

# オペレーティングシステムの機能を使ってみよう

## 第4章 ファイルシステム

# ファイルシステム

- **ファイル**

二次記憶装置に格納された不揮発性のデータ記憶のこと  
(二次記憶：HDD, USB メモリ, CD-ROM, SSD, ...)

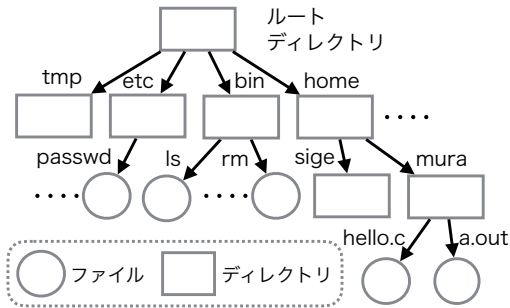
- **ファイルシステム**

二次記憶装置に多数のファイルを記憶・管理する仕組み  
記憶・管理されているファイルの集合

- **UNIX ファイルシステム**

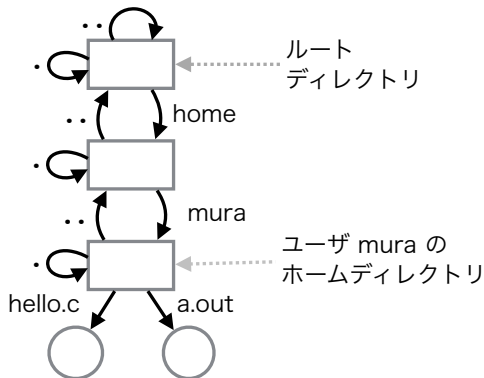
Windows や macOS のファイルシステムも基本は同じ

# ファイル木



- ルートディレクトリを根にした有向の木構造
- 節点（ノード）はディレクトリ（フォルダ）
- 葉（リーフ）はファイル
- 有向枝（エッジ）はリンク
- **ファイルとリンクは独立している**
- ディレクトリもファイルの一種

# 特別なディレクトリ



- ルートディレクトリ (ファイル木の根になるディレクトリ)
- 親ディレクトリ (ルートに近いディレクトリ, 「..」で表す)
- カレントディレクトリ (現在位置, 「.」で表す)
- ホームディレクトリ (ログイン時のカレントディレクトリ)

# パス (Path)

- パスは径の意味 (ファイルへの道)
- ファイルをパスにより特定できる.
- ファイル木のリンクに付いた名前を「/」で区切って書く.
- パスには以下の二種類がある.

## 1 絶対パス

ルートディレクトリを起点にしたパス

「/」で書き始める.

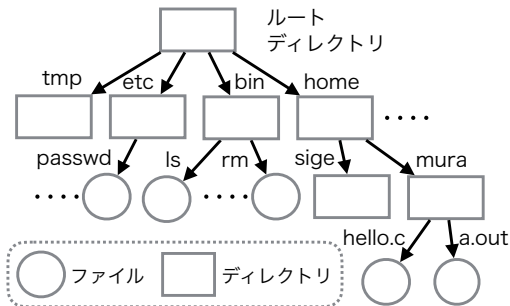
例: /home/mura/hello.c

## 2 相対パス

カレントディレクトリを起点にしたパス「/」以外で書き始める.

例: hello.c

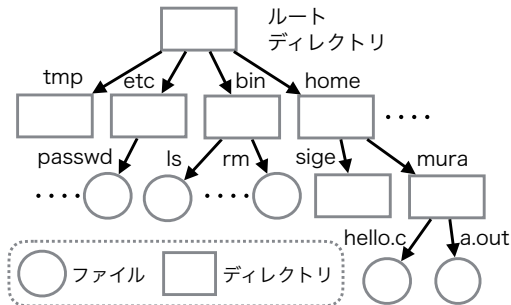
# 絶対パス



ルートディレクトリを起点にしたパス

- `/etc/passwd`
- `/bin/ls`
- `/home/mura` : ディレクトリへのパス
- `/home/mura/hello.c`
- `/home/sige/../mura/./hello.c`

## 相対パス



カレントディレクトリを起点にしたパス  
(カレントディレクトリが/home のとき)

