MC68000用OSの基本設計(1)
－ファイルシステムの設計－

林 好，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

現在我々は，一研究室で使用する規模の，16ビットマイクロプロセッサを用いたスーパーパーソナル・コンピュータ・システムを開発中である。対象機種としては，アドレス空間の大きいMC68000を採用した。

上記のシステムソフトウェアであるオペレーティング・システム（OS）の基本方針として，次のものを挙げた。

（1）将来のスーパーパーソナル・コンピュータ・システムに必要な機能を容易に実現できるように設計すること。
（2）当初はOSの核だけを作成し，十分な拡張性を考慮すること。
（3）日本語情報処理への応用を前提とすること。
（4）OSの常駐部は，できるだけ小さく作成し，ユーザプログラムを圧迫しないように設計すること。

上記の狙いをこめて，我々はそのOSをOmicron（以下OS/oと略記）と名付けた。

我々は，将来あるべき姿のスーパーパーソナル・コンピュータ・システムのOSとして以下の機能が不可欠であると考えた。

（1）マルチユーザ・マルチタスクが可能なこと。
（2）インハウスTSSとして機能すること。
（3）固定ヘッドディスク装置をベースにしたシステムにすること。
（4）日本語情報処理を十分に意識すること。
（5）他システムとの交信が可能なこと。

本論文では，上記のOS/oの核の一つであるファイルシステムの設計について報告する。

2．設計方針

将来の日本型スーパーパーソナル・コンピュータ・システムに不可欠な機能を容易に実現できるようなOSとする必要がある。そこで，そのOSの核となるべきファイルシステムの設計を行なうにあたって次のような方針を立てた。

（1）日本語情報処理が容易に実装できるように考慮する。

日本語文書を計算機に格納することは，それらの文書の保守の面から重要である。しかし，他のシステムとの互換性，ファイルのスペースの効率を考慮すると，日本語ファイルをサポートするためには，1バイトコードと2バイトコードとの混在したファイルをサポートできなければならない。また，文書の中には，図，表等があり，それらの扱いについても考慮する必要がある。

（2）仮想フロッピーディスクの概念を導入する。

現在採用している固定ヘッドディスク装置の容量には，40メガバイトという制限がある。その制限に対処するために，フロッピーディスク装置（以下FDと略記）を用意した。しかし，FDはアクセスが遅いという欠点を持つ。そこで，FDの内容を，ディスク装置上のファイルと同様な方法で，かつ同程度の効率でアクセスできる仮想フロッピーディスクの概念を導入する。

（3）ファイルの世代管理を行なう。

プログラムにかぎらず，設計仕様書等文書情報の修正の時系列を残すことは，システムの保守の面から重要である。しかし，一般に世代管理を完全に実施すると，それに伴うオーバーヘッドが激増する傾向にある。設計の要点としては，アクセスタイム，容量等を考慮し，世代管理用情報として何を残すかを厳選することである。

（4）ファイルの保護を十分に行なう。

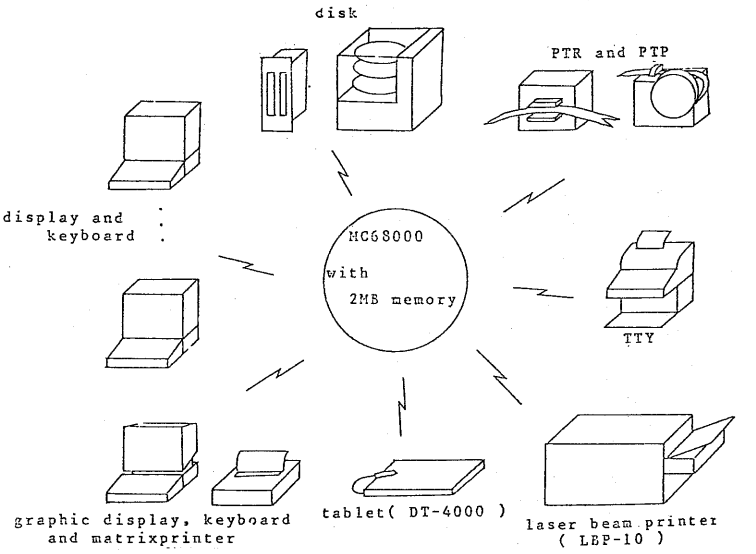

図1　ハードウェア構成

ファイルをアクセスしようとするユーザが本人であることを確認することは，難しい。従来のシステムの多くは，パスワードを用いていたが，それでも完全に確認することはできない。このシステムでは，パスワードを拡張した方式（具体的には，プライバシーに関する質疑応答）を用いて行なう。

