オペレーティングシステム(OS)

柴山　潔

14. ファイルシステム (2)

- ファイル構造とファイルアクセス方式(2)
- ディレクトリ管理
- ファイル割り付け

ファイル構造 —代表例—（再掲）

(A) 逐次アクセスファイル

- ファイルシステムがあらかじめ規定する順序（生成順が代表的）でファイルブロックを並べる

(B) 直接アクセスファイル

- 番号付けしたファイルブロックの集まり

(C) インデックス付きファイル(indexed file)

- インデックス（index; 索引）によって直接アクセス（→直接アクセスファイルの発展形）

インデックス付きファイル (1)

- インデックス(index; 索引)によって直接アクセス
  - 直接アクセスファイルの発展形

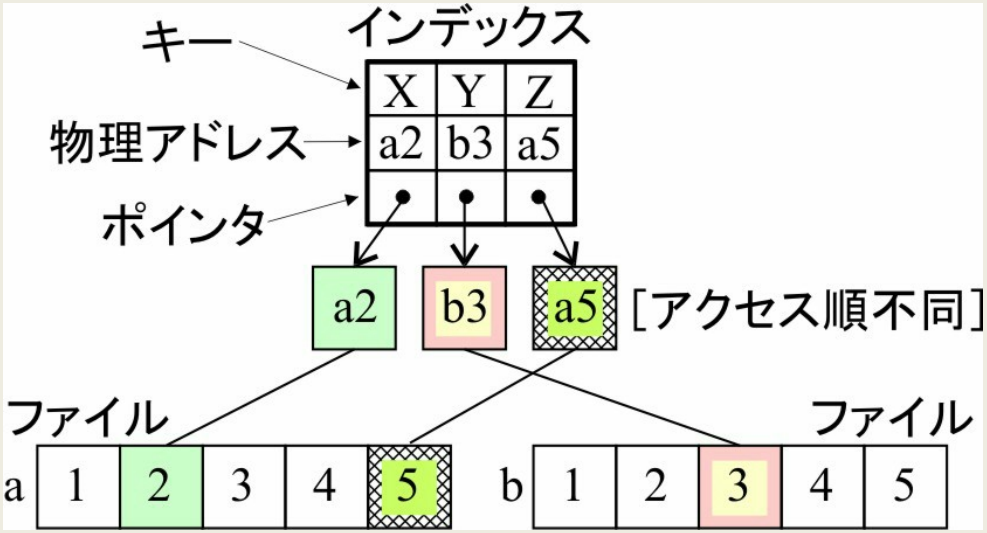
[定義3.12] **インデックス(index)**

- [一般的定義] 配列や表などで均質に並んでいる項目のうち，特定の項目を指す数字の“指標”，または，それを参照するための“キー(key)”(文字列が普通) ＝ 索引

- ブロックの「論理名」を示す“キー”とファイル装置上の“物理アドレス”の対

インデックス付きファイル (2)

- (1) キー（図では，X,Y,Z）によってインデックスを検索（＝連想）；(2) そのキーと対になっている（ファイル装置上の）物理アドレスによって，当該ブロックに直接アクセス；の順でアクセス
  - 直接アクセスファイル(B)と同様の理由で，直接アクセスファイル装置とのマッピングが容易&親和性が高



インデックス付きファイル (3) —特徴—

長所

- キー順（例：図での X→Y→Z）での（論理的）逐次アクセスも，キーによる直接アクセスも，どちらも可能

短所

- インデックス管理やインデックス検索のコスト（時間も空間も）が大

ディレクトリ (1)

[定義3.13] **ディレクトリ(directory)**

- ファイルシステムにおいて，ファイル装置に格納してある「すべてのファイル（の管理や操作）に関する情報」（＝ファイル情報）のうちからファイル管理とファイル操作に必須で重要な項目だけを抜粋&それらを体系化・一元化して構成するリスト（一覧表）

■ディレクトリの構成

(a) ファイル名（論理名）
(b) 当該ファイルを格納するファイル装置（実際には，ボリューム[定義3.15]）上の物理アドレス
(c) ファイル名(a)とファイルがある場所(b)との対応付け（＝ファイル割り付け）

- **OS**（ファイルシステム）自身が，ファイル管理&ファイル操作のために，ディレクトリを参照/更新
- **OS**（ファイルシステム）は，ディレクトリを保持して，その情報をユーザプログラム（ユーザも含む）に提示/提供

ディレクトリ (2)

- ディレクトリへの登録（実際）：(a)～(c)の各情報の確定，特に，ファイル名(a)とボリューム(b)との対応付け（＝ファイル割り付け(c)）
  - マウント(mount)：「ディレクトリ上でのファイル名とボリュームとの対応付け（マッピング）」操作機能のうちの，特に，「ボリュームのディレクトリへの登録」操作
- ディレクトリそのものもファイル装置上に

- ファイルアクセス（＝ファイル装置上のファイルへのアクセス）は，(1) ディレクトリ；(2) ファイルそのもの；への，2回の&この順での「ファイル装置へのアクセス」で構成
  - (1)によって，そのファイルがあるファイル装置上の物理アドレスを得て，その物理アドレスによって，(2)のファイルそのものへのアクセス

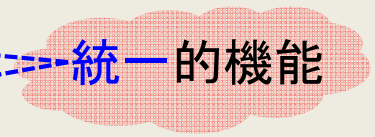
■ディレクトリの利用例

◆ファイル検索：ファイル名（論理名）によって，ファイル装置上でのファイルの格納場所（物理アドレス）を検索

◆ファイル保護：ファイル操作時に，当該ファイル操作に関する権限をチェック

ディレクトリ管理

- ディレクトリを利用する/ディレクトリによる管理
  - **OS**(ファイルシステム)によるファイル管理
  - プログラム(**OS**とユーザを含む)によるファイル操作



