# 11　Windows における階層ディレクトリ

## 1. ドライブ名とは

MS-DOS（エムエス・ドス）や Windows などの OS では，使用するディスクドライブ（Disk Drive：ディスク装置）に，A，B，C，…，Z という名前が自動的に付けられる。

ディスクドライブとは，記憶媒体（記録メディア）を読み書きするドライブ（drive：装置）のことであり，フロッピーディスクドライブ（FDD：Floppy Disk Drive），ハードディスクドライブ（HDD：Hard Disk Drive），CD-ROM ドライブ，DVD ドライブなどがある。

ディスクドライブに付けられるこの名前（アルファベット）を，ドライブ名（drive name）といい，ドライブ名が A のドライブを，「ドライブ A」「A ドライブ」などと呼ぶ。ドライブ名は，ドライブ文字（drive letter）ということもある。

Windows は，ドライブ名によってディスクドライブを区別（識別）し，管理している。ドライブ名は，次のように自動的に付けられる。

○ 3.5 インチ FDD　…　ドライブ A
○ 1 台目の HDD　…　ドライブ C
○ 2 台目の HDD　…　ドライブ D
○ 3 ……　…　ドライブ E

従って，3.5 インチ FDD はドライブ A であり，HDD はドライブ C から始まる。

ハードディスクは，通常は複数の領域（パーティション）に分割して使用する。例えば，1 台のハードディスクのみを使用しそれを 2 つの領域に分割した場合は，2 つの HDD が存在すると解釈され，1 番目の領域がドライブ C，2 番目の領域がドライブ D となる。

A と C 以外のドライブ名は，ユーザが自由に変更できるが，初心者はドライブ名を変更しない方がよい。トラブルの元である。

自分が使用するパソコンのドライブ名は，きちんと認識しておこう。「コンピューター」のアイコンを開くと，利用可能なドライブとドライブ名が表示される。ドライブ C のハードディスクドライブなどは，「ローカルディスク（C:）」と表示される。

## 2. MS-DOS の階層ディレクトリとは

MS-DOS は，Microsoft 社によって開発されたパソコン用の OS である。Microsoft Disk Operating System を略して，MS-DOS（エムエス・ドス）と名付けられた。Windows が登場する以前のパソコン用の OS として，最もよく使用されていたものである。

さて，ハードディスクには大量のファイルを保存できるが，ディスク上の数多くのファイルを効率良く管理する方法として，MS-DOS では「階層ディレクトリ」という方法が導入された。これは，元々は UNIX で使用されていたファイル管理法であり，MS-DOS にも導入され，Windows にも継承されている。大容量のハードディスクを使う場合，このファイル管理

法を知らないとパソコンは有効に使えない。

ディレクトリ（directory）とは，

ファイルを入れる入れ物

と考えればよい。ディレクトリは，ファイルと同様にディスクに記録される情報である。

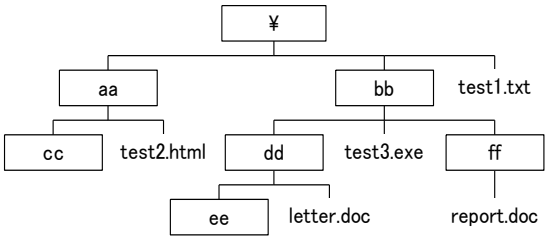
Windows やマッキントッシュでは，ディレクトリを「フォルダ（folder）」とも呼ぶ。ディレクトリとフォルダは同義語である。（フォルダはマッキントッシュで使われていた用語であり，それが Windows にも導入された。）

ディレクトリには，ファイルのみならず他のディレクトリも入れることができる。ユーザはディレクトリを自由に作成できるので，いくつかのディレクトリを適当な名前（ディレクトリ名）を付けて作成し，その中にファイルを入れて分類・整理する。必要ならば，ディレクトリの中に他のディレクトリを複数作成して，大量のファイルを分類して格納していく。

