2016 年 05 月 06 日

計算機実習 III (2016 年度)
第 3 回: コマンドプロンプトとバッチファイル その 3

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

目 次

1 バッチファイルの変数

コマンドプロンプト、バッチファイルでは、**環境変数** と呼ばれる変数を使うことができる。環境変数の一覧表示、設定、削除には、set コマンドを使用する。

操作	コマンド	説明
表示	set	現在設定されている変数の一覧を表示
設定	set [変数名]=[値]	[変数名] の環境変数に [値] を設定
利用	%[変数名]%	その部分がその変数の値 (文字列) に置き換わる
削除	set [変数名]=	その環境変数を削除

表 1: 環境変数の操作

C 言語とは違い、変数の宣言は不要で、変数の値は基本的に「文字列」である。例:

```
@echo off
set a=竹の
set b=茂治
set c=%a% %b%
echo %c%
```

このバッチファイルを実行すると、変数 c の値である「竹の 茂治」という文字列が echo コマンドにより出力される。

注意:

- C 言語とは違い、文字列は引用符で囲まない。逆に引用符をつけると引用符も値の一部になる。
- set コマンドの 「=」の前後にスペースを入れてはいけない。スペースを入れると、変数名や値文字列にスペースが含まれる。例えば「set a =3」とすると、変数名が「a 」(a + 空白) となってしまう (意味を知った上でやるのは構わない)。
- 変数名の大文字小文字は区別されず、例えば str も STR も sTr も同じ一つの変数を指す。
- 変数名には数字も使え、C 言語とは違い「100」のような数字のみの名前の環境変数も作れる (推奨はしない)。
- バッチファイルで設定した環境変数は、バッチファイル内で削除しなければバッチファイル終了後も残る。例えば、

```
@echo off
set str=%str%Y
echo %str%
```

  というバッチファイルは、実行する度に str という環境変数の最後に文字「Y」を追加し続けるので、出力結果は実行の度に長くなっていく。

- 環境変数は、例えば PATH のようにあらかじめ設定されていてコンピュータの動作に影響を与えるものもあるので、set による一覧表示を確認した上で、定義済みの変数は使わないようにすること (意味を知った上でやるのは構わない)。
- バッチファイル内 (コマンドプロンプトではなく) で、echo 等で % という文字自体を表示させるには、%% とする。

変数の値は、%[変数名]% のように変数名を % で囲んで利用する。実際には、コマンドプロンプトやバッチファイルで %[変数名]% を含む行を使った場合、その行の実行前にその部分が変数値へ置き換えられてからその行が実行される。例:

```
@echo off
set st=start
set site=www.niit.ac.jp
%st% http://%site%
```

このバッチファイルの最後の行は、「start http://www.niit.ac.jp」という文字列に置換された後で実行される。もう一つの例:

```
@echo off
set n=1
set s%n%=hoge
echo %s1%
```

この 3 行目は、その実行前に「set s1=hoge」に置換が行われるので正しく s1 という名前の変数が設定され、その値が 4 行目で表示される。なおこの場合、逆に展開する方 (4 行目) で「echo %s%n%%」のように 2 重に % を使うことはできない (後で説明する遅延展開を利用すればできなくはない)。

2 動的環境変数

環境変数には、表 2 に示す、値が自動的に設定される **動的環境変数** がある。これは、単なる set コマンドでの一覧には表示されないし、ユーザが設定することもできず、参照できるだけだが、参照する際に値が自動的に設定される。これらは主にバッチファイルで利用する。

環境変数	値
%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

表 2: 動的環境変数

%random% はその式を呼び出す度に異なる乱数が自動的に設定される。C 言語とは違い、乱数列の初期化は必要ない。

また、「date /t」コマンドの出力と %date% の値は同じだが、「time /t」コマンドの出力と %time% の値は完全に同じではなく、「time /t」の出力は「[時]:[分]」だが、%time% の値は「[時]:[分]:[秒].[1/100 秒]」と、より精度の高い時刻を持っている。

3 環境変数の整数計算

環境変数値は基本的には文字列だが、set コマンドの /a オプションにより、整数値の簡単な計算を行うことができる。

```
Z:¥> set /a [変数名]=[数式文字列]
```

これは、右辺の数式の計算結果の数値 (整数値) を文字列としたものを左辺の環境変数に代入する。右辺の数式には、整数値、環境変数名 (この場合は % で囲む必要はない)、かっこ (丸かっこのみ) 以外に、C 言語と同様の表 3 の演算記号が使える。

