

ディレクトリ

目次

| | | |
|----------|---|----------|
| 1 | ディレクトリとその操作 | 1 |
| 1.1 | ディレクトリの作成 — mkdir | 1 |
| 1.2 | ディレクトリ名の変更とディレクトリ削除 — mv, rmdir | 1 |
| 2 | ホームディレクトリとカレントディレクトリ | 2 |
| 2.1 | ホームディレクトリ | 2 |
| 2.2 | カレントディレクトリ | 2 |
| 2.3 | カレントディレクトリの変更 — cd | 2 |
| 2.3.1 | 例 | 2 |
| 3 | 階層化したディレクトリの作成と利用 | 3 |
| 3.1 | 階層化したディレクトリ | 3 |
| 3.1.1 | 練習 | 4 |
| 3.2 | カレントディレクトリと親ディレクトリの操作 | 4 |
| 3.2.1 | 親ディレクトリと子ディレクトリ | 4 |
| 3.2.2 | カレントディレクトリと親ディレクトリの指定法 — “.” と “..” | 5 |
| 3.2.3 | 練習 | 5 |
| 4 | ディレクトリ間のファイルの移動と複写 | 6 |
| 4.1 | ディレクトリ間のファイルの移動と複写 — mv, cp | 6 |
| 4.1.1 | 練習 | 6 |
| 5 | 問題 | 7 |
| 6 | まとめ | 8 |
| 6.1 | ファイル/ディレクトリ操作の主なコマンド一覧 | 8 |

1 ディレクトリとその操作

Windows におけるフォルダに相当するものを、UNIX ではディレクトリ(directory) またはディレクトリファイル(directory file) と呼びます。ディレクトリはファイルを格納する器です。ディレクトリを利用することで、様々なファイルを、その用途別に分類して整理することができます。

ディレクトリファイルという名称から予想できるとおり、ディレクトリもファイルの一種です。これまでに扱ってきたファイルを「通常のファイル」と呼んで、ディレクトリではないことを明示することがあります。

まず、ディレクトリの作成や削除、ディレクトリ名変更の方法を学びましょう。ファイルと同じくディレクトリにも名前(ディレクトリ名)が必要です。ディレクトリ名の付け方はファイル名と同様です。

1.1 ディレクトリの作成 — mkdir

ディレクトリを作成するには mkdir (make directories) コマンドを使います。引数には作成するディレクトリの名前を指定します。

1. tmpdir という名前のディレクトリを作成してください。

```
mkdir tmpdir
```

2. ディレクトリ tmpdir ができたかどうか確認します。

ディレクトリもファイルの一種なので、ls コマンドで名前を表示できます。ls を実行してください。

3. 通常のファイルとディレクトリが同じように表示されるのは不便ですね¹。両者を区別したいときには、ls にオプション -F を指定します。

```
ls -F
```

ディレクトリの場合にはディレクトリ名の後に / が付加されます。ただし、この / はディレクトリであることを表す印であって、ディレクトリ名の一部ではありません。

4. ディレクトリに対して cat コマンドを実行したらどうなるでしょう。cat tmpdir を試してみましょう。cat コマンドは、通常のファイルの内容を表示するコマンドでしたね。

1.2 ディレクトリ名の変更とディレクトリ削除 — mv, rmdir

ディレクトリ名の変更には、ファイル名変更と同様に mv を使います。ディレクトリの削除は rmdir (remove directories) です。

1. mv tmpdir temp を実行して、tmpdir の名前を temp に変更してください。ls コマンドで結果を確認してください。

2. temp を削除します。rmdir temp を実行してください。結果を確認してください。

¹ファイルの種類を色で区別するように設定されていることもあります

2 ホームディレクトリとカレントディレクトリ

2.1 ホームディレクトリ

UNIX で利用できるファイルやディレクトリは、必ずいずれかのディレクトリ内にあります²。これまでに幾つかのファイルを作成しましたが、それらはユーザー毎に予め用意されたディレクトリ内に作成していたのです。そのディレクトリをユーザーのホームディレクトリ(home directory)と呼びます。ホームディレクトリを記号 `~` で表すことがあります。

