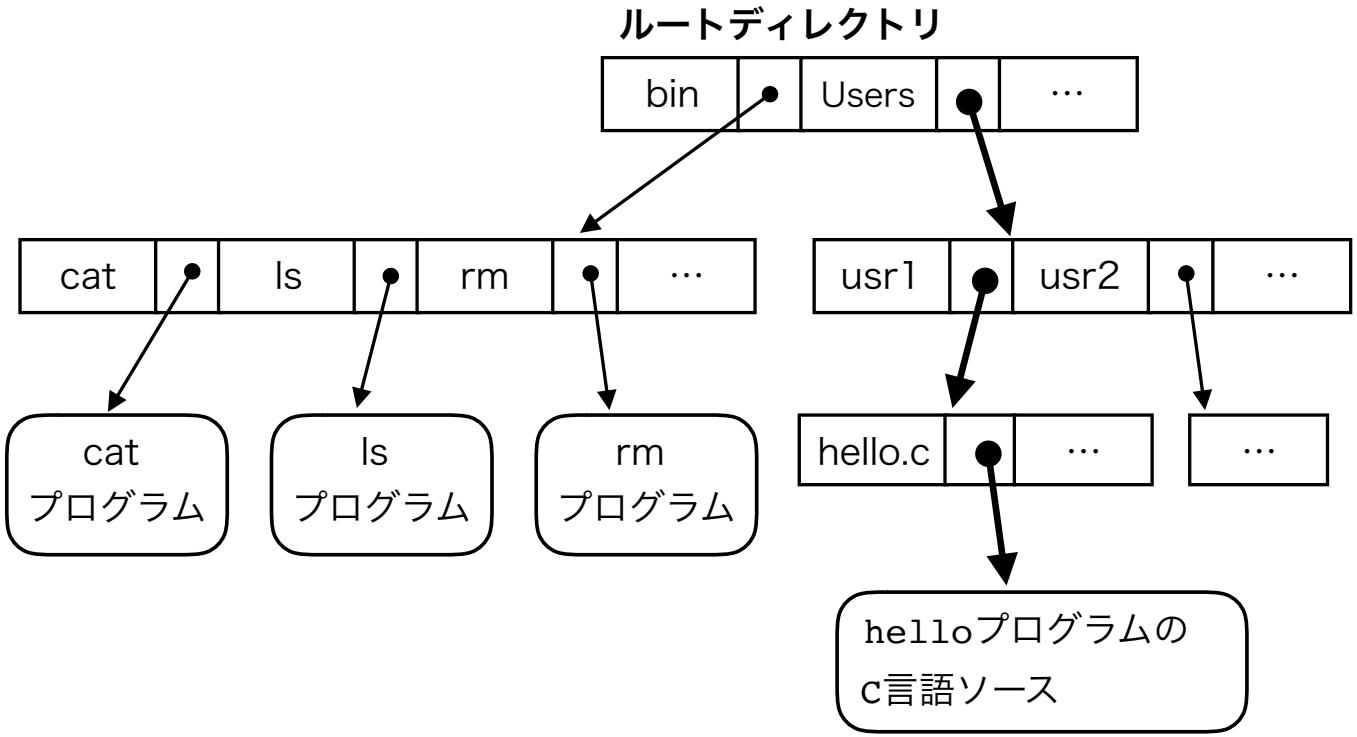
オペレーティングシステム
第１４章 ファイルシステムの概念

https://github.com/tctsigemura/OSTextBook

ファイルシステム

- ファイルシステムは二次記憶装置を
  - 管理する．（どのセクタが，どのファイルの一部？）
  - 抽象化する．（ハードディスク → ファイル）
  - 仮想化する．（1 台のハードディスク → 多数のファイル）
- ファイルは一次元のバイト列（バイトストリーム）
  オペレーティングシステムはファイルの構造を決めない．
- ファイルは名前を持つ．
- 名前とバイト位置でデータが決まる．
  名前＝ファイル名，バイト位置＝ファイル内オフセット

ファイルの名前付け



- ファイルは木構造のディレクトリシステムに格納する.
- ディレクトリは名前とファイル本体のポインタを格納する.
- 階層構造を持った名前 (**パス**) でファイルを特定する.
- **絶対パス**はルートディレクトリを起点にする.
- **相対パス**はワーキングディレクトリを起点にする.