■ ディレクトリ管理の具体的な機能例

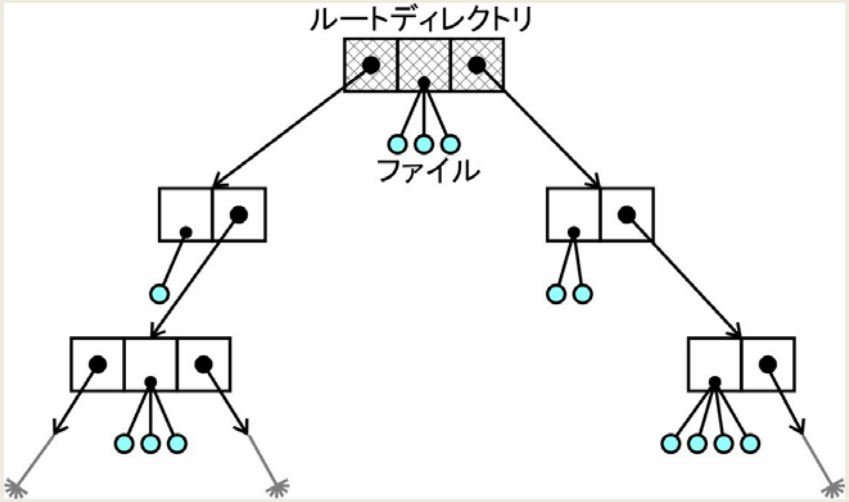
1. 検索:ファイル名(論理名)でディレクトリを検索,ファイルのファイル装置上でのアドレス(= 物理アドレス)を得る
   - ➢ プログラムによるファイルアクセス時には必須の操作
2. 事項の追加と削除:ファイルごとにディレクトリ(の内容)を更新
3. リスト:ファイル情報を一覧表示
   - ➢ ディレクトリをそのまま/並び替えて出力

ディレクトリの構造 ー代表例：木構造ディレクトリー

◆（例）UNIX, Windows

■ 木構造ディレクトリの構成法

1. ディレクトリ全体が1個の木構造
2. その木の唯一ルート（根）（＝ルートディレクトリ(root directory)）とすべてのノード（節）がディレクトリ
3. 各ディレクトリは「それが管理する部分木のルートデイレクトリ」
4. あるディレクトリ（＝その部分木のルート）に属するファイルは，その部分木の葉



木構造ディレクトリのパス名

［定義3.14］ **パス名**

- あるディレクトリ（始点，木構造でのルート/ノード）からアクセス対象（終点）であるファイル（木構造での葉）や別ディレクトリ（木構造でのノード）に至る論理的なパス（path; 経路）上にあるディレクトリ名を，あらかじめ規定してある規則にしたがって，順次連結して作る文字列

■ パス名の付け方

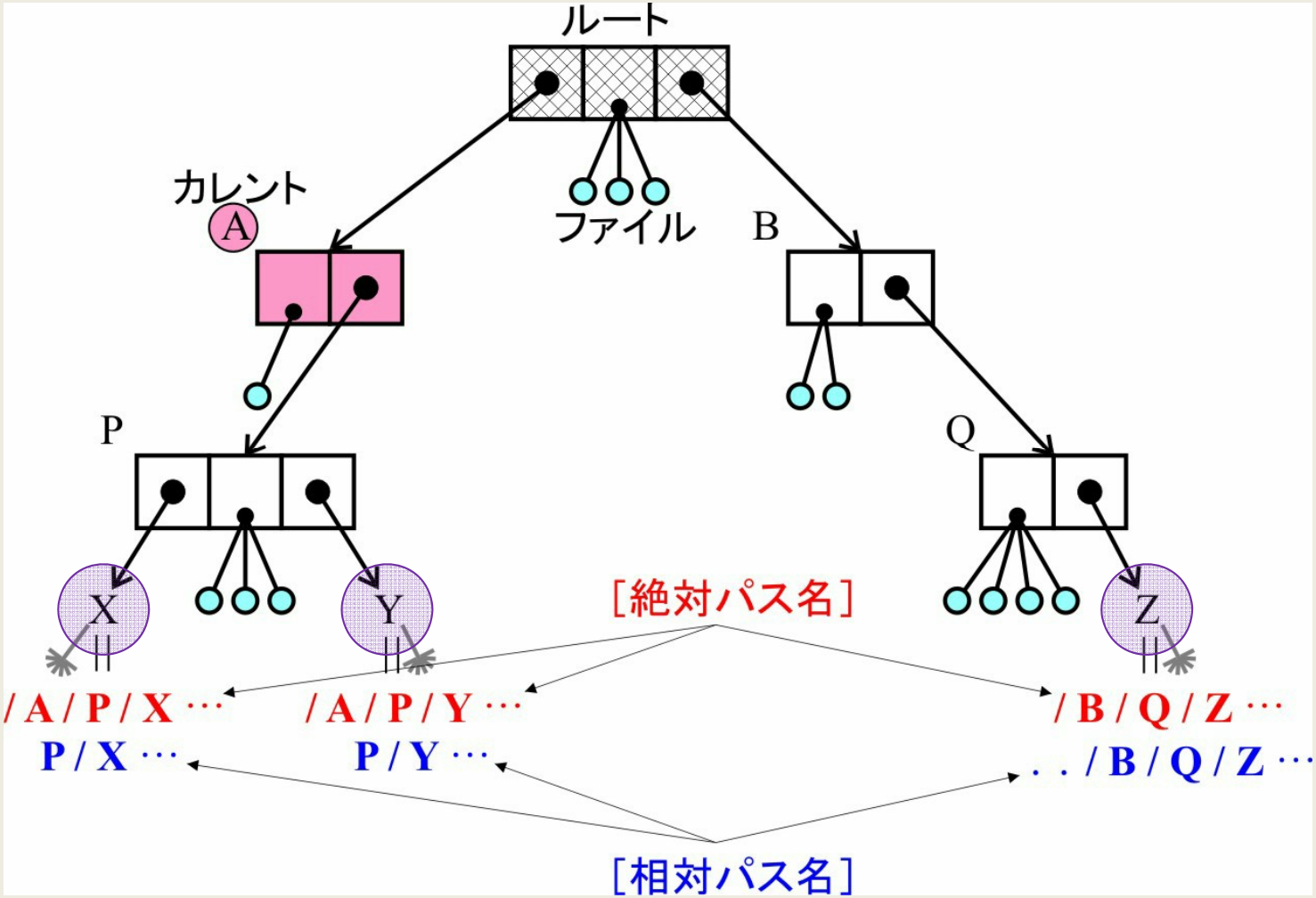
(a) **絶対パス名**: 木全体の唯一ルートであるルートディレクトリ（図でのパス名は冒頭の “/”）から対象ファイルに至るパス（＝絶対パス，完全パス）上のディレクトリ名を “/” で区切って順次連結した文字列

