文献翻訳

# 保護の領域とプロセスの管理*†

R. M. ニーダム
M. V. ウィルクス 著
木村　泉**訳

## アブストラクト

与えられた計算機についてプロセス管理のためのシステムを設計する際には，ハードウェアにどんな記憶保護の機構が組み込まれているかを考慮に入れる必要がある．生ずる問題点を，目下開発中のケンブリッジ大学 CAP システムの視点から論ずる．

現在使われている記憶保護システムの大部分は，動作に二つのモードを設ける，という考えに立っている．一方のモード（非特権モード）では保護システムがはたらき，もう一方のモード（特権モード）ではははたらかないようにするのである．記憶装置のどの部分を利用可能とし，どの部分を保護するかは，1個または数個の基点限界レジスタによって決定される．これらのレジスタの内容を変更したければ，まず特権モードに移らなければならない．また，以上のものよりはもっと複雑な保護システムも作られた例がある．たとえば Multics では，保護の同心円 (ring) に基づいたシステムが使われている．このシステムでは，ある同心円内で走っているプロセスはそれより下位の同心円内の任意の部分に対して利用権を持っているが，上位の同心円内に対しては一切利用権をもたないようになっている．もとの Multics システムでは，同心円は主としてソフトウェア的に作り出されていたが，現在ではハードウェア的に作られている[1]．容易にわかるように，同心円に基づく保護方式は，従来のものと比べれば確かに進歩したものではあるが，ユーザに提供する利便という点から見ればまだ不充分な点が多い．たとえば入力ルーチンには入力バッファ，出力ルーチンには出力バッファの利用権を与えようとすると，それと同時に，入力ルーチンに出力バッファに対する利用権が与えられてしまうか出力ルーチンに入力バッファに対する利用権が与えられてしまうかする．そのようなわけで最近は，もっと一般的な保護システムに対して興味が向けられている．その種のシステムの一つとして，ハードウェア的に作り出された**権限** (capability) の概念に基づく，目下開発中のケンブリッジ大学 CAP システムがある[2]．このシステムの概要についてはのちに手短かに述べるけれども，詳細に立ち入ることは不要である．権限の概念に関する予備知識については文献 3)，ケンブリッジ大学CAPシステムの詳細については文献 4) を見られたい．

ハードウェアに基づく保護システムを作るに際して

---

* A full translation of "Domains of protection and the management of processes" by R. M. Needham and M. V. Wilkes, Computer Journal, Vol. 17, No. 2, pp. 117-120, May, 1974. Translation right granted by the British Computer Society and the authors. Translated by Izumi KIMURA (Tokyo Institute of Technology, Department of Information Science).

** 東京工業大学理学部情報科学科

† **訳注**　この論文は，目下開発中のケンブリッジ大学CAP 計算機における記憶保護の問題について論じたものである．CAP 計算機は権限 (capability) の概念に基づいて設計されている．権限とは，基点番地と限界によって主記憶上の一区域を定めるほか，記憶場所の利用形態（読み，書き，実行など）をも指定するある特別な情報語であって，記憶装置は，処理装置の権限レジスタと称する特別のレジスタに適当な権限が置かれているときでなければ，いっさい利用できない仕組みである．この論文の技術的背景に関しては，和田英一「情報保護の機構」(情報処理，Vol. 16, No. 12, Dec. 1975, pp. 1092～1099) が参考となろう．

この論文は以上のような考えに基づいて作られた計算機システムにおいて生じ得る，ある困難な問題とその一解決法について論じている．論文の中心は最後の「管理領域間での権限の受渡し」の節にあり，それ以前は問題点を理解するために必要な，わくぐみの簡明な説明である．問題は，一般ユーザに自己固有の副システムを作る自由を与えたい，と考えたために生じたものであり，かつ割込み（たとえば時計による）の処理の問題にも密接に関係している．記憶保護の問題の微妙さ（と面白さ）をよくあらわした，エレガントな論文であると言えよう．

目標とすべきことがらは，次の通りである．

1. ユーザが，機械語を含む任意の言語で，しかも過失または故意によってオペレーティングシステムまたは他のユーザに損害を与えるおそれなく，自由にプログラムを作成できるようにすること．

