

ファイルと
ディレクトリ,
パーミッション

目次

- ファイルとディレクトリ
- パーミッション
- まとめ
- 参考文献

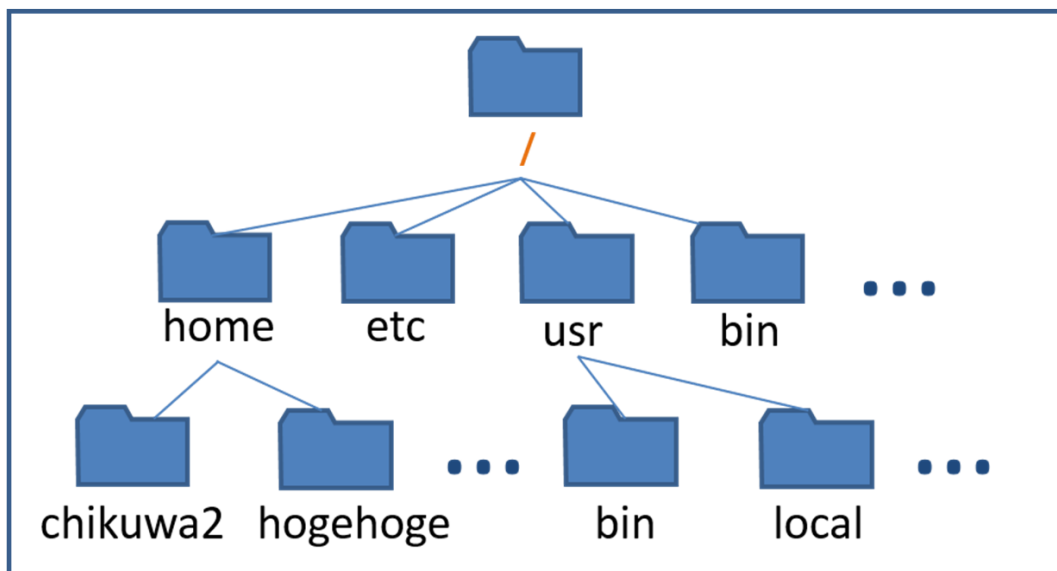
ファイルと
ディレクトリ

Linux におけるデータ管理

- すべてのデータはファイルとして管理される
 - ファイル: 任意のデータを記録し名前をつけたもの
 - ファイルの種類
 - テキストファイル: 人間が読めるファイル
 - バイナリファイル: 機械が読めるファイル
- ファイルはディレクトリで階層的に管理される
 - ディレクトリ: ファイルを格納するファイル
(Windows, mac でいえばフォルダ)
 - ディレクトリの中にディレクトリを格納することも可能

ディレクトリの階層構造

- ルートディレクトリ “/” を起点とした階層構造 (ツリー構造)



