システムソフトウェアと
オペレーティング・システム 85－14
（２０００．８．４）

*Tender*におけるデータの永続化方式

稲本 慎司† 谷口 秀夫‡

†九州大学大学院システム情報科学府 ‡九州大学大学院システム情報科学研究院

〒 812-8581 福岡県福岡市東区箱崎 6-10-1
TEL 092(642)3867
Email : †inamoto@swlab.csce.kyushu-u.ac.jp ‡tani@csce.kyushu-u.ac.jp

要旨

メモリは揮発性記憶媒体であるため、メモリ上の内容を永続化するためには、磁気ディスク装置のような不揮発性記憶媒体にメモリの内容を保存する必要がある。多くのOSで利用されているファイルは、外部記憶装置に存在することを基本としているために、ファイルの内容に対して処理を行うには、その内容をメモリ上に読み込まなければならない。そこで、本稿では、メモリ上の内容を永続化する機能「プレート」について述べる。また、***Tender*** (The ENduring operating system for Distributed EnviRonment) において「プレート」を実現するための機構として、プレート管理処理部、永続化制御、およびUNIXファイル制御について述べる。

キーワード

オペレーティングシステム、永続化、記憶

Data Persistent Method on *Tender*

Shinji INAMOTO† and Hideo TANIGUCHI‡

Graduate School of Information Science and Electrical Engineering, Kyushu University

〒 812-8581 6-10-1 Hakozaki, Higashi-ku, Fukuoka, Japan
TEL 092(642)3867
Email : †inamoto@swlab.csce.kyushu-u.ac.jp ‡tani@csce.kyushu-u.ac.jp

abstract

In case contents of memory should be turn into permanence, contents of memory are preserved in nonvolatile storage such as Magnetic disk, because of memory is volatile storage. Files are used by many operating system as the facility which turn contents of memory into permanence. But, to read/write contents of a file, operating system have to load them to memory, because of files exist in Magnetic disk. So, in this paper, we describe "plate" as the facility which turn contents of memory into permanence. And we describe Plate Management , Persistent Control , and UNIX File Control as mechanisms for "plate" on ***Tender***.

keywords

operating system, persistent, storage

1　はじめに

計算機で行われる処理は、プロセッサによるメモリ上の内容の操作を伴う。メモリは揮発性記憶媒体であるため、メモリ上の内容を永続化するためには、磁気ディスク装置のような不揮発性記憶媒体にメモリの内容を保存する必要がある。不揮発性記憶媒体に保存することにより、計算機の再起動後も保存した内容を再利用することができる。一方、膨大な計算量やメモリ量を必要とする応用プログラム (以降、APとする) に対しては、計算機の停止によって受ける影響を極力抑えるために、処理を継続的に動作させる機能が求められる。

多くのオペレーティングシステム (以降、OSとする) では、メモリ上の内容を磁気ディスク装置のような外部記憶装置にファイルとして保存することにより、メモリ上の内容を永続化している。しかし、ファイルは外部記憶装置に存在することを基本としているために、ファイルの内容に対して処理を行うには、APはその内容をメモリ上に読み込まなければならない。また、APやハードウェアによって提供される処理を継続的に動作させる機能は、AP作成が複雑になるという問題や費用がかかるという問題がある。

そこで、本稿では、***Tender***[1]* においてメモリ上の内容を永続化し、さらに処理を継続的に動作させる機能を支援する機能について述べる。具体的には、メモリ上の内容を永続化する機能として、「プレート」[2] について述べる。また、***Tender*** において「プレート」を実現するための機構として、プレート管理処理部、永続化制御、およびUNIXファイル制御について述べる。現バージョンの ***Tender*** ver.5.0から ***Tender*** ver. 6.0への改版で、これらの機構の再構成や作成を行い、「プレート」を実現する予定である。

2　プレート

プレートとは、データを永続化するもので、既存OSのファイルに相当する。プレートとファイルの比較を表1に示す。両者には、2つの相違点がある。

1つ目の相違点は、プレートがメモリ上に存在することを基本としているのに対し、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ことである。図1に示すように、プレートはメモリ上に存在するのでプレートを外部記憶装置上の領域に対応させ、その領域にプレートの内容を保存することによりプレートの永続化を行う。外部記憶装置上の領域としてファイルを利用する。