- どのファイルに対しても，ルートディレクトリ（唯一）からの絶対パス（名）は唯一

(b) **相対パス名**: カレント（current; 現時点）ディレクトリ（図でのパス名では “.”, “..” は「カレントディレクトリをポイント先とするポイント元のディレクトリ」）から対象ファイルに至るパス（＝相対パス）上のディレクトリ名を “/” で区切って順次連結した文字列

- あるファイルへの相対パス（名）は，カレントディレクトリによって決まり，相異なる列

木構造ディレクトリ（図例）



ボリューム (1)

［定義3.15］ **ボリューム(volume)**

- ファイルを格納するファイル装置において，独立にアドレス指定（＝独立したアドレス空間を構成）可能な単位メモリ領域

■ 代表的なボリューム例

(A) **逐次アクセスファイル装置**：磁気テープメディア（media; 媒体）の1巻（＝リール(reel)）など

(B) **直接アクセスファイル装置**：ハードディスクドライブ装置やフラッシュメモリの1区域（＝パーティション(partition)，論理ディスク），CD-ROMやDVDのメディアの1枚など

- ファイル装置におけるメモリ領域区分（区域）
- 独立して論理的なアドレス指定が可能な（＝独立して論理的なアドレス空間を構成可能な）物理的なファイル装置（の一部）

ボリューム (2)

- **OS**（ファイルシステム）だけではなく，ユーザやユーザプログラムからも，ボリュームは「仮想化されたファイル装置」に見える
- **OS**（ファイルシステム）が，「ファイル装置の管理」機能として，「論理的なボリュームと物理的なファイル装置との対応付け」＝「ファイル装置の仮想化による独立したメモリ領域としてのボリューム」を実現
- ボリュームは，“**OS**の原理”である「物理的なハードウェア機構（ここではファイル装置）の隠ぺい」の実現例

■ **OS**機能の観点でのボリューム

- (1) ファイル構造；(2) ファイルアクセス方式；(3) ディレクトリ；のそれぞれを実現するファイルシステムに対して，（物理的な）ファイル装置を，複数個の独立した（論理的な）ファイル装置に見せかけて，提供

ファイル割り付け

［定義3.16］ **ファイル割り付け**

- ファイル装置のどこにファイルを格納するか
  ＝ ファイルとファイル装置（実際には，ボリューム）との対応付け
  ＝ ファイル装置（実際には，ボリューム）上での領域管理

➢ ファイル割り付けは，ディレクトリによって管理・制御
← ディレクトリが「ファイル（名）とボリューム上での格納場所との対応付け表」（＝ファイル割り付け表）を保持

➢ 「ファイルとボリュームとの対応付け」（＝ファイル割り付け）は，ブロック単位&固定長領域割り付けで ← 大規模領域の管理を一定時間で高速に行うため

ブロックサイズ	大	小
	内部フラグメンテーションが発生	アクセス時間が長 ← ブロック単位転送回数が増

➢「ファイルシステムにおけるファイル割り付け機能の設計」では，↑の「ブロックサイズについてのトレードオフ」を適切に調停することが要点

ファイル割り付け方式 —領域管理方式による分類—

- 「（論理的な）ファイルを，物理的なファイル装置（実は，ボリューム）上の領域や場所（アドレス）にどのように割り付けるか」（＝**領域管理方式**）による分類

(1) **連続ファイル割り付け**：ファイルブロックを，順に，ボリュームの連続する（＝アドレス順の）領域に割り付け

- **連続領域割り付け**：論理的割り付け単位を物理的に連続する領域に割り付け

(2) **不連続ファイル割り付け**：ファイルブロックを，ばらばらに，ボリュームの不連続（＝任意アドレス）の領域に割り付け　＝ **ランダムファイル割り付け**(random file allocation)

