2012 年 04 月 27 日

計算機実習 IV (2012 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 ワイルドカード

複数のファイルを 1 度に指定する方法としてワイルドカードがある。**ワイルドカード** は、「?」と「*」の文字であるが、ファイル名の一部にこれらを使った場合、その部分を任意の文字列と見て、それに一致するすべてのファイルを対象にする、といった仕組みである。

- ? : 0 文字か任意の 1 文字にマッチ (適合) する
- * : 任意の長さの任意の文字列にマッチする

例えば、カレントディレクトリに

```
test.bat, test2.c, test2.exe, test3.bat, test4
```

のファイル (かディレクトリ) があったとすると、

- test?.bat : test.bat と test3.bat がマッチ
- test2.* : test2.c と test2.exe がマッチ
- test?.* : 上の 5 つのファイルすべてがマッチ

となる。なお、拡張子のない「test4」という名前は、「test4.」と同じとみなされることになっているので、test4 も最後のパターンにマッチする。

例えば

```
Z:> copy test2.* dir
```

とすると、これは

```
Z:> copy test2.c test2.exe dir
```

と実行したことになる。

* 自体はすべてのファイル名にマッチし、???.??? は [名前].[拡張子] の形式のファイル名で、名前の部分が 3 文字以下、拡張子の部分が 3 文字以下のファイル名にマッチする。

課題 3-1. a で始まり z で終わるファイル名すべてにマッチするワイルドカードパターンは何か。

課題 3-2. a で始まり z で終わる 6 文字以下のファイル名すべてにマッチするワイルドカードパターンは何か。

課題 3-3. AKB で始まり .jpg という拡張子を持つファイルを、move コマンド 1 回ですべて AKB–JPEG というディレクトリに移動するにはどうすればいいか。

for 文のリスト指定形式の場合、リストにワイルドカード (*, ?) を含む文字列を指定した場合は、ワイルドカードにマッチするファイル名を探してそれをリストの要素とする。

課題 3-4. for %%a in (test¥*.jpg) と ren を用いて、test ディレクトリ内のすべての .jpg の拡張子のファイルのファイル名の最後に .0 を追加するバッチファイル kadai3-4.bat を作成せよ。

2 if 文

if 文は、以下の 4 種類の形式が使用できる。

1. 1 行で if のみ

```
if [条件] [コマンド]
```

2. 1 行で if と else

```
if [条件] [コマンド 1] else [コマンド 2]
```

3. 複数行で if のみ

```
if [条件] (
    [コマンド 1]
    [コマンド 2]
    ...........
)
```

4. 複数行で if と else

```
if [条件] (
    [コマンド A1]
    [コマンド A2]
    ............
) else (
    [コマンド B1]
    [コマンド B2]
    ............
)
```

実行するコマンドが複数ある場合は 3. または 4. の形式を使うが、3., 4. の形式はコマンドが 1 つしかない場合でも使える。

if 文の条件部分には、以下の形式を使う。

- `[値 1]==[値 2]` : 等しければ真
- `[値 1] [比較演算子] [値 2]` : 文字列、数値の比較
- `exist [ファイル名]` : ファイルやディレクトリが存在すれば真
- `errorlevel [番号]` : 直前コマンド終了コードが番号以上なら真
- `defined [変数名]` : その環境変数が定義されていれば真

以上の条件の前に `not` をつけるといずれの場合も真偽が反転されるが、C 言語のように複数の条件を AND (`&&`) や OR (`||`) で結ぶことはできない[1]。

[1] よって、条件の AND や OR はバッチファイルでは if 文の繰り返しとジャンプなどによって実現するしかない。

2 番目の形式では、比較演算子はいずれも 3 文字の文字列で、

- `leq` : 以下 (≤ の意。**l**ess than or **eq**ual の略か)
- `geq` : 以上 (≥ の意。**g**reater than or **eq**ual の略か)
- `equ` : 等しい (= の意。**equ**al の略か)
- `neq` : 等しくない (≠ の意。**n**ot **eq**ual の略か)
- `lss` : より小さい (< の意。**l**e**ss** than の略か)
- `gtr` : より大きい (> の意。**g**rea**t**e**r** than の略か)

