システムソフトウェアと
オペレーティング・システム 83－1
（2000．2．24）

入替え条件を緩和した実行中プログラム部分入替え法の評価
—— 入替え処理速度の高速化 ——

中島雷太† 谷口秀夫†

我々は、実行中プログラムの一部分を入れ替える新たな制御法を提案している。具体的には、プログラム実行状態を分類し、入替え可能条件、プログラム状態の管理法、入替え法、サービスへの影響を軽減する方法を示した。また、出力処理を主に行なうプログラムとプロセッサ処理を主に行なうプログラムについて評価を行ない、プロセッサ処理を多く伴うほど、入替え時間が増加するという結果を示した。さらに、他プロセスが同時走行している場合や入替え条件を緩和した場合にも同様の問題があることを明らかにした。本論文では、入替え処理を高速化することで、増加した入替え時間の短縮を図る。具体的には、入替え処理を高速化する手法を提案し、その効果について論じる。

Evaluation of Mechanism for Exchanging Program Part of Running under the Relaxed Conditions of Exchange
—— speed up of the exchange process ——

RAITA NAKAJIMA† and HIDEO TANIGUCHI†

We have proposed that the mechanism of exchanging program part of running. To be concrete, we have described problems that how to classify the state of executed program parts, to make conditions to be able to exchange program parts clear, how to manage the state of executed program parts, how to guarantee an exchange in finite time, and how to reduce the influence on service. In this paper, we describe the problem about the exchanging time of this mechanism and suggest that how to speed up of the exchange process. we also report the evaluation of the exchanging time and the availability of this suggestion.

1. まえがき

近年、計算機の普及によって計算機を利用する環境が急激に変化しつつある。このような状況において計算機の提供するサービスは、その計算機の利用環境に素早く対応することが望ましい。また、計算機による長期的なサービスの提供では、社会に与える影響を考慮すると、その保守作業などは速やかに完了することが望ましい。

このような背景から我々は、計算機上で実行中のプログラムについて、機能の一部分を動的に入れ替える実行中プログラム部分入替え法を提案した[1]。

同様の関連研究として、プロセスを冗長構成にすることでプロセス単位で入れ替える方法[2]や、プロセスを走行させたままプログラムの一部分を入れ替える方法[3]が提案されているが、少しの変更でもプロセス全体を入れ替える必要があったり、オペレーティングシステム（以降OSと略す）が入替えの契機を考慮する必要があるといった問題点がある。また、OS上での動的再構築に関する研究[4)~6)]も行なわれているが、これらはOS機能を対象としたものであり、アプリケーション（以降APと略す）は対象とはしていない。

これらの関連研究に対し、我々が提案したプログラム入替え法は、次の大きな二つの特徴を持つ。一つは、サービスプログラムが、入替え契機を意識する必要がない点である。もう一つは、入替え対象となっているプログラム部分が、入替え対象でない別のプログラム部分を呼び出している場合にも、プログラムの入替えを可能としている点である。さらに、複数のプロセス間で共有されたプログラム部分の入替えも考慮している[7]。

これまでに、基本方式を提案し、入替え要求から入替え処理が終了するまでの時間（以降「入替え時間」と呼ぶ）の観点から評価を行い、定式化も行なった[7]。具体的には、主に入出力処理を行なうプログラム（以降「入出力処理プログラム」と呼ぶ）を評価対象とし、それに対する入替え時間を示し、評価を行なった。しかし、一

† 九州大学大学院システム情報科学研究科
Graduate School of Information Science and Electrical Engineering, Kyushu University.

般的なプログラムを考えたとき、入出力処理プログラムに対する評価だけでは十分とは言えない。そこで、主にプロセッサ処理を行なうプログラム (以降「プロセッサ処理プログラム」と呼ぶ) に対する入替え時間についての評価を行ない、結果として、入出力処理プログラムに比べ、プロセッサ処理プログラムに対する入替えは、入替え時間が大きく増加することを示した[8]。また、入替え対象であるプログラム部分を共有しないプロセス (以降「他プロセス」と呼ぶ) が同時走行している場合についての評価も行ない、結果として、プロセッサ処理プログラムに対する入替え処理よりも、さらに入替え時間が増加することを示した[9]。また、基本方式における入替え条件の観点からの評価も行ない、入替え条件の定義によっても入替え時間が大きく変化することを明らかにした[10]。