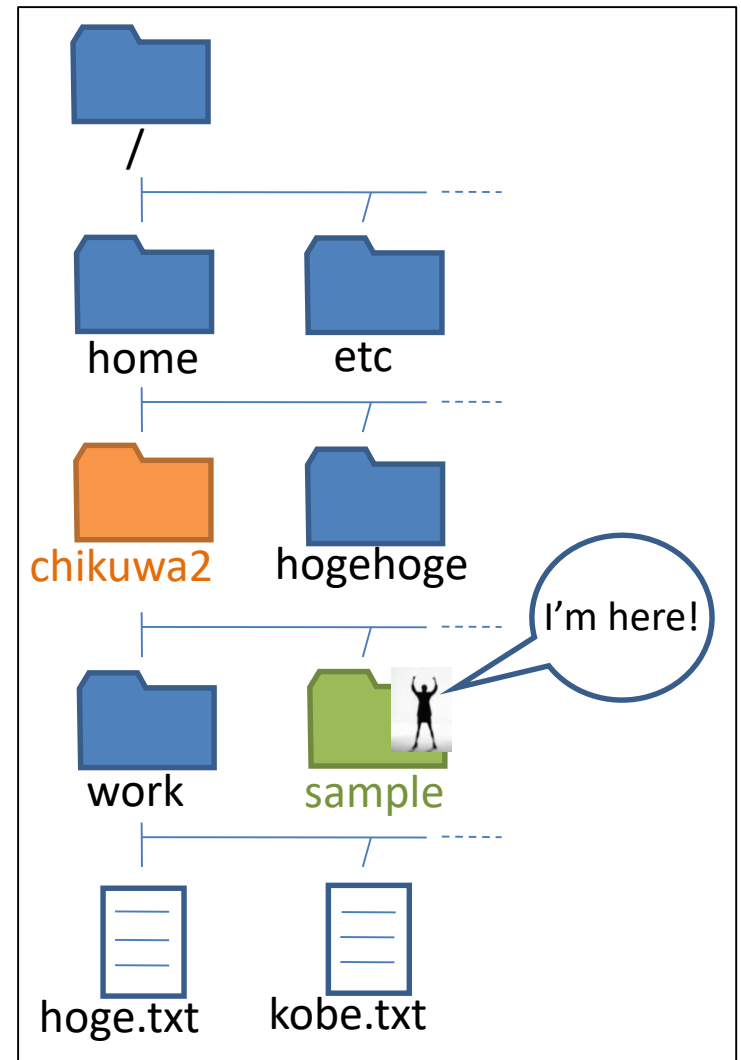
- ツリー構造の利点
 - ファイルやディレクトリを意味のあるまとまりにしておくことで、人間も計算機もファイルにアクセスしやすくなる
 - 別のディレクトリ以下に同じ名前のファイルを作成できる

ファイルを指定する方法

- ファイルの位置 (パス; path) の指定方法には 2 通りある
 - “相対パス”
 - “自分が今いる場所” を起点にして指定
 - “絶対パス”
 - “ルートディレクトリ” を起点にして指定

