解　説

公共情報システムにおけるシステムの信頼性とセキュリティ*

石　田　晴　久**

1. 公共情報システムの信頼性とセキュリティ

近年，社会のいろいろな方面でのコンピュータ利用が広まるにつれて，かなり広い範囲のユーザに計算サービスおよびデータ・サービスを提供する「公共情報システム」というべきシステムが増えていることはいうまでもない．しかし公共情報システムとは一体何かということになると，そのとらえ方は人により，また見方によりさまざまであって，誰にでも納得できる統一的なイメージはまだないというべきであろう．

本稿では，主として技術的な観点から，公共情報システムの条件を次のように考えることにする．

（1） かなり広範囲のユーザに計算処理やデータ・ベース検索などのサービスを提供する共同利用のコンピュータ・システムであること．普通の会社のバッチ処理系のように，当の計算機室の専門要員のみが使用しているシステムは，たとえそれで扱われる情報が公共的なものであっても，ここでは含めない．

（2） ユーザ層がひとつの会社とか大学・官庁に限られず，さまざまな組織にまたがっていること．この観点からは，大学の共同利用機関である大型計算機センターのシステムは公共情報システムの一種である．一方，国鉄や航空会社の座席予約システムとか銀行のオンライン・システムなどは，一組織内のシステムではあるが，規模も大きく，ユーザも多く広域に分布しているから公共情報システムと考えるべきであろう．事実これらのシステムは共通の課題や問題点をかかえている．

（3） ユーザとシステムとがデータ通信線を介して結ばれていることが多く，ユーザ間でファイル（プログラムやデータ）などの共用が行われること．要するにユーザがある程度広域に分布していながら，相互に全く独立ではなく，情報を共用し合うのが公共情報システムのひとつの条件である．いわゆる TSS システムは一応この範囲に含めて考える．とくに，オハイオ州クリーブランド市にあるスーパー・センターの TSS を日本を含む世界何ヵ国から共同利用する GE（ゼネラル・エレクトリック，日本では電通国際情報サービス）の TSS のようなネットワーク・システムは公共情報システムに含めるのが自然であろう．

以上のような公共情報システムでは，信頼性とセキュリティが非常に重要であることは，いまさらいうまでもない．これらの問題のうち，まず信頼性については，いつでも安心して頼りにして使えるという条件が必要であるが，それには，ハードウェアが気づかれぬ形で誤動作しないこと（誤動作が検出できること），ソフトウェアが正しく設計されていて虫がないこと，などは当然として，何よりもシステム・ダウンが極力少なくなければならない．数多くのオンライン端末がついていて，多数のユーザに使われ，ファイル（プログラムやデータ）の共用がひんぱんに行われるシステムでは，システム・ダウンによる被害・損失は非常に大きい．

また，公共情報システムは，多種多様な多数ユーザに，データ通信をも含む形で使われるところから，セキュリティ（安全保障）の確保にも十分考慮が払われていなければならない．

筆者は，本稿を準備するに当たり，参考文献[1]を調べるのに，東大大型計算機センターの TOOL-IR システム[2]により，INSPEC (IEE) の文献データ（1976年分）の検索を利用した．このデータ・ベースには，システムというキーワードをもつ文献が 14,927 件，コンピュータをキーワードするものが 11,097 件，セキュリティをキーワードとするものが 324 件あり，

computer & security は 187 件

computer & system & security は 145 件

* System reliability and security in public information systems by Haruhisa ISHIDA (Computer Centre, University of Tokyo).