2.2 カレントディレクトリ

コマンド操作でファイルやディレクトリを扱うときに必要な概念として、カレントディレクトリ(current directory; current は「現在の」,「現行の」という意味)³があります。カレントディレクトリは文字通り、現在使っているディレクトリを意味する用語です。

例えば、`ls` を引数無しで実行して表示されるのは、カレントディレクトリに存在するファイルの名前です。また、`cp` や `mkdir` 等のコマンドにファイル名やディレクトリ名を与えると、コマンドが処理するのはカレントディレクトリ内のファイルやディレクトリとなります⁴。

ユーザーがログインしたときのカレントディレクトリはホームディレクトリです。これまで、ファイルやディレクトリがホームディレクトリに作成されたのはそのためです。

2.3 カレントディレクトリの変更 — `cd`

ホームディレクトリ (`~`) 以外のディレクトリに存在するファイルやディレクトリを、これまでと同様にファイル名やディレクトリ名で操作するには、カレントディレクトリの変更の操作を行います。

カレントディレクトリを変更するコマンドは `cd` (change directory) です。`cd` の引数には変更先のディレクトリを指定します。引数を指定しなければ、ホームディレクトリがカレントディレクトリになります(ホームディレクトリに戻れます)。

`cd` の引数にディレクトリ名を指定する際にも、そのディレクトリはカレントディレクトリに存在していなければなりません。他の場所にあるディレクトリを `cd` の引数に与える方法は後の資料で扱います。

2.3.1 例

1. カレントディレクトリにディレクトリ `tmp` を作成 — `mkdir tmp`
2. 確認 — `ls -F`
3. カレントディレクトリを `tmp` に変更 — `cd tmp`

何も表示されなければ成功です。プロンプトにカレントディレクトリ名が表示される設定がなされていれば、プロンプトが変わります。

4. カレントディレクトリをホームディレクトリに変更 — `cd`

²ルートディレクトリという特別なディレクトリを除く。

³ワーキングディレクトリ (working directory) や作業ディレクトリと呼ぶこともあります。

⁴カレントディレクトリ以外のファイル操作については後で学びます。

3 階層化したディレクトリの作成と利用

3.1 階層化したディレクトリ

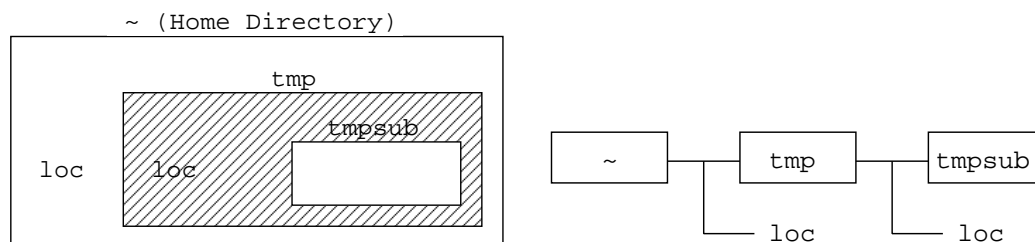


図 1: 階層化ディレクトリの例

- ディレクトリの中には、さらにディレクトリを作ることができます。この仕組みを利用すると、ファイルを大分類・中分類・小分類のように、階層的に分類して整理できます。図 1 の左の図では、ホームディレクトリの中にディレクトリ tmp があり、さらにその中にディレクトリ tmpsub がある様子を示しています。なお、loc は通常のファイルです。

- 右の図は、左の図を別の形式で描いたものであり、ディレクトリの中にあるファイルやディレクトリを右側を実線で結んで描画することにより、ディレクトリの階層的な構造を表しています。今後、ディレクトリの階層構造の表現には、右の図の形式のみを用います。

- 左の図における斜線は、カレントディレクトリが tmp であることを表現したものです。この状態でカレントディレクトリに存在するのは、斜線部に書かれたファイル loc とディレクトリ tmpsub です。これらが右の図ではどれであるか、わかりますか。

コマンド操作で指定するファイル名やディレクトリ名は、カレントディレクトリのファイルやディレクトリである点に留意してください (第 2.2 節参照)。

- 一つのディレクトリに、同じ名前のファイルやディレクトリを複数することはできませんが、ディレクトリが異なれば可能です。従って、図 1 のように、同名のファイル loc がホームディレクトリおよびディレクトリ tmp に存在することは許されます⁵。
- ls コマンドの引数に既存のディレクトリ *directory* を与えて

```
ls [options] directory
```