（5）ファイルの共有を容易に行なえるようにする．

現在開発中のシステムでは，グループ単位でソフトウェアを開発することが多い。そのため，文書あるいはプログラムの共有が頻繁に行なわれる。このことから，共有が容易に行なわれ，かつ十分に保護がなされるべきである。

これらの方針を基に設計されたファイルシステムを，実装するシステムのハードウェア構成を図1に示す。

3．ファイルの種類

3．1ファイルの属性

OS/0がサポートするファイルは，ファイルのディスク装置への割り付けを考慮してすべて直接編成ファイルとした。もちろん，直接編成ファイルは，順編成ファイルとしても扱うことができる。ファイルのアクセス単位は，ユーザが仕事をする単位と考えられるA4用紙1ページ（全角文字で46文字/行×66行）を採用した。

また，それらは大別して，二つのグループになる。一つのグループは，世代管理を行なう文字列の格納されているファイル，もう一つのグループは，世代管理を行なわない2進データの格納されているファイルである。2進データのファイルの例としては，図があるが，その世代管理を行なうならば，1枚の図の単位で行なわなければならない。そうすると，その図を別のファイルにとっておくことと同程度の人もりを要してしまう。また，オブジェクトモジュール，ロードモジュールは，ソースプログラムのレベルで世代管理が行なわれている。以上のことから，2進データファイルは，世代管理を行なわないことにした。

これに加えて，ファイルのイクステンションを設定したので，OS/0がサポート

するファイルの属性を五つとした。これを表1に示す。

表1　ファイルの属性

属性	使用例	エディット	世代管理
文字データファイル	マニュアル	可	行なう
ソースプログラムファイル	言語Cソースプログラム	可	行なう
2進データファイル	図	可	行なわない
オブジェクトモジュールファイル	言語Cオブジェクト	不可	行なわない
ロードモジュールファイル	言語Cロードモジュール	不可	行なわない

3．2日本語ファイルのサポート

OS/omは，日本語情報処理を標準的にサポートする。これは，設計仕様書等マニュアルを計算機で管理することにより，システムの保守を容易に行なうことを目的としたからである。また，プログラムの中に，漢字仮名まじり文でコメントが付けられるようにする意義も大きい。しかし，現在日本語情報処理で言うところのアプリケーションをすべてカバーする設計は難しい。

そこで，OS/omでは日本語情報処理の基盤として，アルファベットやカタカナでなく，日本語が標準言語となるよう，次の方針を立てた。

（1）漢字，ひらがな，カタカナ，英数字等が含まれるテキストファイルを，ファイルシステムで標準的に扱えるようにする。

（2）上記のコード体系はJISに準拠する。また，他のシステムとの互換性，ファイルスペースの効率を考慮して，1バイト，2バイトコードの混在をファイルシステムで制御する。

（3）図・表等の処理を可能にするために，2進データファイルを導入する。

現在，上記のファイルシステムを利用するオンライン手書き文字認識システムJOLIS，日本語文書浄書システムJOSHOが当研究室において開発中である。[1),2)]

4．ファイルの世代管理

OS/omは，日本語文書（特にマニュアル類）をサポートする。これらは，ソースプログラムの修正に伴って，修正が行なわれる。この時，その修正の時系列がわかることは，文書の保守だけでなく，システムの保守の面からも重要である。また，プログラムにしても，以前に開発したソフトウェアの誤りを記録しておくことは，それ以後の開発に役立つ。これらのことを，人手を介さずにやることにより，その情報の信頼性が向上する。このため，ファイルの世代管理を行なう。