2. ユーザが，オペレーティングシステムに何ら変更を加えることなく，記憶保護つきの副システムを作り出すことができるようにすること．

3. ハードウェアの故障またはソフトウェアの虫が生じたとき，記憶情報に及ぶ損害の程度を限定すること．

4. 万一ソフトウェア上の誤りをおかしたとしても破滅的な結果を惹き起すようなおそれなしに，オペレーティングシステムに変更を加えることができるようにすること．

ユーザに無制限の利用を許すいかなる多重プログラミングシステムにとっても，目標1を十分にみたすハードウェア的保護は不可欠である．目標4は実は目標3の特別の場合である．もっとも，この種の保護は絶対的なものではあり得ないから，損害の程度を限定する，と言うよりは，重大な損害が起る危険を減少させる，と言った方がよいかも知れない．

プロセスまたは手続きのまわりに，絶対に必要な最少限度より多くの保護の障壁をめぐらすことは，建物における防火シャッターにたとえられよう．平常は，それらは何の役もしないじゃま物である．だが，ひとたび事故が起れば損害の拡大を防ぐのに役立つ．

## 保護の領域

プロセスは走るための場である手続き——何らかの命令の集り——を必要とする．1回生存する間にプロセスは，いくつもの手続きの上を走る可能性があり，そしてそれらのうちには，そのプロセスに固有なものもあれば，同時期に走っている他のプロセスと共有されるものもあるかも知れない．任意の一時点をとってみると，プロセスはあるデータ区域 (data segment) の組に対して利用権をもっており，またある手続きの組に対して呼出しの権利をもっている．これらのデータ区域および手続きが，その時点におけるそのプロセスの**保護の領域**(domain of protection. 以下単に**保護領域**とも言う) を構成する．われわれは，プロセスの保護領域が，手続きの変更の際に限って変更できるようにする，という方針をとる．

保護領域を規定することは複雑な問題である．特にそれは，プロセスがあるデータ区域または手続きに対して利用権を持っている理由が，必らずしも本来その権利を持っていたためではなく，ある特定の手続きの上を走っているためでもあり得るゆえに著しい．ある場合には，手続きに付属するデータ区域が，その手続きの上を走るすべてのプロセスにとって利用可能であることもある．またある場合には，データ区域がプロセスに付属していて，しかもある特定の手続きの上を走っているときだけそのプロセスにとって利用可能であるようなこともあり得る．保護領域を記述する情報のたくわえかたはいろいろある．そのいくつかを挙げると，

1. オペレーティングシステムに付属する共通的な表を使う方法．

2. データ区域や手続きには錠前を，プロセスには鍵をもたせるという方法．プロセスがデータ区域を利用したり手続きを呼び出したりできるのは，持っている鍵の中に錠前と合うものがあるときに限るとする．

3. 番地構成に基づく方法．データ区域や手続きに対する利用権は，プロセスが適当な番地をすでに持っているかまたは作り出せるときに限って与えられるとする．

ケンブリッジ大学 CAP システムは第3の範疇に属する．それはハードウェア的に構成された権限概念に基づいている．**権限** (capability) は基点番地と限界から成っており，これらによって主記憶上の一区域を定める．そのほかに，許容される実行の型（読出し，読出し書込み，実行のみ，等）を定める少数のビット群がついている．区域に対する利用権は，適当な権限が処理装置の中の**権限レジスタ** (capability register) と呼ばれる特別のレジスタ群のうちの一つに置かれたときに限って得られる．ハードウェアは，権限が累算器に出て来たり，データ語（命令語を含んで言う）が権限レジスタに出て来たりすることのないように設計する．この権限の概念が，保護に関する諸規則を実現させ，保護領域を規定する役をする．権限は，プロセスが手続きに入る際には，引数として自由に受け渡すことができる．保護領域の変更を可能にする特別の型の権限として，**呼出し権限**(ENTER capability)がある．これはただ一つの用途，すなわちそれを所有しているプロセスが**保護された手続き** (protected procedure) に入ってその上で走ることを可能にするという用途に限って用いられる．プロセスが手続きに入ることなしに，その手続きに所属する情報に対する利用権をこの

呼出し権限を用いて得ることはできないようになっている.

