ファイルシステムの実装

- ファイルシステムは記憶装置がどのように利用されるかを記したメタデータ
  - 使用されるファイルシステムの種類や設定
  - ディレクトリの位置、ファイル情報の位置など
- をどこかに記録しておいて、その内容に沿って管理
- 記憶装置のレイアウト
- 記憶装置の使い方には慣例的なルールがある
- ブロックサイズは512バイト
- 先頭のブロックはコンピュータが起動した時、必要な情報が置かれている
  →マスターブートレコード(MBR)
  - 初期起動プログラム
    （イニシャルプログラムローダ:IPL）
  - パーティションテーブル（４領域まで）
- パーティションテーブル
- 記憶装置は基本的にパーティションという領域に分割してから使う→同じ機器でもシステム上は複数のボリューム
- 区切りを示す情報→パーティションテーブル
- 区分けされた各パーティション毎に個別のファイルシステムを指定
- パーティションの内部
- ファイルシステムが必要とする情報を配置
  →フォーマット
  - システムの起動情報：ブートブロック
  - ファイルシステムの基本情報：スーパーブロック
  - ブロックの空き情報
  - 個々のファイル情報：i-node
  - ファイルのリスト：ルートディレクトリ
  - ファイルの実データ

ファイルの格納

- ファイルの実データは記録できる空き領域を選んで格納する
- どこの領域を使い、どのような順番に並んでいるかをファイルシステムがわかる必要がある
- よく知られる管理方法
- 連続ブロック割り当て
- 空き領域の先頭からファイルを詰めていく方法
- ファイルの先頭位置と長さがわかれば良い
- ファイルの消去が行われると不定な長さの空き領域ができる→断片化が発生
- 消去されない状況であれば有効な点は多い
  →CD-ROM, DVD-ROMへのファイル記録
- 連結リスト
- 利用しているブロック＋次のブロックのアドレスによる連結リストをメタデータとして使用する
- 先頭の情報から順に辿っていけば、ファイルのデータが順番に参照できる
- リストをまとめて入手するには、特定の場所に集めてあるほうが便利
  →ファイルアロケーションテーブル
- FAT16, FAT32 はこのタイプ
- インデックスブロック（i-node）
- 使用したブロックを順に列挙した参照用のブロック（間接ブロック）を仲介する
- Unix系のファイルシステム→i-node
- i-nodeはファイルの情報と使用しているブロックの情報から成る
  - 先頭の幾つかのブロックを直接指定する部分
  - 参照用ブロックを指定する部分
- 間接ブロックは１段分、２段分、３段分がある

ディレクトリの実装

- ルートディレクトリは初期化した段階で作成
- 以降のディレクトリはファイル同様、記録するブロックを確保して親ディレクトリに登録
- 各ファイルの記録（ディレクトリエントリ）には、
  - ファイルの情報をまとめて記録するもの
  - ファイルのメタデータへのリンクを記録するもの
- ２通りある→ファイルシステムによって異なる
- FATは前者、Unix系は後者
- リンク
- 同じファイルを異なるディレクトリから参照する方法
- ハードリンク：ファイル本体をそれぞれのディレクトリに登録
- シンボリックリンク：一方はファイルの本体につながるパスを保持したデータ
- どちらもファイルの属性がディレクトリエントリと独立していないと実現が難しい
- 空き領域の管理
- ファイルを新しく作成する際には空いているブロックを知る必要がある
- 連結リストで管理：ファイルの記録状態と共用できる
- ビットマップで管理：1bit=1ブロックなので管理データが小さくて済む
- その他の機能
- ジャーナリング：ファイル操作の手順をログとして持つことで、アクシデントからの復旧を可能に
- クォータ：記録領域の制限を設けることが可能になり、マルチユーザシステムでの不平等を軽減
- 複雑なアクセス権：利用する実態に合わせた権限を設定可能に
- 暗号化：記録内容を暗号化することで安全性を高める