2つ目の相違点は、外部記憶装置に対する内容更新の要求の主体である。プレートを利用していると、図1に示すように、メモリ上のプレートの内容を保存する場合、OSが外部記憶装置への入出力要求と入出力処理を行う。一方、ファイルを利用していると、ファイルに対応するメモリ上の領域の内容を保存する場合、APが外部記憶装置への入出力要求を行い、OSが入出力処理を行う。

プレートを利用することによって、以下の3つの利点を持つ。

(1) データ操作の簡易化
(2) データの永続化の容易化
(3) OSを永続動作させるための処理の容易化

ファイルの内容を操作するには、APはメモリ上に領域を確保し、外部記憶装置からファイルの内容をメモリ上に読み込まなくてはならない。しかし、プレートはメモリ上に存在するこ

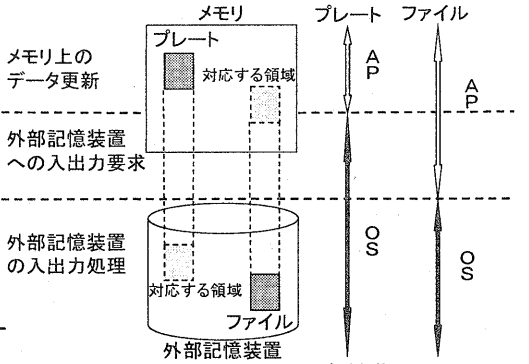

図1　プレートの永続化

* 論文中では、資源「プレート」を、資源「データ」と表記しているが、その後改名した。

表1 プレートとファイルの比較

永続化方式	存在する場所	外部記憶装置に対する内容更新の要求
プレート	メモリ	OSが要求
ファイル	外部記憶装置	APが要求

とを基本としているので、APは外部記憶装置からの読み込みを行わずに、プレートの内容を操作することができる。つまり、APはデータ操作がファイルに比べ簡易になる。

計算機の停止に備えてAPの走行中にAPのデータを保存したい場合がある。この場合、ファイルを利用していると、APが外部記憶装置への入出力要求を行うので、APが自身のデータ保存を意識しなければならない。一方、プレートを利用する場合には、OSが外部記憶装置への入出力要求を行うので、APは自身のデータ保存を意識する必要がなく、APのデータ保存が容易になる。

「OSを永続動作させる」とは、計算機の定期的な停止、もしくは緊急停止が起こった場合、システムを再開したときに、OSの状態を停止する前と同じ状態に復元することである。これは、OSが管理情報を永続化することによってなされる。ファイルを利用していると、メモリ上の内容のうち、どれがOSの永続動作に必要な管理情報であるかを判断できないので、メモリ上の内容全てを永続化しなければならない。しかし、プレートを利用していると、OSの永続動作に必要な管理情報をプレートとして、メモリ上に存在させることにより、必要な管理情報を永続化することができる。

一方、プレートを利用することによって、OS起動時の処理が増加するという欠点が生じる。プレートはメモリ上に存在することを基本としているので、プレート機能を有するOSでは、OS起動時に、外部記憶装置上に保存されている内容を元に、以前、存在したプレートをメモリ上に復元しなければならない。したがって、OS起動時に、プレートを復元するためにプレート用のメモリ空間を確保し、プレートの内容を外部記憶装置からメモリ上に読み込む処理が必要である。このために、OS起動時の処理が増加してしまう。この対処として、オンデマンド・ページング機能の利用が考えられる。OS起動時にはプレートに仮想メモリ空間を割り当て、実際の読み込みは行わない。当該プレートが実際にアクセスされた時点で、プレートの内容を外部記憶装置からメモリ上に読み込む。この際、メモリ常駐が必要なプレートのみ、初期化時に外部記憶装置からメモリ上に読み込むことも考えられる。

なお、プレートは広いメモリ空間を必要とする。これに対し、最近、メモリの低価格化により、計算機が持つメモリが大容量化している。したがって、メモリ上により多くのデータを存在させることが可能となる。また、プロセッサの64ビット化により、広い仮想記憶空間を得ることができる。これらにより、大容量化している外部記憶装置上の全てのデータを仮想記憶空間上に存在させることが可能になると推察する。

3 資源「プレート」

3.1 *Tender*

Tenderのプログラム構造を図2に示す。プログラムは大きく3つの部分に分けられる。それは、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の動作の最も基盤