## 管理の領域

以上の議論から，次のようなことがわかって来る．任意の時点をとってみると，記憶装置の中には1群の手続きと1群のデータ区域が存在している．また1群のプロセスがあって，これらの手続きの上を走り，またこれらのデータ区域を利用している．それぞれのプロセスは，ある保護領域の中に閉じ込められており，その領域内の任意の手続きに入ること，および領域内の任意のデータ区域を利用することを許されている．プロセスが新しい手続きに入る際には，保護領域が新しいものに変ることがあり得る（変らないこともある）．このようにして共存し，おそらくは協同動作をしたり資源を共有したりしつつある1群のプロセスは一つの**管理の領域**(domain of coordination, 以下単に**管理領域**とも言う）の中にある，というように言うことにする．これらのプロセスはある管理プログラム (coordinator, **スケジュラ**—scheduler—とも言う）の支配にしたがう．管理プログラムの機能は次の通りである．

1. それは自己の管理下にあるプロセスの消長を把握する．その際，自由に走り得るプロセスと，何らかの理由で一時停止状態にあるプロセスとを区別する．

2. 自己の管理下にあって，実際に走ってはいないようなプロセス全部について，それらを起動するに必要な情報を保持する．割込みによって中断されたプロセスの場合，この情報のうちには処理装置の状態表示語が含まれる．

3. 各プロセスが消費した処理装置時間を記録する．

4. 呼び出されると，それは（起動すべきプロセスが一つ以上あれば）あるスケジュール決定アルゴリズムを使って，次に起動すべきプロセスを決定する．

5. 必要に応じてプロセスに信号機（セマフォア—semaphore）および通信手段を提供する．

もしスケジュラという言葉を使うつもりならば，多重プログラミングシステムの中でなされるスケジュリングの中には，ここには含まれていないものも種々ある，ということを記憶にとどめる必要がある．管理領域の中には，階層的であると否とを問わず，いかなる順序も存在しない．プロセスが必要に応じてデータ区域を利用したり手続きを利用したりすることを許されるのは，何らかの順序に関して優先的な地位にあるからではなく，適当な権限をもっているからなのである．

明らかに，管理プログラムがその管理上の機能——プロセスの起動や停止，待行列や表の更新等——をはたすことができるのは，それが実際に処理装置上で実行されているときに限る．走っているプロセスが，たとえば終了して管理プログラムに戻る，ということはあるが，走っているプロセスの，任意の点における中断は，外部からのハードウェア的な割込みがなければ起らない．そのようなことが起った場合ハードウェアは，割込み処理プロセスに飛び込む．そのプロセスの最初の任務は，管理プログラムの管理下のある場所に，再起動に必要な情報全部をしまうことである．割込みが時計からのものであった場合には，割込みの処理が終ったあと制御は割り込まれたプロセスに戻らずに管理プログラムの方に戻る．そして現在の状況が，使われているスケジュール方式にしたがって再評価され，次に起動すべきプロセスが選定される．時計の割込みは一定期間ごとに起るから，一つのプロセスがある一定期間より長い間中断されずに走るようなことはあり得ない．管理プログラムはこのような機構によって，他のプロセスが起動されるまでにあるプロセスにどれだけの時間間隔を与えるかを制御し得るのである．

## 管理プログラムの階層

オペレーティングシステムに関する近来の考え方に従えば，ユーザは自己に固有のユーザ群をしたがえた副システムを作り出すことができるべきであり，またそのためには，主オペレーティングシステムの設計者が使用したものとあらゆる点において同じだけ強力な，なお望ましくはまったく同一の，保護機構がユーザにも開放されるべきである．言い換えれば，ユーザは下位の副プロセスを管理するための管理プログラムを書けるのでなければならない．こうしてわれわれは，完全に一般的な管理プログラムによって形造られる任意の深さの階層，という考えに到達する*.

ある管理領域の中で管理プログラムの上を走っているプロセスを**管理プロセス** (coodinating process) と呼ぶ．この管理プロセスの管理下にある副プロセスは，個別の下位管理領域に属しているわけだが，そこで問題にしたいのは，これらの管理領域間のインタフェー

* 本文の理論的な検討では，深さは当然任意と考えるが，実用上は2，3段まで考えれば十分であろう．

スである．管理プログラムは，下位の領域に属する副プロセスを必要に応じて自由に管理できる．だが，上位の領域から見れば，これらの副プロセスはすべて一つのプロセス，すなわち管理プログラムの中を走るプロセスの一部分に他ならない．言い換えれば，上位の管理領域の中では副プロセスの存在は意識されないのである．