ファイルの別名（1）

別名があると便利な例（最新のファイルはいつも同じ名前）

ある日

2017_06_30.log		2017 年 6 月 30 日のファイル
2017_07_01.log		2017 年 7 月 1 日のファイル
2017_07_02.log		2017 年 7 月 2 日のファイル
today.log	→	2017_07_02.log

次の日

2017_07_01.log		2017 年 7 月 1 日のファイル
2017_07_02.log		2017 年 7 月 2 日のファイル
2017_07_03.log		2017 年 7 月 3 日のファイル
today.log	→	2017_07_03.log

ファイルの別名（2）

- **ハードリンク**
  - ファイルシステム仕組みとして OS カーネルに組み込む.
  - ファイルの本体が複数のディレクトリ・エントリから指される.
  - リンクカウントを用いる.
  - ディレクトリをリンクするとループ検出が厄介 → 禁止！
- **シンボリックリンク**
  - ファイルシステム仕組みとして OS カーネルに組み込む.
  - 他ファイルのパスを格納した特別なファイル.
  - リンク切れ状態が許される.（Web ページのリンクに似ている）
- **ファイルシステムの外で実装されるリンク**
  - Windows のショートカット，macOS のエイリアスなど
  - ファイルシステム本体が持つリンク機構は一定ではない.
    → 現代の OS は同時に複数のファイルシステムを使用する.
    → アプリに近い側でどのファイルシステムでも共通の仕組みを提供

ファイルの別名（3）

*HFS+*ファイルシステム上の *macOS* のエイリアスの例

```
1 $ ls -l@ a.txt*
2 -rw-r--r--  1 sigemura  admin     5 Jun 27 10:19 a.txt
3 -rw-r--r--@ 1 sigemura  admin  1012 Jun 27 10:19 a.txt のエイリアス
4         com.apple.FinderInfo         32
```

3 行 拡張属性付きの通常ファイルとしてエイリアスが存在

4 行 拡張属性の名前は `com.apple.FinderInfo`

4 行 拡張属性のサイズは 32 バイト

ファイルシステムのより汎用的な機構である拡張属性を利用して，
エイリアスを実装している．

ファイルの別名（4）

FAT ファイルシステム上の *macOS* のエイリアスの例

```
1 $ ls -la@ ._* a.txt*
2 -rwxrwxrwx  1 sigemura  staff  4096 Jun 27 09:55 ._a.txt のエイリアス
3 -rwxrwxrwx  1 sigemura  staff     5 Jun 27 09:55 a.txt
4 -rwxrwxrwx@ 1 sigemura  staff  1040 Jun 27 09:55 a.txt のエイリアス
5         com.apple.FinderInfo         32
6 $ rm ._a.txt のエイリアス
7 $ ls -la@ a.txt*
8 -rwxrwxrwx  1 sigemura  staff     5 Jun 27 09:55 a.txt
9 -rwxrwxrwx  1 sigemura  staff  1040 Jun 27 09:55 a.txt のエイリアス
```

4,5 行 拡張属性付きの通常ファイルとしてエイリアスが存在

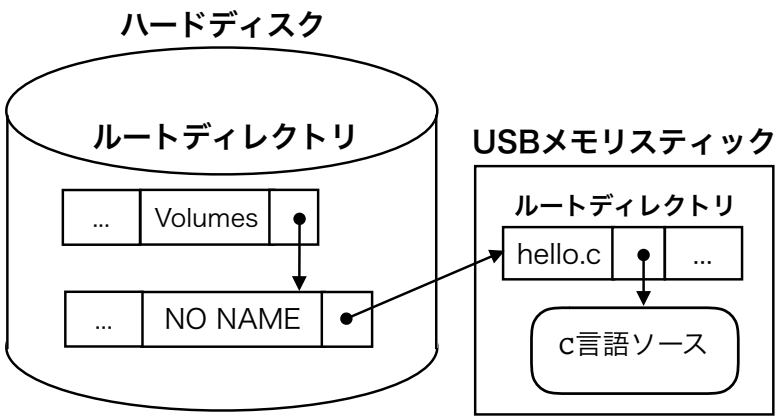
2 行 隠しファイルができている！！

6 行 隠しファイルを消してみる.