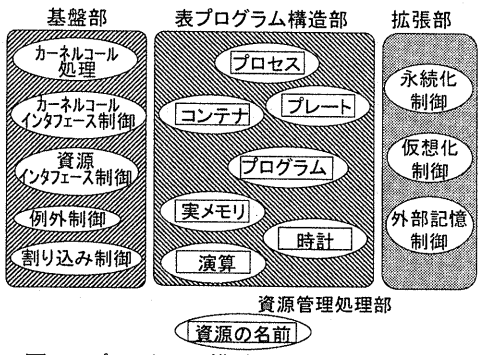

図2 プログラム構造 (***Tender*** ver.5.0)

表 2 AP に提供する資源「プレート」のインタフェース

通番	形式	機能
1	platecreate(name,size)	新規にプレートを生成する。もしくは、既存ファイルを利用してプレートを生成する。
2	platedelete(plateid)	プレートを削除する。
3	plateattach(plateid,vaddr,access)	ユーザがプレートを使用できるようにする。
4	paltedetach(plateid,vaddr)	プレートの内容を永続化し、ユーザがプレートを使用できないようにする。
5	platepersist(plateid)	プレートの内容を永続化する。

となる処理を行う。特に、資源インタフェース制御 (RIC : Resource Interface Controller) は、***Tender*** 特有の部分で、表プログラム構造と名付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 (資源管理処理部) の呼び出しを制御している。ここで資源とは、OS が制御し管理する対象の単位であり、全ての資源に資源識別子が付与され、この識別子により資源を一意に認識することができる。表プログラム構造は、プロセス、プログラム、プレートなどの資源管理処理部の集まりである。各資源管理処理部が提供するインタフェースは操作内容によって open、close、read、write、control の 5 つのタイプに分けられ管理される。拡張部は、OS 動作の拡張機能に位置付けられる処理を行う。

Tender におけるプロセスとメモリ関連資源の関係について、図 3 に示す。資源「仮想領域」は、外部記憶装置あるいは実メモリのデータ格納域を仮想化した資源である。資源「仮想空間」とは、一定の大きさを持つ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に相当する。さらに、資源「仮想領域」を資源「仮想空間」に「貼り付ける」ことにより、資源「仮想カーネル空間」や資源「仮想ユーザ空間」を生成できる。貼り付けるとは、仮想アドレスと実アドレスを対応付けることである。また、仮想アドレスと実アドレスの対応付けを解除することを「剥す」という。資源「仮想カーネル空間」や資源「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

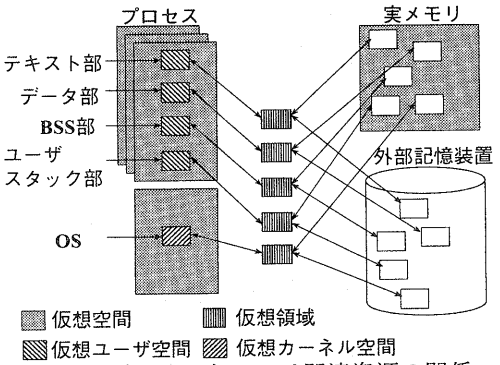

図 3 プロセスとメモリ関連資源の関係

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。資源「仮想カーネル空間」には OS があり、資源「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。

3.2 資源「プレート」の実現方式

Tender ver.6.0 で、AP に提供する資源「プレート」のインタフェースを表 2 に示す。platecreate() は、新規にファイルを生成して資源「プレート」を生成する方法と、ファイルシステム上のファイルを利用して資源「プレート」を生成する方法の 2 つの生成方法を持つ。platedelete() は、資源「プレート」の削除に伴って、対応するファイルの削除も行う。plateattach() は、ユーザが資源「プレート」を使用できるように資源「仮想ユーザ空間」を生成する。また、引数 access を指定することにより、資源「プレート」に書き込みを可能にするか、不可能にするかを決める。platedetach() は、資源「プレート」の内容を対応するファイルに書き

出し、plateattach() によって生成した資源「仮想ユーザ空間」を削除することにより、ユーザが資源「プレート」を使用できないようにする。platepersist() は、資源「プレート」の内容を対応するファイルに書き出し、資源「プレート」を永続化する。永続化の対象は、引数 plateid で指定した資源「プレート」もしくは、全ての資源「プレート」である。

