ＯＳ/ｏ における記憶階層の仮想化

並木美太郎　武山潤一郎　中川正樹　高橋延匡

（東京農工大学　工学部　数理情報工学科）

１．はじめに

近年の半導体技術の進歩により、大容量の半導体メモリや固定ヘッドディスク装置を有するシステムが安価に入手できるようになった。しかるにハードウェアの進歩にかかわらず、パソコンのソフトウェアは10年前と変りがないものが多い。例えば、パソコンのＯＳなどでは、未だ、ファイル名に物理デバイス名を書くような状態のものがある。また、２次記憶装置もフロッピーディスク装置を中心に設計されたものが多く、安価になった固定ヘッドディスク装置に対する考慮がないものもある。すなわち、“仮想化”がなされていないものが多いのである。

情報が価値を持つ現在、マンマシンインターフェイスだけでなく、その情報を保管する２次記憶装置をいかに上手に扱うかが鍵になると思われる。しかし、パソコンでは半導体メモリ、固定ヘッドディスク装置、フロッピーディスクのような性能階層に関するアクセスの仮想化がなされておらず、ユーザが明示的に指定しなければならない。

当研究室では、研究用計算機システムとして開発中のオペレーティングシステムＯＳ/omicron（ＯＳ/ｏ ）に、上記の性能階層に関するアクセスの仮想化を導入し、性能向上を図った。本報告ではその開発方針と実現上の諸問題について述べる。

２．ＯＳ/ｏ の設計方針

ＯＳ/ｏ における、開発の際の問題意識は以下のとおりである。

(1)人工知能の研究、文字認識の研究などへの応用を前提とする。
(2)日本語情報処理に対して、標準的なアーキテクチャを考慮する。
(3)時間的制御を容易にするため実記憶方式を前提とする。

これら問題を解決するために我々は以下の機能を開発している。

(1)並列処理を実現するために、マルチタスクの機能を実現すること。
(2)実記憶方式のため、簡単な記憶保護の機能を実現すること。
(3)実記憶系でのプログラミングを考慮し、ＯＳ/ｏ の下ではリロケータブル、リエンタラブルなプログラムを標準とすること。
(4)日本語情報処理の研究のため、16ビットのJIS漢字コードを標準とすること。
(5)システム記述言語として言語Ｃを採用したこと。
(6)情報の統一的管理を目標としたファイルシステムとすること。

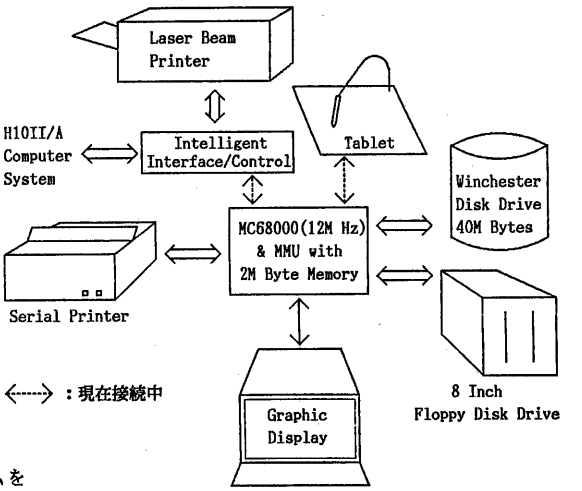

図１　システム構成

我々は、このようなシステムを図１に示すハードウェア上に構築する。 CPUにMC68000を採用したのは16Mバイトに及ぶ広大なアドレス空間を有するアーキテクチャだからである。実記憶方式で狭いアドレス空間ではパフォーマンスが上がらないと思われるからである。

３．ファイルシステムのアーキテクチャ

情報の統一的管理を目指す考え方として、一つは、すべて主記憶装置と考える方式がある。この代表例はMULTICSのセグメンテーションページングに基づく仮想記憶とファイルシステムである。この方式の長所は、ファイルを統一的に扱えることであるが、１セグメントの大きさでファイルの個々の大きさが制限されたり、性能設計が不可能なため、システムのオーバーヘッドが激増する可能性がある。いわゆる、汎用大型計算機向きなのである。