このような構造，すなわち，入れ物 A の中に入れ物 B があり，さらに入れ物 B の中に入れ物 C がある…という構造は，階層的な構造である。階層ディレクトリとは，階層的な構造になっているディレクトリ構造のことであり，1 つのディスクの中にいくつかのディレクトリ（入れ物）を階層的に作成し，数多くの各種のファイルをそれぞれのディレクトリに適切に格納して分類・整理するという考え方である。

階層ディレクトリの構造は，大もとの根（root：ルート）から枝が生え，枝分かれして葉がついているように見えるので，その構造を「Tree 構造」「木構造」などと呼ぶ。

以下は，階層ディレクトリの例であるが，四角で囲まれているものがディレクトリで，それ以外はファイルである。図をさかさまにすれば，木のように見えるだろう。Windows では，どのドライブのディスクも，以下のような 1 つの Tree 構造をなしている。




## 3. 階層ディレクトリにおける基本用語

例えば，フロッピーディスクを例にとって説明しよう（他のディスクでも同じである）。

フロッピーディスクをドライブにセットして，データが書き込めるようにフォーマット（初期化）したとする[1]。このとき，ディスクには，一番大きなディレクトリが自動的に作成され

[1] いかなる記憶媒体も，フォーマットしなければデータを記録できない。ただし，フロッピーディスや USB メモリは，フォーマット済みで販売されている。

る。このディレクトリを

ルートディレクトリ（またはルートフォルダ）

という。単に「ルート」ともいう。

ディレクトリの名前を「ディレクトリ名（フォルダ名）」というが，ルートディレクトリのディレクトリ名は自動的に半角記号の

¥

となる[1]。このディレクトリは削除できず，名前の ¥ も変更できない。

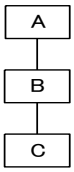
ルートディレクトリはそのディスク上の一番大きな入れ物なので，ディスクに作成されるすべてのディレクトリやファイルは，ルートディレクトリの中に格納される。

ルートディレクトリ以外のディレクトリを

サブディレクトリ（またはサブフォルダ）

と呼ぶ。サブディレクトリは，ユーザが自由に作成・削除できる。

階層ディレクトリでは，上下関係・親子関係で，複数のディレクトリを相対的に見ることがある。例えば，次のようなディレクトリ構造が，ある階層ディレクトリの一部であったとしよう。




つまり，ディレクトリ A の中にディレクトリ B があり，ディレクトリ B の中にディレクトリ C があるとする。このとき，「A の下に B がある」「B の下に C がある」などという。

慣用的に，B は A のすぐ下のディレクトリ，A は B のすぐ上のディレクトリ，あるいは，B は A の配下（はいか）にあるディレクトリともいう。

また，A と B を親子関係でとらえ，

A は B の親ディレクトリ（親フォルダ）
B は A の子ディレクトリ（子フォルダ）

などという。さらに

C は A の孫ディレクトリ（孫フォルダ）

などという。

もちろん，これらの言い方は相対関係に依存し，B と C の上下関係では，次のようになる。

---

1　アメリカ製のパソコンやソフトウェアでは，¥（円記号，円サイン）は，＼（バックスラッシュ）という記号で表わされる。キーボードの「¥」のキーを押しても，画面には「＼」と表示される。両者の記号は見た目は違うが，同じ文字コードである。

B は C の親ディレクトリ（親フォルダ）
C は B の子ディレクトリ（子フォルダ）

なお，カレントディレクトリ（またはカレントフォルダ）という言葉がある。これは

○ 現在，操作の対象になっているディレクトリ
○ 現在，自分がいるディレクトリ

などと一般に説明されるが，例えば上の図で，画面上のディレクトリ B（フォルダ B）をダブルクリックしてフォルダウィンドウを開いた場合，B がカレントディレクトリになる。

ディレクトリに付けられる名前がディレクトリ名（フォルダ名）であるが，Windows では，ファイル名と同様に，作成するディレクトリに自由に名前を付けることができる（ただし，**＊？｜＜＞￥** などの半角記号は使えない）。