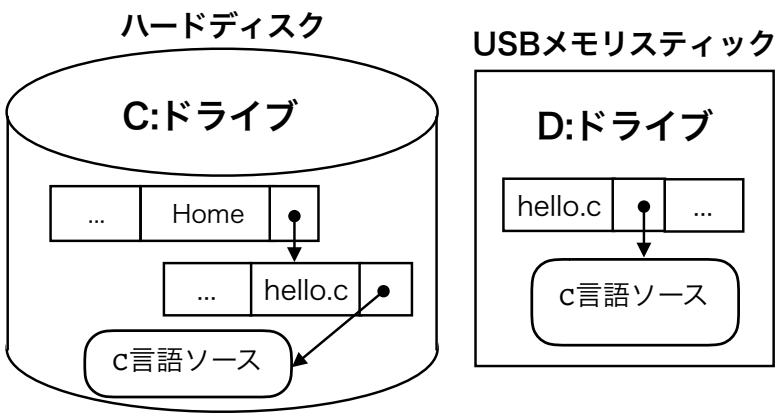
9 行 拡張属性が消えてしまった！！

FAT ファイルシステムの規約の範囲で**エイリアス**を実装している.

ボリュームのマウント



(a) マウント方式



(b) ドライブレター方式

- 二つ目以降のボリュームの接続方法
- マウント方式
  - ボリュームを既存のディレクトリに接続する.
  - `/Volumes/NO NAME/hello.c` が USB メモリの C プログラム
- ドライブレター方式
  - ボリュームを区別するドライブレターを用いる.
  - `D:\hello.c` が USB メモリの C プログラム

ファイルの属性（1）

- **名前**：ファイル名をファイルの属性と考える場合もある.
- **識別子**：ファイル本体の番号など.
- **型（タイプ）**：通常ファイル，ディレクトリ，リンクなど.
- **保護**：`rwxrwxrwx` など.（後で詳しく）
- **日時**：作成日時，最終変更日時など.
- **所有者**：所有者，グループなど.
- **位置**：ディスク上のどこにファイル本体があるか.
  (データ位置のブロック番号など)
- **サイズ**：ファイルのバイト数.
- **拡張属性**：名前付きの小さな追加データ.
  ファイルシステムで用途を定めていない.

ファイルの属性（2）

```
1 $ ls -l@ b.txt*
2 -rw-r--r--  2 sigemura  staff      123 Jun 25 19:38 b.txt
3 -rw-r--r--@ 1 sigemura  staff      836 Jun 25 19:39 b.txt のエイリアス
4         com.apple.FinderInfo            32
5 $ xattr -l b.txt のエイリアス
6 com.apple.FinderInfo:
7 00000000  61 6C 69 73 4D 41 43 53 80 00 00 00 00 00 00 00  |alisMACS........|
8 00000010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  |................|
```

1 行 拡張属性付きでファイルの一覧を表示させる．

4 行 拡張属性を持つファイルがあることが分かる．

5 行 拡張属性の内容を表示してみる．

この例の拡張属性は，以下のようなものであった．

- 属性の名前：com.apple.FinderInfo
- 属性の大きさ：32 バイト
- 目的：ファイルがエイリアスであることを表す．（恐らく）

アクセス制御（1）

ファイルの**保護属性**に基づき，ファイルに誰が何をできるか制御する．

- **ビット表現の保護モード**
  - UNIX で使用される `rwxrwxrwx` のような情報．
  - UNIX の場合，「所有者，グループ，その他」のユーザについて

    r ：読める（Read），
    w ：書ける（Write），
    x ：実行できる（eXecute）

    を指定する．

アクセス制御（2）

- *ACL*（*Access Control List*）
  ファイル毎に，ユーザやグループを指定して細かな制御が可能

```
1 $ ls -le a.txt
2 -rw-r--r--  1 sigemura  staff  4 Jul  5 21:55 a.txt
3 $ chmod +a "group:admin allow write" a.txt
4 $ chmod +a "group:admin deny delete" a.txt
5 $ ls -le a.txt
6 -rw-r--r--+ 1 sigemura  staff  4 Jul  5 21:55 a.txt
7  0: group:admin deny delete
8  1: group:admin allow write
```

1 行 a.txt に ACL が無いことを確認した．
3,4 行 chmod コマンドで a.txt に ACL 追加した．
7,8 行 二行の ACL が確認できる．

- リストの先頭から順に評価する．
- 許可・不許可が決まったら評価を完了する．
- ACL で決まらない場合は rwx を使用する．

ファイルの種類

- ファイルシステム（OS カーネル）で決まっている種類
  (通常ファイル・ディレクトリ・リンクなど)
- アプリケーションなどが決めている種類
  (通常ファイルの拡張子で区別する)