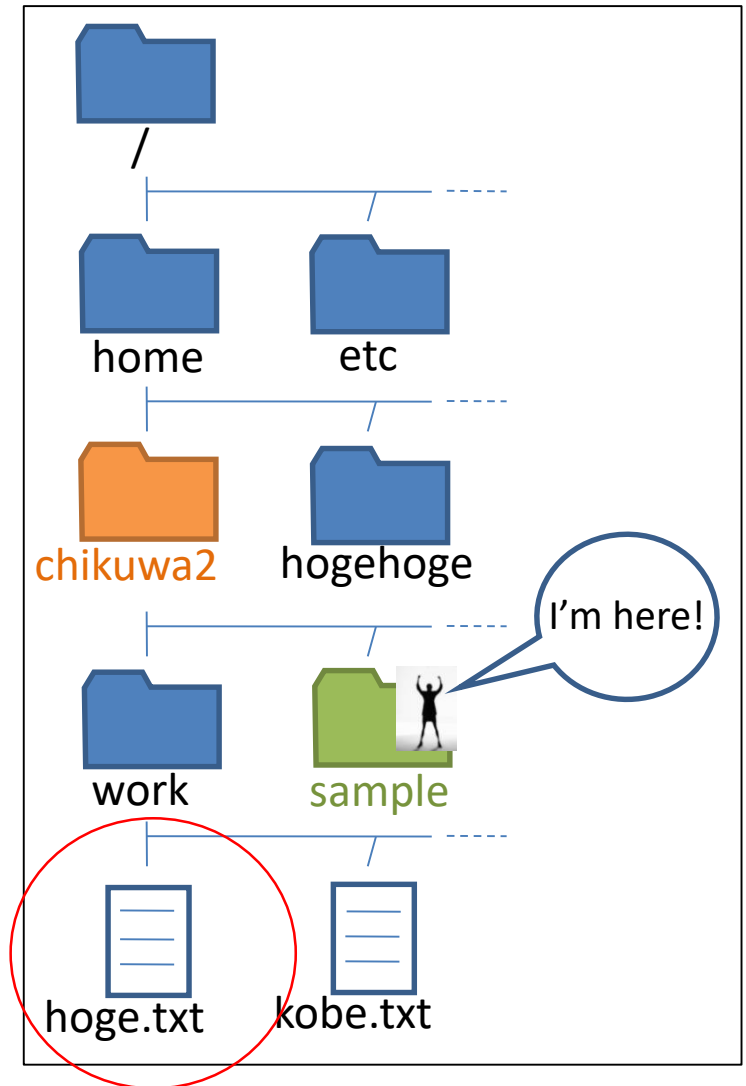
相対パスによる ファイルの指定

- 現在のディレクトリ:
 - カレントディレクトリ
 - “.” (ドット) で表す
 - 右の例では “sample”
- 一段上のディレクトリ:
 - 親ディレクトリ
 - “..” (ドットドット) で表す
 - 右の例では “chikuwa2”



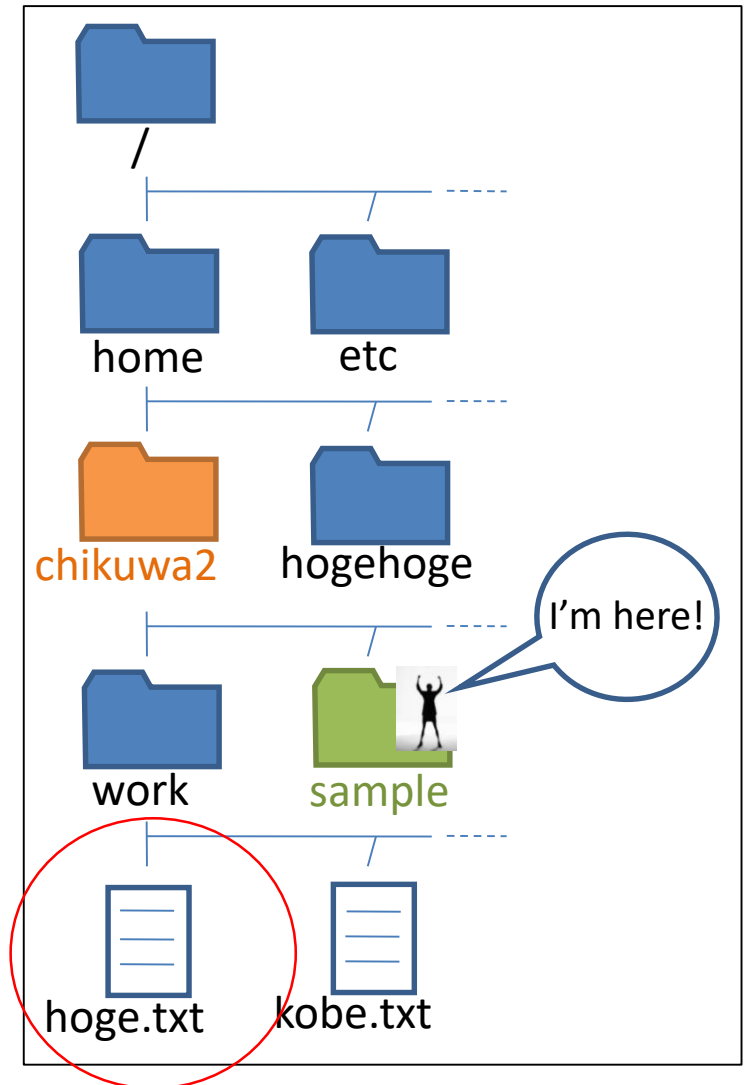
相対パスによる ファイルの指定

- ファイル “hoge.txt” への道順
 - “sample” → “chikuwa2” → “work” → “hoge.txt”
- “hoge.txt” の指定
 - “./../work/hoge.txt”
- “.” = “sample”
- “..” = “chikuwa2”
- “/” = ディレクトリ間の切れ目



