2016 年 04 月 22 日

計算機実習 III (2016 年度)
第 2 回: コマンドプロンプトとバッチファイル その 2

(http://takeno.iee.niit.ac.jp/%7Eshige/
math/lecture/comp4/comp4.html)

目 次

1 ディレクトリとツリー構造

MS–Windows では、図 1 のように、ファイルは **ドライブ** 毎に木構造 (正確には「木の根」構造) で管理されている[1]

コマンドプロンプトを立ち上げると、ユーザはどこかのドライブの木構造のどこかに「いる」ことになっていて、それがプロンプト部分に表示される。必ずしも最初にトップディレクトリにいるとは限らない。

用語:

- 「ドライブ」= USB メモリ、フロッピーディスク、ハードディスクなどの物理的な記憶装置毎に割り当てる記号 (A: ∼ Z:)。ハードディスクを論理的に分割して各分割単位 (= パーティション) 毎に別のドライブとすることもできる。
- 「ディレクトリ」= フォルダ。その中に複数のファイルやディレクトリを持つことができる (図 1 では太枠で示しているもの)。

  例えば、図 1 の (4) のディレクトリ内には (5) のディレクトリがあり、(5) のディレクトリ内には (6), (7) の 2 つのファイルがある。

[1] この図はこの実習室のコンピュータの状況を表しているわけではない。また、この実習では原則ファイル名やディレクトリ名には日本語 (全角文字) は使わない。

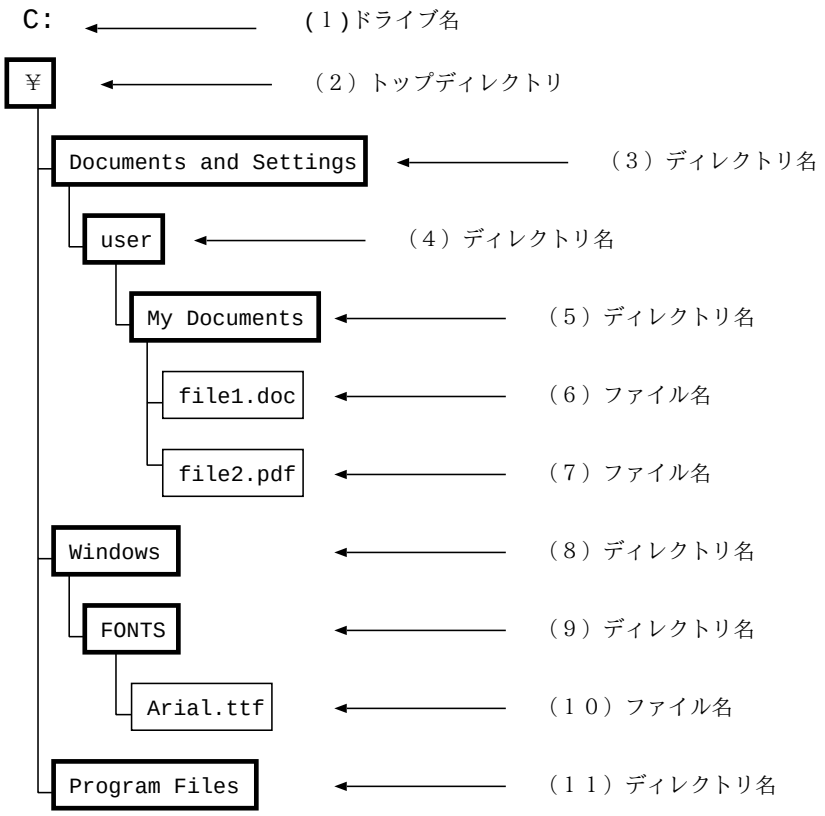

図 1: C: ドライブのツリー構造の一部

- 「カレントディレクトリ」= 現在自分がいるディレクトリ (「カレント」= 現在の)。ピリオド 1 つ (.) で表す。同じように、現在自分がいるドライブを「カレントドライブ」という。
- 「親ディレクトリ」= 一つ上の階層のディレクトリ。ピリオド 2 つ (..) で表す。