ここで問題となるのは，世代管理情報として何を残すかである。当然ユーザの修正と世代管理情報とのマッチングがとられている必要がある。

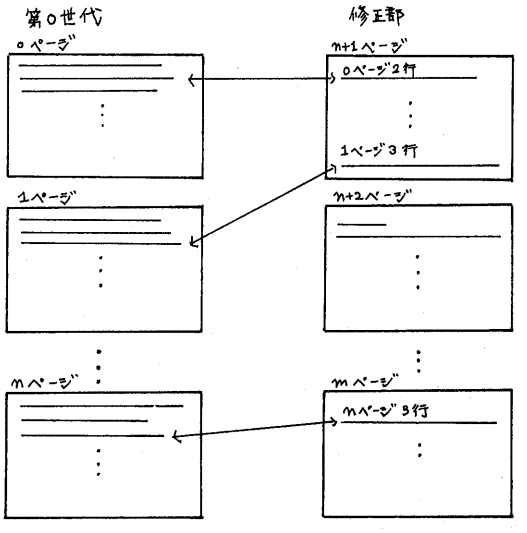

図2　世代管理用情報を含めたファイルの出力形式
（第0世代がmページの途中で終了した場合は，改ページされる。）

(例えば，行の挿入をユーザが行なったならば，世代管理情報として“行の挿入”が行なわれたことが残らなければならない。)世代管理する単位としては，文字単位，行単位，段落単位，ファイル単位等がある。 文字単位とするならば，1文字ごとに世代管理のための情報が必要なので，それらのための情報が大量になる。反対に段落単位，ファイル単位とすると，1文字修正するごとにその単位の情報を持たなければならないので，ファイルの容量が大きくなる。そこで，行単位にとることにした。つまり，世代管理用情報としては，この行は，次のように修正されたという情報を残すことになる。

世代管理情報を併せてファイルを出力した例を，図2に示す。

5．ファイルのアクセス権

ファイルの保護するために，そのファイルをアクセスしようとするユーザが，本人であることを確認できなければならない。しかし，これは大変難しい。これを従来の多くのシステムでは，パスワードを用いて行なっていた。この方法の問題としては，パスワードが一度漏れると，本人であることが確認できなくなることである。このため，公開鍵方式を用いたシステムも多い。[3] しかし，このシステムは一研究室で使用する規模を想定しているので，上記の方法を用いる必要はない。

そこで，このシステムでは，パスワードを拡張した方式を採用した。つまり，ファイルをアクセスしようとするユーザに対して，そのユーザのプライバシーに関する質問を行なって，本人であることを確認する。

プライバシーとしては，生年月日，電話番号，ニックネーム等がある。ところが，個人によってプライバシーというものは違うので，個人の責任において，質問事項を決めてもらうことにした。

この方法を用いるならば，ほぼ完全に近い保護の体制をとることができる。例えば，質問の数を多くすることにより，本人以外のアクセスはほとんど不可能になるだろう。また，質問の数を零にすれば，ファイルを完全に開放することになる。つまり，ユーザが，対象となるファイルの機密の重さに応じて，保護の度合いを決めることができる。

6．ファイルの共有

ファイルの共有に関しては，次のような方針を立てた。

(1) データの機密保全を十分に行なう．

(2) ファイルの共有が容易に行なうことができる．

ファイルには，情報が格納されている。それらの情報の中には，特定のユーザ以外には，公開したくないものもある．このように特定のユーザにだけアクセスを許すような機構が必要である。また，ユーザの誤操作により，他人の情報が破壊されるようなことがあってはいけない。以上から，方針(1)を挙げた。

しかし，機密保全を十分に行なうために，オーバーヘッドが増大してもいけない。例えば，ユーザが他のユーザに共有を許すために，そのファイルを共有しているグループの全てのユーザに同意を求める必要があると，機密保全は十分なされるが，使い勝手は悪くなる。以上から方針(2)を挙げた。

ただし，OSが開発中であることを考えて，方針(2)を重視する．

このことから，ファイルのコピーを譲渡するだけでなく，同一ファイルを共有できるようにした。

7．ファイルシステムの構造

7．1 階層構造

ファイルシステムの構造としては，計算機を使用する組織の形態と同様なファイル管理が行なえるように，階層構造をとることにした。これによって，次のような利点が得られる。

（1）ファイルの保護が自然に行なえる。
（2）ファイルのアクセスが高速に行なえる。
（3）ユニークなファイル名を容易に作ることができる。
（4）各種のシステム管理情報を収集しやすい。

またファイルシステムは，ファイル実体に関する種々の情報を持っている必要がある。種々の情報とは，次のようなものがある。

（1）ファイルの属性
（2）ファイルの共有を許すかどうかの情報
（3）ファイルの大きさ
（4）ファイル実体へのポインタ
（5）ファイルの所有者
（6）ファイルの作成年月日
（7）その他