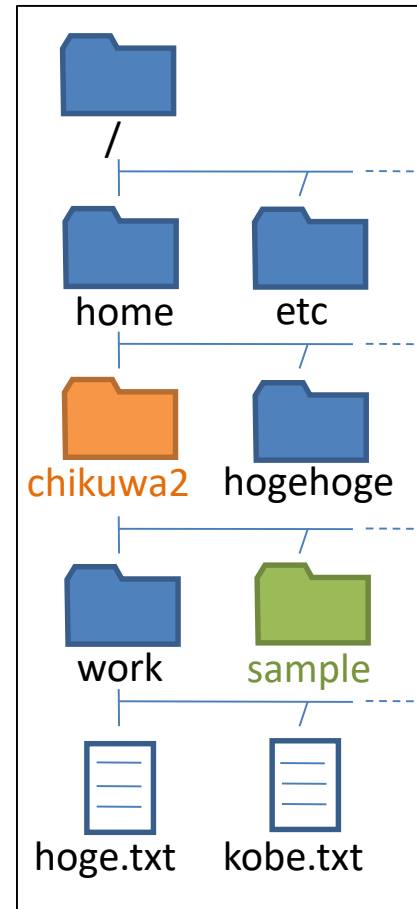
絶対パスによる ファイルの指定

- ファイル“hoge.txt”への道順
 - “/” → “home” → “chikuwa2” → “work” → “hoge.txt”
- “hoge.txt”の指定
 - home/chikuwa2/work/hoge.txt



絶対パスによる ファイルの指定

- 毎回 “/home/chikuwa2”
を書くのは面倒?
- “~” を用いて、ホーム
ディレクトリまでを省
略できる
 - chikuwa2 さんが hoge.txt
を指定
 - ~/work/hoge.txt
 - hohogoho さんが
hoge.txt を指定
 - ~chikuwa2/work/hoge.txt



ディレクトリに関する コマンド

<code>pwd</code>	現在のディレクトリの場所を絶対パスで表示
<code>ls</code>	ディレクトリにあるファイルの一覧を表示
<code>cd</code>	ディレクトリを移動する
<code>mkdir</code>	ディレクトリを作成
<code>rmdir</code>	空のディレクトリを削除
<code>rm</code>	ファイルやディレクトリを削除

コマンドを調べるには

- よく分からない, もっと詳しく知りたいコマンドに出会ったら?
 - ネットで検索、周りの人にきく
 - man を使う
 - コマンドのマニュアルを表示するコマンド
 - manual の略
 - 使い方は `man [コマンド名]`
 - 例)
 - `$ man rm`
 - `rm`: ファイルやディレクトリを削除するコマンド