- mura/hello.c
- sige : ディレクトリへのパス
- ../bin/ls
- sige/../../mura/hello.c

# カレントディレクトリ

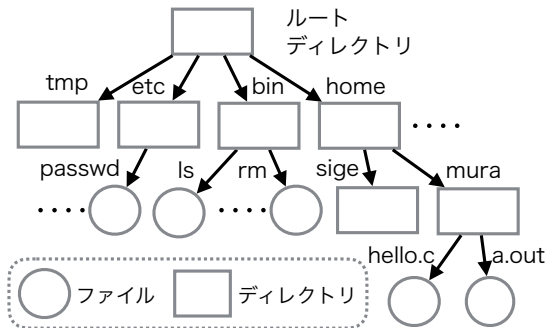
プロセス毎にカレント（現在の）ディレクトリがある。

- カレントディレクトリの変更は他のプロセスに影響はない。
- 他のターミナル（ターミナルもプロセス）に影響はない。
- 次回のログインにも引継がれない。
- 以下のコマンドで変更と確認ができる。
  - **変更** (*cd* コマンド)  
\$ cd パス
  - **確認** (*pwd* コマンド)  
\$ pwd

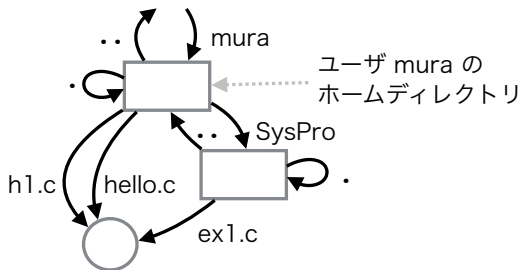


# cd, pwd コマンドの実行例

```
$ pwd
/home/mura
$ ls
a.out  hello.c
$ ls .
a.out  hello.c
$ cp hello.c h.c
$ cp /home/mura/hello.c i.c
$ ls
a.out  h.c
i.c    hello.c
$ cd ..
$ pwd
/home
$ cd ../bin
$ pwd
/bin
$ cd ~
$ pwd
/home/mura
```

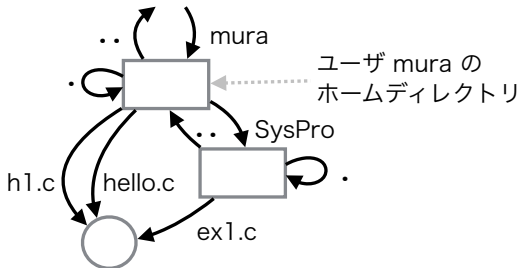


# リンク (ハードリンク)



- ファイルに別名を付ける.
- ハードリンクとシンボリックリンクの二種類がある.
- **ハードリンク**
  - 従来のリンクと同じもの
  - 一つのファイル本体に複数のリンクが可能
  - 元々あったリンク, 後で追加したリンクに区別はない

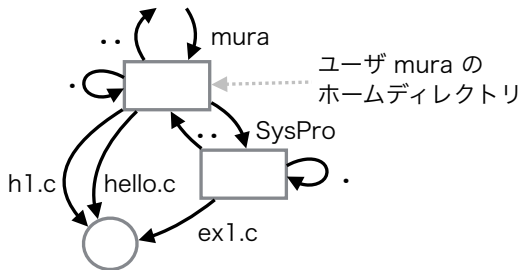
## リンク（ハードリンクの作成）



*ln* コマンドを用いる.

```
$ pwd
/home/mura                                # カレントディレクトリはここ
$ ln hello.c h1.c                         # hello.c にリンク h1.c を追加
$ mkdir SysPro                             # SysPro ディレクトリを作る
$ ln hello.c SysPro/ex1.c                 # リンク ex1.c を追加
$ cat h1.c                                # hello.c の内容が表示される
$ cat SysPro/ex1.c                        # hello.c の内容が表示される
```

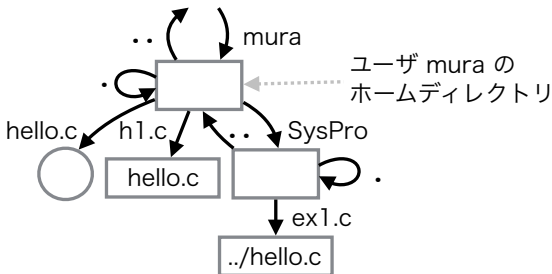
# リンク（ハードリンクの削除）



*rm* コマンドを用いる.

```
$ pwd
/home/mura          # カレントディレクトリはここ
$ rm h1.c           # リンク h1.c を削除
$ rm SysPro/ex1.c   # リンク ex1.c を削除
$ rmdir SysPro      # SysPro ディレクトリを削除
```