以上のことから次に示すような，2種類のディレクトリを用意する。

（1）ファイルシステムの構造を示すディレクトリ
（2）ファイル実体の種々の情報を有するディレクトリ

（以下では，(1)をディレクトリ，(2)をターミナルディレクトリと呼ぶ。）

それぞれの構造を図3，4に示す。この構造を決めるにあたっては，次のことに注意した。第1には，アクセスするために，できるだけ1セクタ（512バイト）におさめるようにしたことである。第2には，ファイルシステムのエラーに対処するために，相対ポインタを用いたことである。

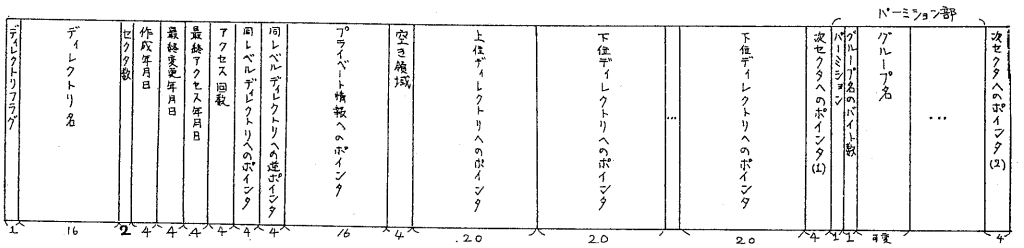

図3　ディレクトリの構造（下の数字はバイト数）

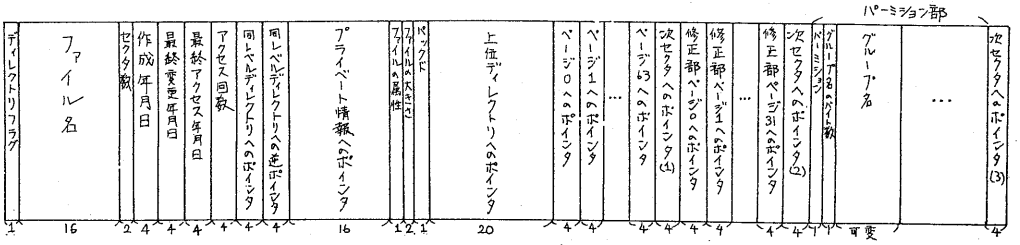

図4　ターミナルディレクトリの構造（下の数字はバイト数）

また，ファイルの保護，アクセスの高速化，および拡張性を考慮して，基点から五つのディレクトリ（ユーザ，ライブラリ，コマンド，システム，データベース）に分岐する．図5にファイルシステムの構造例を示す．

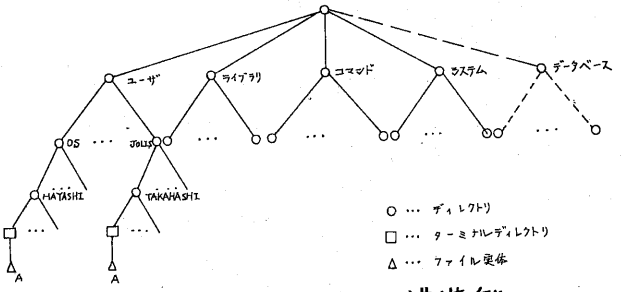

図5 ファイルシステムの構造例

7．2 ファイル実体の構造

3.1節で示したように，ファイル実体は，大別して二つのグループになる。そのグループは，世代管理を行なう文字データファイルと，世代管理を行なわない2進データファイルになる．2進データファイルには，単に2進のデータがつめこまれる。

文字データファイルの構造を決めるにあたっては，世代管理を行なうと同時に，最新情報へのアクセスが遅くならないように注意した。前にも述べたように，世代管理は行単位に行なう。そこで，各行に四個のポインタを用意する。