  例えば、図 1 の (4) がカレントディレクトリの場合は .. は (3) のディレクトリを指すことになる。
- 「サブディレクトリ」= 階層的に見て下のディレクトリ。

  例えば図 1 の (4), (5) は (3) のサブディレクトリ。
- 「拡張子」= 多くのファイル名の最後についている「. (ドット) + 数文字のアルファベット」の部分。ファイルの種類を区別するのに使用する。ディレクトリには拡張子はない。

  例えば図 1 の (6) の拡張子は「.doc」。
- 「ファイル名」= この実習では、拡張子はファイル名の一部とする。

例えば図 1 の (6) のファイル名は「file1.doc」であり、file1 の部分は、この実習では「ファイル名の拡張子以外の部分」と呼ぶ。

[**注意**: エクスプローラではデフォルトで拡張子は非表示の設定になっている。エクスプローラで、[整理]→[フォルダーと検索のオプション]→[表示] と進んで [登録されている拡張子は表示しない] のチェックを外せば拡張子も表示できるようになる。]

MS–Windows では、例えば図 1 の (4) のディレクリや (6)、(10) のファイルの場所を、それぞれ以下のように表すことができる (**絶対パス指定** と呼ぶ):

(4) = C:¥Documents and Settings¥user
(6) = C:¥Documents and Settings¥user¥My Documents¥file1.doc
(10) = C:¥Windows¥FONTS¥Arial.ttf

例えば上の (6) は、「ドライブ C: の、¥Documents and Settings というディレクトリ内の、user というサブディレクトリ内の、My Documents というサブディレクトリ内の、file1.doc という名前のファイル」を意味する。

このように、[ドライブ名]:[ディレクトリ]... のような形式でファイルやディレクトリの場所を指定する文字列を **パス** と呼ぶ (「パス」= 通り道、道順)。パスでは、ディレクトリやファイルの区切り文字、およびトップディレクトリ (= 最上位のディレクトリ) を表すのに「¥」を使用する。

一方、カレントディレクトリが、例えば (5) の

```
C:¥Documents and Settings¥user ¥My Documents
```

である場合は、(4), (6), (10) の場所は、自分の居場所を起点としてそれぞれ

(4) = .. (..¥..¥user も同じものを指す)
(6) = file1.doc (.¥file1.doc も同じものを指す)
(10) = ..¥..¥..¥Windows¥FONTS¥Arial.ttf

のように表すこともできる (**相対パス指定** と呼ぶ)。

用語:

- 「パス」= ディレクトリやファイルの位置を示す文字列
- 「絶対パス」= 「[ドライブ名]:¥」から始まるパス (**フルパス** とも呼ぶ)
- 「相対パス」= カレントディレクトリからの位置を指定するパス

2 ディレクトリ操作コマンド

ディレクトリ関連コマンドを表 1 に紹介する。オプションの [dir] は具体的なディレクトリのパスを表すが、[ ] は実際にはつけずに実行する。

コマンド	説明
cd	カレントディレクトリ名を表示
cd [dir]	ディレクトリ [dir] へ移動
md [dir]	ディレクトリ [dir] を作成
rd [dir]	ディレクトリ [dir] を削除
rd /s [dir]	[dir] 内のファイルやディレクトリもすべて削除
dir [dir]	[dir] 内のファイル、ディレクトリの一覧表示
dir /w [dir]	一覧の簡易表示
dir /p [dir]	一覧の 1 画面ずつの表示
pushd [dir]	カレントディレクトリをスタックに保存して [dir] へ移動
popd	スタックからディレクトリを一つ取り出してそこへ移動

表 1: ディレクトリ関連コマンド

注意:

1. 表 1 の [dir] には、対象となるディレクトリの絶対パスか相対パスを指定するが、

```
Z:¥> cd C:¥Windows
Z:¥>   (← その結果)
```

   のように [dir] にカレントドライブではないパスを指定した場合、カレントドライブは変わらない。カレントドライブを変更するには、単に

```
Z:¥> E:
E:¥>   (← その結果)
```