一方、パソコンでこのような統一的管理を行うために、我々は性能階層の異なる各種記憶装置をアクセスする方式を仮想化することで、ファイルのサイズの拡大等にも対処可能な拡張性のある方式を提案する。これを我々は性能階層に関するアクセスの仮想化方式と呼ぶ。

ＯＳ/ｏ では、ファイルシステムのアーキテクチャとして、この性能階層に関するアクセスの仮想化を基本とした。

４．ファイルシステムの特徴

ＯＳ/ｏ ファイルシステムは以下の特徴を有する。

(1)情報の統一的管理の指向

ＯＳ/ｏ はユーザの一つとして日本語情報処理の研究開発を考えている。そのため、入出力装置として通常のキー

ボードやプリンタの他にタブレット、レーザビームプリンタなどが接続される。ＯＳ/ｏ としては、このようなデバイスを扱いやすい形でユーザに開放する必要がある。また、大容量化・低価格化している半導体メモリを２次記憶のstaging areaとして用い、ファイルアクセスの高速化を図る。さらに、ＯＳ/ｏ ではコンパイラ、エディタのように頻繁に使用されるプログラムはROMにして常駐する予定である（ROMライブラリ）。ＯＳ/ｏ ファイルシステムでは以上すべてについてファイルの概念を拡張し、すべてファイルという形式で統一的に取り扱う。すなわち、磁気ディスク上に構築されたファイルは磁気ディスクファイルであるが、それに対し、I/Oデバイスに対するファイルをI/Oファイル、CPUのアドレス空間上に実体が置かれたファイルをLSIファイルと呼ぶ。

(2)木構造化されたファイル

ＯＳ/ｏ ファイルシステムは木構造を採用した(図２)。これは、ファイル名を一意に決定できること、ファイルの保護が比較的容易にできること、さらにはユーザがファイルの管理をしやすいことなどの長所がある。また、分散処理傾向に備え、ディスクは任意のディレクトリの下にマウント可能とした。

(3)ファイルの共有・保護

ファイルの共有はパーミションを用いて行なう。保護はパスワード方式を拡張したプライベート情報に関する質疑応答方式によってなされる。これは自分の生年月日、恋人の名前などをプライベート情報とし、ユーザが任意個登録しておいてアクセスの際に質問するものである。情報の個数、難易度によって機密の度合いをユーザが設定できる利点がある。また、上記の方式の他、物理的な方式として、よりいっそう機密を要するものについてはフロッピーディスクを用いユーザの責任において管理することも可能にした。

(4)ファイルの世代管理

ＯＳ/ｏ ではファイルの世代管理を行なう。これにより、プログラムだけでなく種々の文書の時系列情報をファイルシステム上に保存でき、保守性の向上が期待できる。なお、世代管理の単位はファイルシステムの統一性からファイル単位である。

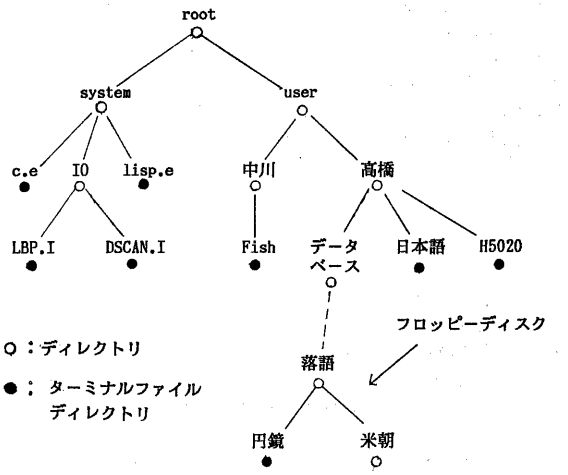

図２　木構造化されたファイル

(5)ディスクアクセスの高速化技法の確立

パソコンではアクセス速度の遅いフロッピーディスクをいかに扱うかがパフォーマンス向上の鍵となる。ＯＳ/ｏ ファイルシステムでは仮想フロッピーディスクとLSIファイルの概念を導入し性能階層の仮想化を図る。