の形式で実行すれば、*directory* 内のファイル一覧が出力されます⁶。

もしも、カレントディレクトリにディレクトリ tmpsub があるならば、

```
ls tmpsub
```

は tmpsub に存在するファイルの一覧を出力します。

⁵ファイル名 loc は単語 location からとった名前です。後の練習では、このファイルを実際に作成し、カレントディレクトリがどこであるかを確認するために使います。

⁶ls にオプション -d を指定すると、ディレクトリの内容ではなく、ディレクトリ自身を表示します。

3.1.1 練習

図 1 に示したディレクトリとファイルをホームディレクトリ (~) に作ります。自分が今どのディレクトリにいるのか (カレントディレクトリがどこなのか) を常に意識しながら、次の操作をしてください。

1. 通常のファイル loc をホームディレクトリに作ります。loc の中身は Here is my home とします。

```
echo Here is my home > loc
```

正しくできたか、ls コマンドおよび cat loc で確認してください。

2. カレントディレクトリをホームディレクトリから tmp に変更します。ディレクトリ tmp が存在することを確認してから、

```
cd tmp
```

を実行してください。今後、ディレクトリ変更の操作を、単に「tmp に移る・行く」のように表現することがあります。

3. ls を実行して、カレントディレクトリである tmp に存在するファイルを調べてください。tmp には、まだ何も作っていませんから、何も表示されませんね。
4. echo コマンドを使って、ディレクトリ tmp の中に loc という通常のファイルを作ってください。ファイルの中身は Here is tmp とします。正しくできたか確認してください。
5. mkdir コマンドを使って、ディレクトリ tmp の中に tmpsub というディレクトリを作ってください。結果を確認してください。
6. ホームディレクトリに戻るために、引数無しで cd を実行してください。ちゃんとホームに帰れましたか？ファイル loc の中身を確認してください。
7. rmdir tmp を試してください。ディレクトリ tmp を削除できませんね。rmdir は中身が空のディレクトリを削除するコマンドだからです。tmp は消さないでおきましょう。
8. ホームディレクトリをカレントディレクトリとしたままで、ディレクトリ tmp の中身 (tmp に存在するファイル) を確認してみましょう。ls の引数にはディレクトリ名を与えることができるのでしたね。ls tmp と ls -F tmp を実行してください。

3.2 カレントディレクトリと親ディレクトリの操作

3.2.1 親ディレクトリと子ディレクトリ

図 2 は、前節の 3.1.1 で作成した、ホームディレクトリ (~) 内のディレクトリ/ファイルの構造を、再度示したものです。

あるディレクトリの一つ上位のディレクトリを 親ディレクトリ (parent directory) と呼びます。図 2 では、tmpsub の親ディレクトリは tmp であり、tmp の親ディレクトリは ~ です。逆に、tmpsub を tmp の子ディレクトリ (child directory) またはサブディレクトリ (subdirectory) と呼びます。tmp は ~ の子ディレクトリです。

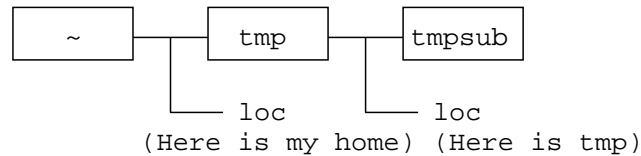


図 2: 親ディレクトリと子ディレクトリ

3.2.2 カレントディレクトリと親ディレクトリの指定法 — “.” と “..”