** 東京大学大型計算機センター

とかなり多かった．そしてセキュリティについて興味深かったのは，このことばが，防犯機器や警備保障に用いられるコンピュータにも，コンピュータ自体のセキュリティという意味にも使われていることであった．本稿ではもちろん後者の意味のみを対象とする．

2. システム・ダウンの原因とその対策

すでにふれたように，システムの信頼性確保の上から，最も重要なのは，システム・ダウンを極力少なくする（理想的にはゼロにする）ことである．しかしこれは，同時にまた実現が非常に難しい問題でもある．システム・ダウンの形態もシステムが大規模・複雑になると，多様になり，可用性 (availability) が一律には表現できなくなる．とくにシステムの各要素(CPU，メモリ，入出力機器など）が複数個あって，たとえば故障したCPU が自動的に切り離せるシステムでは，フェイル・ソフト(システムの一部が欠けた形で運転)の状態が存在する．そうすると，全システムのダウン回数や平均故障間隔 (MTBF＝Mean Time Between Failures)，平均修復時間 (MTTR＝Mean Time To Repair) のほかに，いろいろな形のパート・ダウン（フェイル・セイフ状態）についても，ダウン回数やMTBF，MTTR などを問題にしなければならない．

東大大型センター・システムでは，システム・ダウンを次のように分けて，分析している．

（1） オープン・バッチ (含 TSS) 時の全システム・ダウン
（2） TSS サービス時のみの全システム・ダウン（夜間）
（3） ユーザが直接使用していないときの全システム・ダウン（深夜・早朝・週末）
（4） パート・ダウン（4台の CPU，8バンクの主記憶，100台を越す周辺機器のいずれかがダウン）

このうち，(1)のオープン・バッチというのは，ユーザが4台のカード読取機，9台のラインプリンタ(用紙の自動カッタ付)，4台の磁気テープ装置，2台のXY プロッタ，2台の特殊プリンタ，紙テープ入出力機器などをセルフ・サービスで使う方式である．センターにとってのダウンの影響は(1)がもちろんもっともひどいが，皮肉なことに，入出力機器をほとんど止めて，入出力はファイルからのみ行う(3)の積み残しジョブ処理のときが，ダウンはもっとも少ない．この例のように，同じダウンでも，システムの使用形態によって，その影響が異なることに注意する必要がある．

また従来，余り注意が払われてこなかったパート・ダウンも要注意である．デュアル・システムやデュプレックス・システムのように，同じシステムを2台用意して同じことをやらせたり，もう一台を万一の場合の待機系として用意したりするシステムでは，切り換えが本当に自動的にできれば，全システムのダウン率は非常に低くできるが，パート・ダウンは表面から隠されてしまう恐れがある．

さて，コンピュータ・システムのダウンの原因であるが，とくに2重化などの対策はせず，むしろ常にシステムのレベル・アップや拡張を行っている前記東大システム[3]の例をあげると，システム・ダウンの原因は**表-1**のようになっている．これを詳細に分析してみると，大規模で複雑なシステムでは，コンピュータ技術の進歩により，誤動作自動検出・自動再試行・自動構成制御（故障部分の自動切り離し）などの高信頼性確保技術が使われているため，個々の要因による故障は比較的少なくなっているが，要因の種類の方が非常に多いために，全体としてシステム・ダウンをゼロにするのはなかなか難しいことが分かる．ダウン要因は

（1） ハードウェアの故障
（2） ソフトウェアの虫あるいはダウン対策不足・不良
（3） オペレータの操作ミス
（4） 保守・修理上のミス
（5） レベルアップ作業・長時間停電

などに分けられるが，巨大システムでは，これらを更に細分化して，そのひとつひとつをつぶすような設計（とくにシステム構成）をしなければならない段階にきている．

表-1 システム・ダウンの原因

（東大システムの例，1976 年度）

CPU（4台構成）障害*	15%
主記憶（6バンク構成）障害	4%
入出力処理装置（4台構成）障害	4%
ドラム・ディスクその他	24%
OS ソフトウェア不良（改良上の虫が多い）	29%
センター・ルーチン不良	8%
操作ミス・保守ミス	12%