なお，後述する相対パス名のところで登場するが，親ディレクトリやカレントディレクトリに対して，（実際の名前の他に）次のような名前も使われる。

| 対　象 | ディレクトリ名 |
|---|---|
| 親ディレクトリ | **..**（半角ピリオド 2 つ） |
| カレントディレクトリ | **.**（半角ピリオド 1 つ） |

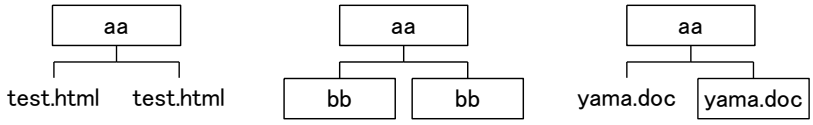
例えば上の図で，ディレクトリ B を開いたとしよう。すると，B がカレントディレクトリになり，B の親ディレクトリは A なので，

A のディレクトリ名は　**..**
B のディレクトリ名は　**.**

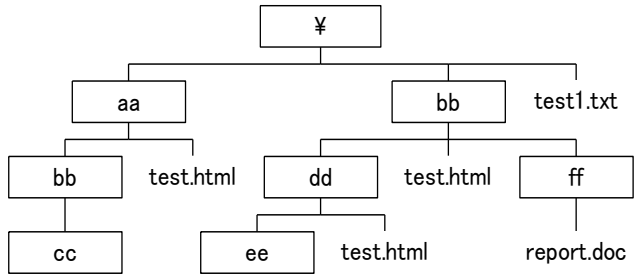
となる。つまり，カレントディレクトリ（自分）はピリオド 1 つ（**.**），その親ディレクトリはピリオド 2 つ（**..**）で表現するわけである。

## 4. 階層ディレクトリにおけるパス名

1 つのディレクトリの中に，（内容は違っていても）同じ名前のものを複数入れることはできない。例えば，次のようなディレクトリ構造を作ることはできない。




しかし，入れる場所が違えば可能である。例えば，次のように，test.html を異なるディレクトリに入れることはできる。




この階層ディレクトリには，ファイル test.html は 3 つ，ディレクトリ bb は 2 つ存在するが，では，これらのファイルやディレクトリはどのようにすれば特定できるのだろうか。単に test.html というファイル名だけでは，OS はどの test.html なのか判断できず困ってしまう。(プログラミングの世界では，ディレクトリやファイルに対して何らかの命令を実行することがあり，その場合，それらを特定しなければならない。)

つまり，階層ディレクトリにおいては，ファイルやディレクトリの名前だけでは，それらの対象を特定できないということである。そこで，パス（path：経路，道順）という概念が登場する。これは，ディレクトリパスともいう。

パスとは,「ディレクトリ」(起点）から他の「ディレクトリやファイル」(終点）までの道順のことである。階層ディレクトリにおいては，起点から終点まで進む時に,

途中で同じディレクトリを 2 度経由してはいけない

というルールがある（同じことだが，同じ道を 2 回通ることができないということ)。そうなると，起点と終点を定めれば，その間のパス，すなわち

起点のディレクトリ → ディレクトリ → …… → ディレクトリ
→ 終点のディレクトリまたはファイル

というパス（道順）は 1 つしか存在しない。と言うことは，起点を定めておけば，ディレクトリやファイルは，そこまでのパスで特定できることになる。

つまり，ディレクトリやファイルという対象を特定するときに，単にそれらの名前だけでは特定できないが，パスを指定すれば対象を一意に特定できるということである。

このようなパスをパス名で表現する。1 つのパス名の形式は，次のようになる。

**（起点のﾃﾞｨﾚｸﾄﾘ名）¥ﾃﾞｨﾚｸﾄﾘ名¥……¥ﾃﾞｨﾚｸﾄﾘ名¥（終点のﾃﾞｨﾚｸﾄﾘ名またはﾌｧｲﾙ名）**

すなわち, 最初に起点のディレクトリ名, 最後に終点のディレクトリ名またはファイル名，その間には経由するディレクトリ名を順に書き，ディレクトリの間を半角記号の ¥ で区切るわけである。ディレクトリ間の区切り記号として ¥ が使われるが，これはルートディレクトリの意味ではない。