- **不連続領域割り付け**：論理的割り付け単位を物理的に不連続な領域に割り付け

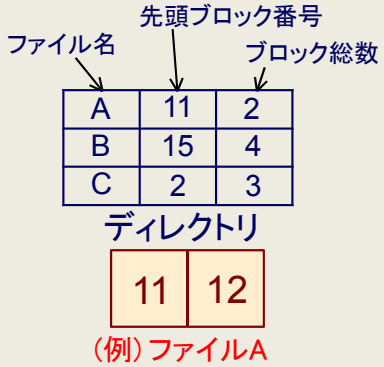
連続ファイル割り付け

◆ ディレクトリ: (1) ファイル名; (2) 先頭ブロック番号; (3) ブロック総数

■特徴

- ○ 高速アクセス可 ← 物理的に連続(＝物理アドレス順で)格納
- ○ ディレクトリの作成･操作が容易 ← ディレクトリに先頭ブロック番号とブロック総数を保持
- × ユーザプログラムやユーザがファイル生成時にファイルサイズを指定する必要
- × 外部フラグメンテーションが発生し易い → 空き領域の詰め直し(＝デフラグ)が必要

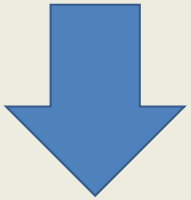
➢ 逐次アクセスファイルとの親和性が高，逐次アクセスファイル装置(代表例：磁気テープ装置)のボリュームに適用



不連続(ランダム)ファイル割り付け (1)

- 直接アクセスファイルとの親和性が高，直接アクセスファイル装置(代表例： ハードディスクドライブ装置)のボリュームに適用

- 不連続ファイル割り付けの短所である「余分な領域の必要性(空間性能)」や「リストアクセス(リストたどり)の遅さ(時間性能)」に対しては，現代のコンピュータシステムの高性能(空間性能も時間性能も)ハードウェア機構によって，それらの顕在化を防止可能
- 現代のコンピュータシステムの主要な(メイン(main))ファイル装置(代表例：ハードディスクドライブ装置)は不連続ファイル割り付けとの親和性が高い直接アクセスファイル装置



現代のOS(ファイルシステム)のほとんどが不連続ファイル割り付けを採用

不連続(ランダム)ファイル割り付け (2) —代表例—

(B-1) リンクファイル割り付け(linked file allocation):各物理ブロックに次の(リンクする)ブロック番号(ポインタ)を付加

- ファイル装置に割り付けるファイルは「ファイルブロックのリスト」

(B-2) インデックスファイル割り付け(indexed file allocation):すべての ポインタをインデックスとして, 特定ブロック(= インデックスブロック)で保持

- インデックスブロックの参照によって, ファイルへのポインタを得る

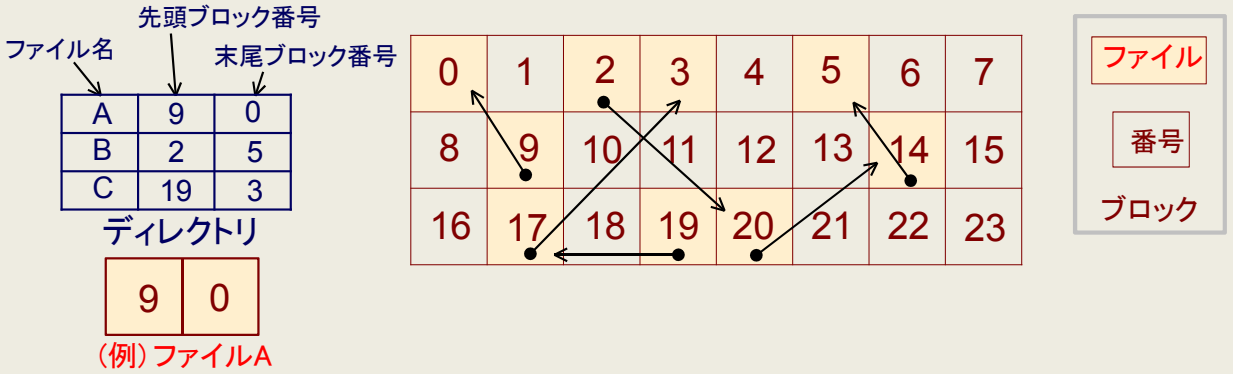
リンクファイル割り付け

◆ ディレクトリ: (1) ファイル名 ; (2) 先頭ブロック番号（ポインタ）; (3) 末尾ブロ ック番号（ポインタ）

■ 特徴

- ○ 外部フラグメンテーションは不発生
- ○ ユーザによるファイル生成時のファイルサイズ指定は不要
- × 各物理ブロックにポインタ領域が余分に必要，ポインタの破壊（＝リンク切れ）は致命的な障害（例：ファイルの喪失）
- × リスト途中の特定ブロックへのアクセスが遅 ← リストたどりが必須

◆（例）FAT (File Allocation Table)