〔同じ原因で何回もダウンした場合もそのまま数えている〕

次に，高信頼性システムを実現させる上での注意点をいくつかあげておく．

（1） 極度の集中化（密結合）システムはやめ，ある程度分散型(疎結合)のシステム構成にする．ただ，

分散型もあまりに広域に分散させると，その運転や保守により多い人間が必要となり，システム開発や運営管理の力が結集しにくくなる．

（2） 多重プロセッサ系では，故障したCPUやメモリ・バンクは自動的な切り離しおよび修理後の自動的な組み入れができるようにする．これは自明のようでもあるが，われわれの経験では，実際にこれを完全に行うのは，頭で考えるほど容易ではない．CPUの場合，故障した時点で，そのCPUがユーザ・タスクを実行していた場合は，それを放棄するのは簡単だが，OSのシステム・タスクを実行しているときや，入出力の割込み待ちの状態にあるときは，それらを一瞬のうちに放棄して他のCPUに渡すのはなかなか難しい．メモリの方も，現在の仮想メモリでは論理アドレスと物理アドレスが切り離されているとはいえ，一部には物理アドレスが直接使われていることもあり，さらに物理アドレスの方は，1バンク内で連続ではなく，各バンクにまたがってつけられており，ページングによって連続した論理アドレスが別々にバラバラの物理アドレスに拡がっていることもあって，故障したメモリ・バンクの自動切り離しは非常に困難である．

これらの点は，今後アーキテクチャやOSの核部分の設計に当たっては，一層の研究開発を必要とする[4]．しかし，この種の機構は仮にできたとしても，CPUもメモリもめったに故障しないから，コスト・パフォーマンスは恐らく極めて悪い．経済的には，めったに起きない障害に対して，どだれけの備えをするかという問題になる．

（3） オペレーションや保守上のミス，つまり人間のミスによるシステム・ダウンが起きないような構成にする．オペレータがシステムの起動時刻を入れまちがえたために，後でシステムを止めて実質上ダウンさせ，時刻の修正を行うといったミスが起こるシステムは，オペレータのミスというより，システムの設計が悪い．ケーブルのつなぎ忘れといった保守上のミスも，システム・ダウンにつながらぬような設計にしなければならない．とにかく，コンピュータのハードウェアおよびソフトウェアの信頼性が向上すると，システムの複雑化に対応できない人間のミスが増える可能性が強く，人間がミスしようのないようなシステムの設計が重要となる．このためには，操作をできるだけ自動化し，かつ人間の介入を求めるときは，できるだけシステムから問いかけをして人間の応答を求め，入ってきた応答が妥当かどうかをシステム側でよく調べるといった配慮が必要であろう．

（4） 障害が発生してもシステムをできるだけ止めないようにOSを構成する．一般にシステム・ダウンは突如発生するわけではなく，むしろOSが異常事態を検出して，いわばOSの「どうしようもない」あるいは「このまま運転を続行しては危険」という「判断」のもとにシステムを停止させることが多い．システムをできるだけ止めないという方針で設計されているOSの例としては，Tandem社のT/TOS (Tandem Transaction OS) がある[5]．これは2台ないし16台のミニコンよりなる多重プロセッサNon Stop用のOSで，データメーション誌1977年7月号161ページには，「1台のCPUがダウンしても誰も気付かない」システムとして紹介されている．保守員が定期点検をしたところ，CPUのひとつが1週間前からダウンしていたのに誰も気付かなかったというので，ダウンしたときはブザーがなるように設計変更中だという．

このNon Stopシステムの例はともかくとして，普通には，システムを止めないことは，OSが危険をおかすことにもなって，セキュリティの概念と相反する面もあり，Non Stop OSにはまだ研究すべきことが多い．

3. 公共情報システムにおけるセキュリティ問題

コンピュータ・システムはそれがひとつの計算機室内で少数の専門要員によって使われている限りは，それほどセキュリティ上の問題はないといえる．しかし広域に分布する多数のユーザが情報を含むコンピュータ資源を共用するようになると，セキュリティ上のいろいろな問題が出てくる．とくにTSSのようにデータ通信が介入して，不特定多数に近いユーザが，計算機センターからは見えない不特定の場所で電話回線端末を使う場合には，セキュリティ対策を十分たてておかないと，いろいろな混乱の起きる可能性がある．