カレントディレクトリの子ディレクトリは「カレントディレクトリに存在するディレクトリ」です。そのため、第 2.2 節で述べたとおり、子ディレクトリの名前をそのままコマンドの引数として指定すれば、それがコマンドの処理対象となります。

一方、カレントディレクトリ自身や「カレントディレクトリの親ディレクトリ」は、カレントディレクトリに存在するディレクトリではありません。したがって、ディレクトリ名をコマンドの引数に与えても、コマンドの処理対象にはなりません。これらをコマンドの引数にしたければ、次の記号を使います。

- . (ドット) : カレントディレクトリ自身
- .. (ドット二つ) : カレントディレクトリの親ディレクトリ

3.2.3 練習

1. カレントディレクトリを、図 2 の tmpsub に変更してください。

tmpsub はカレントディレクトリに存在するディレクトリではないので、`cd tmpsub` だけではうまくいきません。まず tmp に移ってから、tmpsub に移りましょう。

2. tmpsub (カレントディレクトリ) に存在するファイルを `ls` コマンドで調べましょう。

何も表示されませんね。

3. `ls` に `-F` と `-a` の二つのオプションを付けて実行してみましょう。`ls` に複数のオプションを付けるときには、それらを `-aF` のようにまとめて書くことができます。

```
ls -aF
```

`-a` は `.` で始まるものを含めて全てのファイル名を表示する `ls` のオプションでしたね。カレントディレクトリ自身とその親ディレクトリの存在を確認できましたか。

4. カレントディレクトリを、親ディレクトリである tmp に変更します。

まず、`cd tmp` を試みましょう。

コマンドの引数には、カレントディレクトリに存在するファイル/ディレクトリ名を指定するのでしたから、これではうまくいきません。

そこで、先ほどカレントディレクトリに存在することを確認した、親ディレクトリの記号 `..` を使います。

```
cd ..
```

エラーメッセージは出ませんね。成功です。

5. カレントディレクトリの内容を `ls -aF` で表示してみてください。loc というファイルが存在するはずですから、ファイルの中を見てください。

今はちゃんと tmp にいますね。

6. `ls -aF .` の結果を予想して、試してください。

`ls` の引数に `.` を指定すると、カレントディレクトリに存在するファイルの一覧が得られます。したがって、結果は `ls -aF` と同じです。

7. `ls -aF` や `ls -aF .` の表示では `..` も現われましたから、親ディレクトリ (ホームディレクトリ) の指定には、再度 `..` を使うことができます。

`ls` コマンドの引数に `..` を与えて、親ディレクトリに存在するファイルの一覧を表示してみましょう。

8. `..` を使ってホームディレクトリに戻りましょう。

4 ディレクトリ間のファイルの移動と複写

4.1 ディレクトリ間のファイルの移動と複写 — mv, cp

前の資料では、`mv` と `cp` の引数にファイル名を与えてファイル名の変更とファイルの複写を行いました。が、`mv` や `cp` の最後の引数に 既存の ディレクトリ名 *directory* を指定すると、`mv` と `cp` は次のとおり動作します。

`mv file directory` — *file* を *directory* に移動する

`cp file directory` — *file* を *directory* に、同じ名前のファイルとして複写する

directory として `.` や `..` を指定することも可能です。なお、`mv` の場合、*file* としてディレクトリ名を与えれば、ディレクトリを別のディレクトリに移動することもできます⁷。

4.1.1 練習

1. 準備

- (a) カレントディレクトリをホームディレクトリにするために、`cd` コマンドを引数無しで実行してください。
- (b) ホームディレクトリに通常のファイル `now` がありますか？中身はどうなっていますか？ファイル `now` が無ければ `date > now` で作成してください。
- (c) ディレクトリ `tmp` がホームディレクトリに存在することを `ls -F` で確認してください。

2. ファイルの移動

- (a) ファイル `now` をディレクトリ `tmp` に移動します。 `mv now tmp` を実行してください。

⁷`cp` でディレクトリを複写するにはオプション `-r` が必要です。

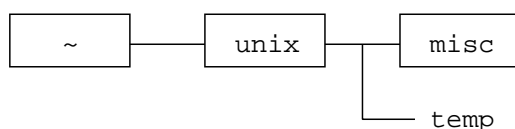
- (b) `ls` を実行して、ホームディレクトリから `now` が無くなったことを確認しましょう。
- (c) `ls tmp` を実行して、ディレクトリ `tmp` に `now` が存在することを確認しましょう。
- (d) カレントディレクトリを `tmp` に変更してから、ファイル `now` の内容を見てみましょう。
`now` はディレクトリ `tmp` に移動できていますね。