本論文では、基本方式で提案している入替え条件に着目し、文献10) において明らかにした入替え時間が増加する問題への対処法について述べる。また提案した対処法について、入替え時間の観点から評価を行ない、その有効性を示す。

2. 入替え法

2.1 基本方式の概要

プロセスの走行中に、そのプログラムの一部分の入替えを可能にするには、以下の五つの課題について、どのように対処するかを明らかにする必要がある[1]。

(1) プログラム実行状態の分類
(2) 入替え可能条件
(3) プログラム実行状態の管理法
(4) 有限時間内での入替えを保証する制御法
(5) 対象プログラムが提供するサービスへの影響を軽減する方法

本論文では、特に重要な課題 (1)、課題 (2) についてのみ述べる。

実行中であるプログラムの一部分を入れ替えるには、どのような実行状態で入替えを行なうかを判断することが重要である。文献 1) では、図1に示すように入替え対象プログラム部分の実行状態を、以下の3つに分類している。

(1) 未使用: 実行する前、あるいは、実行を終えた状態
(2) 走行中: 実行中の状態
(3) 呼出中: 別のプログラム部分を呼び出している状態

また、プログラム部分の実行状態によって、以下の入替え条件を満たす必要がある。

(条件1) 呼び出しと返却のインタフェースの一致

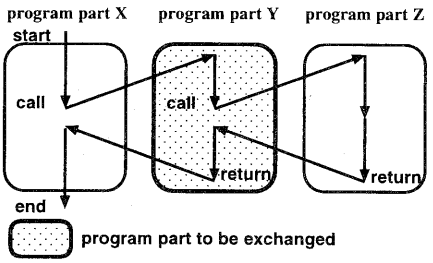

図1 状態と入替え対象プログラム部分の関係

表1 プログラム部分状態と入替え可能条件の関係

通番	実行状態	入替え可能条件
(1)	未使用	(条件1), (条件2)
(2)	走行中	入替え不可能
(3)	呼出中	(条件1), (条件2) (条件3), (条件4) (条件5), (条件6)
(4)	未使用かつ走行中	入替え不可能
(5)	未使用かつ呼出中	(条件1), (条件2) (条件3), (条件4) (条件5), (条件6)
(6)	走行中かつ呼出中	入替え不可能
(7)	未使用かつ走行中 かつ呼出中	入替え不可能

(条件2) 処理矛盾の回避
(条件3) 戻り値のアドレスを変更しない
(条件4) スタティックリンクの場合、メモリ上の外部変数の参照アドレスが同じであること
(条件5) 外部変数の値が入替えの前後で同じであることの保証が必要か否か
(条件6) アドレス渡しによる外部変数や内部変数の参照や更新がどのように行なわれているか

入替え対象プログラム部分が、複数のプロセスによって共有されている場合も含めたプログラム部分の実行状態と入替え可能条件の関係を、表1に示す。

2.2 入替え可能条件の緩和について

表1に示したように、入替えの契機として、「呼出中」の場合の入替えを考慮するか否かは、新たに入れ替えるプログラムを記述する際に要求される入替え可能条件に大きな影響を与える。

入替え条件として、入替え対象プログラム部分の実行状態が「未使用」または「呼出中」の時に入替えを可能とした場合と、「未使用」の時にのみ可能とした場合の長所、短所を考える。入替え可能条件の観点から見た場合、表1から「呼出中」の場合の入替えを考慮すると

表2　入替え可能な入替え対象モジュールの実行状態の違いによる相対的特徴

入替え可能な入替え対象モジュールの実行状態	入替え可能条件	入替え時間
「未使用」または「呼出中」	多い (6 条件)	短い
「未使用」のみ	少ない (2 条件)	長い

(条件 1) から (条件 6) の六つを満たしていなければならない。これに対し、「未使用」の時にのみ入替えを可能とした場合は (条件 1)、(条件 2) の二つのみであり、入替え可能条件が緩和されている。このため、提案する入替え法を利用者が利用しやすくなるという長所があると言える。