５．ファイルシステムの構成

５．１　ファイルの属性、ページサイズ

ＯＳ/ｏ でサポートするファイルは、ディスク装置への割り付け等を考慮して直接編成ファイルとした。無論、順編成ファイルとしてもアクセス可能である。ファイルの内容としては、文字（キャラクタ）、バイナリの２種類を指定できる。ファイルのアクセス単位（ページサイズ）はディスクの使用効率、バッファのサイズ等を考慮して、1KBとした。ＯＳ/ｏ ファイルシステムでは磁気ディスクに関し、ディスクのアドレスをページ番号で指定する。すなわち、ディスクに対し、1KBずつ分割し、それに番号付けしたものが論理的なディスクアドレスとなる。

５．２　ディレクトリ名

ディレクトリ名は、JIS漢字コードで最大16文字であり、ファイルではさらに拡張子 1文字が使用可能である。パスネームは各ディレクトリ名を“／”で区切って作成する。例えば、図３中の“美太郎.C”というファイルをアクセスする場合は

／数理情報／第１講座／高橋研究室／美太郎.C

とすれば良い。これはルートディレクトリからのアクセスとなる。もし、ワーキングディレクトリが“第１講座”であるならば、

高橋研究室／美太郎.C

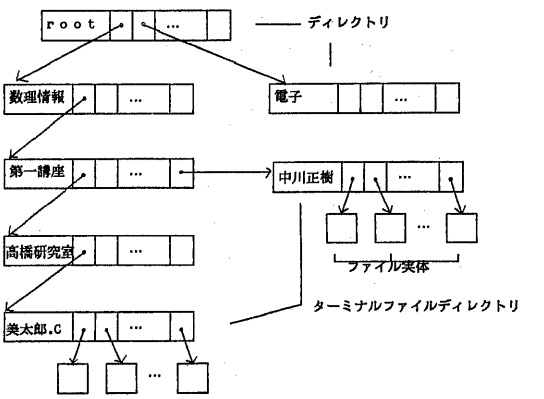

図３　ディレクトリ、ターミナルファイルディレクトリ、ファイル実体

とすれば良い。

5．3　ディレクトリの構造

ディレクトリはファイルシステムの構造を表わす“ディレクトリ”とファイル実体の種々の情報を有する“ターミナルファイルディレクトリ”の２種類がある。構造を図４に示す。なお、ディレクトリのサイズはアクセスの統一性を考慮して１ページとしてあり、これを越えると次のディレクトリが割り当てられる。

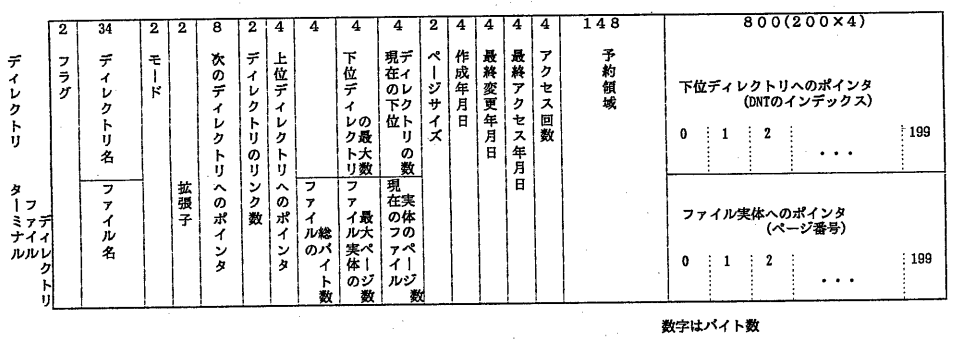

図4　ディレクトリの構造

5．4　ファイルシステムの管理表

ファイルシステムは次の５つの表から構成されている。

- DMT (Disk Mount Table)
- DNT (Directory Name Table)
- FPT (Free Page Table)
- AFT (Active File Table)
- LSIFT (LSI File Table)

これらの表は、大まかに、図５のような関係となっている。

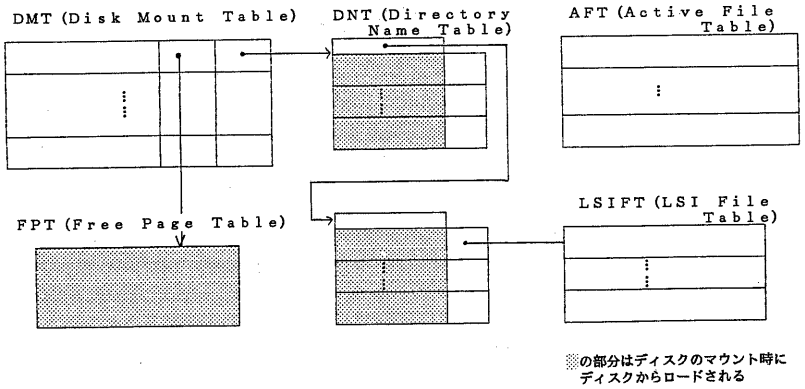

図５　各表の関係

以下に表の概略を示す。

(1)DMT (Disk Mount Table)