公共情報システムでのセキュリティの破られ方としては，次のような点（一部はコンピュータ犯罪）があげられる[6]．

（1） TSSを利用して他人（他社）のソフトウェアを盗む．

（2） TSS端末を通して他人のデータを盗む．あるいは悪用する．この(1),(2)にはデータ通信の盗聴も含まれる．

（3） 他人のユーザ番号やパスワードを利用し，他

人になりすまして，自分はタダでコンピュータを使う．プログラムで課金をゼロにする手が使われたこともあるが，そうしたことが可能なシステムは，システムの設計が悪い．東大システムでは初期の頃，メモリの初期クリアをしていなかったために，メモリ内を年中調べていると他人のパスワードのイメージが拾えるという弱点があった．この弱点はその後，はじめに必ずメモリ・クリアを行うようシステムを改良して解消したが，メモリ・クリアなどはセキュティ確保の第一歩である．

（4） システムの運用に妨害を与える．たとえばOS の弱点をついてシステム・ダウンを起こさせるのがその1例である．後述の「トロイの木馬」問題もある．よく話題になる話に，磁石で磁気テープの内容を故意に消去する話があるが，これは NBS (米標準局) などでの実験により，強い磁石を磁気テープ・リールに直接押しつけるようにしない限り消去ができないことが分かっている[7]．

（5） システムが物理的に破壊される．これには暴動などによるほか，火事や地震による破壊も考えられる．対策としては，耐火・耐震の建物を作り，計算機室への入室を制限するなどのほか，万一の場合の復旧をいかにすみやかに行うかの考慮もいる．復旧対策をしているという話では，小田急電鉄が一ヵ月に一回数パックのディスクを秩父地方の飯能にあるワンビシ倉庫に預けているという実例がある (IBM ユーザーズ誌，No. 181)．

（6） コンピュータ要員の悪質行為を含むコンピュータ犯罪．これについては，コンピュータ犯罪を専門に研究しているドン・パーカー（スタンフォード研究所）の著書[6]をごらん頂きたい．コンピュータ犯罪への最大の対策は，コンピュータ要員への適切な人事管理であり，技術対策だけでは防ぎきれない．

本稿では，次章以降で，セキュリティ確保のための技術について論ずるが，その前に現在のシステムにはセキュリティ対策を入れるのは容易ではないことを指摘しておきたい．それは，ひとつには，セキュリティがまだそれほど問題にもならず，重視もされていないため，「セキュリティは売れない」からである．また最近はなばなしく登場している新機種でも，過去のシステムとの互換性を重んじているため，ハードウェアは新しいとはいっても，OSは古いものが多く，古いOSにはセキュリティ対策が入れにくいという事情もある．セキュリティ機構は OS の根本的な設計哲学に関するものであるから，既製の OS に後から入れるのは非常に困難である．

一般にシステムは，信頼性が高くなり，可用性もあがれば，それだけセキュリティも向上するが，余りにもセキュリティを向上すると，次のようなマイナス面が出てきやすい．

（1） コストが高くなる．これは結局万一の場合に備えてどれだけ投資しておくかという保険の問題になる．

（2） システムの柔軟性が失われる．悪く設計すると，ガンジガラメで変更や改良を行う自由度のないシステムとなる．

（3） 情報 (データやプログラム) の共用 (sharing) がしにくくなる．極端な話，共用を全くやめれば，安全なシステムが作れるが，それでは公共情報システムにはならない．

（4） 制約がきつく，使い勝手が悪くなる．

（5） 他システムとの互換性が失われる．

こうしたマイナス面を強調しすぎると，治安がよくてやたらにカギをかける習慣のないわが国では，セキュリティ対策がなおざりにされかねないが，公共情報システムの重要性は今後増す一方であるから，少なくとも経済性の許す限りでは，セキュリティ対策はあって然るべきである．今後コンピュータのハードウェアのコストが低下してくれば，ハードウェア上の対策はより容易になるであろう．またコンピュータをアメリカのように治安の悪い国へ輸出しようという場合には，当然高いセキュリティのシステムが要求される．研究の面でも，セキュリティの問題は大学や研究所においてももっと研究されるべきであろう．

4. TSS におけるセキュリティ技術

多数のユーザが遠くの端末からコンピュータ・システムを共同利用する TSS では，さまざまなセキュリティ（アクセス制御）技術が施されている．次にそのいくつかについてのべよう．