単項演算子		
記号	意味	例など
!	!a = a が 0 以外なら 0、a が 0 なら 1	!11=0
~	~a = a のビット毎の反転	$-a-1$ と同じ
-	-a = $-a$ (符号の反転)	
四則演算		
記号	意味	例など
*	a*b = a と b の積	14*5 (=70)
/	a/b = a を b で割った商	14/5 (=2)
%	a%b = a を b で割った余り	14%5 (=4)
+	a+b = a と b の和	14+5 (=19)
-	a-b = a と b の差	14-5 (=9)
ビット演算 (引用符で囲む必要あり)		
記号	意味	例など
<<	"a<<b" = a を 2^b 倍	"14<<5" (=448)
>>	"a>>b" = a を 2^b で割った商	"14>>5" (=0)
&	"a&b" = a と b のビット毎の AND	"14&5" (=4)
|	"a|b" = a と b のビット毎の OR	"14|5" (=15)
^	"a^b" = a と b のビット毎の XOR	"14^5" (=11)

表 3: set /a の右辺で使える演算記号

整数値は通常は 10 進数であるが、先頭が 0x の場合は 16 進数、先頭が 0 の場合は 8 進数とみなされる。

また、表 3 の他に、set /a の後ろの代入を意味する等号 (=) の代わりに、以下の演算つき代入記号も利用できる:

*=, /=, +=, -=, %=, <<=, >>=, &=, ^=, |=

これらの意味も C 言語と同様であり、例えば「set /a x/=3」は「set /a x=x/3」と同じ意味になる。例:

```
@echo off
set x=10
set y=4
set /a y+=3 * x % 7
echo %y%
```

というバッチファイルは、4 に、$3 \times x$ (=$3 \times 10 = 30$) を 7 で割った余り (=2) を加えた 6 を表示する。

コマンドプロンプト上で set /a を使用すると、代入だけでなく右辺の値も表示されるので、例えば

```
Z:¥> set /a x=0x3f * 2
```

と実行すると、右辺の 16 進数 (0x3f×2 = $63 \times 2 = 126$) を 10 進数に変換した「126」が表示される。これは簡単な 16 進数電卓がわりになる (ただし、逆に 10 進数から 16 進数への変換は難しい)。同様に、

```
Z:¥> set /a x=0342 * 2
```

のようにして 8 進数から 10 進数への変換もできる。

注意:

- 上の例のように、set /a の右辺の数式には適宜スペースを入れても構わない。
- set /a では小数計算は行えない。
- 「set /a」の右辺で環境変数値を使う場合は、その変数名を % で囲む必要はない (囲んでも構わない)。ただし、%random% などの動的環境変数は set /a の右辺でも % で囲む必要がある。
- /a がなければ、右辺の数式がそのまま文字列として変数に設定される。逆に、set の右辺が数式ではない文字列の場合、/a があると計算に失敗し変数には 0 のような適当な整数値が代入されるので、単に文字列を代入する場合は /a は使わないこと。

%random% と set /a を組み合わせると、色々な範囲の乱数を利用することができる。例えば、

```
Z:¥> set /a r=%random% % 4 + 9
```

とすると、r にはほぼ均等の割合で 9 以上 12 以下のランダムな整数が代入される。それは、

- 整数を 4 で割った余り = 0 以上 3 以下の整数
- 「0,1,2,3,4,5,6,7,8,9,...」を 4 で割った余り=「0,1,2,3,0,1,2,3,0,1,...」
  (0,1,2,3 の繰り返しなのでほぼ均等の割合で現れる)

なので、

- %random% % 4 = ほぼ均等な確率の 0 以上 3 以下のランダムな整数
- %random% % 4 + 9 = ほぼ均等な確率の 9 以上 12 以下のランダムな整数

が得られることになる。

注意:

- 余り計算の % をバッチファイル内で使う場合は %% と重ねて書く必要がある (上の例はバッチファイルではなくコマンドプロンプト)。しかし、変数を囲む % の方は重ねない。例:

  ```
  @echo off
  set /a x=%random% %% 6 + 1
  echo %x%
  ```