一方、入替え時間の観点から見た場合、「未使用」の時にのみ入替えを可能とすると、入替え契機が少なくなり、結果として入替え時間の増加につながると推察できる。

入替えが可能な入替え対象モジュールの実行状態について、入替え可能条件と入替え時間の相対的な特徴を表2に示す。

3. 入替え処理速度の高速化

3.1 問 題 点

基本方式が持つ問題点として、文献 8)、9) で入替え時間が増加する場合があることを示した。具体的には、入替え対象モジュールを含むプログラムの処理において、プロセッサによる処理時間が長いほど、入替え時間は長くなるという問題があることを明らかにした。さらに、文献 8) では、入替え条件の定義による入替え時間の違いについて、入替え条件を「未使用」の時にのみ入替え可能とした場合も、入替え時間が増加する問題点を明らかにしている。本章では、入替え条件の定義によって生じる入替え時間増加の問題点について述べる。

まず、文献 8) の測定結果を示す前に、その測定条件について説明する。図 2が測定に用いたプログラムの流れであり、module Y が入替え対象モジュールである。

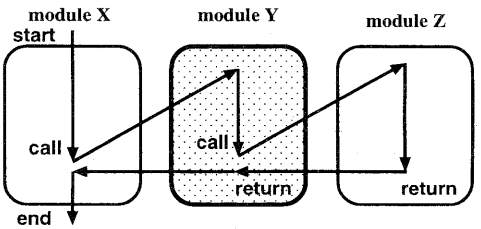

図2　プログラムの処理の流れ

モジュール処理時間の関係を表 3に示す。各モジュールの処理時間の総和を 10 秒、module Y の処理時間を一定 (1 秒) とした A から H の八つの場合を想定している。表 3のような設定にしたのは、入替え対象モジュールである module Y と、それが呼び出す module Z の処理時間の総和の違いによって入替え時間にどのような影響があるのかを明らかにするためである。各プロセスは疑似乱数を用いて不定間隔で起動し、module Y を共有するプロセス数が 1 から 10 までの場合について module Y に対して入替えを行ない、その入替え時間を測定している。

表3　プログラムの各モジュール処理時間 (秒)

type	module X	module Y	module Z
A	1	1	8
B	2	1	7
C	3	1	6
D	4	1	5
E	5	1	4
F	6	1	3
G	7	1	2
H	8	1	1

測定には二種類のテストプログラムを用いている。具体的には、入出力処理プログラムとして表 3で設定した処理時間だけ sleep 処理を行うプログラムを、プロセッサ処理プログラムとして表 3で設定した処理時間だけ特定の変数を 1 加算する処理を行うプログラムを用意した。また、他プロセスの存在が入替え時間にどのような影響を与えるかを明らかにするために、他プロセスが 1 つ同時走行している場合の入替え時間についても測定を行なっている。他プロセスは、プロセッサ処理プログラムと同様に、特定の変数を 1 加算する処理を繰り返すプログラムの実行プロセスである。測定は Pentium 90MHz の計算機上で行い、20 回の平均値を測定結果として記録したものである。

文献 8) の図 10 と図 11 について、縦軸の入替え時間の目盛り幅を統一になるように修正しグラフ化したものを図 3、図 4に示す。

図 3、図 4において、共有プロセス数が 10 の時の各 type での入替え時間に着目する。入替え条件を、「未使用」または「呼出中」の時に入替え可能とした場合の入替えでの入替え時間を 1 とした時、「未使用」の時にのみ入替え可能とした場合の入替えでは、6 ～ 10 倍の

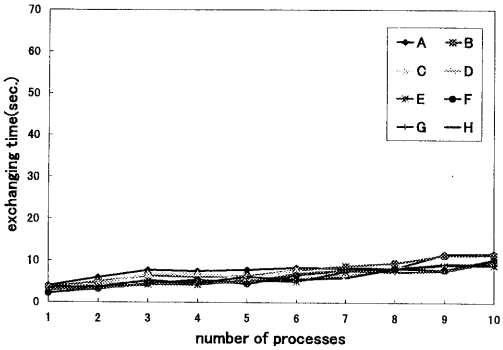

図3　プロセス数と入替え時間の関係
（入替え条件:「未使用」または「呼出中」の時入替え可能）

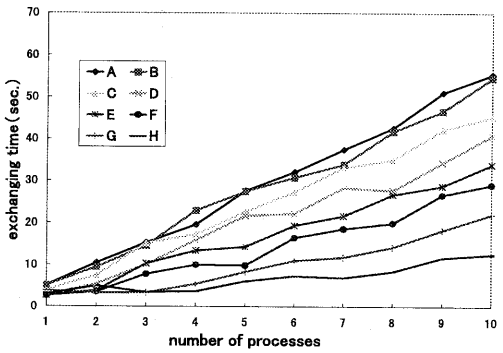

図4　プロセス数と入替え時間の関係
（入替え条件:「未使用」の時にのみ入替え可能）

入替え時間を要することがわかる。また、他プロセスが1つ同時走行している場合には、図3、図4とほぼ同様の傾向を示し、7～12倍の入替え時間となり、一層遅くなることが明らかとなっている[9]。以上のことから、入替え条件の違いによって、入替え時間が大幅に増加してしまう場合がある。したがって、この増加した入替え時間の短縮化を図る必要がある。