のいずれかが指定できる。

両辺の値がどちらも整数値を表す文字列であれば、整数値として比較が行われる。値が文字列ならば辞書式順序で比較され、デフォルトではアルファベットの大文字と小文字は区別される。辞書式順では、数字 (0–9) よりも大文字 (A–Z) が大きく、それよりも小文字 (a–z) の方が大きい[2]。なお、2 番目の比較演算子を用いた形式では `if` と条件の間に `/l` (エル) を置くと、アルファベットの大文字と小文字は区別せずに比較する。

例えば以下のバッチファイルは、ほぼ 4 割の確率で「ヒット」、6 割の確率で「内野ゴロ」と表示する:

```
@echo off
set /a x=%random% %% 10
if %x% lss 4 (
    echo ヒット
) else (
    echo 内野ゴロ
)
```

バッチファイルのオプションは `%1`, `%2`, ... 等で参照できるが、オプションが足りていないことも if 文で判定できる:

[2]基本的には、文字コード (ASCII コード) 順の比較を行う。

```
@echo off
if "%1"=="" (
    echo オプションが一つもない
) else (
    if "%2"=="" (
        echo オプションは 1 つだけ [%1]
    ) else (
        echo オプションは 2 つ以上ある [%1][%2]
    )
)
```

なお、== の左辺や右辺が空文字列にならないよう (なるとエラー)、ここではあえて " を用いていることに注意する。また、if–else をつなげたい場合は、C 言語のように「`else if (...)`」とは書けないので、この例のように else ブロックの中に新たに if 文を書く。

バッチファイルを途中で中断するにはコマンド「`exit /b`」を用いる。なお、オプション /b をつけないとコマンドプロンプト自体終了してしまう。これを使えば、上のバッチファイルは else を使わずに以下のように書くこともできる:

```
@echo off
if "%1"=="" (
    echo オプションが一つもない
    exit /b
)
if "%2"=="" (
    echo オプションは 1 つだけ [%1]
    exit /b
)
echo オプションは 2 つ以上ある [%1][%2]
```

この方法なら、if 文の入れ子を増やさずにすむ。

課題 3-5. 半々の確率で「明日は晴れでしょう」か「明日は雨でしょう」という文字列を表示するバッチファイル kadai3-5.bat を作成せよ。

課題 3-6. 1 つ目と 2 つ目のオプションを整数値とみて、その差の絶対値を表示するバッチファイル kadai3-6.bat を作成せよ。

課題 3-7. 1 つ目のオプションを整数値とみて、それが 10 以上でかつ偶数のときだけ表示するバッチファイル kadai3-7.bat を作成せよ。

課題 3-8. 1 つ目のオプションを整数値とみて、それが 10 未満かまたは奇数のときに表示するバッチファイル kadai3-8.bat を作成せよ。

課題 3-9. オプションが 1 つだけならその値を、オプションが 2 つならその和の値を、オプションが 3 つならそれらの和の値を表示するバッチファイル kadai3-9.bat を作成せよ。

課題 3-10. 1 つのオプションをファイル名とみて、そのファイルがあればそれをメモ帳 (notepad) で開き、なければ「ファイル [ファイル名] はありません」と表示するバッチファイル kadai3-10.bat を作成せよ。なお、メモ帳で特定のファイルを開くには、ファイル名をオプションとして与えて「notepad [ファイル名]」のように実行する。

課題 3-11. for %%a in (%*) を用いて次のようなバッチファイル kadai3-11.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、そのファイルが存在すれば「ファイル [ファイル名] が存在します」と表示し、そのファイルがなければ「ファイル [ファイル名] はありません」と表示する

