UNIXにおける追記型光ディスクシステムの扱い

6Y-1

鈴木達郎　深海悟

NTT　電気通信研究所

1. はじめに

最近光ディスク装置が使われ出しており、その大容量性と低価格さで、オフィスにあふれる紙文書に取って代わるものとして注目されている。

光ディスクは追記型(write once)のものと書き換え型(erasable)のものに分けられるが、現時点で媒体技術として実用になっているのは追記型である。

追記型と書き換え型を比較すると、当然書き換え型の方が使いやすいが、追記型は書き換えできないという性質を逆に利用して重要保存書類を扱うとか、価格、信頼性、寿命の点での長所を活すとかの用途も考えられ、少なくとも現時点ではシステムの構成要素としての利用価値は高い。

追記型光ディスクはwrite-onceというこれまでなじみのなかった性質を持つことが特徴である。この性質は従来の記憶媒体で強いてあげればカード、紙テープといったものしかなく、FD、磁気ディスクと大きく異なっている。UNIX*を始めとするほとんどのOSは追記型媒体に対応できるような構造を取っていない。即ち、例えばファイル管理プログラムのアルゴリズムは二次記憶媒体が書き換え可能であることを前提として作られている。

そこでwrite-once媒体をUNIXのような現在のOSでどういう位置付けにし、どう扱うかが問題となる。

2. 光ディスクのデータ構造

まず、光ディスクに蓄えるデータ構造をどうするかを考える。

(1) ディスク一枚を一つのファイルとみなす。(紙テープの世界)

(2) ディスク一枚の中に多数のファイルが存在し、構造管理される。(磁気ディスクの世界)

紙テープ的使い方も充分ありうる。しかし光ディスクは切り貼りもできないし、大容量を活すためにも、何らかの形でファイルシステム化した使い方が望ましい。従って、write-onceの媒体でどうやって効率よくファイルシステムを構築するかが重要な点となる。

---

*: UNIXはAT&Tのベル研究所が開発したOS

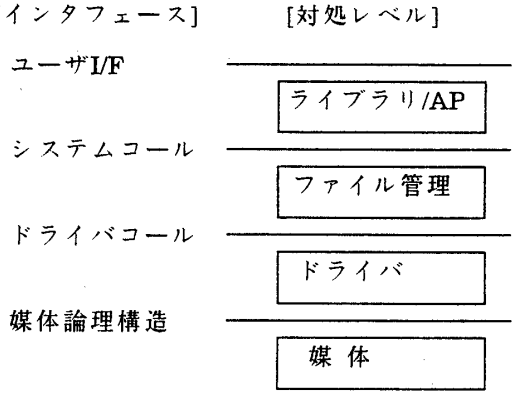

図1.　UNIXでの対処レベル

3. 追記型であることの対処

ここでUNIXの構造を階層モデルで考えて見ると図1.のようになる。現在UNIXでは追記型であることの対処を行っていないわけだから、各レベルのうちどこかで追記型であることを意識し対処しなければならない。次に各レベルでの対処法について検討する。

3.1 ライブラリ/APレベル

OS/カーネルの世界には手を加えず、ライブラリ/APのレベルで追記型媒体用のファイルシステムを構築することになる。UNIXの上に構築するとすれば、OSはrawデバイスとして提供し、APがデータレベルで構造を持たせることとなる。

この方式の欠点は、光ディスク上のファイルが、直接OSのファイルシステム上で走る既存のAPの対象とならないことである。専用のAPを作るか又は一度磁気ディスク上のファイルシステムに移してから使うことになる。特にUNIXのように、各種の流通APがある場合や、すでに磁気ディスクを対象としてOAサービスを構築しているシステムにとっては致命的である。文献[1]の例でもかなりの苦労をしたあげく、find, ls, cp などのコマンドはそのまま使えず、「容易に変更可能」と言うに留まっている。

3.2 カーネルレベル

カーネルレベルで追記型に対する対処を行った場合はシステムコールの互換性が保たれ、既存のAPがそのまま使えることとなる。

How to deal with write-once type optical disk system in UNIX

Tatsuo SUZUKI, Satoru FUKAMI

NTT　Electrical communications Laboratories

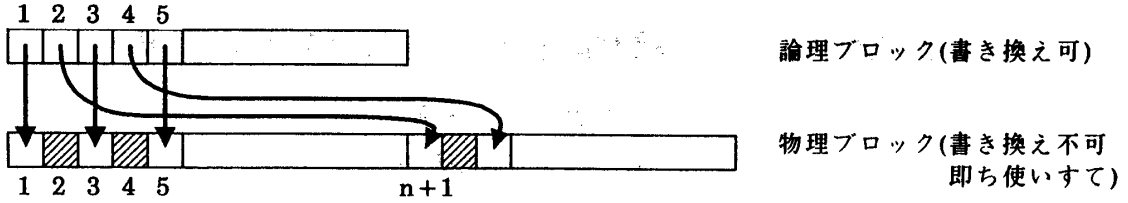

図2. 論理/物理ブロックアドレス変換

<ファイル管理レベル>
現在のファイル管理を作り替えて、追記型媒体上でもファイルシステムが構築できれば理想的であり、この方法がもっとも正攻法と言える。UNIX上の話ではないが、追記型媒体用のアクセスメソッドを導入した例もある[2]。一般にブロックを再利用せずに使いすて方式を基本にすれば可能である。