拡張子	意味
.c, .java, .s 等	ソース・プログラム（C 言語，Java 言語，アセンブリ言語）
.py, .pl, .php 等	スクリプト言語のプログラム（python，perl，PHP）
.txt, .html, .xml 等	プレーンテキスト，マークアップ言語
.jpg, .png, .bmp 等	画像データ
.mp3, .m4a, .wma 等	音声データ
.mpg, .mp4, .wmv 等	動画データ
.pdf, .ps, .eps 等	印刷・表示用の文書ファイル
.zip, .tar, .tbz 等	アーカイブファイル
.exe, .app, 拡張子無し	実行形式プログラム（Windows，macOS，UNIX）
.doc, .docx	MS Word 文書

.app だけはディレクトリの拡張子

ファイルシステムの操作（1）

ディレクトリ操作

機能	対応する UNIX の API
ファイルの作成	`creat`, `open(... O_CREAT ...)` システムコール
ディレクトリの作成	`mkdir` システムコール
ファイルの削除	`unlink` システムコール
ディレクトリの削除	`rmdir` システムコール
リンクの作成	`link`, `symlink` システムコール
リンクの削除	`unlink` システムコール
名前の変更（移動）	`rename` システムコール
ディレクトリエントリの読出し	`opendir`, `readdir`, `closedir` 関数

- ファイルの作成は creat システムコールでもできる.
- ディレクトリの読み出しはライブラリ関数で行う.
- rename システムコールはファイルの移動もできる.

ファイルシステムの操作（2）

ファイル操作

機能	対応する UNIX の API
ファイルを開く	`open` システムコール
データを読む	`read` システムコール
データを書く	`write` システムコール
読み書き位置を移動	`lseek` システムコール
ファイルを閉じる	`close` システムコール
ファイルの切り詰め	`truncate`, `open(... O_TRUNC)` システムコール
ファイルのプログラムを実行	`execve` システムコール
ファイルの属性変更	`chmod`, `chown`, `chgrp`, `utimes` システムコール
ファイル属性の読出し	`stat` システムコール

- open はファイルの保護属性をチェックする.
- 切り詰めは専用の truncate システムコールも使える.
- ファイルの属性の読み書きができるべき.

ファイルシステムの操作（3）

ファイルの共有とロック

```
#include <sys/file.h>
#define   LOCK_SH   1    // 共有ロック
#define   LOCK_EX   2    // 排他ロック
#define   LOCK_NB   4    // ブロックしない
#define   LOCK_UN   8    // ロック解除
int flock(int fd, int operation);
```

- LOCK_SH：**共有ロック**（*shared lock*）
- LOCK_EX：**排他ロック**（*exclusive lock*）
- LOCK_NB：ロックできない時，ブロックしないでエラー
- open システムコールにもロックの機能がある．