3.2 対処法

増加した入替え時間を短縮するには、入替え対象モジュールが入替え可能でない状態を早期に脱出し、入替え可能状態にすることが必要である。このため、入替え対象モジュールが入替え可能状態でない場合には、その入替え対象モジュールを走行中のプロセスのみを走行させることによって、早期に入替え可能状態に遷移することができる。したがって、以下の対処が有効である。

(1) 入替え可能状態にあるプロセスの早期停止

(2) 入替え可能状態でないプロセスの優先実行

入替え可能状態にあるプロセスの早期停止とは、入替え可能状態にあるプロセスの処理を強制的に一時停止させることである。これにより、入替え可能状態でないプロセスが走行する確率が増加する。図5にプロセス早期停止の様子を示す。入替え対象モジュールはモジュールYとする。各プロセスにおいて、入替え可能でない状態から入替え可能な状態に遷移した直後にその処理を一時停止させる。なお、図5では、基本方式で述べた有限時間内での入替えを保証する制御法として、入替え要求後入替え対象モジュールに処理を移行しようとしたプロセスについて処理を一時停止させる様子についても、併せて示している。

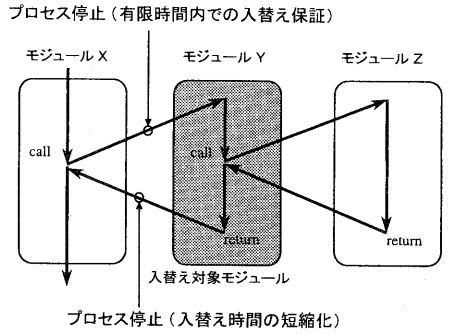

図5　プロセス早期停止の様子

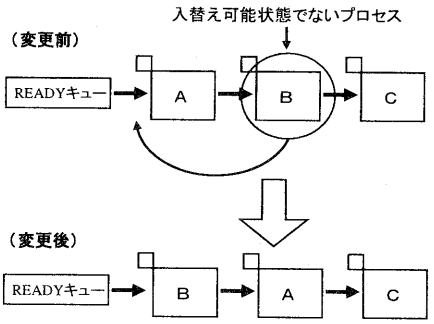

図6　readyキューの繋ぎ変えの様子

入替え可能状態でないプロセスの優先実行とは、入替え可能状態でないプロセスを優先的に走行させることである。これにより、早期に入替え可能な状態に移行できる。図5において、モジュールYあるいはモジュールZを実行していることが検出されたプロセスが処理の対象となる。具体的には、入替え可能状態でないことが検出できた時点で、当該プロセスが繋がっているreadyキューについて、当該プロセスをreadyキューの先頭に繋ぎ変えることで優先的に走行させるようにする。図6にreadyキューへの繋ぎ変えの様子を示す。入替え要求後、入替え法独自に作成したプロセス管理表を参照し、入替え対象モジュールを走行中のプロセス、すなわち入替え可能状態でないプロセスが存在するか否かを確

認する。存在しない場合は直ちに入替えが行なわれる。存在する場合には、そのプロセスについて、その時点でのreadyキューへの繋がりをキューの先頭に変更する。これにより、対象プロセスの優先実行を実現する。

4. 評価と考察

文献10)では、提案した入替え処理速度高速化手法のうち、プロセスの早期停止による効果について、測定結果を示し、その有効性を示した。本章では、もう一つの提案手法であるプロセスの優先実行を適用した場合の効果について、入替え時間の観点から行なった測定結果を示し、提案した手法の有効性について考察する。

