

Tender における異種 OS インタフェースの共存手法

3U-03

田端 利宏†

野口 直樹‡

中島 耕太†

谷口秀夫††

†九州大学大学院システム情報科学府

‡九州大学工学部電気情報工学科

††九州大学大学院システム情報科学研究院

1 はじめに

我々は、**Tender**オペレーティングシステム (The ENduring operating system for Distributed EnviRnment)^[1]を開発している。**Tender**には、我々が提案した新たな機能をいくつか実装しており、独自のインタフェースを持つ。**Tender**の機能と他のオペレーティングシステム (以降、OS と略す) の機能の比較を考えたとき、OS のインタフェースが異なるため、同一条件での評価が難しい。また、他 OS の応用プログラム (以降、AP と略す) を**Tender**用に修正し、移植すると、AP の振る舞い変更されるため、機能の比較をうまくできない可能性がある。

そこで、他 OS と同一条件での機能比較を実現するために、他 OS のインタフェースを**Tender**に実装し、同一の AP で機能を比較評価できる環境を構築する。本稿では、プロセスの生成処理を例に挙げ、**Tender**に UNIX インタフェース (ここでは、BSD/OS を例に挙げる) も共存させる方式を述べる。また、プロセスの生成と消滅処理に関しての比較評価結果を報告する。

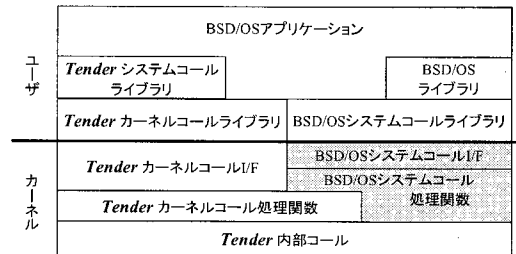
2 **Tender**オペレーティングシステム

Tenderでは OS の操作する対象を資源として、分離し独立化した。資源には、資源名と資源識別子を付与し、資源操作のインタフェースを統一している。さらに、各資源を操作するプログラム部品を資源毎に分離し、共有プログラムを排除した。また、各資源の管理情報も資源毎に分離し、各資源の管理表の間のポインタを禁止した。このように、資源の分離と独立化を行うことで、資源の事前生成や保留により、資源の作成や削除を伴う処理を高速化している^[1]。

3 OS インタフェース共存手法

OS インタフェース共存に対する要求条件として、AP のソースコードを修正せず、実行ファイルをそのまま利

Coexistence of Other OS Interface on **Tender**
Toshihiro TABATA†, Naoki NOGUCHI‡, Kohta NAKASIMA† and Hideo TANIGUCHI††
†, ††Graduate School of Information Science and Electrical Engineering, Kyushu University
‡Department of Electrical Engineering & Computer Science, Kyushu University

図1 **Tender** と BSD/OS AP の関係

用できることがある。この要求を満たすために、以下の対処を行う。

- (1) 実行ファイルの形式をサポートする。
- (2) BSD/OS のシステムコール互換のインタフェース (以降、I/F と略す) を実現する。

そこで、BSD/OS のシステムコール互換の I/F を**Tender**に実装した。その様子を図1に示す。**Tender**では、BSD/OS 3.1の実行形式 (a.out) をサポートしているため、BSD/OS プログラムから、プロセスを生成し実行することができる。また、BSD/OS システムコール処理関数を新たに**Tender**に実装した。これにより、カーネルはBSD/OS AP の発行するシステムコール番号と引数を取得することができる。また、**Tender**カーネルコール I/F は、これとは独立して存在する。

BSD/OS システムコール処理関数では、発行されたBSD/OS のシステムコールに相当する処理を実行し、戻り値を返す。この処理は、**Tender**の内部コールやカーネルコール処理関数を利用して実現している。

4 プロセスの生成と消滅のインタフェースの比較

ここでは、プロセスの生成と消滅の処理を例に挙げ、インタフェースの違いについて述べる。UNIXでは、fork システムコールにより、システムコールを発行したプロセス (親) と同じプロセス (子) を生成する。それから、必要に応じて exec システムコールを発行し、新たにプロセスを実行させる。一方、**Tender**では、表1に示すように、指定したプログラム (plateid) から、新規のプロセスを生成する。つまり、UNIXのように、親プロ