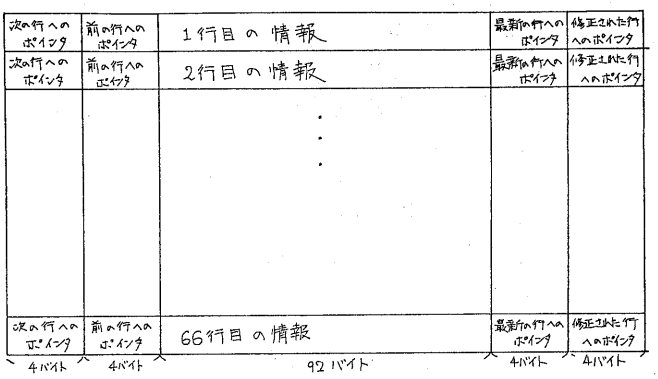

図6 ページの構造

（1）次の行へのポインタ
（2）前の行へのポインタ
（3）最新の行へのポインタ
（4）修正された行へのポインタ

各々のポインタは，ページ番号（3バイト），行番号（1バイト）からなる。つまり，各々のポインタの大きさは，4バイトである．また，行長は固定長とし，その大きさを92バイトとする。

1ページの大きさ，つまり直接編成ファイルのアクセス単位は，アクセスする際のキーがユーザにわかりやすいということから，A4サイズの日本語文書1ページ分とした。これにより，1ページの大きさは，108バイト×66行＝7128バイト≒14セクタとなる。（A4には，全角文字で，46文字/行×66行が出力できる。）図6に，ページの構造を示す。

8．仮想フロッピーディスク

8．1 仮想フロッピーディスクの概念

OS/omでは，比較的大容量の2次記憶（40メガバイトの固定ヘッドディスク装置）をサポートしている。多くのユーザは，このディスク装置をベースにしてシステムを使用する。しかし，ディスク装置が大容量といっても，40メガバイトという制限がある。そこで，その制限を緩和するために，FD（フロッピーディスク装置）も実装することにした．表2に，OS/omで用いるディスク装置とFDの性能比較を示す。

表2 OSで使用するディスクとフロッピーディスクの比較 4),5)

比較項目	ディスク	フロッピーディスク
容量	40メガバイト	1メガバイト/枚
取りはずし	不可	可
転送速度	800キロバイト/秒	62キロバイト/秒
割付け法	共有	専有
ファイル形式	直接編成	順編成

表2から明らかなように，仕事の効率を考慮するならば，FDは単なる情報のセーブにむいている。そこで，次に示す特徴を持つ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ない。
（2）FDのアクセスタイムをディスク装置のそれと同程度にする。
（3）ディスク装置上のファイルと同じ方法でアクセスできる．
（4）FDのドライバーが1台の時は，それを共有装置とする。

これらの特徴を出すために，図7に示すように仮想フロッピーディスクを行なう。つまり，ディスク装置の特定の領域を，FDのステージングエリアとして使用する。そして，その領域を介して，FDにあったファイルをアクセスする。仮想フロッピーディスクが終了した後は，ふたたびFDに，領域内の情報が再格納される。

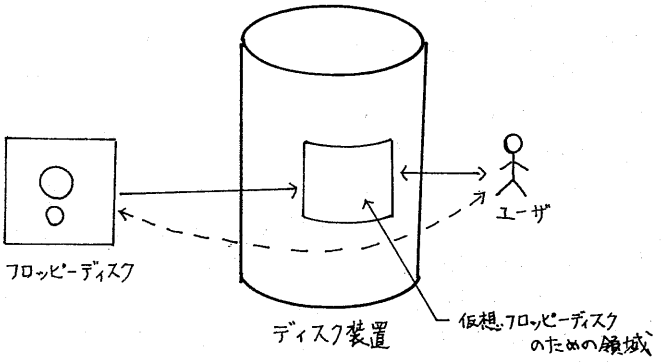

図7 仮想フロッピーディスク

仮想フロッピーディスクを用いるファイルとしては，次のようなものがある。
（1）大容量のテキストファイル（日本語文書のための文字辞書）
（2）完成されたソースプログラム
このように，大容量で，そうたびたび使わないが，使いはじめると頻繁にアクセスするファイルに用いる。

8．2仮想フロッピーディスクの実現方式

仮想フロッピーディスクを実現するにあたって，ディスク装置に格納した時に，ファイルシステムの構造にどのように組み込むかが，問題になった。ユーザの使い勝手を考慮するならば，ディスク装置上のファイルと同様なアクセスができる方が良い。そこで，次の二方法を考えた。

