2012 年 04 月 20 日

計算機実習 IV (2012 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

目 次

1 動的変数と変数の演算

環境変数には、以下の動的な環境変数もある。これは、単なる `set` コマンドでは表示されないし、ユーザが設定することもできず、参照の度に値が自動的に設定される。

- `%cd%` : ドライブ名付きのカレントディレクトリ
- `%date%` : 現在の日付 (例: 2010/06/26)
- `%time%` : 現在の時刻 (例: 16:48:54.26)
- `%random%` : 0 から 32767 の間の乱数
- `%errorlevel%` : 直前のコマンドの終了コード

変数値の整数計算は `set /a` によって行うことができる。

```
Z:> set /a [変数名]=[数式文字列]
```

とすると、右辺の数式の計算結果 (整数演算) の数値を文字列としたものが環境変数に代入される。式には、C 言語などと同様の以下のものが利用できる:

- 整数、および整数値を持つ変数名 (`%` で囲まなくてよい)
- カッコ: ( )
- 四則演算: `*`, `/`, `+`, `-`, `%` (`%` は剰余)

- ビット演算: <<, >>, &, ^, |
- 演算つき代入: *=, /=, +=, -=, %=, <<=, >>=, &=, ^=, |=

例えば、

```
Z:> set x=10
Z:> set y=4
Z:> set /a y+=3*x%7
Z:> echo %y%
```

とすると、4 に 3×10 を 7 で割った余り (=2) を加えた 6 が表示される。

```
Z:> set /a r=%random%%3
Z:> echo %r%
```

とすれば、0 か 1 か 2 がほぼ均等の割合でランダムに出力されることになる。なお、% はバッチファイル内では特別な意味を持つので、剰余計算の % をバッチファイル内で使う場合は、%% と書く必要がある[1]。

なお、set で変数に小数値を「文字列」として設定することはできるが、set /a では小数計算は行えない。

課題 2-1. 1 つ目と 2 つ目のオプションの値の和をコマンドプロンプト画面に出力するバッチファイル kadai2-1.bat を作成せよ。

課題 2-2. 1 つ目と 2 つ目のオプションの値の積をコマンドプロンプト画面に出力するバッチファイル kadai2-2.bat を作成せよ。

課題 2-3. 動的変数 %date% を用いて「今日の日付は [日付] です。」と表示するバッチファイル kadai2-3.bat を作成せよ ([日付] の部分に実際に今日の日付が出力されるように)。

課題 2-4. 変数 x と変数 y に適当に整数を設定し、$z = 3x(x + 4y)$ の値を set /a による数式を用いて計算させ、その結果を表示させてみよ。

課題 2-5. 動的変数 %random% を用いて、実行するたびに 0 か 1 を半々の確率でコマンドプロンプト画面に出力するバッチファイル kadai2-5.bat を作成せよ。

[1] 例えば、バッチファイル内では set /a r=%random%%%3 のように書く。

課題 2-6. 動的変数 %random% を用いて、実行するたびに 1 から 6 までの数字を等分の確率でコマンドプロンプト画面に出力するバッチファイル kadai2-6.bat を作成せよ。

課題 2-7. 動的変数 %random% を用いて、実行するたびに 0 か 1 を、0 が 7 割、1 が 3 割の確率でコマンドプロンプト画面に出力するバッチファイル kadai2-7.bat を作成せよ。

課題 2-8. 0 から 9 までの乱数を 3 回作ることで小数 3 桁の 0.000 から 0.999 までの範囲の乱数を画面に表示するバッチファイル kadai2-8.bat を作成せよ。

2 ファイル/ディレクトリ操作コマンド

ディレクトリやパスに関する基本的な事柄を以下に列挙する。

- ディレクトリ = フォルダ
- この講義では、原則ファイル名、ディレクトリ名には日本語 (全角文字) は使わない。
- カレントディレクトリ = 現在いるディレクトリ。ピリオド 1 つ (.) で表す。
- 親ディレクトリ = 一つ上のディレクトリ。ピリオド 2 つ (..) で表す。例えば、カレントディレクトリが

  ```
  C:¥Document and Settings¥hoge¥My Documents
  ```

  の場合は、.. は

  ```
  C:¥Document and Settings¥hoge
  ```

  を意味し、..¥.. は

  ```
  C:¥Document and Settings
  ```

  を意味する (¥はディレクトリ区切り文字)。
- ディレクトリやファイルの位置を示す文字列を **パス** というが、上の例のようにドライブ名 (C:) と一番上のディレクトリ (¥) から始まるパスを **絶対パス**、..¥test¥file.jpg のように、カレントディレクトリから指定するパスのことを **相対パス** という。