（1） パスワード

多くのシステムでは，一定の文字列からなる文字パスワードが使われている．このパスワードは全2重通信系（入力と出力が独立）では，エコーバックを抑止することにより，プリンタへの印字が防げるが，半2重通信系では，キーインするとプリントされてしまうため，あらかじめ無意味な文字を重ね打ちして紙を黒くしてから，パスワードをキーインさせている．しか

しこの方法では端末機としてCRTディスプレイを使うと，パスワードが明確に表示されてしまい，隠せなくなってしまう．

もっとスマートな方法は，HITAC 8800/8700のOS 7で使っているような乱数から計算される数字パスワードである[3]．これは各ユーザがあらかじめ計算公式をシステムに登録しておき，TSSを使う都度システムから出力される乱数に対して，先の計算式をあてはめて，システムとユーザが答を出し，相互にマッチしたら合格というやり方である．ただ大学では，「ゼロをかける（答は常に0）」とか「最後の2桁をとる」とかいう簡単な式しか登録しない人が多く，登録した計算公式を秘密にしておくことになっていないケースが多い．これは式を複雑にすると，覚え切れず，また計算も複雑になるからであり，ここにもセキュリティと使いやすさの衝突がみられる．なおこのOS 7方式では，ある式に対する数種のパスワード・サンプルがあれば，コンピュータを使ってその計算公式が推定できるという弱点のあることがある大学院生より指摘されている．

（2） 端末識別符号

これは使用承認を受けた端末以外は排除するという目的には必要であり，識別符号は端末から自動的に送出させるようにすることもできる．しかしこの自動送信はコスト高になり，端末の登録も面倒（とくに同じ端末機からあちこちのシステムを使う場合）なところから，多くのTSSでは端末IDは使われていない．したがって利用資格のない人が変な端末から利用を試みることはありうることになる．この辺はコスト低減・柔軟性への要求と安全性とが対立するところである．

（3） 回線断によるジョブ終了

電話網を使うTSSでは，電話を切ったとき（切れたとき）には，その回線から入っていたTSSジョブは異常終了になるよう設計されている．ところがシステムによっては，そうした場合，ごく稀にであるが，そのジョブがまだ（異常終了）にならないうちに，たまたま同じ回線に他の端末がつながると，もとのジョブがたまたま生き残って，新しくログオンしようとするユーザにひきつがれてしまう場合がある．これを仮装侵入（masquarade）というが，次のユーザが前のユーザのふりをしてTSSの使用を続行できるとなったら大変である．したがって電話網利用は電話が切れるとすぐ終了するようになっていなければならない．一方，ユーザからは，「何かのはずみで電話が一時切れたときには，すぐまた電話をかけ直したら前のジョブがそのまま実行できるようにしてくれ」という要求もあるが，かけ直したときに同じ回線につながるとは限らないという問題はさて置くとしても，仮装侵入の問題を考えると，このような要求は好ましくない．

（4） データ通信における暗号の利用

データ通信の盗聴は，専用線の場合はやや困難としても不可能ではなく，電話線の場合はその気になればやってできぬことはない．アメリカではデータ通信の盗聴は現実に起きているため，一部では銀行も含め，データ伝送に暗号が使われており，暗号化の研究もかなり行われている[5]．暗号化にはハードウェアとしてはデータ・スクランブラーといった装置が商品化されている．しかしわが国では問題になるようなスパイ活動がないためもあって，暗号を使うという話は聞かれない．

（5） ダウン後の一時ファイルの回復

TSSでとくにファイルを編集している間に，システム・ダウンが発生すると，そこまでの編集結果（通常一時ファイル中にある）が失われて非常に困ることがある．これを防ぐには，編集中の内容の入っている一時ファイルを準恒久的ファイルとして特別扱いし，ダウン回復後にそれが再び取り出せるようにするとよい．システムによってはある程度この種の回復手段をもつものもあるが，完全とはいい難く，今後まだ努力を要しよう．

（6） ユーザ資格のチェックと権限設定

これについては，OS 7の例をとれば，ユーザは，システム管理者(オールマイティ)，システム・プログラマ，特定ユーザ，グループ管理者，一般ユーザの5レベルに分けられ，それぞれコマンドの利用可能範囲(権限)が下のレベルほど狭くなるように設定されている．