パーミッション

パーミッションとは

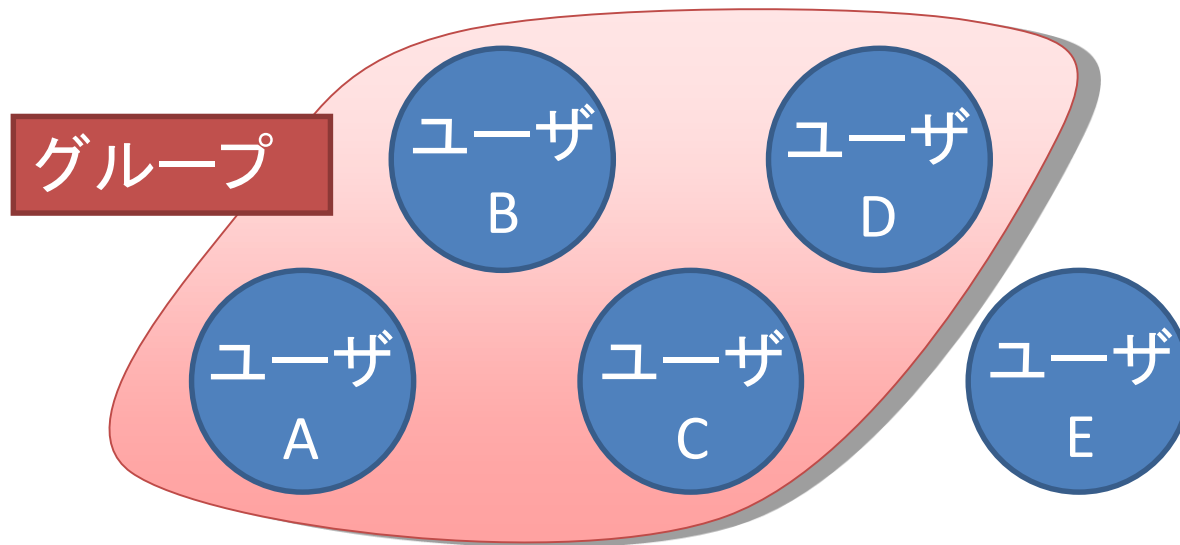
- ファイル, ディレクトリの“利用権限”
 - すべてのファイル, ディレクトリにそれぞれパーミッションが設定されている

パーミッションの設定

- 誰に許可するか
 - ファイル所有者 (User)
 - ファイル所有グループ (Group)
 - その他 (Others)
- 何を許可するか
 - ファイルを読み取る許可 (Read)
 - ファイルを書き換える許可 (Write)
 - ファイルを実行する許可 (eXecute)

ファイルの所有グループ

- 複数のユーザを束ねて管理する単位
 - 所有グループが同じだと, 共同作業をする際に便利
 - Linux では多くの場合, 所有グループは所有者と一致



ファイルモード

d **r w x** **r - x** **r - x**

File type

User

Group

Others

- ls -l コマンドで表示

ファイルのモード

所有者と所有グループ

```
rin@ika-itpass:~$ ls -l
total 24
drwxr-xr-x 2 rin rin 4096 Jul  3 13:48 chikwa2
-rw-r--r-- 1 rin rin   9 Jul  3 13:47 hoge.txt
```