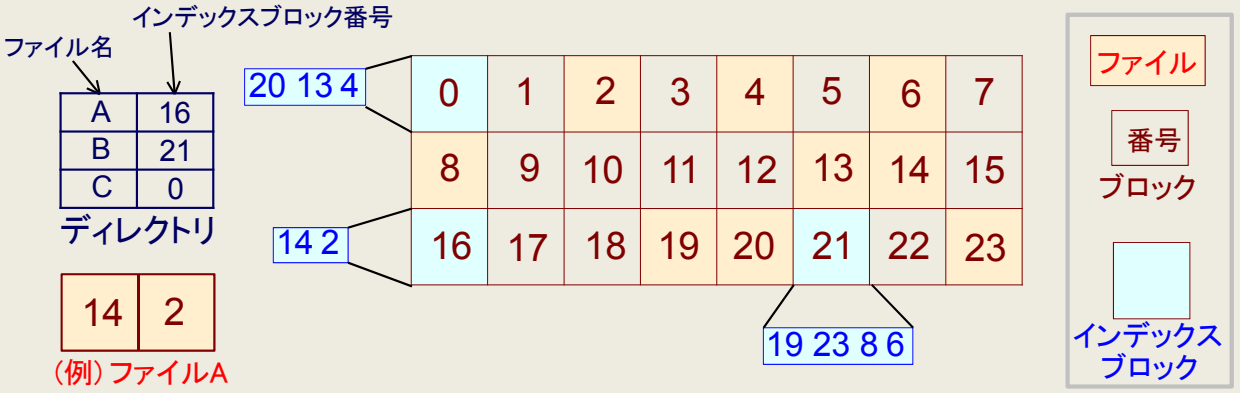
インデックスファイル割り付け

◆ ディレクトリ: (1) ファイル名 ; (2) インデックスブロック番号

■ 特徴

- ○ インデックスブロック(＝ボリュームの目次, VTOC (Volume Table Of Contents)によって特定ブロックへの直接アクセスが可 → リンク割り付け の短所を解消
- × インデックスブロック領域が余分に必要

◆（例）UFS (Unix File System)



ファイルシステムの実例 －不連続ファイル割り付け－

(1) FAT (File Allocation Table)：（例）初期のWindows

- リンクファイル割り付けでの「ファイル内容」と「次ブロックへのポインタ」とを分離
  - ➢「次ブロックへのポインタ」をリスト（一覧表，＝FAT (File Allocation Table)）で一括保持
    - ◆ FAT32：4G (=2^{32})ブロックを対象に管理可

(2) UFS (UNIX File System)：（例）UNIX

- インデックスファイル割り付けの代表例
  - ➢ ディレクトリにインデックスブロックそのものを取り込んだ高機能ディレクトリ（＝iノード）で管理

(3) NTFS (NT File System)：（例）比較的新しいWindows

- インデックスファイル割り付けに部分的に連続領域割り付けを取り入れ，高セキュリティ&高信頼性機能を付加
  - ➢ インデックスそのものも木構造ディレクトリ
    - ◆ 約180億G (≒16Exsa) (=2^{64})ブロックを対象

（上級）

OSによるファイル保護

- 物理的（ハードウェア）障害orソフトウェアによる不正操作/不正アク セスからファイルを保護

➢ ファイル保護 は，メモリ保護と，「情報への不正アクセスの防御」という基本的な点では同じ

↓ but,（実際上の観点）

■ 保護対象　メモリ保護　：メインメモリで保持しているプロセス

↓ ↑

ファイル保護　：ファイル装置に格納してあるファイル

➢ メインメモリが「プロセス（情報）を（OSが）一時的に保持するメモリ装置」であるのに対して，ファイル装置は「ファイル（情報）を（ユーザが，意図的に削除するまで）半永久に格納しておくメモリ装置」であることに起因

（例） メインメモリ&ファイル装置というメモリ階層を利用する仮想メモリ機構でも，ファイル装置はメインメモリのバックアップ装置として機能

➢ メモリ保護では「アクセス権の管理」だけで十分 but, ファイル保護では，アクセス権の管理の外にバックアップ機能も必須

（上級）

OSによるファイル保護機能（例）

(A) アクセス制御(access control)：ユーザ（広義，プログラムも含む）によるファイルアクセスごとに，ファイル情報（← ファイル保護に関する情報が主）をもとにして，種々のアクセス権限をチェック

- （例） アクセス権による保護では，(a) 共用ファイルについては，OSによる「書き込み」を許すが，ユーザには「読み出し」だけを許可；(b) 非共用ファイルについては，所有者（ユーザ）自身のアクセスは「可（自由）」，他ユーザのアクセスは「不可」；などのような制御

(B) バックアップ（backup; 退避）：主要な（メイン）ファイル装置（例：ハードディスクドライブ装置）上にあるファイルを定期的に別の補助ファイル装置（例：超大容量ハードディスクドライブ装置やDVDなど，主要なファイル装置よりも大容量のメモリ階層に位置するファイル装置）へ退避（コピー）

- (B)のバックアップの逆操作はリカバリ(recovery)（＝回復）といい，「バックアップしてあるファイルを補助ファイル装置から主要なファイル装置へ戻す（回復する）」機能
- リカバリ（回復）操作は「ファイルのファイル装置への再割り付け」機能も併備

（上級）

OSによるファイルアクセス制御（例）
—アクセス権による方式(1)—

(a) アクセス制御リスト：各ファイルごとに許可する操作（アクセス権）を単純なリスト（一覧表）にして保持

- ○ 仕組みが単純 → 実現が簡単
- × リスト（一覧表）は，疎&サイズは大 → 冗長

- アクセス権の対象操作例 —アクセス制御方式による設定—
  (1) 読み出し（リード(read)）；(2) 書き込み（ライト(write)）；(3) 更新（内容の変更や追加，アップデート(update)）；(4) 実行；(5) 生成/削除；など

（上級）

OSによるファイルアクセス制御（例）
—アクセス権による方式(2)—

(b) ユーザクラス(user class)：ユーザをクラス（＝グループ）に分け，各クラス単位で（ごとに）許可する操作をあらかじめ決めておく

- ○ コンパクトなサイズ&簡単な仕組み → 「複数ユーザによるアクセス権の共有」などの多彩なアクセス制御が可能
- × ユーザクラスの設定次第で仕組みが複雑 → オーバヘッドが顕在化