しかしUNIXの場合、従来との互換性を取るためにはinodeまわりにかなり手が入り、媒体の使用率もかなり悪くなってしまう。また、追記型かどうかで処理を変えないと、共存する磁気ディスク等の書き換え機能を殺すことになる。

<ドライバレベル>
ドライバの中で、追記型媒体をあたかも書き換え型媒体かのようにファイル管理に見せることができれば良いわけである。このための一つの方法として、論理/物理ブロックアドレス変換がある。図2.のように、上位には書き換え可能な論理ブロックを見せておき、その実体は追記型即ち使いすての物理ブロックにマッピングを取るという方式である。

この場合の問題はマッピングテーブルである。テーブル自身書き換えを伴うわけだが、この容量もかなり大きいものになる。例えば、媒体総量が1GBで1ブロック1KBと仮定すると、アドレスに4B取った場合テーブルは4MBになる。これをすべてメモリ上に置くのは大変だし、磁気ディスク等の別の書き換え可能媒体上に置くと、アクセスが余分に必要になる。

一般に上位のプログラムは書き換え可能だと思っているわけだから、どんどん書き換え要求が来て媒体の有効利用率はかなり減ることを覚悟しなければならない。またある日突然物理的にパンクする(たとえ論理的に余裕があっても)か、容量が可変の(どんどん減っていく)ボリュームになる。

3.3媒体レベル

光ディスク媒体上の論理構造が通常の磁気ディスクの場合と一致させることができれば、OSを何もいじらなくても(もちろんドライバは必要)readの場合従来のプログラムと完全に互換性がある。(UNIXの場合 read only のマウントが可能である。)

追記型媒体の上に書き換え型の媒体と同じ構造を作るのは簡単でボリュームコピーすればよい。ただし、この場合書き換えはもちろんできないし、作成の単位も論理ボリュームになり、ファイル単位の追加はできなくなる。

具体的には例えば図3.のように論理ボリューム単位のバッファ用磁気ディスクを用意し、充分たまった時点でまとめて光ディスクに移すことになる。この時、APからのインタフェースの保存や、ディスク上の配置を整理し、連続領域にするなどの配慮を行うのが望ましい。

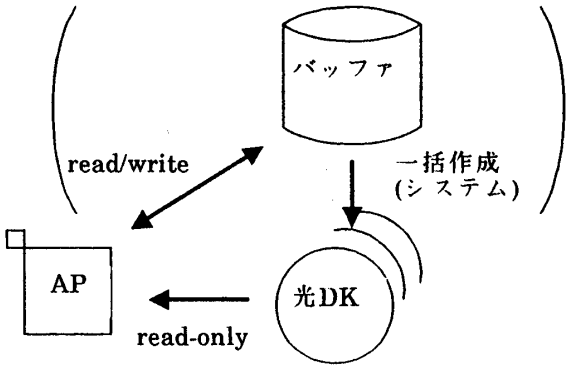

図3. 論理構造の一括作成

4. 考察

追記型光ディスク上にUNIXのファイルシステムを作る方法を検討してきたが、どの方法にもそれぞれに問題点があり、システムごとに使用目的や書き換え型の導入予測などを考慮して扱いを考えねばならない。

追記型光ディスク用のAPをこれから新規に作るのならAPレベルで対処するか新しいファイルシステムを作ってもよい。しかしUNIXのファイルシステムと互換性を保ちながら、見かけ上書き換え機能をサポートするのはかなり無理がある。

もし、作成した後はread-onlyでもかまわないのなら、論理ボリューム単位の一括作成が最も簡便な方法である。

いずれの場合でも、光ディスクは大容量なので、上位のユーティリティ(検索など)の充実、他のOAサービスとの連動などが不可欠なのは言うまでもない。

<参考文献>
[1] P. S. Kivolowitz, "Optical Storage Management under Unix Operating System", USENIX Summer Conference, Salt Lake City, pp.297-311 (1984).
[2] 土井 他:光ディスク記憶装置, 日立評論 vol. 66, no. 8 (1984).