着脱可能なディスクを管理する表である。例えば、フロッピーディスクをドライバにマウントした時、この表にドライバのデバイス番号、そのディスクの容量を表わす最大ページ数、FPT(後述)、どのディレクトリの下にマウントされたかなどの情報が書き込まれる。この表のインデックス０はルートディレクトリのあるディスクを表わしている。

(2)DNT (Directory Name Table)

DNTはディレクトリ名とディレクトリのあるページ番号（論理的なディスクアドレス）の対応をとる表である。この表はディスクに一つ必ず存在し、通常のファイルの形式で格納されている。ディスクのマウント時にディスクからメモリにロードされ、必要な限りメモリに常駐する。葉でないディレクトリが、より下のディレクトリを指す場合、この表のインデックスを指している。これにより、ディレクトリのサーチの高速化を図っている。DMT,DNTの関係を図７に示す。１ページ１KB中に26個分のディレクトリが入る。もし、26個を越える場合は１ページ分毎にリストでつながれてメモリにロードされる。なお、インデックス０はそのディスクのトップレベルのディレクトリである。

ディレクトリのサーチは次の手順で行なう。例えば、図６において

数理情報／第一講座／高橋研究室／美太郎.C

というディレクトリをサーチすると次の様になる。ここで“数理情報”のディレクトリはメモリにロードされており、DMTインデックス０のディスクとする。

(ｉ)“数理情報”というディレクトリ中の下位ディレクトリへのDNTインデックスを用いて、“第一講座”というディレクトリをDNT上でサーチする。図６ではインデックス8、12をチェックする。

(ii)“第一講座”というディレクトリが見つかったら、ペアになっているページ番号を用いてディスクよりディレクト

リをロードする。図６では14ページをロードする。
(iii)“第一講座”がターミナルファイルディレクトリでなければディレクトリ中の下位ディレクトリへの DNTインデックスを用いて“高橋研究室”というディレクトリをDNT上でサーチする。もし見つからない場合には、“第一講座”の下にマウントされているディスクがあれば、ディスクのDMTインデックスと、そのインデックスによって指されるディスクのDNTインデックス０を用いて、そのディスクのトップレベルのディレクトリが“高橋研究室”かどうか比較する。図７では“第一講座”の下にDMTインデックス１のディスクがマウントされており、ディスクのトップレベルのディレクトリが“高橋研究室”となっているので、以後はDMTインデックス１のDNTを使用する。
(iv)以下、同じようにして“美太郎.C”をサーチする。