3.2.1 処理構成

図2に示したように、***Tender*** ver.5.0 では、資源「プレート」をプレート管理処理部が管理し、プレートの永続化を永続化制御が行っている。ここで、図4に示すように、永続化制御の機能は、資源「プレート」を永続化 (persistent) する機能、UNIX ファイルシステムを操作する機能、および OS を継続的に動作させる永続化 (enduring) 機能の3つに分けることができる。そこで、***Tender*** ver.6.0 では、資源「プレート」を永続化する機能を永続化制御、UNIX ファイルシステムを操作する機能を UNIX ファイル制御、OS を継続的に動作させる機能を動作継続制御とし、分離することとした。各部分の構成位置について、以降に説明する。

先にも述べたように、基盤部は OS の動作の最も基盤となる処理を行い、拡張部は OS 動作の拡張機能に位置付けられる処理を行うものである。さらに、データを保存する観点から基盤部に必要な最低動作の要件を以下に述べる。

(1) AP やデータは不揮発性記憶媒体上に存在し、それらを利用してユーザプロセスが動作する。

(2) AP が生成したデータを不揮発性記憶媒体上に保存する手段を提供する。

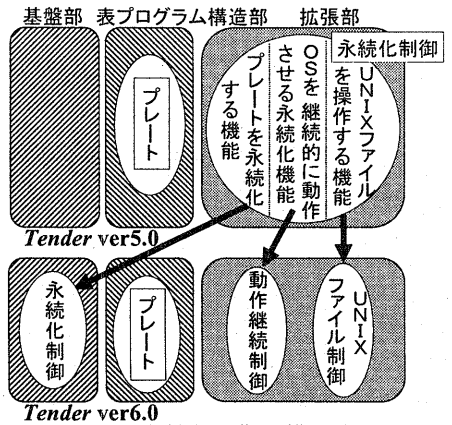

図4 永続化制御の構成変更

＜永続化制御の位置付け＞

最低動作を実現するためには、不揮発性記憶媒体上のデータの利用や不揮発性記憶媒体上へのデータの保存が必要不可欠である。したがって、不揮発性記憶の格納形態を管理する唯一の制御部である永続化制御を、基盤部として実現する。

なお、永続化制御と同様な制御部として、メモリと外部記憶装置の間でのデータ入出力を制御する仮想化制御がある。仮想化制御は、オンデマンド・ページング機能を提供する。オンデマンド・ページング機能がなくても必要な情報が全てメモリ上に存在すれば、最低動作を実現できる。したがって、仮想化制御を拡張部として実現する。

＜ UNIX ファイル制御の位置付け＞

最低動作を実現するには少なくとも1つの不揮発性記憶の格納形態を提供しなければならない。UNIX ファイルシステム以外の不揮発性記憶の格納形態が提供された場合、UNIX ファイル制御は必ずしも必要ではない。したがって、UNIX ファイル制御を拡張部として実現する。

＜動作継続制御の位置付け＞

動作継続制御は、OS を継続的に動作させるための制御部であり、最低動作を実現するには必要のない制御部である。したがって、動作継続制御を拡張部として実現する。

Tender ver.6.0 で、資源「プレート」を実現するには、プレート管理処理部、永続化制御、UNIX ファイル制御が必要である。これらについて以降に述べる。

3.2.2 プレート管理処理部

プレート管理処理部は、メモリ上の資源「プレート」の管理と永続化を行う。また、プレート管理処理部は、表プログラム構造部の一部として実現する。

メモリ上の資源「プレート」の管理と永続化のためにプレート管理処理部は次の機能を持つ。

(1) プレートの生成
(2) プレートの削除
(3) プレートの貼り付け
(4) プレートの剥し
(5) プレートの書き出し
(6) プレートの大きさ変更

プレートの生成は、新規に生成する方法、指定したファイルを利用する方法、資源「プレート」の資源識別子を指定しファイルを利用する方法、資源「仮想カーネル空間」を利用する方法、資源「仮想ユーザ空間」を利用する方法の5つの方法がある。資源「プレート」の資源識別子を指定しファイルを利用する方法は、OS起動時の資源「プレート」の復元を行う際に利用され、資源「プレート」の復元の前後で資源識別子が変化しないようにする。これは、資源は資源識別子によって識別されており、資源識別子が変化すると利用に支障をきたすためである。