   のようにドライブ名をコマンドのように実行する[2]。

2. dir コマンドで [dir] を省略した場合は、カレントディレクトリが対象となる。

3. cd, md, rd には、それぞれ chdir, mkdir, rmdir という長いコマンド名もある。

4. rd は、「ごみ箱」への移動ではなく本当に消し去るので、復帰はできない。

5. rd は、そのディレクトリがアクセス中の場合は削除できない。また、/s オプションなしでは [dir] にファイルかディレクトリがある場合は削除できない。

[2]本実習では、すべての作業を Z: ドライブで行うので、カレントドライブの変更はほぼ必要ないが、コマンドプロンプトで USB メモリのファイル、ディレクトリの操作を行う場合は必要になる。

6. pushd, popd は **ディレクトリスタック** という、先入れ、後出し形式のディレクトリ保存場所を使用する。例えば以下のようになる (> の左がカレントディレクトリ、右が実行コマンド)。

```
Z:¥hoge1¥hoge2> pushd ..¥hoge3¥hoge4
Z:¥hoge1¥hoge3¥hoge4> cd ..¥hoge5
Z:¥hoge1¥hoge3¥hoge5> popd
Z:¥hoge1¥hoge2>
```

1 行目の pushd で Z:¥hoge1¥hoge2 がスタックに保存され、3 行目の popd でそれが取り出されて、2 行目の cd とは無関係にそこにカレントディレクトリが移動 (復帰) しているのがわかる。

3 ファイル操作コマンド

ファイル関連コマンドを 表 2 に紹介する。オプションの [dir] 等はディレクトリのパス、[file] はファイルのパス、[name] 等はファイルかディレクトリのパスを指すが、実際には実行する際は [ ] はつけない。

コマンド	説明
copy [file1] [file2]	[file1] のコピー [file2] を作成
copy [file1] ... [fileN] [dir]	[file1], ..., [fileN] のコピーを [dir] に作成
copy [file1]+...+[fileN] [file]	[file1], ..., [fileN] を連結した [file] を作成
copy nul [file]	0 バイト (空) の [file] を作成
del [file1] ... [fileN]	[file1], ..., [fileN] を消去
ren [name1] [name2]	[name1] の名前を [name2] に変更
move [file1] [file2]	[file1] を [file2] に移動
move [name] [dir]	[name] を [dir] 内に移動

表 2: ファイル関連コマンド

注意:

1. 「コピー」は「複製」を意味し、元のファイルを変更しないが、「移動」は元のファイルが元の場所からなくなることを意味する。

2. copy [file1] ... [fileN] [dir] は、元のものと同じ名前のコピーを [dir] 内に作成し、名前は変更しない。

3. nul は通常のファイルではなく、**ヌルデバイス** という特殊ファイル。

4. del は、ごみ箱への移動ではなく本当に消し去るので、復帰はできない。

5. 表 2 の [dir], [file], [name] 等にはパスを指定するが、ren の [name2] には名前部分だけを書き、¥ が含まれるパスは指定できない ([name1] の方は指定できる)。

6. move の [file2] にはパスも書けるが、これは元の [file1] を削除する。よって、同じディレクトリ内の move は ren と同じで、他のディレクトリへの move は copy した後で元のファイルを del したのと同じになる。

   例えば、図 1 の (8) のディレクトリで、

   ```
   C:¥Windows> move FONTS¥Arial.ttf FONTS¥hoge
   ```

   とすれば (10) のファイルが hoge という名前に変わり、さらに (8) で

   ```
   C:¥Windows> move FONTS¥hoge "..¥Program Files"
   ```

   とすればその hoge というファイルが (11) の中に移動する。

   なお、パスに空白が含まれるときは、この最後の例のようにパスを引用符 " で囲む必要がある (TAB 補完を使うと自動的に囲んでくれる)。

4 別ディレクトリ内のコマンドの実行

カレントディレクトリが例えば Z:¥ である場合、バッチファイル (例えば test1.bat) を自分が作成したディレクトリ (例えば Z:¥comp3-bat) の中に保存すると、それを

```
Z:¥> test1.bat
```