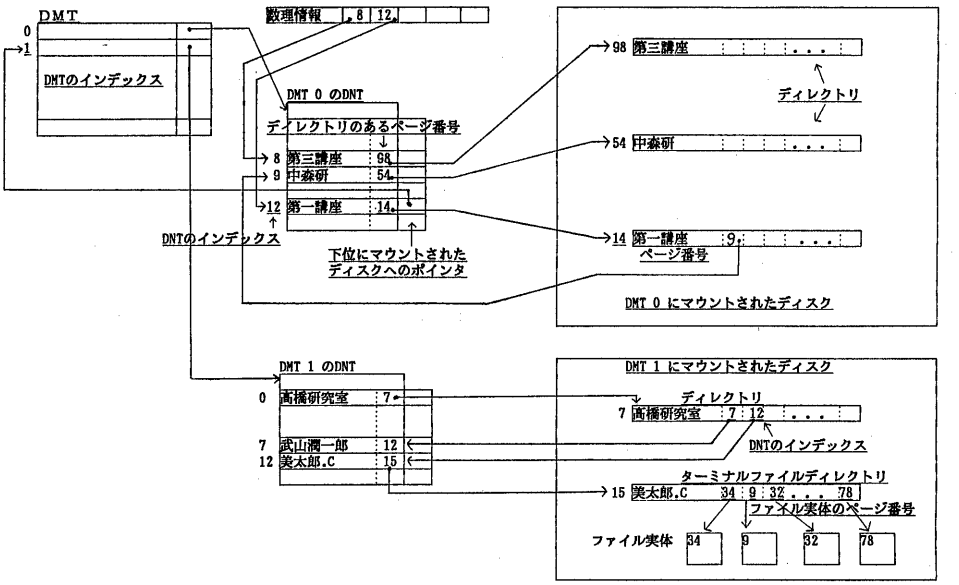

図６　ＤＭＴとＤＮＴの関係

このようにディレクトリ名がメモリにロードされているのでファイルのサーチは高速に行なわれるだけでなく、任意のディレクトリの下にディスクがマウント可能となっている。なお、このDNTはLSIファイルを実現する上で重要な役割を持っている。
(3)FPT (Free Page Table)
そのディスクの空きページを管理する表であり、ビットマップのテーブルとなっている。この表はファイルとしてディスクに格納されており、ディスクのマウント時にメモリにロードされる。
(4)AFT (Active File Table)
オープンされたファイルに関する情報が登録される。例えば、バッファのアドレス、どのモードでオープンされたかなどが書かれる。AFTのインデックスをファイル記述子と呼び、ファイルのオープン時にこの値がオープンしたプログラムに返される。以後のファイル入出力はこのファイル記述子を用いて行なわれる。
(5)LSIFT (LSI File Table)
LSIファイルに関する表である。第６章で述べる。

６．ＬＳＩファイル

６．１　ＬＳＩファイルの概念

LSIファイルとは、通常、磁気ディスク上に置かれるディレクトリやファイル実体を半導体メモリ(RAM,ROM)上に置き、アクセスの高速化を目指したものである。原理は簡単であるがファイルシステムへの取り込みかたにより、いくつかの方法がある。
(1)CPUのアドレス空間以外にメモリを置く方法
この方法はファイル用メモリをCPUのアドレス空間におかず、他のデバイスと同じようにある特定のチャネルを介してデータ転送を行なう方法である。利点はCPUのアドレス空間を圧迫しないこと、欠点はハードウェアに工夫が必要なこと、転送のオーバーヘッドがあることである。
(2)CPUのアドレス空間におき、I/Oハンドラを通じてデータ転送を行なう方法
この方法はファイル用メモリをCPUのアドレス空間におき、I/Oハンドラ(実際はメモリ間転送ルーチン)でデータを転送する方法である。利点はファイルだけでなくディレクトリもLSI化が可能なことがあげられる。欠点は転送のオ

ーバーヘッドがあることである。
(3)CPUのアドレス空間におき、ファイル実体をバッファと見なしてアクセスする方法
　この方法はメモリ上にあるファイル実体をファイルバッファと見なし、ユーザプログラムとの間でデータ転送を行なう。利点はオーバーヘッドが少ないことである。
　(1)、(2)の方法はファイルシステムの変更なしで実現できるので、手を入れることの出来ないＯＳには有効な手段であろう。I/O管理にハンドラとして登録すれば比較的手軽に実現できる。CP/M-68K上で実現した所、確かにアクセスは高速化された[8]。しかし、(1)、(2)の方法はファイルシステムの「外の世界」で実現されているので、不都合なことが無いわけではない。例えば、記憶保護の観点から見た場合、ファイル実体となるべき主記憶の保護をI/Oハンドラレベルで行なうよりは、より高次のファイルシステムで磁気ディスクなどと統一的に扱ったほうが良いはずである。　さらに、仮想化のレベルがあくまで、「ディスク」レベルであり、「ファイル」レベルでないこともあげられる。たとえば、磁気ディスク上のソースファイルをLSIファイルに持ってきてデバッグしようとすると、磁気ディスク<->メモリ（LSIファイル）というデータ転送を「ファイルコピー」という形で明示的に行なう必要がある。これは、とりもなおさず、異なる記憶階層間のデータ転送を明示していることに他ならない。ユーザインターフェイスの観点からは、これは望ましいことではない。
　ＯＳ/ｏ ファイルシステムでは(3)の方法を採用し、ファイルシステムにLSIファイルを取り込んでいる。