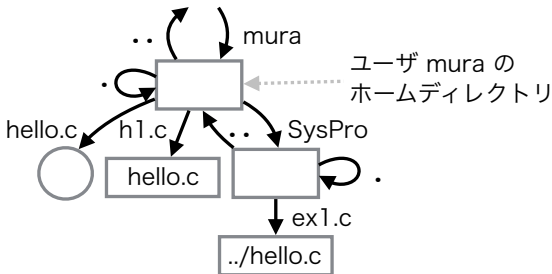
## リンク (シンボリックリンク)



- シンボリックリンク

- パスを格納した特殊なファイル
- ソフトリンクとも呼ぶ
- パス名の解釈時に字面で評価される
- 字面での評価なので制約が少ない（別ディスク，リンク切）
- ファイルが置換わると新しいファイルを参照する

## リンク（シンボリックリンクの作成）

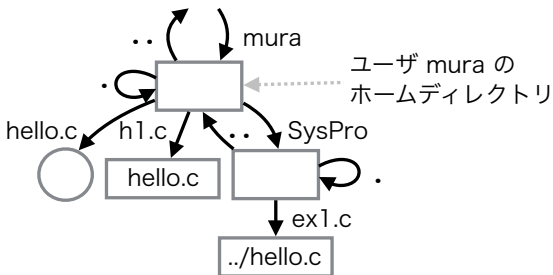


***ln -s* コマンドを用いる.**

```
$ pwd
/home/mura
$ ln -s hello.c h1.c
$ mkdir SysPro
$ ln -s ../hello.c SysPro/ex1.c
$ cat h1.c
$ cat SysPro/ex1.c
```

# カレントディレクトリはここ  
# リンク h1.c を作成  
# SysPro ディレクトリを作る  
# ex1.c を作成  
# hello.c の内容が表示される  
# hello.c の内容が表示される

## リンク（シンボリックリンクの削除）



*rm* コマンドを用いる.

```
$ pwd
/home/mura
$ rm h1.c
$ rm SysPro/ex1.c
$ rmdir SysPro
```

# カレントディレクトリはここ  
# リンク h1.c を削除  
# リンク ex1.c を削除  
# SysPro ディレクトリを削除

# ファイルの属性

## 主な属性

**種類** 普通ファイル，ディレクトリ，シンボリックリンク等

**保護モード** open システムコールで紹介した `rw-rw-rw-`.

**リンク数** ファイルを指しているハードリンクの数. リンク数が0になるとファイル本体が削除される. (例えば，ハードリンク例の `hello.c` ファイルの場合は3になる)

**所有者** 所有者のユーザ番号.

**グループ** 属するグループのグループ番号.

**ファイルサイズ** ファイルの大きさ (バイト単位).

**最終参照日時** 最後にアクセスした時刻.

**最終変更日時** 内容を最後に変更した時刻.

**最終属性変更時刻** 属性を最後に変更した時刻.



# 属性の表示方法

`ls -l` コマンドを用いる.

```
$ ls -l a.txt
-rw-r--r-- 1 mura  staff 10 May  1 18:18 a.txt
```

**ファイルの種類** 一文字目の「-」はファイルが普通のファイルであることを表している. 一文字目が「d」はディレクトリであること, 「l」はシンボリックリンクであることを表す.

**ファイルの保護モード** open システムコールで紹介したもの (rwxrwxrwx).

**リンク数** 1 はリンク数が1であることを表している.

**所有者** mura はファイルの所有者がユーザ mura であることを表している. メタ情報の内部表現はユーザ番号であるが, ls コマンドがユーザ名に変換して表示している.

# 属性の表示方法

`ls -l` コマンドを用いる.

```
$ ls -l a.txt  
-rw-r--r--  1 mura  staff 10 May  1 18:18 a.txt
```

**グループ** `staff` はファイルがグループ `staff` に属することを表している. メタ情報の内部表現はグループ番号であるが, `ls` コマンドがグループ名に変換して表示している.

**ファイルサイズ** `10` はファイルのサイズが `10` バイトであることを表している.

**最終変更日時** `May 1 18:18` はファイルの最終変更日時である.

**パス** `a.txt` はファイルへ到達するために使用したパスである.  
**パス名 (ファイル名)** はファイルの属性ではない.

# 属性の変更方法

*chmod* コマンドを用いる.

```
$ chmod 000 ファイル...      # 書式 1
$ chmod ugo+rwx ファイル...  # 書式 2
$ chmod ugo-rwx ファイル...  # 書式 3
```