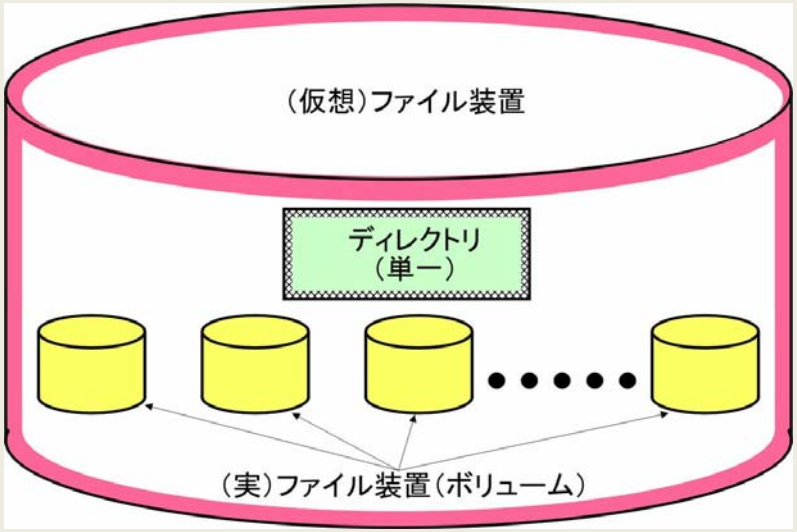
- ユーザクラスの例：(1) 所有者（owner; 作成者）；(2) グループ(group)；(3) 使用者（user; ユーザ）；(4) 共有；など
- 代表的なOSであるUNIX：所有者/使用者(User)，グループ(Group)，その他すべて(Others)という3ユーザクラスごとに，読み出し(Read)，書き込み(Write)，実行(eXecution)のアクセス権を独立に設定可能

（上級）

仮想ファイル装置とは？

●「異種複数台のファイル装置（群）を 単一のファイル装置 に見せかける」機能は“OSの原理”としての“仮想化”の一種
＝ファイル装置の仮想化

- 物理的なファイル装置（ハードウェア装置）の台数や装置ごとに相異なる仕様（容量や動作速度など）を仮想化によって隠ぺい → OS（ファイルシステム）自身を含む広義のユーザプログラムに，「論理的に統一した仕様（例：ファイル操作，ファイル構造，ファイルアクセス方式など）を備える単一の仮想的なファイル装置」として見せかける
- 「論理的に統一した仕様を仮想的に備えるファイル装置」＝ 仮想ファイル装置（＝仮想ストレージ）



（上級）

仮想ファイル装置とボリューム

- 現代のコンピュータシステムでは，（単一の）**OS**（ファイルシステム）によって，異種複数台の物理的なファイル装置（実際にはボリューム）で構成する（1台の）仮想ファイル装置を管理

- ボリューム（定義3.15）も「物理的なファイル装置の隠ぺい」という点では，仮想ファイル装置と同じファイル装置の仮想化
  - しかし，両者での「仮想化の対象」となる物理的単位は相違
    - ボリューム：ボリュームそのものである（1台の）物理的なファイル装置（上）のメモリ領域区分（区域）（例：ハードディスクドライブ装置上の1個のパーティション，DVDドライブ装置上の1枚のメディアなど）
    - 仮想ファイル装置：（1台の）物理的なファイル装置全体（例：1台のハードディスクドライブ装置，1台のDVDドライブ装置など）

（上級）

ファイル装置の管理 —仮想ファイル装置として—

- 実際のOS（ファイルシステム）では，ボリュームと仮想ファイル装置を組み合わせて，ファイルやファイル装置を管理
  ＝ 異種複数台の物理的なファイル装置（それぞれが複数のボリュームをもつ）によって（1台の）仮想ファイル装置を構成

➢ 実際のファイルシステムによるファイル管理機能（例：ファイル操作，ファイル構造，ファイルアクセス方式，ディレクトリ管理など）は，「仮想ファイル装置におけるボリューム管理」として対処し実現

■ OS（ファイルシステム）の観点

- 仮想ファイル装置は，「異種複数台の物理的なファイル装置を隠ぺい，それらに格納するファイルを単一のファイルシステム/ディレクトリによって管理」の効果を実現
  ➢「論理的に単一化した（＝1台の仮想ファイル装置の）単一ディレクトリ」を管理するファイルシステムは単一

（上級）

仮想ファイル装置の機能 (1)

■「単一ファイルシステム（＝単一ディレクトリ）によって管理する仮想ファイル装置」で実現できる具体的な機能

(A) バックアップ：物理的ファイル装置（＝ボリューム）の相互に完全なコピーを取っておく

- バックアップメモリとしてのファイル装置の冗長性を利用 → 仮想ファイル装置全体としての高い耐故障性&信頼性を実現
- 物理的ファイル装置（＝ボリューム）の台数や容量などの空間的性能を隠ぺい

(B) キャッシュ：物理的ファイル装置（＝ボリューム）の一部を，より高速なメモリ階層（例：RAM，フラッシュメモリなど）にコピー（＝キャッシュ）しておく

- 参照局所性の高いプログラム（広義で，データも含む）をキャッシュすることによって，仮想ファイル装置全体としての高速アクセスを実現
- 物理的なファイル装置のそれぞれの時間的性能を隠ぺい