ディレクトリ関連コマンドを以下に紹介する。

- `cd [dir]` (または `chdir [dir]`)
  ディレクトリ [dir] へ移動する。[dir] を省略した場合は移動はせず、カレントディレクトリ名を表示する。

- `md [dir]` (または `mkdir [dir]`)
  ディレクトリ [dir] を作成する。

- `rd [dir]` (または `rmdir [dir]`)
  ディレクトリ [dir] を削除する。[dir] 内にファイルやディレクトリが残っている場合は削除できないが、`/s` オプションをつけるとその中のファイルやディレクトリも含めてすべて削除する[2]。

- `dir [dir]`
  ディレクトリ [dir] 内のファイル、ディレクトリの一覧を表示。[dir] を省略した場合は `dir .` と同じ。オプション `/w` は簡易表示。`/p` は 1 画面ずつ表示。

- `pushd [dir]`/`popd`
  `pushd [dir]` はカレントディレクトリをディレクトリスタックに保存して [dir] へ移動する。`popd` はディレクトリスタックからディレクトリを一つ取り出してそこへ戻る。例えば以下のようになる (`>` の左はカレントディレクトリの絶対パス)。

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

ファイル関連コマンドを以下に紹介する。

- `copy [file1] [file2]`
  ファイル [file1] のコピー (複製) を [file2] という名前で作成する

- `copy [file1] [file2] ... [fileN] [dir]`
  ファイル [file1], [file2], . . . , [fileN] のコピーを、ディレクトリ [dir] 内に作成する (名前は変更しない)

- `copy [file1]+[file2]+ ...+[fileN] [tofile]`
  ファイル [file1], [file2], . . . , [fileN] の内容をすべて連結した結果を [tofile] の名前で作成する

[2] よって `rmdir /s` の実行には十分慎重に。

- `copy nul [file]`
  中身が空 (0 バイト) の [file] を作成する (`nul` はヌルデバイス)
- `del [file1] [file2] ... [fileN]`
  ファイル [file1], [file2], ..., [fileN] を消去する (ごみ箱に入れるのではなく本当に消去する)
- `ren [name1] [name2]`
  ファイル (またはディレクトリ) [name1] の名前を [name2] に変更する ([name2] には ¥の含まれるパスは書けない)
- `move [file1] [file2]`
  ファイル [file1] を [file2] に移動する ([file2] にはパスも書ける)。同じディレクトリ内のファイルの移動は `ren` と同じ意味になり、他のディレクトリへのファイルの移動は `copy` した後で元のファイルを `del` したのと同じ意味になる。
- `move [name] [dir]`
  ファイル (またはディレクトリ) [name] を [dir] 内に移動する。

課題 2-9. テスト用のディレクトリ test を作成し、`cd` でそちらに移動して `dir` を実行して、また元の場所に戻ってみよ。

課題 2-10. test の中に空ファイル test1.jpg を作成し、そのファイル名を test1.gif に変更してみよ。

課題 2-11. test の中で、test1.gif のコピー (複製) test2.gif, test3.gif を作成せよ。

課題 2-12. test の中にディレクトリ test2 を作成し、`move` コマンドを使って test1.gif, test2.gif, test3.gif を全部 test¥test2 に移動させよ。`move` コマンド 1 回だけでできるかどうかも考えよ。

課題 2-13. test¥test2 内のファイル test1.gif, test2.gif, test3.gif を、`del` コマンドを使って全部消去せよ。`del` コマンド 1 回だけでできるかどうかも考えよ。

課題 2-14. test¥test2 の中にディレクトリ test3 を作成し、
`pushd test¥test2` と `pushd test3` との実行後に `popd` を 2 回実行し、それぞれどこに戻るか確認せよ。

課題 2-15. test 内のディレクトリ test2 を、その中にあるファイルやディレクトリとともに削除せよ。

3 for 文

バッチファイルで繰り返しの作業を行うのに、C 言語と同様 for 文が使える。for 文は、主に以下の形式で使用する。

1. 例示形式のリストに対して実行

```
for %[変数名] in (リスト]) do (
    [コマンド]
    ..........
)
```

2. 単調な整数のリストに対して実行 (C 言語の for 文と同様)

```
for /l %[変数名] in ([初期値],[増分],[終了値]) do (
    [コマンド]
    ..........
)
```

3. ファイルの各行に書かれたリストに対して実行

```
for /f %[変数名] in ([ファイル]) do (
    [コマンド]
    ..........
)
```

実行するコマンドが 1 つしない場合は、1 行で

```
for %[変数名] in ([リスト]) do [コマンド]
for %[変数名] in ([リスト]) do ([コマンド])
```