区切り記号 ¥ は,「から」「通って」と翻訳すればよい。例えば a¥b¥c というパス名なら，

「a から b から c まで」または「a を通って b を通って c まで」と解釈する。

パス名は，終点を特定する概念であることから，終点に与えられる名前である。a¥b¥c というパス名なら，これを「c のパス名」と呼ぶ。また，我々が「a¥b¥c に対して○○をせよ」と OS に指示を与えると，その命令は終点の c に対して実行される。

パス名において，起点は常にディレクトリであること，終点はディレクトリまたはファイルのいずれでもよいことに注意しよう。なお，パス名には，「絶対パス名」と「相対パス名」がある。

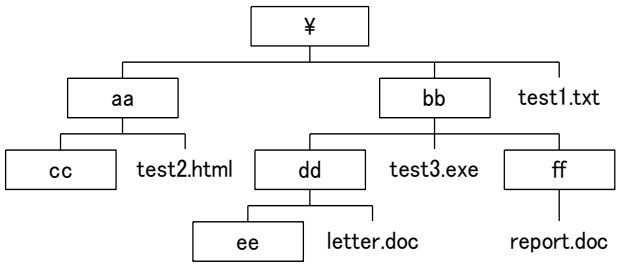
## 5. 絶対パス名

絶対パス名とは

ルートディレクトリ ¥ を起点とするパス名

のことである。カレントディレクトリの位置にかかわらず，起点は必ずルートディレクトリにする。絶対パス名でパス名を指定することを，「絶対パス指定」という。

次の階層ディレクトリで説明しよう。これは，ドライブ A 上にあるものとする。(一番上の ¥ はルートディレクトリであり，四角で囲まれているものがディレクトリ，そうでないものがファイルである。)




例えば，cc の絶対パス名は，ルート ¥ から aa から cc までなので

¥aa¥cc

となる。最初の ¥ はルートディレクトリのことで，それ以降の ¥ はディレクトリの区切り記号である。パス名の形式に従えば，ルート¥と aa の間に区切り記号 ¥ を入れて

¥¥aa¥cc

と書くべきであるが，これは誤りと解釈される。

この¥aa¥cc を，cc の絶対パス名と呼ぶ。いくつか例をあげよう。

○ aa の絶対パス名 … ¥aa
○ ee の絶対パス名 … ¥aa¥bb¥dd¥ee
○ ファイル letter.doc の絶対パス名 … ¥bb¥dd¥letter.doc
○ ルートディレクトリの絶対パス名 … ¥

階層ディレクトリにおいて，絶対パス名が分かれば，どのディレクトリやファイルを指定しているのかが分かる。ただし，階層ディレクトリは複数のディスクに作成できるので，¥aa¥cc という指定だけでは，どのドライブ（ディスク上）のパス名なのかが分からない。（単に¥aa¥cc と指定すると，OS はカレントドライブ上のパス名と解釈する。カレントドライブとは，現在，作業の対象になっているドライブのことである。）

そこで，一般には，絶対パス名の前にドライブ指定を行う。ドライブ指定の形式は

ドライブ名:

である。つまり，ドライブ名の後に半角のコロン（:）を付けてドライブ指定を行う。例えば，ドライブ A のドライブ指定は「A:」であり，ドライブ C のドライブ指定は「C:」となる。ドライブ名は小文字でもよく，「a:」や「c:」と記述しても構わない。以下では，大文字で書く。

さて，上記の階層ディレクトリはドライブ A 上にあると仮定したので，（ドライブ指定を伴った）絶対パス名は以下のようになる。（カレントドライブがドライブ A であれば，¥aa などは A:¥aa と解釈される。）

① ディレクトリ aa の絶対パス名 … A:¥aa
② ファイル test1.txt の絶対パス名 … A:¥test1.txt
③ ファイル letter.doc の絶対パス名 … A:¥bb¥dd¥letter.doc
④ ルートディレクトリの絶対パス名 … A:¥