６．２　LSIファイルの実現

　ＯＳ/ｏ ファイルシステムでは次のようにLSIファイルを実現している。
(1)ディレクトリの形式は磁気ディスクファイルと全く同じ形式をとる。ただし、ファイル実体へのポインタはページ番号ではなく、ファイル実体となるメモリのアドレスである。磁気ディスクファイルかLSIファイルかの区別は、ディレクトリのモード中のビットで示される。
(2)LSIFT (LSI File Table)では、次の情報が書き込まれる。
　(ｉ)メモリ上にあるディレクトリのアドレス
　(ii)兄弟のレベルにあるファイルへのポインタ(双方向)
(3)DNT (Directory Name Table)では、ディレクトリ欄毎に一つLSIFTへのポインタが割り当てられており、これにより、木構造を決定する。ただし、このポインタはテンポラリなものであり、磁気ディスクには格納されない。ディスクをディスマウントすると、この構造は消失する。
　以上の関係を表わしたものが図７である。
　ファイルをサーチする時は、DNT上にあるLSIFTへのポインタを用いてファイル名を得る。

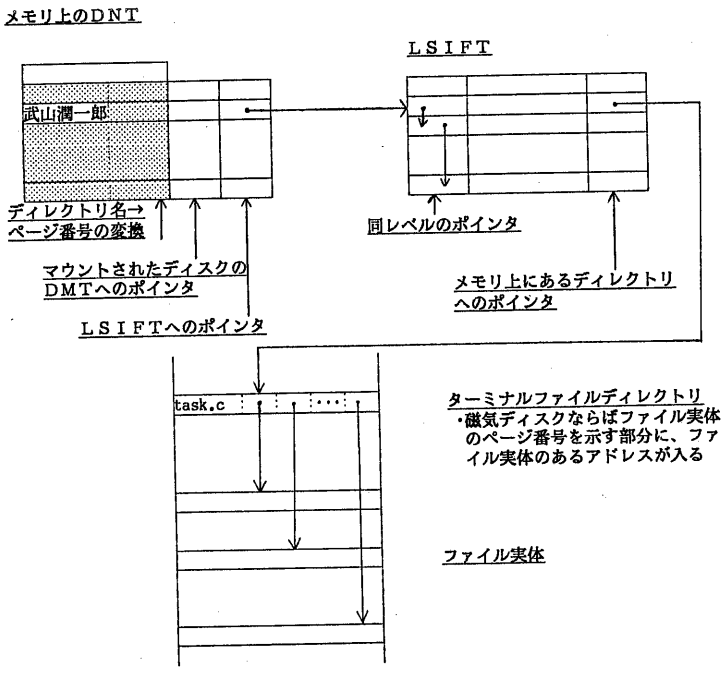

図７　LSIファイルの実現

例えば、図７において

高橋研究室／武山潤一郎／task.c

をサーチする時は、“高橋研究室”からメモリ上にあるDNT中の“武山潤一郎”のインデックスを得る。この次のステップは、“武山潤一郎”のディレクトリをロードする前に、DNT上のLSIFTへのポインタより、LSIFTにあるディレクトリで“task.c”があるかをチェックする。もし、なければ“武山潤一郎”のディレクトリをロードし、再び、DNT上のディレクトリ名をチェックして行く。

６．３　ファイルによる磁気ディスク・LSIメモリの性能階層の仮想化

６．１で述べた通り、磁気ディスク<->メモリ（LSIファイル）というデータ転送を「ファイルコピー」という形で明示的に行なうのは好ましいことではない。そこで、ＯＳ/ｏ ファイルシステムでは、これを「ファイルのモード変更」という形で行なうことにした。ファイルのモードとは、例えば、許されるオープンのモード（リード、ライトなど）などであるが、この中にLSIファイル／磁気ディスクファイルかを示すビットもある。通常、モードはファイル生成時に指定するが、SVC(SuperVisor Call)により、変更可能である。このモードをLSIファイル→磁気ディスクファイル、磁気ディスクファイル→LSIファイルと変更することにより、データ転送を暗黙に行なう。

