間になるが、1ミリ秒以下の時間であるといえる。

5. おわりに

Tender おいて、データに持続性を持たせる資源「プレート」について述べ、プレートを利用して ***Tender*** の資源を永続化する機構を述べた。さらに、実装と評価により、制御機構の性能を示した。仮想記憶空間上にデータが存在することを基本とするプレートの利用により、資源の永続化が簡単に行える。入出力時間を除くと、新たにプレートを作成する処理時間は数ミリ秒であり、既存の仮想カーネル空間や仮想ユーザ空間を利用してプレートを作成する場合の処理時間は約0.5ミリ秒である。また、ページ例外時に行うプレートの書き出し処理時間は数十μ秒である。さらに、プレートを利用して仮想カーネル空間や仮想ユーザ空間を作成する処理時間は、利用しない場合に比べ約500μ秒増加する。

今後は、処理を継続動作させるための永続化機能を実現する予定である。

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).