4.1 測定条件

本稿で提案した対処法の効果を明らかにするため、測定は**3.1 問題点**で述べた条件と全く同じ条件下で行った。

4.2 プロセス優先実行の効果

入替え可能状態でないプロセスの優先実行による対処を行った場合の入替え時間について、他プロセスが存在しない場合を図7、他プロセスが一つ同時走行している場合を図8に示す。また、各typeでの特徴がよくわかるように、type A,E,Hの場合についてのみ示す。また、本手法を適用した場合を、type A',E',H'として実線で示し、本手法を適用しない場合を、type A,E,Hとして破線で示す。図7に示すtype A,E,Hは図4に、図8に示すtype A,E,Hは論文9)の図9にそれぞれ対応している。

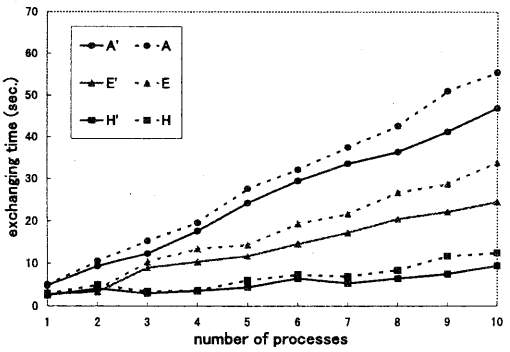

図7 プロセス数と入替え時間の関係
(プロセッサ処理プログラムに対する入替え)

図7の測定結果からわかる特徴について、type別に以下に述べる。

(1) type H'の場合、プロセス数が多くなると本手法の効果が大きくなっている。本手法を適用しない場合に比べて最大約30 %の時間短縮となっている。

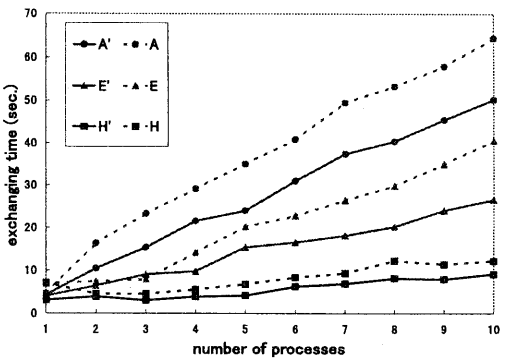

図8 プロセス数と入替え時間の関係
(プロセッサ処理プログラムに対する入替え、他プロセスあり)

プロセス数が多くなると、入替え可能状態でないプロセスの存在確率は高くなる。しかしこの場合、入替え対象モジュールであるmodule Yの処理時間が1秒で、総処理時間に対して非常に短い。このため、走行しているプロセス数に占める入替え可能状態であるプロセスとそうでないプロセスの割合は、入替え可能状態であるプロセスの割合の方が大きく、プロセス数が増加するほど大きくなると考えることができる。このため、本手法の効果はプロセス数の増加に伴って大きくなると言える。

(2) type E'の場合、本手法の効果が見られる。本手法を適用しない場合に比べて最大約20%の時間短縮となっている。この場合、module Yの処理時間とその他のモジュールの処理時間の総和は等しい。したがって、入替え要求時に入替え可能状態であるプロセスとそうでないプロセスの割合もほぼ50 %であると言える。このため、type H'に比べて効果が小さくなる。

(3) type A'の場合も同様に本手法の効果が見られる。本手法を適用しない場合に比べて平均で約16 %の時間短縮となっている。ただし、module Yの処理時間は1秒でその他のモジュールの処理時間の総和に比べ非常に短い。したがって、入替え要求時に入替え可能状態であるプロセスに比べてそうでないプロセスの割合の方が高くなる。したがって、他のtypeに比べて本手法の効果は現れにくくなってしまう。

次に、図7と図8 type A',E'に着目することで、次の特徴がわかる。

(4) 図7と図8を比較したとき、図8の方が、共有プロセス数に関係なく入替え時間の短縮率がおよそ10 ~ 15%高くなっている。このことから、本手法の適用によって、他プロセスが入替え時間に与える影

響を取り除くことが出来ると推察する。本手法の適用対象は、入替え対象モジュールを共有するプロセスのうち、入替え要求時に入替え可能状態でないプロセスであり、入替え対象モジュールを共有しない他プロセスは適用対象外である。このため、適用対象プロセスが本手法の適用によって優先的に実行されている間は、他プロセスが実行されることはなく、他プロセスが入替え時間に影響を及ぼすことはない。したがって、他プロセスが同時走行している状況での入替えにおいて、本手法を適用することは、入替え時間を短縮する上で、非常に効果的であると言える。しかしながら一方で、入替えとは無関係であるプロセスの処理が、入替え処理が終了するまで一時的に停止してしまうという問題点を挙げることもできる。