（1）ディレクトリ"USER"の下に，VFというディレクトリを設け，その下にFD上のファイルを登録する。（図8-(a)から図8-(b)に変換される。）
（2）仮想フロッピーディスクとして使用するFDの所有者に割り当てられているディレクトリの下に，FD上のファイルを登録する。（図9-(a)から図9-(b)に変換される。）

方法(1)では，ファイルをアクセスする場合に，そのファイルを共有の形で行なわなければならない。これは，FDとディスク装置との2レベルストレージの考え方をユーザに強要する。また，マルチボリューム・マルチファイルの場合に管理

が難しくなる。反対に，方法(2)では，アクセスしようとする場合には，ファイル名だけで良い。また，マルチファイルの場合には管理は難しいが，マルチボリュームの場合には，問題がない。しかし，FD上のファイルをディスク装置に格納した時に，同一のファイル名があった場合には，エラーになる。そのため，ユーザは，FD上のファイル名に十分注意しなければならない。この点については，次のように対処する。仮想フロッピーディスクとしてアクセスするファイルは，ディレクトリにだけ登録しておく。つまり，ディスク装置上には，対応するファイルのターミナルディレクトリしか存在しない。そして，FDには，ファイル実体とそのターミナルディレクトリを併せて格納する。これにより，方法(2)を採用した。

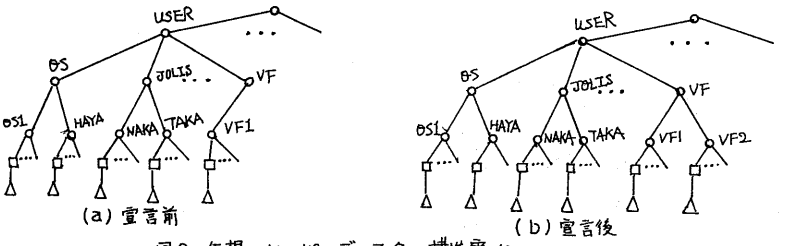

図8 仮想フロッピーディスクの構造案(1)

○… ディレクトリ　□… ターミナルディレクトリ　△… ファイル実体

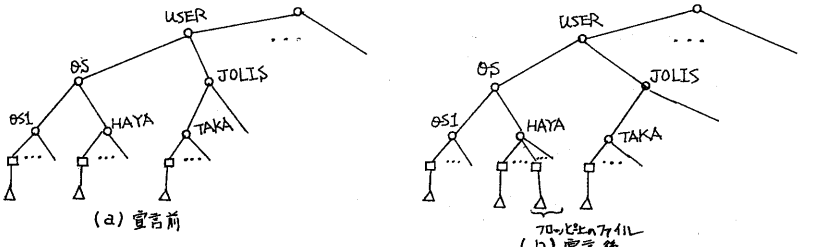

図9 仮想フロッピーディスクの構造案(2)

注) ディレクトリ OS/HAYA に割り当てられているユーザが宣言した場合

○… ディレクトリ　□… ターミナルディレクトリ　△… ファイル実体

なお，仮想フロッピーディスクの使用を宣言した時に，双方のターミナルディレクトリを比較して，同一ファイルであることが確認できる。

8．3 仮想フロッピーディスクの制限

仮想フロッピーディスクを使用するにあたっては，次の制限を設けた。

(1) 仮想フロッピーディスクファイルとするには，宣言しなければならない。
(2) ファイル名は，すべてのディスクファイルおよび仮想フロッピーディスクファイルを通じてユニークでなければならない。
(3) 仮想フロッピーディスクとして同時に使用できるFDは，4枚までとする。
(4) 仮想フロッピーディスクとして使用するFDは，あらかじめシステム管理者により，初期化されていなければならない。

9．エラー処理

9．1 エラーについて

ファイルシステムにおいて，最も重要なことは，いかに信頼性を保証するかである．そのため，ファイルシステムを開発するにあたって，次のような方針を立てた。

(1) データの抽象化を行なう。
(2) デバッグしやすくするために，読み易さに重点をおいたコーディングを行なう。
(3) 机上デバッグを十分に行なう。

これはインプリメントの問題としては，当然としても，それだけでは完全な信頼

性を得ることはできない。また，ディスク装置の故障も十分考えられる。そこで，ファイルシステムのエラー検出と回復が重要になる。

9．2 エラー検出

ファイルシステムを破壊するような致命的なエラーの例を次に示す。

(1) すでに書き込まれているセクタに情報を書き込んだ。
(2) 情報を書き込もうとしたセクタと違うセクタに書き込んだ。
(3) 読み出した情報が書き込んだ情報と違う。
(4) 新しく登録されたユーザに割当てたディレクトリが，指定された箇所と違うところに格納された。
(5) 記憶媒体へのアクセスが不可能になった。
(6) 不良セクタをアクセスした。