と実行しようとしても、コマンドプロンプトはそのバッチファイルを見つけられないため実行できない (エラーになる)。その場合、そのバッチファイルを実行するには、以下のようないくつかの方法がある。

1. カレントディレクトリを、そのバッチファイルがあるディレクトリに移動してから実行する

   ```
   Z:¥> cd comp3-bat
   Z:¥comp3-bat> test1.bat
   ```

2. そのバッチファイルのパスを指定して実行する

   ```
   Z:¥> comp3-bat¥test1.bat
   ```

3. コマンドプロンプトにコマンドの置き場所を教えてやる (上級)

   ```
   Z:¥> PATH=%PATH%;Z:¥comp3-bat   (← 一度やればよい)
   Z:¥> test1.bat
   ```

最後のものは、環境変数 PATH の変更という作業を行っている (環境変数については第 3 回で説明)。実は、カレントディレクトリにはない notepad.exe などが上の 1. や 2. の方法でなくてもファイル名だけで実行できるのは、notepad.exe などの置き場所が環境変数 PATH に既に設定されているからである。逆に、PATH に設定されていないディレクトリ内にあるコマンドは、上の 1. か 2. の方法でないと実行できない。

コラム: コマンドプロンプトの便利な利用法

コマンドプロンプトは、通常のエディタ画面などとやや勝手が違うが、便利な利用法をいくつか紹介する。

- 日本語入力

  Windows XP 等では、単に「半角/全角」や「変換」キーなどを押しても日本語入力はできなかったが、Windows 7 では単に「半角/全角」や「変換」で日本語を入力可能。

- 別ウィンドウからの貼り付け (ペースト)

  エディタなど、他のウィンドウでコピーした文字列は、コマンドプロンプト画面上で右クリックして現れるメニューの「貼り付け」を選択すれば、コマンドプロンプトに貼り付けられる。

- 別ウィンドウへのコピー

  逆に、コマンドプロンプトに表示されている文字列を他のウィンドウにコピーする場合は、

  1. まずコマンドプロンプト画面上で右クリックして現れるメニューの「範囲指定」を選択する。
  2. コピーしたい文字列をドラッグして選択する (反転表示される)。
  3. コマンドプロンプトのタイトルバー (上部のバー) で右クリックして「編集」$\Longrightarrow$「コピー」と進む。

  とする。これでコピーされるので、任意のウィンドウで貼り付けできる。

  なお、「範囲指定」を選択するのが面倒ならば、以下のようにして「簡易編集モード」を有効にすれば、「範囲指定」を選択しなくても常時ドラッグで選択することができるようになる。

  1. コマンドプロンプトのタイトルバーで右クリックして「プロパティ」を選択。
  2. 「オプション」タブの「編集オプション」の項目にある「簡易編集モード」を有効にする。

3. 「OK」をクリック。

- 入力文字列の編集機能

  コマンドプロンプトで入力した文字列の編集 (訂正/追加) は、基本的には左右の矢印キーでカーソルを移動して行えるが、他にも以下のような編集操作が利用できる。

キー操作	意味
Ctrl + 左矢印	単語単位でカーソルを左へ移動
Ctrl + 右矢印	単語単位でカーソルを右へ移動
Home	カーソルを行頭へ移動
End	カーソルを行末へ移動
Ctrl + Home	行頭からカーソル位置まで削除
Ctrl + End	カーソル位置から行末まで削除
Esc	その行の入力を全部削除

- コマンド履歴

  前にも説明した通り、上下の矢印キーで過去に入力したコマンドの履歴 (ヒストリ) をたどることができるが、F7 キーでコマンドの履歴の一覧を表示させることもでき、その上で上下の矢印キーで選択することができる。

  コマンド履歴 (ヒストリ) に取っておけるコマンドの数などは、コマンドプロンプトの「プロパティ」 (タイトルバーで右クリック) の「オプション」タブで設定できる。

ただ、コマンドプロンプトでの編集機能は、エディタでの編集機能に比べればやはり劣るので、面倒ならばエディタでバッチファイルを作って、それをコマンドプロンプトで実行するのが効率的であろう。