プレートの貼り付けは、資源「仮想ユーザ空間」を生成することにより、ユーザが資源「プレート」を利用できるようにする。また、プレートの貼り付けの機能を提供することにより、資源「プレート」を複数の仮想記憶空間の間で共有することが可能となる。

プレートの剥しは、プレートの貼り付けで生成した資源「仮想ユーザ空間」を削除することにより、ユーザが資源「プレート」を利用できないようにする。

プレートの書き出しは、資源「プレート」の内容を対応するファイルに書き出す。書き出しの対象は、特定の資源「プレート」もしくは、全ての資源「プレート」である。

プレートの大きさ変更は、資源「プレート」の大きさの拡張と縮小を行う。

3.2.3 永続化制御

永続化制御は、抽象化された不揮発性記憶の格納形態を管理し、メモリ上の資源「プレート」の内容を永続化する。

＜不揮発性記憶の格納形態の抽象化＞

Tender ver.5.0では、不揮発性記憶の格納形態としてUNIXファイルシステムを利用している。永続化制御は資源「プレート」をファイルに対応付けることが可能であれば良いので、不揮発性記憶の格納形態はUNIXファイルシステムに限定される理由はない。***Tender*** ver.6.0では、永続化制御で不揮発性記憶の格納形態を抽象化して管理することによって、複数種類の不揮発性記憶の格納形態を管理することを可能にする。これにより、資源「プレート」の可搬性を増大させることができる。現在、不揮発性記憶の格納形態として、UNIXファイルシステムやFAT(File Allocation Table)ファイルシステムを想定している。図5に示すように永続化制御はUNIXファイル制御以外の不揮発性記憶の格納形態を提供する制御部を複数呼び出すことができる。ただし、資源「プレート」の内容を永続化するには、永続化制御は少なくとも1つの不揮発性記憶の格納形態を管理しなければならない。***Tender*** ver.6.0では、UNIXファイル制御によって、不揮発性記憶の格納形態としてUNIXファイルシステムを提供する。

＜永続ユニット＞

永続化制御では、永続ユニットを管理対象とする。永続ユニットとは、永続化を行う単位である。

Tender ver.6.0では、不揮発性記憶の格納形態を抽象化して管理する構造が必要である。そこで、永続ユニットを利用する。

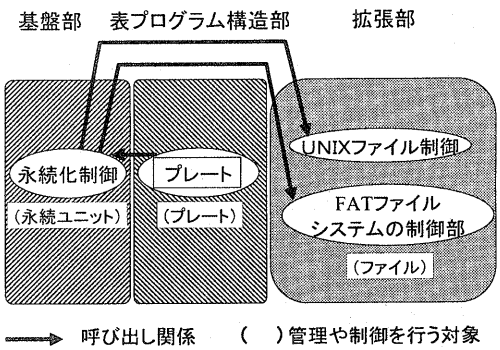

図5 各制御部と管理部の関係

＜永続化制御の機能＞

永続化制御は次の機能を持つ。

(1) 永続ユニットの生成
(2) 永続ユニットの削除
(3) 永続ユニットの読み込み
(4) 永続ユニットの書き出し
(5) 永続ユニットの大きさ取得
(6) 不揮発性記憶の格納形態上の領域の予約
(7) 不揮発性記憶の格納形態上の領域の解放
(8) 資源「プレート」の復元

永続ユニットの生成と永続ユニットの削除は、資源「プレート」に対応するファイルの生成、もしくは削除を行うようにUNIXファイル制御を呼び出し、永続ユニットを生成、もしくは削除する。永続ユニットの読み込みは、資源「プレート」に対応するファイルの内容をメモリ上の資源「プレート」に読み込むようにUNIXファイル制御を呼び出す。永続ユニットの書き出しは、資源「プレート」に対応するファイルへ資源「プレート」の内容を書き出すようにUNIXファイル制御を呼び出す。永続ユニットの書き出しは、特定の資源「プレート」のみの書き出し、もしくは存在する全ての資源「プレート」の書き出しを行う。永続ユニットの大きさ取得は、資源「プレート」に対応するファイルの大きさを取得するようにUNIXファイル制御を呼び出す。不揮発性記憶の格納形態上の領域の予約は、資源「プレート」に対応するファイルを生成するための領域を予約するようにUNIXファイル制御を呼び出す。不揮発性記憶の格納形態上の領域の予約は、資源「プレート」に対応するファイルを生成するために予約していた領域を解放するようにUNIXファイル制御を呼び出す。資源「プレート」の復元機能は、システム再開時に行われ、以前に存在した資源「プレート」に対応するファイルの内容をメモリ上に読み込み、資源「プレート」を復元する。この機能は、資源「プレート」がメモリ上に存在することを基本としているために必要な機能である。