(1)磁気ディスクファイル→LSIファイルというモード変更

この場合はLSIFT上にLSIファイルを新たに作成し、磁気ディスクファイルの内容をコピーする。この後、磁気ディスク上のファイルは消去せずに残しておく。これは、安全性を考慮したからである。こうすると、同じファイルが2つ存在することになるが、６．２で述べたサーチルールにより、必ずLSIファイルが先に探索されるので問題は無い。

(2)LSIファイル→磁気ディスクファイルというモード変更

(ｉ)磁気ディスクファイルにそのファイルが存在しない場合

磁気ディスク上にファイルを作成し、コピーした後、LSIファイルを削除する。

(ii)磁気ディスクファイルにそのファイルが存在する場合

これは、先の(1)で変更したファイルを再び磁気ディスクファイルに戻す場合である。これは、コピーした後、LSIファイルを削除するだけである。

以上のように、ＯＳ/ｏ ファイルシステムでは、磁気ディスク<->LSIメモリのデータを明示的に行なわず、「ファイルのモード変更」という形で仮想化を図った。

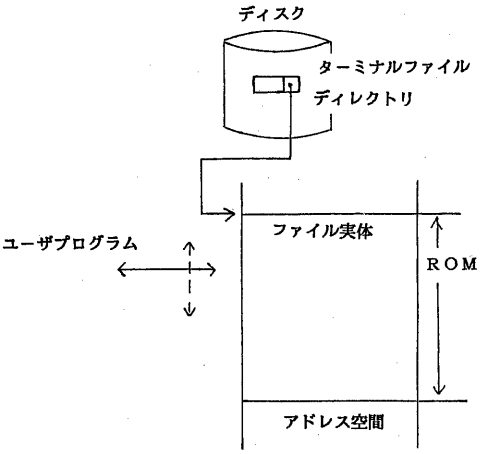

図８　ROMに対するLSIファイル

６．４　ROMに対するLSIファイル

ＯＳ/ｏ ではROMライブラリを実現するが、ファイルシステムではROMもファイルとして仮想化を図った。ROMに対するLSIファイルはファイル実体が常駐していること、ファイルの大きさが一定であることから、ターミナルファイルディレクトリはディスク上に、ファイル実体は主記憶領域に取られる。したがってROMのデータは主記憶のアドレスを直接参照することでアクセス可能となる(図８)。

７　仮想フロッピーディスク

７．１　仮想フロッピーディスクの概念

近年、マイクロウインチェスターディスクのような固定ヘッドディスク装置が安価に入手出来るようになり、パーソナルコンピュータに標準的に装備されるようになった。固定ヘッドディスク装置はアクセスが速いと言う利点があると同時に容量に制限がある(5MB~40MB)という欠点がある。一方、フロッピーディスク装置はアクセスが遅く、一枚当りの容量が少ないという欠点はあるが、交換可能であること、ビット当りの単価が安いという利点がある。それぞれの利点･欠点を見ると、固定ヘッドディスク装置はデバッグ中のプログラムやデータベース、ライブラリファイルの格納など効率を重視したケースに向いている。対して、フロッピーディスク装置はファイルのバックアップや個人情報の管理に向いている。

ＯＳ/ｏ ファイルシステムでは固定ヘッドディスク装置、フロッピーディスク装置、それぞれの長所・短所を合わせて以下の特徴を有する仮想フロッピーディスクの概念を導入する。

(1)容量の制限が事実上無い。

(2)フロッピーディスクのアクセスタイムを固定ヘッドディスクと同程度にする。

７．２　仮想フロッピーディスクの実現

基本的には、固定ヘッドディスク装置をフロッピーディスクのstaging areaとして用いることで実現される。ただ、staging areaとして用いるならば、フロッピーディスク上のファイルを使用前に全て固定ヘッドディスク装置にコピーし、使用後はフロッピーディスクに戻してやれば良いのだが、これではアクセスしないファイルまでコピーすることになり、コピーにかなりの時間を必要とする。そこで、ＯＳ/ｏ ファイルシステムでは必要になった場合にのみ、コピーを行なう。そのため、ディスクを仮想フロッピーディスクとしてマウントした場合は、DMT (Disk Mount Table)に固定ヘッドディスク<->フロッピーディスクのページの変換表を付加する。この表にはフロッピーディスクのページに対応する固定ヘッドディスクのページ番号の他、未だ固定ヘッドディスクに割り当てられていない場合を