文字	意味
u	所有者 (ユーザ)
g	グループ
o	その他のユーザ
+	権利を与える
-	権利を取上げる
r	読出し
w	書込み
x	実行

**書式 1** 000 は 3 桁の 8 進数である. 8 進数で保護モードを指定する. 8 進数の値のは open システムコールの書式 2 と同じである.

**書式 2, 3** ugo+-rwx の文字を組合せて保護モードの変更方法を記述する. 各文字の意味は上の通りである. 例えば, 所有者とグループに書込み権と実行権を与える場合なら ug+wx のように書く. その他のユーザの読出し権を取上げるなら o-r のように書く.

# 属性の変更方法

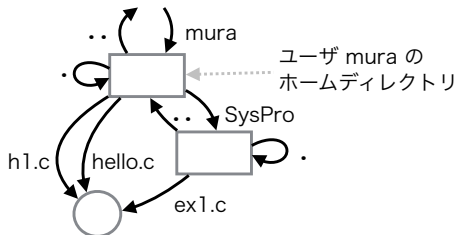
## *chmod* コマンドの使用例

```
$ ls -l a.txt
-rw-r--r--  1 mura  staff 10 May  1 19:42 a.txt
$ chmod 640 a.txt
$ ls -l a.txt
-rw-r-----  1 mura  staff 10 May  1 19:42 a.txt
$ chmod g+w a.txt
$ ls -l a.txt
-rw-rw----  1 mura  staff 10 May  1 19:42 a.txt
$ chmod g-r a.txt
$ ls -l a.txt
-rw--w----  1 mura  staff 10 May  1 19:42 a.txt
```

# 課題 No.3 (1/4)

## 1. パスとハードリンク

- (a) 自分のホームディレクトリのパスを調べる.
- (b) ハードリンクに関する課題を行うために、ディレクトリ/tmp にカレントディレクトリを移動する<sup>1</sup>.
- (c) /tmp 以下に図のディレクトリやファイルを作る.
- (d) `ls -l` を用いてファイルの種類やリンク数を確認する.



<sup>1</sup>PC 教室の Mac はユーザのホームディレクトリを共有ドライブ上に置いている。/tmp はローカルハードディスク上にある。

## 課題 No.3 (2/4)

- (e) 相対パスを用いて `hello.c` の内容を表示する.  
(表示には `cat` コマンドを用いると良い.)
- (f) 絶対パスを用いて `hello.c` の内容を表示する.
- (g) `h1.c` を利用した相対パスを用いて `hello.c` の内容を表示する.
- (h) `ex1.c` を利用した相対パスを用いて `hello.c` の内容を表示する.
- (i) カレントディレクトリを `SysPro` に変更する.
- (j) `ex1.c` を利用した相対パスを用いて `hello.c` の内容を表示する.
- (k) `hello.c` を利用した相対パスを用いて `hello.c` の内容を表示する.
- (l) ディレクトリのハードリンクができるか試す.
- (m) 課題のために作成したファイルやディレクトリを全て削除する.

### 2. シンボリックリンク

- (a) シンボリックリンクを作ってみる.
- (b) シンボリックリンクを `ls -l` を用いて確認する.
- (c) シンボリックリンクを用いてファイルをアクセスできることを確認する.
- (d) リンク切れのシンボリックリンクを使用するとどうなるか確認する.
- (e) ディレクトリに対するリンクを作って使用できることを確認する.
- (f) 他のディレクトリにあるファイルをリンクして使用できることを確認する.
- (g) シンボリックリンクのループを作る. (`a.txt` → `b.txt`, `b.txt` → `a.txt`)
- (h) ループしているシンボリックリンクを使用するとどうなるか確認する. (`$ cat a.txt`)

### 3. 保護モード

- (a) テキストファイル (`hello.c`) と実行可能ファイル (`a.out`) を準備する.
- (b) `chmod` で保護モードを変化させてみる.
- (c) `ls -l` で変化を確認する.
- (d) 保護モードを変化させて、ファイルの読出しや実行ができるか確認する.
- (e) ディレクトリの保護モードは何の意味を持つか考える.