なお，フォルダウィンドウのタイトルバーやアドレスバーには，開かれているフォルダの絶対パス名が表示される。

## 6. 階層ディレクトリにおける相対パス名

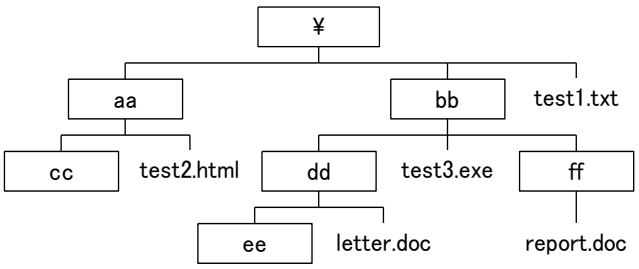
相対パス名とは

カレントディレクトリを起点とするパス名

のことであり，相対パス名でパス名を指定することを「相対パス指定」という。

現在，開かれているディレクトリ（自分自身がいるディレクトリ）をカレントディレクトリと呼んだが，カレントディレクトリの位置によって相対パス名の形は違ってくる。一方，絶対パス名では，カレントディレクトリの位置とは無関係にそのパス名の形は常に同じである。

相対パス名には，次の(1)～(3)の 3 つのルールがある。前と同じ図でいくつか例をあげよう。なお，相対パス名では，ドライブ指定は不要である。




### (1)　カレントディレクトリの名前はピリオド１つ（.）にする

つまり，カレントディレクトリの名前（自分自身の名前）は本当の名前を使わずに，ピリオド１つ（.）にする。

例えば，現在，カレントディレクトリ（自分の居場所）が bb であったとしよう。このとき，ee の相対パス名は，カレントディレクトリ（.）から dd から ee までなので，

**.¥dd¥ee**

となる（2 つの ¥ はどちらもディレクトリの区切り記号である）。

カレントディレクトリが dd であれば，ee の相対パス名は

**.¥ee**

となる。

絶対パス名と同様に，「**.¥dd¥ee**」や「**.¥ee**」を指定して何らかの操作を行った場合，終点の ee に対してその操作が実行される。

ところで，多少混乱する話だが，「カレントディレクトリから（自分から）」という指定

**.¥**

は常に省略できる。従って，上の「**.¥dd¥ee**」は「**dd¥ee**」と書いてもよい。以下の例では，省略した形を後半に書く[1]。

（例）　カレントディレクトリが bb の場合

①　ディレクトリ dd の相対パス名　…　**.¥dd**　または　**dd**
②　ファイル test3.exe の相対パス名　…　**.¥test3.exe**　または　**test3.exe**
③　ファイル letter.doc の相対パス名　…　**.¥dd¥letter.doc**　または　**dd¥letter.doc**
④　カレントディレクトリ bb 自身の相対パス名　…　.　または　　（何も書かない）

---

1　省略しない形，省略した形のどちらで書くべきかは各人の自由だ。しかし，今までの話の流れとしては，省略した形は頭を混乱させる。従って，無理に省略することもない。

**(2) 上の階層のディレクトリを通過するには，親→親→…とたどっていく。各親の名前はピリオド 2 つ(..)にする。**

親ディレクトリの名前は，必ずピリオド 2 つ（..）にしなければならない。

（例）　カレントディレクトリが ee の場合

① ディレクトリ dd の相対パス名　…　**.¥..**　または　**..**
② ディレクトリ bb の相対パス名　…　**.¥..¥..**　または　**..¥..**
③ ルート ¥ の相対パス名　…　**.¥..¥..¥..**　または　**..¥..¥..**

**(3) あるディレクトリから，それと同じ階層のディレクトリやファイルに行くには，その親ディレクトリを経由しなければならない。**

同じ階層のディレクトリやファイルに行くには，親を通過しなければならない[1]。

（例）　カレントディレクトリが dd の場合

① ディレクトリ ff の相対パス名　…　**.¥..¥ff**　または　**..¥ff**
② ファイル test3.exe の相対パス名　…　**.¥..¥test3.exe**　または　**..¥test3.exe**
③ ファイル report.doc の相対パス名
…　**.¥..¥ff¥report.doc**　または　**..¥ff¥report.doc**
④ ファイル test2.html の相対パス名
…　**.¥..¥..¥aa¥test2.html**　または　**..¥..¥aa¥test2.html**