以上のことから、本手法は、入替え対象モジュールを含むプログラムの総処理時間に対する入替え対象モジュールの処理時間の割合が小さく、共有プロセス数が多い場合に最も効果があると言える。その結果として、最大約 30 % 入替え時間の短縮を図ることができた。また、他プロセスが同時走行しているような状況下での入替えでは、本手法を適用することによって、他プロセスが全く存在しない場合の入替え時間にまで短縮することができ、さらに有効であることがわかった。しかしこの場合、入替えとは無関係であるプロセスの処理が一時的に停止してしまうという問題点がある。

5. あとがき

実行中プログラムの部分入替え法について、プログラムの一部分を入れ替える時、そのプログラムの処理主体が、入出力処理であるかプロセッサ処理であるかによって、入替え処理に要する入替え時間が大幅に増加する場合がある。また、入替え対象となっているプログラムがどのような実行状態の時に入替えを可能とするかによっても、入替え時間に大きな差が生じる。本論文では、このような問題への対処として、入替え処理速度を高速化する手法を提案した。具体的には、入替え可能状態にあるプロセスの処理を早期に停止する手法と、入替え可能状態ではないプロセスを優先実行する手法を提案した。

また、プロセスを優先実行する手法を適用した場合の入替え処理における入替え時間の測定結果を示し、次のことを明らかにした。入替え可能状態でないプロセスの優先実行を適用する場合、入替え対象モジュールを含むプログラムの総処理時間に対して、入替え対象モジュールの処理時間の割合が小さく、共有プロセス数が多い時に最も効果があり、最大約 30% 入替え時間を短縮することができる。また、他プロセスが入替え時間に与える影響を取り除くことができ、他プロセスの数が多いほど、その効果は大きくなる。

今後の課題として、プロセスを優先実行する手法の適用によって、入替えとは無関係であるプロセスの処理が、入替え処理が終了するまで一時的に停止してしまうという問題への対処がある。

参考文献

1) 谷口秀夫 , 伊藤健一 , 牛島和夫 : “プロセス走行時におけるプログラムの部分入替え法 ”, 信学論 (D-I), Vol.J78-D-I, No.5, pp.492-499, Mar. 1995.
2) Hiroshi Muramatsu, Masahiro Date, Hiroshi Yoshida, Masaharu Kitaoka, and Norio Kurobane : "Operating System SXO for Continuous Operation", IFIP 92, vol.1, pp.615-621, Jan. 1992.
3) B.Snead, F.Ho, and B.Engram : "Operating System Features Real Time and Fault Tolerance", Computer Design, pp.177-185, Aug. 1984.
4) Hideyuki Tokuda, Tatsuo Nakajima, Prithvi Rao : "RT-Mach:Towards a Predictable Real-Time System.", In *Proceedings of USENIX Mach Workshop*, October 1990.
5) S. Oikawa, K. Sugirura, H. Tokuda : "Adaptive Object Management for a Reconfigurable Microkernel.", In *Proceedings of International Workshop on Object Orientation in Operating Systems*, 1996.
6) S. Moriai and H. Tokuda : "Dynamic Loadable Object Support for Real-Time Mach Kernels.", In *Proceedings of International Conference on Worldwide Computing & Its Applications '97*, March 1997.
7) 谷口 秀夫, 後藤真孝 : “実行中プログラムの部分入替え法における入替え時間の評価”, 信学論 (D-I), vol.J82-D-I, No.8, pp.998-1007, Aug. 1999.
8) 中島雷太, 谷口秀夫 : “入替え条件を緩和した実行中プログラム部分入替え法の評価”, 情処研報, vol.98, no.71, pp45-52, Aug. 1998.
9) 中島雷太, 谷口秀夫 : “実行中プログラム部分入替え法の評価 — 他プロセスの影響 —”, 情報処理学会コンピュータシステムシンポジウム論文集, Vol.98, No.15, pp79-86, Nov. 1998.
10) 中島雷太, 谷口秀夫 : 実行中プログラム部分入替え法における入替え処理速度の高速化, 情処研報, Vol.99, No.65, pp1-8, Aug. 1999.