（7） 特定のユーザをつかまえる機能

これは東大のOS 7では，津川雅彦邸誘拐事件[6]で犯人がキャッシュ・カードを使った瞬間にその事実をコンピュータで検出したのにヒントをえて，実施に移した．具体的には，つかまえるべき相手のユーザIDをオペレータが指定しておくと，当事者がシステムを使い出すと同時にそのことがオペレータ・コンソールに表示されるようになっている．これなどは典型的なセキュリティ侵犯の検出機能である．

5. ファイルおよびメモリの共用とセキュリティ

前章の TSS ではもちろん，一般の処理形態でも，ファイル（データセット）のユーザ間での共用の問題は非常に重要である．共用が混乱なく行われるためには，OS に次のような制御が必要である．

（a） ファイルに対してユーザが勝手な名前をつけても，システム内ではすべてのファイルが正しく識別できる．これを可能にするため，OS 7 ではファイルには内部的にユーザ ID がつけられる．一方 UNIX（ベル研）[9] では，ハイアラーキーをなすディレクトリ系があり，ファイル名の上には何段ものディレクトリ名がつくようになっている．そこで，たとえばあるファイル名 file は，ディレクトリが a. b. c のときはただ file といえばよいが，コマンドでディレクトリを a. b に変えると，ファイル名は c. file となるということで一義性を保たせている．

（b） 他人のファイルをアクセスするには，持主の許可がいる．OS 7 では持主が，許可する相手をパラメータとして指定して PERMIT コマンドを発行してこれを行っている．許可の条件は読み出しのみ，書込み可，実行可のいずれかだが，MULTICS では追加可という条件もある．このように許可する相手（ユーザ ID）が指定できる OS は今日でもまだ稀である．

なお，共用を許可されたファイル（file の名前）は，OS 7 では使わせてもらう側からみて % userid. file と長くなり，これに自分で勝手な（短い）名前をつけるには SHARE コマンドで新しい名前を登録しなければならない．しかし UNIX では単にディレクトリをその場で相手のものに切り換えればよいようになっている．

なお UNIX では，共用時のアクセス制御は，実行許可の出ているプログラム・ファイルの場合，「そのプログラムが呼ばれると，その中では有効なユーザ ID がそのプログラムの持ち主のユーザ ID に変わる」という形で行われている．つまり自分が持主の身代りになって使う，すなわち持ち主と同じことができる（逆にそれしかできない）ということであり，単純ながら面白いやり方である．持主からみれば，自分でもできないようなイタズラを他人にやられることはないということになる．

（c） あるファイルがアップデート（書き込み）されている間は，他のユーザはそのファイルにはアクセスできないようにする．これについては後述する．

6. OS におけるセキュリティとインテグリティ

（1） インテグリティ

OS の用語では，セキュリティということばが，「不当にデータをのぞかれることがない」という意味で使われるのに対し，インテグリティ（integrity）ということばは，「データが不当に改変されることがなく，データの正しさが保持される．」という意味で使われる．したがって両者の間には，図-1 のような関係がある[10]．とくにセキュリティが犯されても，インテグリティは守られうる（のぞかれはしても改変はされない）し，セキュリティは守られても，インテグリティが破られる（有資格者が背任行為をする）こともありうる点に注意する．インテグリティを保持するには，セキュリティが破られたことを検出したら変更を不可能にし，変更を行う有資格者が，まちがってあるいは故意に不当な変更を加えることがないようにしなければならない．これは「悪意のあるユーザからいかにしてシステムを守るか」という問題で，先の TSS であげたアクセス制御ももちろんその防護策の一部になっている．

（2） Capability

OS におけるアクセス制御を最少特権の原理（プログラム実行に必要な環境を最少限必要なだけに制限）にもとづいて行う方法のひとつに capability（特権切符）という考え方がある[4]．これは 1966 年に Dennis と Van Horn により提案されたもので CAL-TSS (CDC-6400)[11]，BCC 5000，SUE（トロント大 360），HYDRA，Cambridge Capability System，Plessy 250 などで試みられている．この考え方では，プロセス，ジョブ，プログラムなどを一般にサブジェクト（ユーザ）と考え，ファイル，プログラム，ディレクトリ，