ファイルモード

d **r w x** **r - x** **r - x**

File type

User

Group

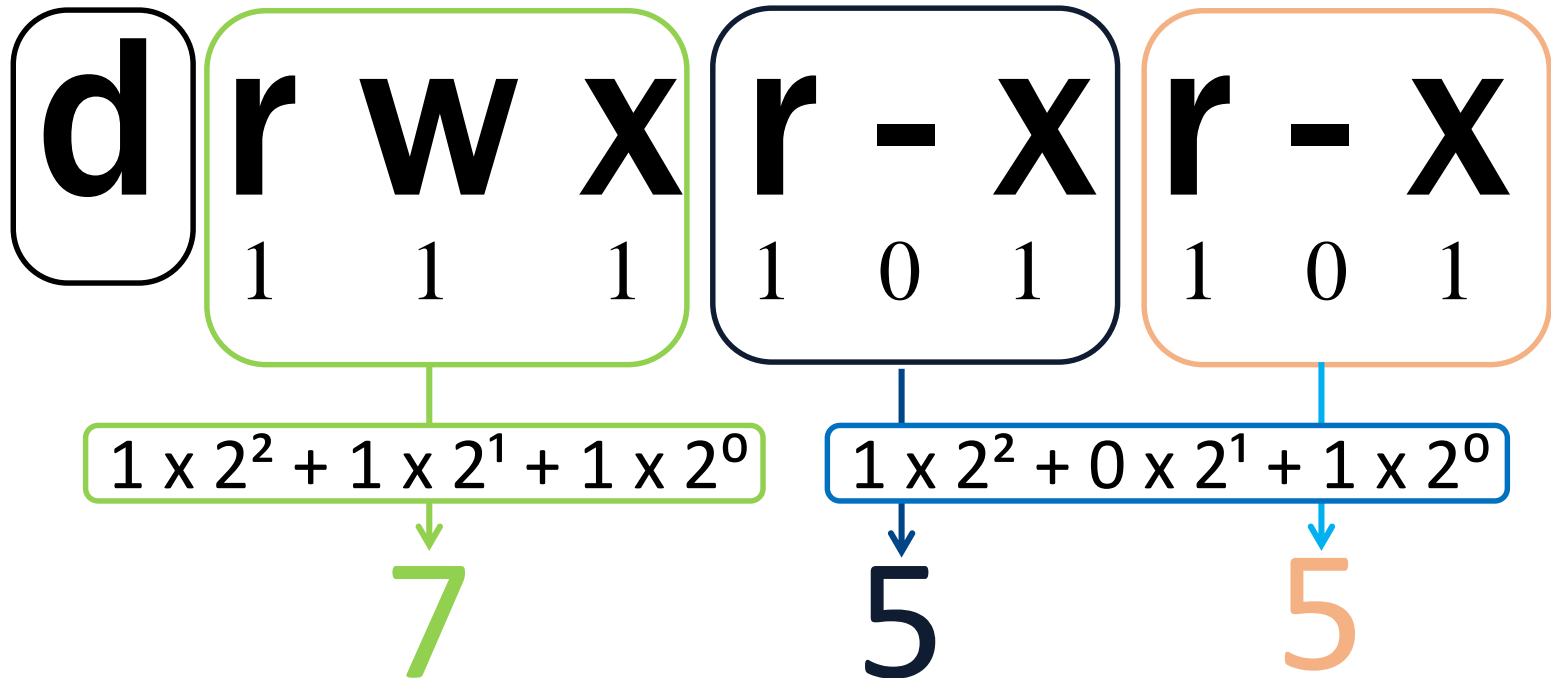
Others

- “d” = ディレクトリ
- “-” = 通常ファイル

- “r” = 読み取り
- “w” = 書き込み
- “x” = 実行
- “-” = 不許可

ファイルモード

- 計算機は各権限が許可されているかどうかを2進数で表現している



※便宜上 r を 4, w を 2, x を 1 と考えると簡単

ファイルモードの変更

d **r w x** **r - x** **r - x**

File type

User

Group

Others

7

5

5

- chmod コマンドで変更できる

例) “755” を “775” に変更する

- \$ chmod 775 [ファイル名]
 - User および Group は全権限アリ
 - Other は読み込み, 実行権限アリ
- \$ chmod g+w [ファイル名]
 - “group” に “write” 権限を与える

パーミッションの意義

- マルチユーザであるからこそ...
 - 見られたくないファイルを見られてしまう可能性
 - メール, 発表前の研究成果 ...etc.
 - 人に重要なファイルを消されてしまう可能性
 - /etc/shadow ...etc.
- ファイル, ディレクトリのパーミッションを適切に設定する必要がある
 - 例)
 - 他人には見せない, 侵入させない
 - 閲覧は許可するが, 書き換えは許可しない... などなど

スーパーユーザ

- 唯一の例外 “スーパーユーザ”
 - 計算機の管理者であるスーパーユーザ (root) はすべてのファイル, ディレクトリにアクセスが可能
 - ファイルの所有者やパーミッションの変更も自由自在
 - “root” は絶大な権限が与えられる代わりに, その計算機に対する責任も背負う

まとめ

- ファイルとディレクトリ
 - ツリー構造で階層的に管理
 - 相対パス・絶対パスで指定
- パーミッション
 - ファイルやディレクトリの利用権限
 - 利用権限の種類は `r, w, x`
- ファイルとディレクトリの構造を理解し, それぞれについて適切なパーミッションを設定しよう