のように書くことも可能。

3. の `for /f` の形式では、ファイルは複数指定できるし、ファイルの代わりに ‘[コマンド]‘ のようにコマンド名を逆クウォートで囲むことでコマンドの出力をリストとすることもできるし、そのファイルのどの部分を使用するか、などの細かい制御も可能であるが、その詳細は省略し (詳しくは help for 参照)、ここでは最初の 2 つの形式 (1., 2.) を中心に説明する。

for 文の局所変数 %[変数名] の変数名はアルファベット 1 文字である。また、バッチファイル内で使うときは、%[変数名] は %%[変数名] と % を重ねる必

要がある。この局所変数は、値を参照するときも、set /a の右辺に書く場合でも %[変数名] (バッチファイル内では %%[変数名]) として使用する。

上の 1. の形式の場合は、リストには複数の文字列を書き並べ、それが各変数に代入されてその分だけコマンドが実行される。リストの各要素の区切りは、スペースかカンマ (,) を使用する。例えば

```
Z:> for %a in (dog cat mouse) do echo %a
```

とすると、%a には順番に dog, cat, mouse という文字列が代入されて、そのそれぞれに対してコマンド「echo %a」が実行され、結果として

```
dog
cat
mouse
```

と表示される。

上の 2. の形式は、C 言語の for 文とは増分指定 (C では 3 つ目)、終了条件 (C では 2 つ目) の指定の順番が違う点を除けばほぼ同様。例えば

```
Z:> set x=0
Z:> for /l %a in (1,2,100) do set /a x=x+%a
Z:> echo %x%
```

とすると、100 以下の正の奇数全体の和 (=2500) が表示される。

増分や整数値には負の値も使用でき、例えば、

```
Z:> for /l %a in (9,-2,-4) do echo %a
```

とすると、%a には順に 9,7,5,3,1,−1,−3 が代入され、do の後ろのコマンドが実行される。

注意: for 文の中で (局所変数ではない) 環境変数を使用する場合、デフォルトでは % で囲んだ展開 (変数から変数値への置き換え) は逐次行われるのではなく、for 文の「実行前に行われてしまう」ので、予期しない結果を生むことがある。例えば、

```
Z:> set s=abc
Z:> for %a in (dog cat mouse) do set s=%s% %a
Z:> echo %s%
```

とすると、for 文内では %s% の値は常に for 文前の abc という文字列なので、for 文の部分は

```
Z:> set s=abc dog
Z:> set s=abc cat
Z:> set s=abc mouse
```

をやったのと同じことになり、結局 s には「abc mouse」が代入されることになる。この問題を解消するには、コマンドプロンプトの遅延展開機能オプションを用いるか[3]、後で説明する if と goto によるループに書き換えればよい。

なお、前の例の「set /a x=x+%a」の場合は、右辺の x には % を使っていないので、値は正しく更新される。

課題 2-16. for /l を用いて、1 から 20 まで表示するバッチファイル kadai2-16.bat を作成せよ。

課題 2-17. for /l を用いて、1 から 50 までの和を表示するバッチファイル kadai2-17.bat を作成せよ。

課題 2-18. for を用いて、「月」から「日」までの曜日を表示するバッチファイル kadai2-18.bat を作成せよ。

課題 2-19. for /l を用いて、test ディレクトリ内に file1.jpg, file2.jpg, . . . , file20.jpg の 20 個の空ファイルを作成するバッチファイル kadai2-19.bat を作成し、それを実行せよ。

課題 2-20. for /l %%a in (1,1,10) do の for 文の実行部分で、「set /a x+=%%a」と「set /a y=y+%%a」と「set /a z=%z%+%%a」とした場合ではどのような違いがでるか確認せよ。

バッチファイル中で、%* を for 文のリスト指定に使用すれば、そのバッチファイルの実行時に指定したすべてのオプションに対する処理を行うことができる。例えば、

```
@echo off
for %%a in ( %* ) do (
    echo [%%a]
)
```

[3]コマンドプロンプトを cmd /v:on で起動し、遅延展開する変数を % でなく ! で囲む。詳しくは help cmd 参照。

というバッチファイルは、指定されたオプションを一つ一つ [ ] で囲んで表示する。

課題 2-21. test2 ディレクトリを作成し、`for %%a in ( %* )` を用いて次のようなバッチファイル kadai2-21.bat を作成せよ:

> オプションを 1 つ以上受けとって、それぞれをファイル名とみなし、それを test2 ディレクトリ内に移動し、「ファイル [ファイル名] を test2 に移動しました」と表示する

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

  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いが、「`Alt + 半角/全角`」で日本語入力モード (右下に「全あ連ローマ」などと表示される) になる。元の半角文字入力に戻すには、再び「`Alt + 半角/全角`」とする。

- 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「選択」 $\Longrightarrow$ 「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。

  - 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。