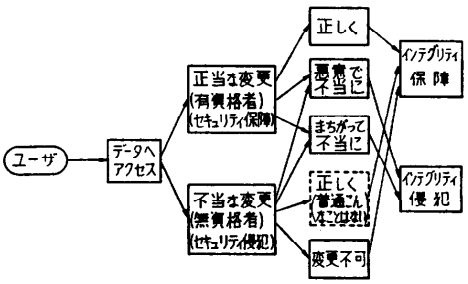

図-1 セキュリティとインテグリティ

セマフォア(制御ブロック)，メモリ・ブロック，ディスク・トラック，端末などを一般にオブジェクトとみなし，capability を次のように性格づける．

(a) サブジェクトはあるオブジェクトへの capability (切符) をもつときのみそれにアクセスできる．

(b) どんなアクセスができるかは capability によってきまる．

(c) capability はシステムの核で作られ，特権をへらす方向にしか変更できない．しかし capability の持主であるサブジェクトは，その capability を転送したり，コピーしたり，パラメータとして渡したりすることはできる．

Capability に関連した有名な問題に，(ホーマーの叙事詩オデッセイからきた)「トロイの木馬」問題がある．これは，プログラム (トロイ側) から呼出す (他人の) サブルーチン (木馬) に悪意 (敵意の古代ギリシャ兵がかくれている) があったときどうするかという問題である．悪意としては，たとえば，そのサブルーチンが，(a)秘密データを改変する，(b)他のファイルに秘密データを隠す，あるいはデータ通信で他のプログラムにモニタさせる(閉じこめ問題)，などがある．つまり何気なくサブルーチンを呼ぶと，呼んだ側が全く気付かぬうちに (正常だと信じているうちに) サブルーチンの側でセキュリティ侵犯をやっている場合である．これを防ぐには capability が有効である．

(3) 統計分析によるプライバシー侵害の防止

データ・ベースで厄介な問題のひとつに，「Aは医師である」ことが分かっているとき，「この町に離婚している医師は何人いるか」と聞いて，答えが「1人」だったとき，Aは離婚しているということ (プライバシー) がバレてしまうというような問題がある．これを妨ぐには，データ・ベースへのアクセス制御だけでは不十分であって，データ・ベースにアクセスするユーザ・プログラムを特殊な言語で書かせるなどして，ユーザとは切り離す (intentional resolution という) 必要がある[12]．この場合，当の言語には，上記のような統計分析的な質問があったとき，答の数が余り小さいときには，答を教えないといった機能をもたせるのである．

(4) データ・アクセスとデータ更新の同期

ファイルやデータベースを同時に多くのユーザの間で共用するときに生ずる厄介な問題は，データへのアクセスとデータ更新の衝突である．これを避けるには，たとえばデータ更新中は，他人によるアクセスを禁止するなどのいわゆるロック機構を設ければよいが，ロックのかけ方が不適当であると，デッドロック現象が起きやすくなる．この種のデッドロック問題はすでにいろいろな角度から研究されているからここでは述べないが，実際のシステムでは次のような機構のあることが望ましい．

i) データをアクセスしたとき，誰かがそれを更新中であれば，更新中につきアクセスできない旨のメッセージを出す．現在のシステムでは記号でしかメッセージが出ないものがある．

ii) 更新がすんだら，すんだことをユーザに知らせる．これを具体的にどうやるかは，いろいろな方法が考えられるが，「更新がすんだら知らせてくれ」というコマンドを出した端末ユーザには，更新がすむと同時にその旨のメッセージが伝わるとよい．

iii) 更新作業を行うユーザには，更新作業中に何人のユーザからアクセスがあったかを知る手段があるとよい．そうすればアクセスが多いときには，更新作業を一時中断して，後ですいているときにやるなどの対策がとれる．