（上級）

仮想ファイル装置の機能 (2) —仮想メモリとの比較—

- 仮想ファイル装置は，「OSによるメモリ階層の仮想化」という点で，仮想メモリと類似
  - 一方で，仮想メモリとアーキテクチャ上の相違

■ 仮想メモリ：メインメモリ（階層）の空間的性能を，隣接する別のメモリ階層であるファイル装置（階層）によって改善

■ 仮想ファイル装置：ファイル装置（階層）の空間的&時間的性能を，大まかには同じメモリ階層に位置付けられるファイル装置（群）によって改善

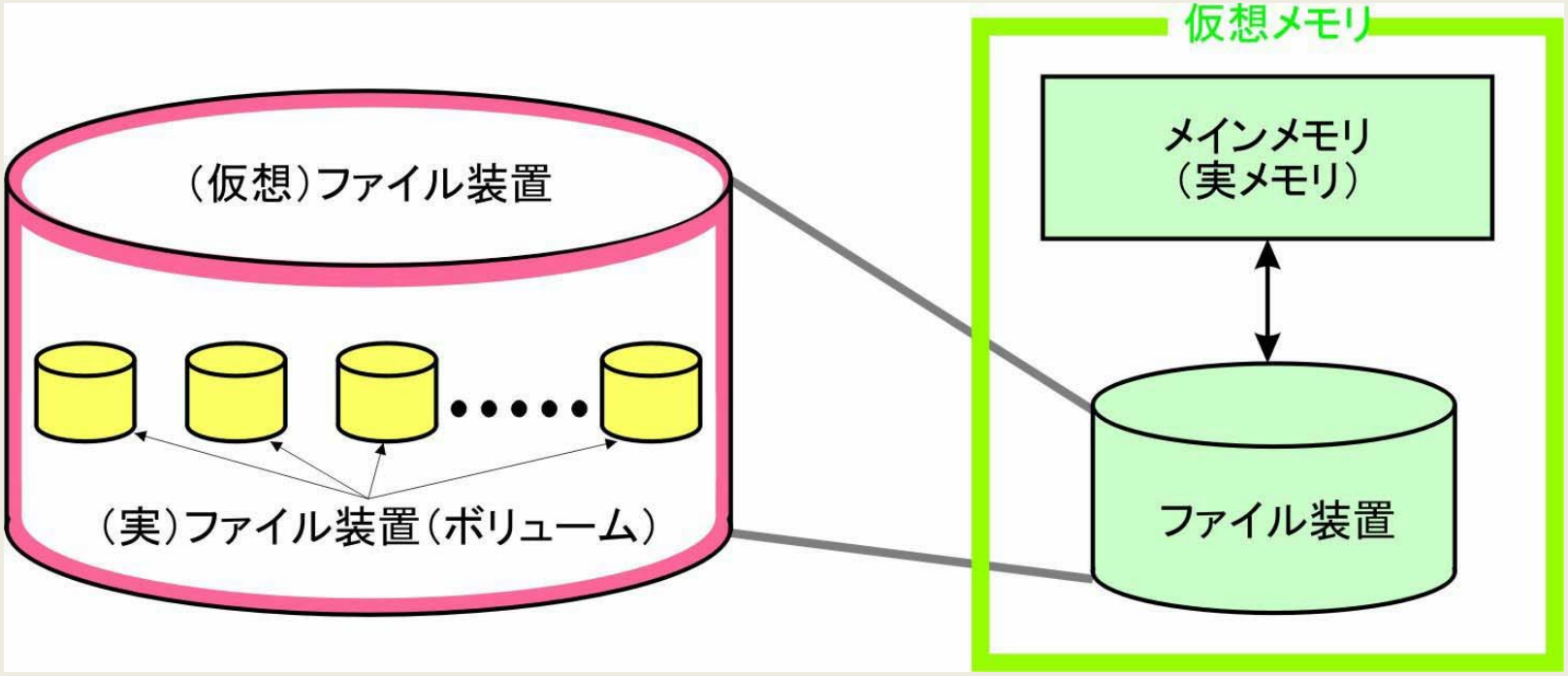
■ OS機能の観点

- 仮想メモリが主としてメモリ管理，仮想ファイル装置が主としてファイル管理，それぞれの機能の実現/適用にあたる

- 現代のコンピュータシステムでは，仮想メモリのバックアップメモリであるファイル装置を仮想ファイル装置として実現

（上級）

仮想ファイル装置の機能 (3)
—仮想メモリとの比較（図説）—



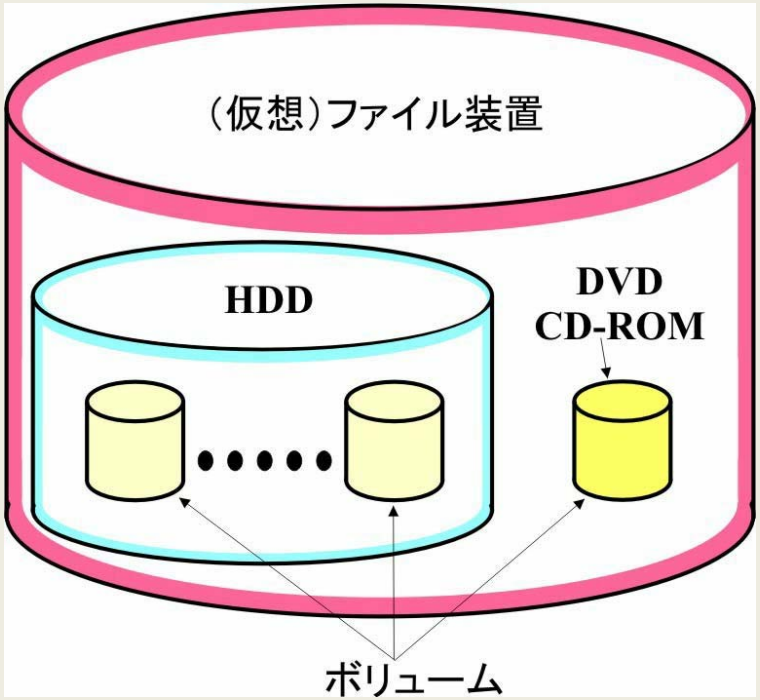
（上級）

ファイル装置とボリュームとの対応付け

- 仮想ファイル装置の実現では，「物理的なファイル装置そのものと論理的なファイル装置（実際には，ボリューム）との対応付け」が必要

- （代表例）ファイル装置とボリュームとの対応付け
  - 代表的なハードディスクドライブ装置においては，ボリュームは，そのハードディスクドライブ装置上に確保するパーティション/論理ディスクと呼ぶメモリ領域区分（区域）
  - DVDドライブ装置やCD-ROMドライブ装置においては，ボリュームは，ある時刻では，その時にドライブ装置にセットしてあるメディア

- （普通）これらのドライブ装置では，メディアは可換，**OS**が動作中の間はドライブ装置そのものをボリュームに



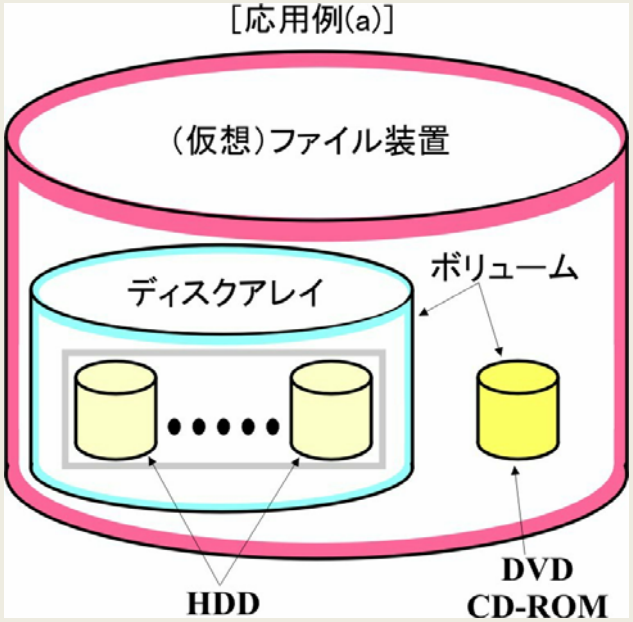