ワーキングディレクトリの変更

```
#include <unistd.h>
int chdir(const char *path);
```

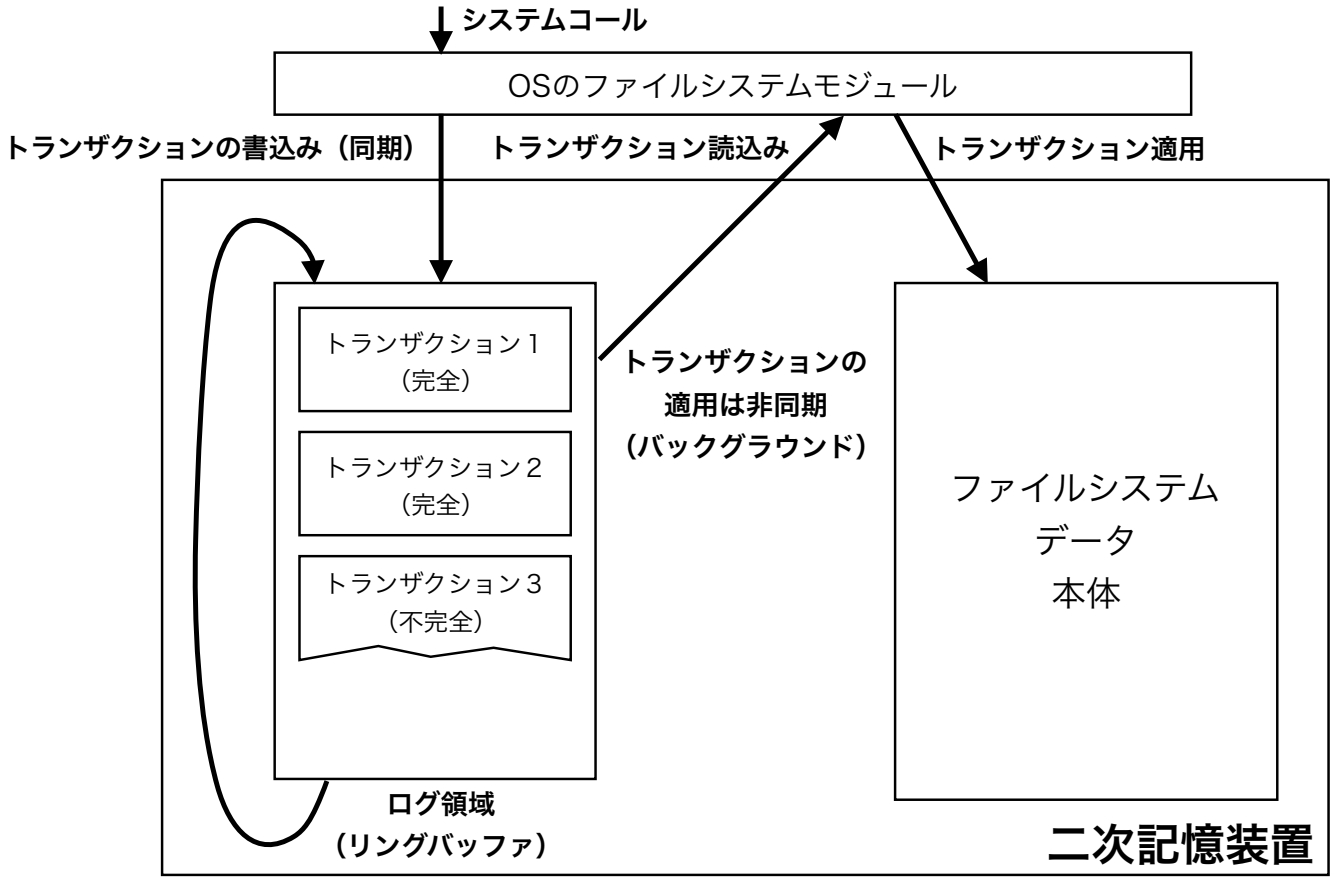
ファイルシステムの健全性（1）

一貫性チェック

- 正常終了時にはファイルシステムにアンマウントの印をする.
- OS の起動時に印がなかったら一貫性チェックをする.
- **メタデータ**の矛盾を解消するだけ.
- ファイルが消えたり，データが消えたりは修復できない.

ファイルシステムの健全性（2）

ジャーナリング・ファイルシステム



- データベースの WAL（Write Ahead Logging）のアイデア.
- NTFS，ext3，ext4，HFS+ 等が該当する.

練習問題（1）

1. 次の言葉の意味を説明しなさい.
   - ディレクトリシステム
   - パス，絶対パス，相対パス
   - ディレクトリ，ファイル
   - ハードリンク，シンボリックリンク
   - ショートカット，エイリアス
   - マウント，ドライブレター
   - 拡張属性，ACL
2. 自分のオペレーティングシステムについて調査しなさい.
   (GUI より CLI のコマンドを用い方がより詳しい観察ができる.)
   - ショートカット（Windows），エイリアス（macOS）
   - ファイルの属性（保護，日時，所有者，サイズ等）
   - 拡張属性が使用できるオペレーティングシステムか？
   - ACL が使用できるオペレーティングシステムか？
   - USB メモリにはどのようなパスで到達できるか？
   - ファイルシステムの一貫性をチェックするコマンドは何か？

練習問題（2）

3. 自分が使用しているオペレーティングシステムで試してみなさい.
   - ショートカットやエイリアスを作成し試してみなさい.

```
# macOS の場合の実行例
$ echo aaa > a.txt
$ open a.txt
$ open a.txt のエイリアス        <--- エイリアスは GUI で作る
$ cat a.txt
$ cat a.txt のエイリアス
```

   - UNIX や macOS で実行して結果が異なる理由を考察しなさい.

```
# ハードリンクの場合
$ echo aaa > a.txt
$ echo bbb > b.txt
$ ln a.txt c.txt
$ mv a.txt d.txt
$ mv b.txt a.txt
$ cat c.txt
```

```
# シンボリックリンクの場合
$ echo aaa > a.txt
$ echo bbb > b.txt
$ ln -s a.txt c.txt
$ mv a.txt d.txt
$ mv b.txt a.txt
$ cat c.txt
```

   - ショートカットやエイリアスの振る舞いを調べる.
     (リンク先ファイルを削除・移動・別ファイルに置換えした場合など)
   - ACL の追加・削除とその効果を確認する.