最上位の管理プログラムは，一部分はハードウェア，他の一部分はソフトウェアから成ると考えるのがもっともふさわしいが，下位の管理プログラムはソフトウェアのみによって構成される．このソフトウェアは外部からの割込みに関する情報を更新する能力をもっていなければならない．これをどのようにして実現するかについてのくわしいこと，およびその際効率上の考慮から生ずるめんどうな点については，ここの話には関係がない．おもな動作を一般的に概観すると次のようになる．時計の割込みが受理されると，ハードウェアはそのとき走っていたプロセスを停め，割込み処理用のプロセスを起動する．その結果として，各管理プログラムが自己に所属する副プロセスに関して保持している総消費時間の記録が更新される．また，割り込まれた副プロセスを再起動するに必要な全情報——処理装置の状態情報を含む——が，最下位の管理プログラム（すなわち割り込まれた副プロセスを直接管理している管理プログラム）が保持しているはずの，他の副プロセスに関する同種の情報の集りにつけ加えられる．より上位の各管理プログラムに対しても同様のことがおこなわれるが，ただしそれらについては，再起動点として，副プロセスが割り込まれた点を記録する代りに，直下の管理プログラムの入口が記録される．その結果，システムがふたたび動き出したときには，階層内の各管理プログラムが呼び出され，下位の副プロセスのうちのどれを起動すべきかに関して独自の判断をすることになる．この判断をどのような規則に基づいておこなうかは，各管理プログラムの設計者の考えにまかされる．

以上は時計の割込みが生じたときに起ることがらについての説明である．いかなる割込みも，その割込みがあったときには管理プログラムによる判断をなすべきだ，と主システムの設計者が考える限りにおいて，同様に処理することができる．周辺装置からの要求によって，動作の完了を示す割込みが生じたときには，以上の他にも(または以上のことがらに代って)，しなければならないことがある．これらの場合には，情報

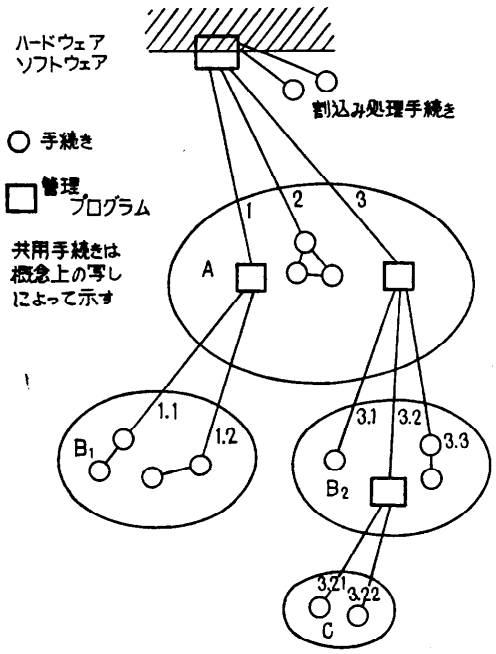


図-1　プロセスと管理に関する考え方

の送り先は要求を出したプロセスを管理する管理プログラムとなる．しかしこの話は保護の問題には関係がないので，ここでは論じないことにする．

プロセスと管理に関する本文の考え方をあらわす図を図-1 に示す．管理プログラムを四角印であらわし，その他の手続きは丸印で示す．これらの四角や丸を結びつけている線は，プロセスが一つの手続きから他の手続きに移る経路を示す．適当な権限をもってさえいれば，いかなる管理領域にあるプロセスも，そのときたまたま主記憶内にある任意の手続き，すなわち命令群の上で走ることができる．純粋手続き（書き換わる部分をもたない手続き——pure procedure）は，各プロセスがいかなる管理領域内にあるかにかかわりなく，プロセス間で共用が可能である．図-1 のような図では，このようにして共用されている手続きをそのまま画くと図が見にくくなるので，概念上いくつもの写しができているかのように考えて図を画くことにする．

図では，最上位の管理プログラムが，ハードウェアとソフトウェアの両方にまたがっている．その下には5個のプロセスがぶらさがっている．そのうちの2個は割込み処理手続きの中を走るものなので，管理プログラムのハードウェアに属する部分から伸び出している．他の3個はソフトウェア部分から伸び出していて，