3.2.4 UNIXファイル制御

UNIXファイル制御は、UNIXファイルシステムの操作を行い、永続化制御に不揮発性記憶の格納形態としてUNIXファイルシステムを提供する。

＜UNIXファイル制御の機能＞

UNIXファイル制御は次の機能を持つ。

(1) ファイルの生成
(2) ファイルの削除
(3) ファイルの読み込み
(4) ファイルの書き出し
(5) ファイルの大きさ取得
(6) UNIXファイルシステム上の領域の予約
(7) UNIXファイルシステム上の領域の解放

ファイルの生成は、指定された名前で指定されたサイズのファイルを生成する。ファイルの削除は、指定された名前のファイルを削除する。ファイルの読み込みは、指定されたファイルの内容をメモリ上の指定されたアドレスに読み込む。ファイルの書き出しは、指定されたファイルにメモリ上の指定されたアドレスの内容を書き出す。ファイルの大きさ取得は、指定されたファイルのサイズを取得する。UNIXファイルシステム上の領域の予約は、指定されたサイズの領域を予約する。UNIXファイルシステム上の領域の解放は、指定された領域を解放する。

3.3 実装と評価

Tender ver.5.0を改版し、資源「プレート」を実現するために必要なプレート管理処理部、永続化制御、およびUNIXファイル制御を実装し、評価した。

測定は、PC/AT互換機 (Pentium 90MHz、MEM 80MB、HDD 1.7GB(IDE)) を使用した。不揮発性記憶の格納形態であるUNIXファイルシステムには、BSD/OSver.3.1のFFS(Fast File System) を使用した。測定カウンタは、ハードウェアクロックカウンタを使用し、2ヶ所のカウンタ値の差分をクロック数で割ることにより処理時間を算出した。処理時間の測定は、同じ処理を20回繰り返し、1回当たりの処理時間を求めた。

ユーザプログラム上で資源「プレート」を新

規に生成する処理時間を図6に示す。この処理時間は外部記憶装置との入出力時間を除いた時間である。図6より、入出力時間を除いた資源「プレート」の生成処理時間は、生成する資源「プレート」の大きさに比例することがわかる。これは新規に資源「プレート」を生成する場合、メモリを確保するための処理が必要であることに起因する。メモリはページ単位で確保されるため、確保するメモリの大きさに比例してメモリの確保処理時間も大きくなる。

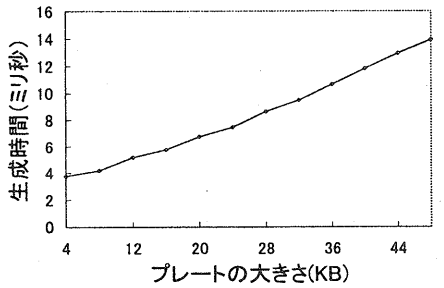

図 6 新規にプレートを生成する処理時間 (入出力時間を除く)

4 おわりに

メモリ上の内容を永続化する「プレート」と、***Tender***において資源「プレート」を実現するための機構として、プレート管理処理部、永続化制御、およびUNIXファイル制御について述べた。「プレート」はメモリ上に存在することを基本とするという特徴と、メモリ上の「プレート」から外部記憶装置上への内容の更新をOSが要求し、実施するという特徴を持つ。この特徴により、データ操作の簡易化、データの永続化の容易化、およびOSを永続動作させるための処理の容易化という利点を持つ。また、永続化制御は抽象化した不揮発性記憶の格納形態を管理するため、資源「プレート」の内容を格納する不揮発性記憶の格納形態として、複数種類のファイルシステムを利用することができる。また、新規に資源「プレート」を生成する処理時間を示した。

今後、資源「プレート」を利用して、***Tender***上の処理を継続的に動作させる永続化 (enduring) 機能を実現する予定である。

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] 谷口秀夫, 市川正也, "*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 ", 情処研報, Vol.99, No.32, pp.7-12 (1999).