（上級）

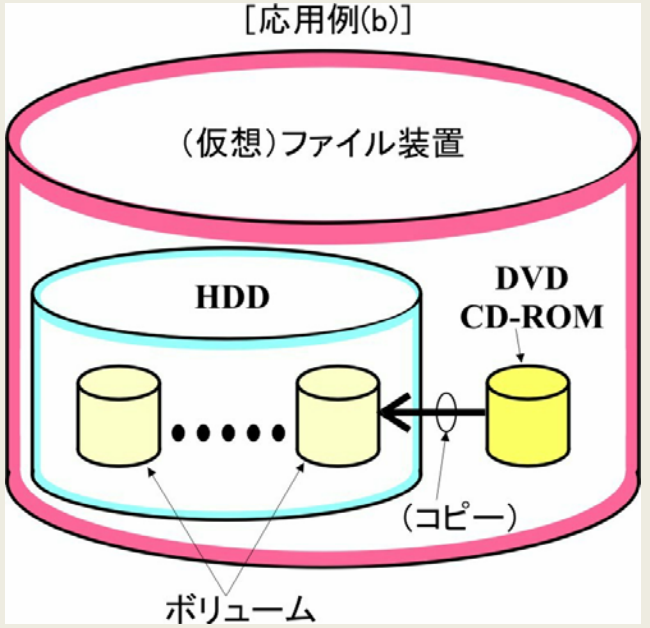
ファイル装置とボリュームとの対応付けの応用例

(a) ディスクアレイ：複数台のハードディスクドライブ装置で1個のボリュームを構成

(b) 仮想ドライブ装置：DVDドライブ装置やCD-ROMドライブ装置（上）のメディアをハードディスクドライブ装置（のパーティション）に丸ごとコピー&ボリュームに

- （仮想）ファイル装置におけるバックアップとキャッシュの両機能の具体的な適用/応用





（上級）

【まとめ】ファイル装置の管理 (1)

- 仮想ファイル装置の実現においては，次の2種類の「**OS機能による支援**」が必須
  - ハードウェア装置の仮想化：異種複数台のファイル装置の隠ぺい & それらの管理方式の統一
  - **OS**（ファイルシステム）の単一化：異種複数台のファイル装置に対する，ディレクトリの単一化や併合（＝単一ディレクトリ管理）

  - 特に，「**OS**（ファイルシステム）の単一化」においては，形式的な（＝幾何学的形状や構造上での）ディレクトリの単一化や併合だけではなく，「そのディレクトリを用いて行うディレクトリ管理機能の単一化」という「意味的な（＝論理的な）単一化」が必要

- 単一ディレクトリ管理によって「ファイル単位の仮想化」&「ファイル単位での管理」が実現可

（上級）

【まとめ】ファイル装置の管理 (2)

- 現代の代表的なファイルシステムのほとんどが，仮想ファイル装置を含めて木構造ディレクトリを採用

↓（理由）

- 木構造は「部分木の集まり」であるので，ある部分木の任意のノード（ルートや葉も含む）を別の部分木のルートとすることによって，簡単に部分木どうしを併合可能
- 「複数の木構造ディレクトリを併合して，単一の木構造ディレクトリを構成する」ディレクトリの管理は比較的簡単