示すページフォルトビットを有している。これを示したのが図９である。
なお、固定ヘッドディスクの内容をいつフロッピーディスクに戻すかについては、以下のルールをとっている。
(1)ディレクトリ、ターミナルファイルディレクトリのようにファイルシステムの構造を決定するものについては即座にフロッピーディスクに戻す。
(2)ファイル実体についてはフロッピーディスクのディスマウント時にまとめてフロッピーディスクに戻す。ただし、変更されなかった場合は戻されない。なお、固定ヘッドディスクの容量が不足した場合は、随時戻される。

８．今後の計画

現在、ファイルシステムはLSIファイルまで実現されている。以下が課題として残されている。
(1)仮想フロッピーディスクの実現
(2)ファイルの共有･保護の実現
(3)世代管理の実現

９．おわりに

性能階層の仮想化を盛り込んだＯＳ/ｏファイルシステムについて述べた。オフィスオートメーションの発展を考慮すると、パソコンの２次記憶装置に対する深い考察が必要なのではなかろうか。

最後に、ＯＳ/ｏ の設計会議の出席者、特に中森眞理雄助教授に深謝する。また、本研究の基礎を築いた大学院生、武部桂史（現日立）、林努（現日立）、藤森英明（現日電）の諸氏に謝意を表わす。

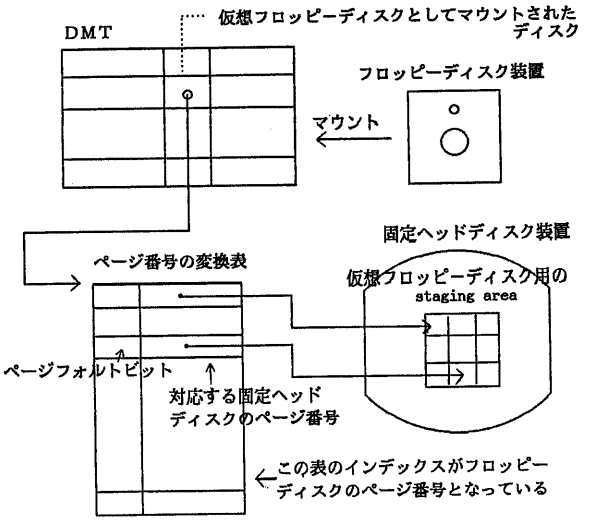

図９　仮想フロッピーディスクの実現

参考文献

［１］　林努，高橋延匡，“MC68000用ＯＳの基本設計(1) －ファイルシステムの設計－”，情報処理学会マイクロコンピュータ研究会資料25-3，1982.12
［２］　高橋延匡，武山潤一郎，並木美太郎，中川正樹，“MC68000 用小型ＯＳ：ＯＳ/ｏ の開発”，情報処理学会計算機システムの制御と評価研究会資料21－6，1983.12
［３］　中川正樹，篠田佳博，藤森英明，高橋延匡，“MC68000 ユニ＆マルチ・プロセッサ・システム用システム記述言語Ｃ処理系の開発”，情報処理学会計算機システムの制御と評価研究会資料21－7，1983.12
［４］　高橋延匡，“ＯＳ／omicron の設計思想”，情報処理学会第29回全国大会予稿，1984.9
［５］　並木美太郎，中川正樹，高橋延匡，“ＯＳ/ｏ のファイルシステム”,情報処理学会第29回全国大会予稿,1984.9
［６］　武山潤一郎，並木美太郎，中川正樹，高橋延匡，“ＯＳ/ｏ のタスクとタスク管理”，情報処理学会第29回全国大会予稿，1984.9
［７］　中川正樹，篠田佳博，高橋延匡，“ＯＳ/ｏ のプログラム実行環境とプログラム・リンケージ”，情報処理学会第29回全国大会予稿，1984.9
［８］　並木美太郎，中川正樹，高橋延匡，“フロッピーディスクのLSIファイルによるＯＳの一性能向上方式 － CP/M-68Kにおける適用結果 －”，本学会第29回全国大会予稿，1984.9
［９］　高橋延匡，並木美太郎，武山潤一郎，中川正樹，“ＯＳ/ｏ のアーキテクチャと第一版の実現”，情報処理学会計算機システムの制御と評価研究会資料24-11，1984.9