表1 *Tender* のプロセスの生成と消滅の I/F

形式	機能
open_proc(plateid, arg, vmid)	plateid で指定されたプログラムを vmid で指定された仮想空間上にプロセスとして生成する。
close_proc(rflag)	rflag で指定された資源を保存し、プロセスを消滅させる。

セスから子プロセスを生成する機能はない。プロセスの消滅処理では、再利用する資源を指定することができる。

5 性能評価

Tender 上に、fork システムコール処理関数を実装し、BSD/OS 3.1 と処理時間を比較した。fork システムコール処理では、open_proc 関数で新規にプロセスを生成し、プロセスの状態を親プロセスと同じ状態に変更する。このとき、プロセス構造体の一部を複製し、親プロセスから資源を引き継ぐ処理を実現している。

Tender 上で、評価用プログラムを実行し、その処理時間を測定した。測定には、PentiumIII 750MHz の計算機を利用した。このプログラムは、fork システムコールを実行し、子プロセスの実行終了を待つ処理を1000回繰り返す。測定対象のプログラムは、テキスト部20KBで固定とし、BSS部の大きさを固定(0KB)としデータ部の大きさを可変とした場合と、データ部の大きさを固定(4KB)としBSS部の大きさを可変とした場合について処理時間を測定し、その一回当りの処理時間を算出する。また、*Tender* ではプロセスを構成する資源を再利用できるため、資源再利用ありとなしの場合について測定した。BSD/OS については、プロセスの大きさに依存せずにプロセスを生成できるため、一度測定した結果を図に載せている。データ部を可変とした場合の処理時間を図2、およびBSS部を可変とした場合の処理時間を図3に示す。

評価結果から、データ部とBSS部の大きさに比例して処理時間が増加することがわかる。*Tender* では、プロセス生成時にメモリにプロセスの内容をすべて読み込む。このため、プロセスの生成後にデータ部とBSS部の内容を親プロセスから複製すると、2回複製することになるため、処理のオーバーヘッドが大きい。すべての資源を再利用した場合には、データ部とBSS部の大きさが小さい場合には、BSD/OSのプロセス生成よりも高速である。例えば、WebサーバApache 1.3.12のプ

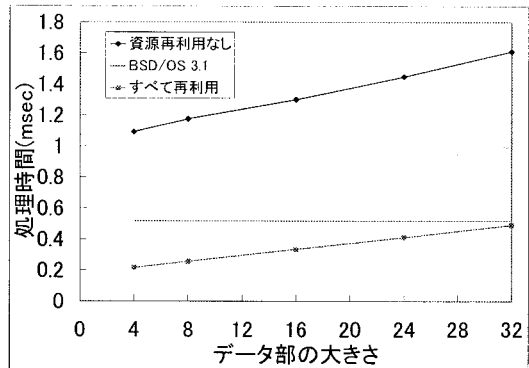


図2 プロセスの生成と消滅時間 (データ部可変)

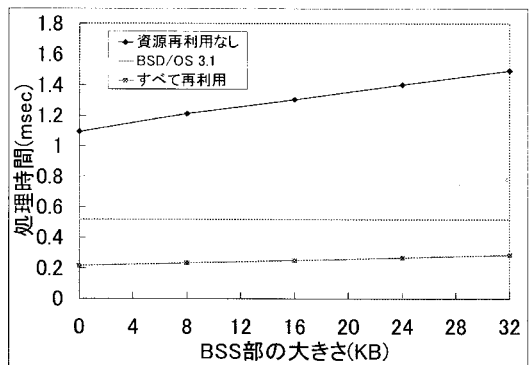


図3 プロセスの生成と消滅時間 (BSS部可変)

ログラムのデータ部の大きさは32KBで、BSS部の大きさは約34KBであった。また、*Tender* では、プロセス生成時にプロセスを実メモリ中に読み込んでいるため、ページ例外は発生しない。さらに、データ部とBSS部に関する複製回数を減らすことで、処理を高速化できる。

6 おわりに

本稿では、OSの機能比較のための*Tender*へのBSD/OSシステムコール互換のI/Fの実装方式とプロセスの生成と消滅処理の評価結果を述べた。今後の予定として、実アプリケーションでの評価がある。

参考文献

- [1] 谷口秀夫, 青木義則, 後藤真孝, 村上大介, 田端利宏: 資源の独立化機構による*Tender*オペレーティングシステム, 情報処理学会論文誌, Vol.41, No.12, pp.3363-3374 (2000).