  これは 1 から 6 までの整数をランダムに (ほぼ均等に) 出力する。

4 環境変数の文字列処理

環境変数の値文字列から文字列の一部分だけを取り出したり、一部分を変更した文字列を作ることもできる (表 4)。

式	意味
%var%	環境変数 var の文字列全体
%var:~[m]%	$[m]$ 文字目から最後までの部分文字列
%var:~[m],[n]%	$[m]$ 文字目から $[n]$ 文字分の部分文字列
%var:[s1]=[s2]%	var 内の [s1] をすべて [s2] に変更した文字列

表 4: 環境変数の文字列処理

[m] は、文字列の先頭を「0 番目」と数えることに注意。[m] に負の値を指定した場合は、文字列の先頭からではなく後ろから数える (一番後ろの文字が (−1) 番目)。また、[n] として負の値を指定すると、最後の [n] 文字を削除したものとなる。

例えば、環境変数 var が「1234567890123」という文字列の場合、それぞれ表 5 のようになる。

環境変数式 = 結果文字列	環境変数式 = 結果文字列
%var:~5% = 67890123	%var:~-5,2% = 90
%var:~5,2% = 67	%var:~-5,-2% = 901
%var:~5,-2% = 678901	%var:123=aBc% = aBc4567890aBc
%var:~-5% = 90123	%var:123=% = 4567890

表 5: 「%var%=1234567890123」の場合の例

なお、これらは % で囲んだ部分がそのような値になるだけで、元の環境変数の値は変わらない。よって、var の値はこれらの後でも 1234567890123 のままである。

これを使えば、例えば動的環境変数 %time% から現在時刻を数値として取得したり、2 つの時刻を比較してコマンドの実行時間の計測などを行うのに利用できる。例:

```
@echo off
set ts=%time%
set /a hr=1%ts:~0,2% - 100
set /a mn=1%ts:~3,2% - 100
set /a sc=1%ts:~6,2% - 100
echo 現在は %hr%時 %mn%分 %sc%秒です。
```

なお、この例では、あえて先頭に 1 をつけて 100 を引いているが、それは時、分、秒が「08」「09」という 2 文字の場合でも整数値として取得するための対策である。文字列として 2 文字を取得するだけならこのようにする必要はないが、「set /a」で整数値として取得する場合、「08」は先頭が 0 なので 8 進数とみなされるが、「08」という 8 進数はないのでエラーとなる。上のようにすれば「08」であっても「108 − 100」により正しく「8」という整数値が取得できる。

コラム: エディタ

プログラムソースファイルのようなテキストファイルを作成/修正 “だけ” 行うソフトを **エディタ** と呼ぶ。MS–Windows でいうと「メモ帳」(コマンド名は notepad) など

がそれに当たるが、MS–Word のようなワープロソフトが保存するファイル (.doc) はテキストファイルではないので、通常ワープロソフトはエディタとは呼ばない。

エディタは、色々な機能がついたソフトに比べて軽快であることが長所である。Visual-Studio のような統合環境ソフトや Thunderbird のようなメール作成ソフトにもエディタ機能がその一部として組み込まれているが、それぞれ別の性質を持つので、それらでそれぞれ長い文章を書くのはあまり適切とは思わない。むしろ、普段良く使うエディタを一つ決めておいて、文章を書く作業だけはそちらで行う方が効率はよいように思う。

さらにプログラマ向けには、エディタは、

1. プログラム言語のキーワードやコメント部分の色づけ機能
2. フォントサイズやフォントの種類のカスタマイズ
3. かっこの対応のチェック
4. 自動インデント機能
5. カーソル移動や編集機能のショートカットの充実
6. 全角空白 (日本語を使う場合)、改行、ファイル末尾の表示機能

などの機能を備えているのがより望ましい。しかし「メモ帳」にはこれらの機能はほとんどなく、少なくともプログラムを書くのには便利ではない。

フリーのエディタでこのような機能の多くを備えているものに Notepad++ や SciTE などがあるそうであるが、対応していないプログラム言語用の 1. のカスタマイズや 2. が容易でないこと、メニューやヘルプが英語であることなどから、本講義ではそれらではなく otbedit を選択した。otbedit は、完全ではないが上記のすべてをある程度は持っているし、本講義で使用するバッチファイルや AWK 用の設定は用意することができた。もちろん、普段自分で使っているエディタがあればそれを使ってもよい。

ソフトコースの学生は、今後も長いプログラムを書く機会もあるだろうから、良い機能を持つ、手に良くなじむエディタを一つ持っているといいだろう。