6. 安全なコンピュータ・システムを作るには

最後に，信頼性が高くて安全なシステムを作るための心構えをのべて結びとしたい．

(1) 余りにも野心的で巨大かつ複雑なシステムは作らない．危いシステムは作らなければそもそも問題はないわけである．

(2) 運転要員やシステム開発・管理要員の人事管理に注意する．コンピュータ・システムの最大の弱点はそれを取り巻く人間である．データの扱いについては，東大のような官庁関係では「電子計算機処理データ保護管理要綱」が制定されている．

(3) セキュリティの立場からみたとき，現在の OS の最大の弱点は入出力機能にあり，そこの強化が必要である．ユーザ空間が個別に分かれている多重仮想記憶系をもつシステムでも，入出力チャネル関係には意外な弱点のあるものが多いとされている[13]．また端末側にかなりの権限移譲をした形のリモート・バッチ処理系は，端末側からの妨害行為には非常に弱い．

(4) データ・ベース[14]でとくに個人に関する情報を扱う場合には，とくに慎重な配慮が必要である．デ

ータ・ベースは，いろいろな問題があるところから，やたらに使うべきでないという意見すらある[15]．

（5） ソフトウェアの作り方については，ソフトウェア工学の方から，いくつかのガイドラインが出ている[15]．システム・プログラムの作成に当たっては，それらに従い，虫の混入の防止，虫や異常状態の検出，診断，虫の修正や異常状態の回復などに十分な考慮を払って，信頼性の高いソフトウェアを狙うべきである．

（6） セキュリティに関する研究開発を推進する．

今後は今までよりも複雑で大きな公共情報システムが続々と登場しそうであるから，わが国で今まで欠けていたこの面の研究を推進する必要があると思われる．

参 考 文 献

1) J. A. Scherf : Computer and data security : A comprehensive annotated bibliography, Project MAC Technical Reports TR-122 MIT (1974)
〔1973 年までの主な文献がほとんど出ている〕
2) 根岸，山本：オンライン文献検索システム TOOL-IR におけるマン・マシン・インターフェース，情報処理学会誌, Vol. 17, No. 5, pp. 402～409 (1976)
3) 石田，村田：超大型コンピュータ・システム，産報図書 (1975)
4) T. A. Linden : Operating system structures to support security and reliable software, ACM Computing Surveys, Vol. 8, No. 8, pp. 409～445 (1976)
5) J. Kruper : A computer system for non-stop high security operation, ONLINE, Vol. 14, No. 5, pp. 324～327 (1976)
6) ドン・B・パーカー（羽田訳）：コンピュータ犯罪，秀潤社 (1977)
7) S. B. Geller : The effects of magnetic fields on magnetic storage media used in computers, NBS Technical Note 735, National Bureau of Standards (1972)
8) G. E. Mellen : Cryptology, computers, and common sense, AFIPS Proc. of NCC, Vol. 42, pp. 569～579 (1973)
9) 石田：ベル研の軽装 OS-UNIX，情報処理, Vol. 18, No. 9, pp. 942～949 (1977)
10) C.S. Chanderseharan etc. : On virtual machine intergrity, IBM Sys. J., Vol. 15, No. 3, pp. 264～269 (1975)
11) B. W. Lampson & H. E. Sturgis : Reflections on an operating system, CACM, Vol. 19, No. 5, pp. 251～265 (1976)
〔失敗の記録として貴重〕
12) N. Minsky : Intentional resolution of privacy protection in database systems, CACM, Vol. 19, No. 3, pp. 148～159 (1976)
13) C. R. Attanasio etc. : Penetrating an operating system : a study of VM/370 integrity, IBM Sys. J., Vol. 15, No. 1, pp. 102～116 (1976)
14) D. C. Tsichritzis : Data Base Management Systems, Academic Press (1977)
15) G. Shussel : When not to use a data base, Datamation, Vol. 21, No. 11, pp. 82 (1975)
16) D. E. Morgan & D. J. Taylor : A survey of methods of achieving reliable software, IEEE Computer, Vol. 10, No. 2, pp. 44～52 (1977)

（昭和 52 年 6 月 14 日受付