3. ファイルの複写

- (a) ディレクトリ `tmp` のファイル `now` を親ディレクトリ (ホームディレクトリ) に複写します。`cp now ..` を実行してください。
- (b) `ls` を実行して、カレントディレクトリ (`tmp`) に `now` が残っていることを確認しましょう。
- (c) カレントディレクトリ (`tmp`) のファイル `now` の内容を見てみましょう。
- (d) カレントディレクトリを親ディレクトリ (ホームディレクトリ) に変更して、複写した `now` の内容を見てみましょう。

5 問題

1. ホームディレクトリに存在する通常のファイルやディレクトリの名前一覧を表示しましょう。
2. その中でディレクトリはどれでしょう? ファイルの種類を表す記号 (ディレクトリの場合は `/`) がつくように、ファイルやディレクトリ名の一覧を表示しましょう。
3. ホームディレクトリに存在するファイル `loc` の名前を `loc.home` に変更してください。
4. ホームディレクトリの子ディレクトリである `tmp` に存在するファイル `loc` の名前を `loc.tmp` に変更してください。
5. ディレクトリ `tmp` に存在するファイル `now` を、ディレクトリ `tmpsub` に移動してください。
6. ディレクトリ `tmp` をカレントディレクトリにした状態で、ディレクトリ `tmpsub` にファイル `now` があることを確認してください。
7. ディレクトリ `tmpsub` をカレントディレクトリにした状態で、ディレクトリ `tmp` にファイル `now` が無いことを確認してください。
8. ホームディレクトリに存在するディレクトリ `tmp` の名前を `junk` に変更してください。
9. ディレクトリ `junk` と、その中のすべてのファイルやディレクトリを削除してください。ディレクトリの削除には `rmdir` コマンドを使ってください。
10. 次回の準備 次の図に示すように、ホームディレクトリにディレクトリ `unix` を作成し、そこにディレクトリ `misc` を作成してください。また、ホームディレクトリに存在するファイル `now` を `unix` に移動し、その名前を `temp` に変更してください。



6 まとめ

この資料では、ファイルやディレクトリの最も基本的な操作法である、カレントディレクトリに存在するファイルの操作や、カレントディレクトリの親および子ディレクトリへのファイル複写や移動の方法について学びました。

カレントディレクトリがどこであるかに拘らず、任意のディレクトリに存在するファイルを操作することもできます。その方法は次回の資料で学習します。

6.1 ファイル/ディレクトリ操作の主なコマンド一覧

| コマンド | 機能 |
|---------------------------------------|---|
| <code>ls [directory ...]</code> | カレントディレクトリ (または <i>directory ...</i>) に存在するファイルの名前を表示 |
| <code>ls -F [directory ...]</code> | ファイルの種類を表す記号を付加して、カレントディレクトリ (または <i>directory ...</i>) に存在するファイルの名前を表示 |
| <code>ls -a [directory ...]</code> | ドットで始まるファイルを含め、カレントディレクトリ (または <i>directory ...</i>) に存在するファイルの名前を表示 |
| <code>cd [directory]</code> | カレントディレクトリを <i>directory</i> に変更 (<i>directory</i> を省略した場合はホームディレクトリに変更) |
| <code>mkdir directory ...</code> | <i>directory ...</i> の作成 |
| <code>rmdir directory ...</code> | 空のディレクトリ <i>directory ...</i> の削除 |
| <code>cat file ...</code> | <i>file ...</i> の内容を表示 |
| <code>cp file1 file2</code> | <i>file1</i> を <i>file2</i> に複写 |
| <code>cp file ... directory</code> | <i>file ...</i> を <i>directory</i> に複写 |
| <code>mv file1 file2</code> | <i>file1</i> の名前を <i>file2</i> に変更 |
| <code>mv directory1 directory2</code> | <i>directory1</i> の名前を <i>directory2</i> に変更 |
| <code>mv file ... directory</code> | <i>file ...</i> を <i>directory</i> に移動 |
| <code>rm file ...</code> | <i>file ...</i> を削除 |
| <code>rm -r file ...</code> | <i>file ...</i> をディレクトリ階層を含めてすべて削除 (誤操作に注意) |
| <code>pwd</code> | カレントディレクトリ名を絶対パス名で表示 |

file ... は複数のファイルを指定できることを、`[]` は中の要素が省略可能であることを表します。

ここには次回紹介するコマンドも含んでいます。