

## 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

オペレーティングシステムの機能を使ってみよう

1 / 24

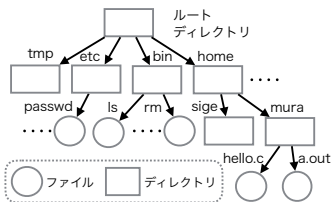
### ファイルシステム

- ファイル  
二次記憶装置（ストレージ）に格納された不揮発性のデータ記憶  
（二次記憶装置：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

オペレーティングシステムの機能を使ってみよう

2 / 24

### ファイル木

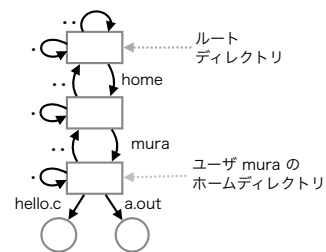


- ルートディレクトリを根にした有向の木構造
- 節点（ノード）はディレクトリ（フォルダ）
- 葉（リーフ）はファイル
- 有向枝（エッジ）はリンク
- ファイルとリンクは独立している
- ディレクトリもファイルの一種

オペレーティングシステムの機能を使ってみよう

3 / 24

### 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）

オペレーティングシステムの機能を使ってみよう

4 / 24

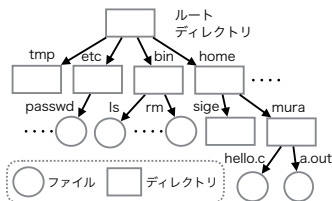
### パス (Path)

- 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

オペレーティングシステムの機能を使ってみよう

5 / 24

### 絶対パス



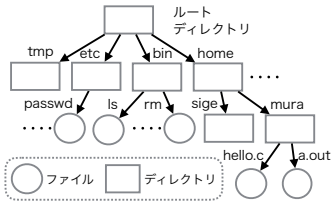
ルートディレクトリを起点にしたパス

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

オペレーティングシステムの機能を使ってみよう

6 / 24

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

オペレーティングシステムの機能を使ってみよ

7 / 24

## 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。

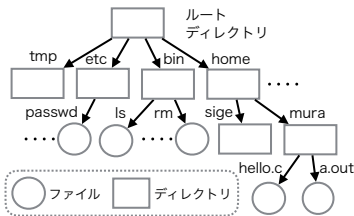
- カレントディレクトリの変更は他のプロセスに影響はない。
- 他のターミナル（ターミナルもプロセス）に影響はない。
- 次回のログインにも引継がれない。
- 以下のコマンドで変更と確認ができる。
  - 変更 (cd コマンド)
    - % cd パス
  - 確認 (pwd コマンド)
    - % pwd

オペレーティングシステムの機能を使ってみよ

8 / 24

## cd, pwd コマンドの実例

```
% 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
```



オペレーティングシステムの機能を使ってみよ

9 / 24

## 演習 4-1

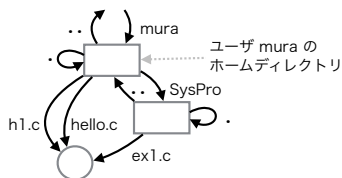
ファイル木を理解する

- Linux の人: 0410\_演習 1 \_... (Linux 版).pdf をやってみる。
- Mac の人: 0420\_演習 1 \_... (Mac 版).pdf をやってみる。

オペレーティングシステムの機能を使ってみよ

10 / 24

## リンク (ハードリンク)

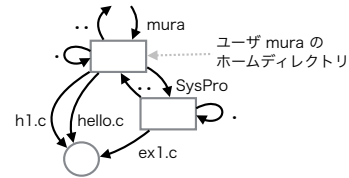


- ファイルに別名を付ける。
- ハードリンクとシンボリックリンクの二種類がある。
- ハードリンク
  - 従来のリンクと同じもの
  - 一つのファイル本体に複数のリンクが可能
  - 元々あったリンク、後で追加したリンクに区別はない

オペレーティングシステムの機能を使ってみよ

11 / 24

## リンク (ハードリンクの作成)



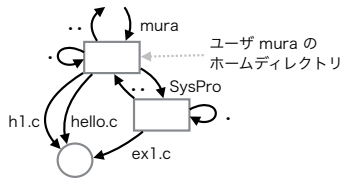
ln コマンドを用いる。

```
% pwd
/home/mura
% ln hello.c h1.c      # カレントディレクトリはここ
% mkdir SysPro        # hello.c にリンク h1.c を追加
% ln hello.c SysPro/ex1.c # SysPro ディレクトリを作る
% cat h1.c            # リンク ex1.c を追加
% cat SysPro/ex1.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
```

オペレーティングシステムの機能を使ってみよ

12 / 24

## リンク (ハードリンクの削除)



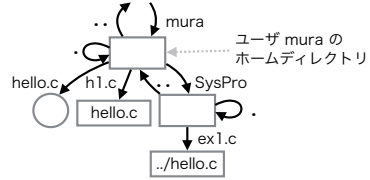
rm コマンドを用いる,

```
% 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
```

オペレーティングシステムの機能を使ってみよ

13 / 24

## リンク (シンボリックリンク)



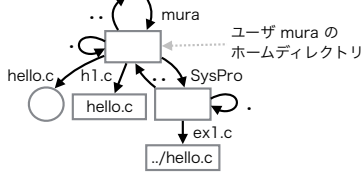
● シンボリックリンク

- 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

オペレーティングシステムの機能を使ってみよ

14 / 24

## リンク (シンボリックリンクの作成)



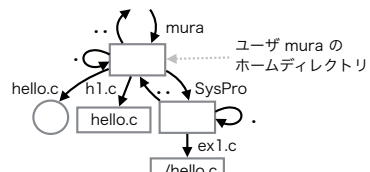
ln -s コマンドを用いる,

```
% pwd
/home/mura          # カレントディレクトリはここ
% ln -s hello.c h1.c # リンク h1.c を作成
% mkdir SysPro       # SysPro ディレクトリを作る
% ln -s ../hello.c SysPro/ex1.c # ex1.c を作成
% 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
```

オペレーティングシステムの機能を使ってみよ

15 / 24

## 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          # 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
```

オペレーティングシステムの機能を使ってみよ

16 / 24

## 演習 4-2

リンクに関する演習

- 0440\_演習2\_リンクと....\_演習.pdf の「1. リンクを作る」をやってみる.

オペレーティングシステムの機能を使ってみよ

17 / 24

## 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-rwx`.

**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.

オペレーティングシステムの機能を使ってみよ

18 / 24

## 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
```

## 演習 4-3

保護属性に関する演習

- 0440\_演習2\_リンクと.... 演習.pdf の「2. 保護属性 (rwxrwxrwx) の効果を確認する」をやってみる。

## 課題 No.3

ファイルシステム

1. 演習 4-1, 4-2, 4-3 を行う。
2. 0440\_演習2\_リンクと.... 演習.pdf の「3. 1,2の結果を提出する」に従い Github に結果を提出する。