これらのエラーに対処するために，以下の点において信頼性設計を行なった。

(a) マップ表とディレクトリのポインタを比較する機能。
(b) ディレクトリに登録されているポインタと，実際に書き込んだセクタ番号を比較する機能。
(c) ディレクトリに登録されているポインタと，実際に読み出したセクタ番号を比較する機能。
(d) ディレクトリを修正する際に，そのディレクトリが格納されているセクタ番号と修正した部分を示す機能。
(e) 木構造を示す機能。
(f) ダイアノスティックプログラムを実行させる機能。

機能(a)は，ディレクトリが破壊されているかを検出するのに用いる。マップ表とは，各セクタごとに何が(ディレクトリ，ターミナルディレクトリ，ファイル実体)格納されているかを示す表である。そのため，各セクタに対して4ビット割当てる。この機能は，エラー(1)，(3)，(5)，(6)の検出に用いる。

機能(b)，(c)は，ルーチン間のインターフェースが正しいかどうかを検出するのに用いる。つまり，エラー(2)，(3)の検出に用いる。

機能(d)は，ディレクトリがいつ破壊されたかを知るのに用いる。

機能(e)は，自分に割当てられているディレクトリがどこにあるかを知るのに用いる。つまり，エラー(4)の検出に用いられる。

機能(f)は，ハード的にエラーのきざしさがないかどうかを確かめるのに，用いる。

9．3 エラー回復

エラーを検出した場合は，エラー回復が行なわれる。しかし，回復不可能な場合は，エラーを起こした情報を捨て，ディスク装置の再構成を行なう。

ディレクトリが破壊された場合は，マップ表と他の破壊されていないディレクトリを用いてエラーの回復が行なわれる。

ターミナルディレクトリ，ファイル実体が破壊された場合は，その情報は捨てられ，ディスク装置の再構成が行なわれる。そのため，各ユーザの情報は，FDにセーブしておくことが必要である。なお，システムソフトウェア等，重要な情報は，二重にしておく，あるいはシステムが一定の間隔でFDにセーブしておくことにより対処する。

10. おわりに

BS/0のファイルシステムの設計について報告した。これは，ファイルの共有を許し，保護を十分に行なうことにより，ディスク装置をマルチユーザ・マルチタスクの環境で使用できることを可能にした。また，1バイトコード，2バイトコードの混在する日本語ファイルをサポートすることにより，日本語があらゆるアプリケーションにおいて使用できる構成となった。これが，現在当学科で開発中のオンライン手書き文字認識システム，日本語文書浄書システムに果す役割は，非常に大きい。

以上のことより，日本型スーパーパーソナル・コンピュータ・システムに不可欠な機能を実現していく基盤が提供できるものと思う。

謝辞

システムのハードウエアを設計・開発して頂いた斎藤忠男教授，阿刀田史一助教授，鵜澤繁行助手，井上賢君，五十嵐智君に深く感謝する。また，設計の際御指導頂いた中森眞理雄助教授，中川正樹助手に深く感謝する。

さらに，研究の基礎をきずいて頂いた武部桂史君（現日立）に感謝する。

最後に，議論に参加していただいた，中森・高橋研究室の方々に感謝する。

参考文献

1) 中川正樹，他：オンライン手書き文字認識システムJOLIS-1の設計と試作，情報処理学会日本文入力方式研究会資料3-1，pp.1-12(1982)

2) 真鍋俊彦，他：マイクロプロセッサを用いたレーザビームプリンタのインテリジェント化ーハードウェアの方式設計についてー，夏のプログラミングシンポジウム(1982)

3) Robert Morris, Ken Thompson, "Password Security: A Case History", CACM, Vol.22, No.11, (Nov.1979)

4) DK801/811 ディスク装置機器仕様書，(株)日立製作所(1980)

5) 翼寿一，他：実戦的フロッピーディスク装置入門，電子科学ブルーブックス，産業出版