領域 $A$ の中を走る．それらには，1，2，3 と番号がつけられている．プロセス2は領域 $A$ 内の3個の手続きの上を走る．これに対し，プロセス1および3は管理プログラムの上を走るものである．それらは，それぞれ領域 $B_1$ および $B_2$ の中で走る副プロセスを管理している．領域 $B_2$ の中の副プロセスの一つは，やはり管理プログラムの上を走っており，領域 $C$ の中の副プロセス群を管理している．領域 $A$ から，すなわち主管理プログラムの視点から見れば，副プロセス 1.1 または 1.2 が走っている場合，走っているのはプロセス1であるかのように見える．同様に，もし副副プロセス 3.22 が走っているとすると，領域 $B_2$ の中では副プロセス 3.2 が走っているものとみなされ，領域 $A$ の中ではプロセス3が走っているものと見られる．

## 管理領域間での権限の受渡し

保護の概念とちがって，管理の概念は完全に階層的な概念である．管理領域の階層という考えに到達したいま，保護領域はどの程度多数の管理領域上にひろがり得るものであるかを考えてみる必要がある．下位の副プロセスというものは，純粋な意味でそれを作り出した管理プログラムの所属物であるから，上記の問題は，管理プログラムが自己の保有する権限を下位のプロセスに与えるについて制限があるとしたらそれはどういうことか，という問題に帰着する．制限がまったく不要なら言うことなしであるが，そうは行かない．下位のプロセスは，もし適当な権限を与えられれば，それを使ってシステム内の任意のデータ区域や手続きを利用することができるわけである．特に，主オペレーティングシステムの構成モジュールを使用できることになってしまうであろう．

検討の結果によれば，上で述べたところにしたがって作られた管理プログラムにおいては，管理領域の境界を越えて読出し権限，読書き権限を受け渡せるようにしても別段困難は生じない．命令群をプログラムとして実行することを可能にするような権限の受渡しを許容しても，困難が生じないことは同様である．しかしながら，特別の注意を払わないかぎり，もし呼出し権限の受渡しを許すと，たちまち保護が失われることになる．その理由は，外部割込みシステムが存在していて，割込みが起ったとき管理プログラムが自分自身にとって読出し可能な場所に，割り込まれたプロセスを再起動するために必要な情報を書き込むようになっている，という事実に関連している．

いま，ある保護領域内で走るべきルーチンの作者がある手続きへの呼出し権限を利用することを許されているものとし，その手続きには保護すべき情報が付属しているものとしよう．問題のルーチンの作者は呼出し権限を使って手続きを呼び出すことができるが，保護された手続きの原作者が利用を許していない情報には，何の手出しもできない．いずれは許容された動作が終了し，ないしは許容された情報が受け渡されて，制御はもとのルーチンに戻って来ることになろう．すべては一つの保護領域の中で起り，保護のみだれは生じない．だが，いま，問題のルーチンの作者が，そのルーチンを管理プログラムにしようと決心したとし，別の管理領域の中で働いている下位のプロセスに問題の呼出し権限を受け渡すように，その管理プログラムを作ったとしよう．この呼出し権限は，その別個の保護領域の中で，保護された手続きを呼び出すのに使われる．どんなプロセスでもそうであるように，この下位のプロセスも，保護された手続きの上を走っている間に外部からの割込みが起って中断される，という可能性をもっている．すると，処理装置内の各レジスタの内容およびプロセスを再起動するためのもろもろの必要な情報が，このプロセスを管理している管理プログラムの中の適当な処理待行列に置かれることになる．割込みが起った瞬間に問題のプロセスは，保護された手続きの上を走っていたのであるから，処理待行列に置かれる情報のうちには，呼ばれた側の手続きが持っている秘密の情報が入っている可能性がある．だが，その秘密の情報は上位の保護領域の中の，管理プログラムに所属する待行列の中に置かれているのだから，上位のプログラムはこれを自由に利用できることになってしまう．

上記のことは，特別の場合について考えてみればもっとはっきりさせることができる．いま，くわしい情報の入っている表に対して利用権をもつ手続きがあったとし，その情報は公開されないことになっていたとする．ただしこの手続きは，表の中の情報に基づいて統計データを作り出すようになっており，呼出し権限によって正当にこの手続きに入って来たプロセスに対してこの統計データを与えるようなものであるとする．この呼出し権限を利用できる立場にあるユーザが，表全体を読み出してやろう，と思ったとする．それには次のようにすればよい．まず自分に元来与えられた保護領域の中の，あるプロセスの一部として走るよう