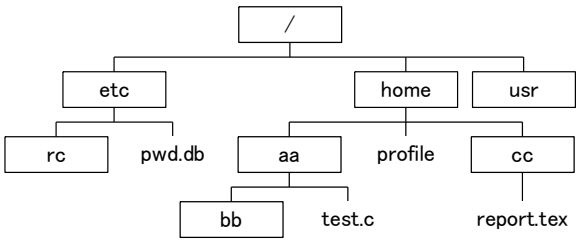
1　よく考えれば，これは当たり前のルールだ。例えば，カレントディレクトリが dd の場合，dd からその横の ff まで行くとしよう。そのとき，dd からその親の bb を経由せずにいきなり ff に行けるとしたら，ff の相対パス名は「**.¥ff**」と書ける。しかし，これは dd の下のディレクトリまたはファイル ff の相対パス名と解釈されてしまう。

# 12. UNIX における階層ディレクトリ

## 1. UNIX の階層ディレクトリ

UNIX という OS には，ドライブ名はない。複数のディスクドライブを使用していても，システムの全体が，以下のような 1 つの Tree 構造で表現される。ルートディレクトリ（/）は，システムの中で 1 つしかない。

UNIX では，ルートディレクトリも，ディレクトリの区切り記号も，スラッシュ（/）で表される。

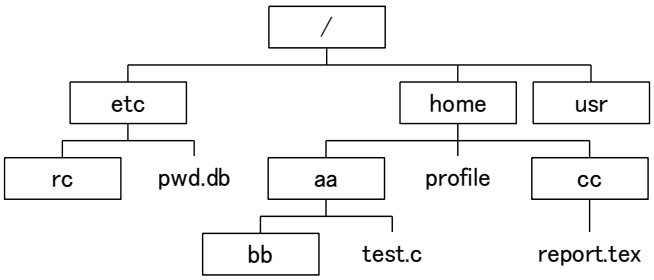



## 2. UNIX での絶対パス名・相対パス名

UNIX での相対パス名・絶対パス名の概念は，Windows と全く同様である。

○ 絶対パス名は，ルートディレクトリ（**/**）を起点とするパス名
○ 相対パス名は，カレントディレクトリを起点とするパス名

次の階層ディレクトリで確認しよう。




例えば，rc の絶対パス名は，ルート（**/**）から etc から rc までなので

**/etc/rc**

となる。

① etc の絶対パス名　…　**/etc**
② rc の絶対パス名　…　**/etc/rc**
③ bb の絶対パス名　…　**/home/aa/bb**
④ report.tex の絶対パス名　…　**/home/cc/report.tex**
⑤ ルート（ / ）の絶対パス名　…　**/**

（例）　カレントディレクトリが home の場合

① aa の相対パス名　…　**./aa**　または　**aa**
② profile の相対パス名　…　**./profile**　または　**profile**
③ report.tex の相対パス名　…　**./cc/report.tex**　または　**cc/report.tex**
④ home 自身の相対パス名　…　**.**　または　　（何も書かない）

（例）　カレントディレクトリが bb の場合

① aa の相対パス名　　…　**./..**　または　**..**
② home の相対パス名　　…　**./../..**　または　**../..**
③ ルート（/）の相対パス名　…　**./../../..**　または　**../../..**

（例）　カレントディレクトリが aa の場合

① cc の相対パス名　　…　**./../cc**　または　**../cc**
② profile の相対パス名　…　**./../profile**　または　**../profile**
③ report.tex の相対パス名
　　…　**./../cc/report.tex**　または　**../cc/report.tex**
④ pwd.db の相対パス名
　　…　**./../../etc/pwd.db**　または　**../../etc/pwd.db**