なルーチンを書く．このルーチンを管理プログラムとみなす．ユーザは誰でも管理プログラムを書いて副システムを作り上げることができるのであるから，彼にはそうする権利がある．次に，下位の管理領域の中で働く，ある下位のプロセスの一部として動作するようなもう一つのルーチンを書き，そのルーチンから繰り返し問題の保護された手続きを呼び出しては，正当なやりかたで情報を要求するようにする．するとこの下位のプロセスは——どんなプロセスでもそうであるように——，繰り返し割り込まれる．そしてある場合には，プロセスが保護された手続きの上を走っている間に割込みが起って，その保護された手続きに関する情報が，ユーザの方で書いた管理プログラムの中の待行列に書き込まれることもあると考えられる．そこで，割込みのあと，この管理プログラムへの戻りが起ったところで，その都度下位のプロセスに関する情報を調べ，ほしい秘密の情報にたどりつく手段がないかどうか探すことにする．もちろん暗闇で手探りをするわけだし，せっかく情報をあばき出してもその意味を知ることができるとは限らないのだから，これが簡単に成功する気づかいはない．また，わざと悪いことをすることによって情報のみだれをひき起す，というような話は，重点を置いて論ずべきことでもない．それより心配なのは，ソフトウェアの誤りまたはハードウェアの故障によって，おかしいことが起ることである．明らかに，ポインタなどが予定外の形で利用可能になってしまったら，どんな大変なことでも起るおそれがある．

以上の議論から見て，二つのやりかたが考えられる．一つは呼出し権限を下位のプロセスに渡すことを禁止するか，または——結局同じことになるが——呼出し権限がそのようにして渡されても使えないようにすることである．そのようにすれば，下位システムの作者は，主システムの作者に与えられていたと同一の保護機構を与えられることになる．彼は主システムの一部として存在する手続き——命令群——の利用権を与えられるが，保護された手続きに対する利用権は与えられない．保護された手続きを使いたいと思ったら，自分で作らなければならない（その際，もしその方が楽だと思ったら，自分が利用権をもっている範囲の既存の命令群を使って作ることは許されるが）．これがケンブリッジ大学 CAP 計算機で使われているやりかたである．

もう一つのやりかたとは，割り込まれたプロセスを再開させるに必要な情報を，割込み処理ルーチンからそのプロセスを管理する管理プログラムに渡す際に，不正な使い方ができない形で渡すようにする，というものである．この考えを実現するための，もっとも簡単なやりかたは，割込み処理ルーチンにある保護された手続きを作り出させるようにし，その手続きを呼び出すと割り込まれたプロセスが再開されるように仕組む，というものであろう．その場合，管理プログラムの方には，この手続きの呼出し権限が渡されることになろう．このようにした場合には，管理領域の境界を越えて呼出し権限を渡すことを制限すべき理由は消滅する．

だが，呼出し権限を下位の管理領域に渡すことが許容されるものであるとしても，それをするかしないかは完全に各ユーザの判断の問題であり，この機能をなるべく使わないように心がけることも考えられる．特に，同時に二つ以上のプロセスから利用をすることは危険であるような情報を読み書きするための保護された手続きへの呼出し権限に関しては，問題が起ることも考えられる．これは，よくある話であって，通常何らかのロックアウト動作をもちいて処理される．その場合，下位の管理領域内で動作するプログラムの作者がプログラミングの正しい作法に注意を払わなかったために，情報が永久的にロックされてしまう，という危険が存在する．

## 参 考 文 献

1) M. D. Schroeder & J. H. Saltzer : A hardware architecture for implementing protection rings, Comm. ACM, Vol. 15, No. 3, pp. 157～170 (1972).
2) R. S. Fabry : Preliminary description of a supervisor for a computer organised around capabilities, Quart. Prog. Rep., No. 18, Sect. IIA, Inst. Comp. Res., Chicago, (1968).
3) M. V. Wilkes : Time-sharing computer systems, Second Ed., Macdonald, London and American Elsevier, New York,(1972).
4) R. M. Needham : Protection systems and protection implementations, AFIPS Conference Proceedings, Vol. 41, pp. 571～578 (1972).

（昭和 50 年 12 月 8 日受付）