3 goto 命令

goto 命令は、コマンドの実行をバッチファイル内の任意の箇所へジャンプさせるもので、

```
goto [ラベル名]
```

の形式で用いる。ジャンプ先のラベルは、

```
:[ラベル名]
```

の形の行であり、任意の位置に置くことができる[3]。

[3]なお、ラベル自体は何もコマンドを実行しないから、これを逆用して、コメント行を書くのに rem の代わりに行頭に「::」や「: 」(コロン + スペース) を書く、という裏技もある。

goto 命令と if 命令の組で簡単なループを実現できる。例えば

```
set j=1
:start1
if %j% gtr 10 goto last1
[コマンド 1]
...........
set /a j+=2
goto start1
:last1
```

とすれば、これは [コマンド 1] から `set /a j+=2` の手前までの行を、1,3,5,7,9 の `j` の値に対して行うループになっていて、`for /l %%j in (1,2,10)` とほぼ同等であるが、if によるループは for 文とは違い、環境変数が先に展開されてしまうことはない。

なお、この goto 命令を使うと簡単に無限ループが作れるが、無限ループとなったバッチファイルの実行は、コマンドプロンプトでは `Ctrl-C` で終了できる。

課題 3-12. `if` と `goto` を用いて、1 から 20 までを順に表示するバッチファイル kadai3-12.bat を作成せよ。

課題 3-13. 「バッチファイルなんていやだ」という行を無限に出し続けるバッチファイル kadai3-13.bat を作成せよ。

課題 3-14. 0 から 9 までの整数をランダムに 10 回表示するバッチファイル kadai3-14.bat を作成せよ。

課題 3-15. `if` と `goto` 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 kadai3-15.bat を作成せよ。

課題 3-16. `if` と `goto` を用いて、C 言語の while 文, do–while 文と同様のことをやるにはどうしたらいいか考察せよ。

課題 3-17. `if` と `goto` を用いて、C 言語の switch–case 文と同様のことをやるにはどうしたらいいか考察せよ。

バッチの流れの制御用のコマンドとして、pause や set /p のように一時停止のためのコマンドもある。

- pause
  そこでバッチを一旦停止し、「続行するには何かキーを押してください」というメッセージを表示して入力待ちになり、何かキーを入力すると pause を終了して次へ進む。

- set /p [変数名]=[表示メッセージ]
  そこでバッチを一旦停止し、[表示メッセージ] を表示して入力待ちになり、何か入力して enter キーを打つと、入力したものを変数に代入して次へ進む。

課題 3-18. set /p を用いて整数値を受けとり、それを 100 倍した整数を表示するバッチファイル kadai3-18.bat を作成せよ。

課題 3-19. バッチファイルを実行すると、2 度入力待ちになり、そこで入力した 2 つの整数の和を表示するバッチファイル kadai3-19.bat を作成せよ。

課題 3-20. バッチファイルを実行すると、「Y か N かを入力してください」と表示し入力待ちになり、ユーザの入力が Y か y ならば「終了します」と表示してバッチファイルを終了し、入力がそれ以外ならば再び最初の入力待ちに戻るようなバッチファイル kadai3-20.bat を作成せよ。

課題 3-21. 変数 x に 0 から 9 までの整数をランダムに設定し、その後ユーザのキー入力がその値に一致するまでユーザの入力待ちを繰り返すバッチファイル kadai3-21.bat を作成せよ。

課題 3-22. バッチファイルのオプションとして指定したファイルが存在したら、次のこと (ファイルのバックアップ) を行うバッチファイル kadai3-22.bat を作成せよ:

> 指定したファイルに対して、そのファイル名の後ろに .n (n は 0 以上の整数) をつけた [ファイル名].n という名前のファイルにコピーする。なお、n は、[ファイル名].n という名前のファイルが存在しないような一番小さい n を取る。例えば、[ファイル名].0, [ファイル名].1 というファイルが既に存在したら、[ファイル名].2 というファイルにコピーする。

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。

BCPad や VisualStudio のような統合環境ソフトや AL–Mail のようなメール作成ソフトにもエディタがその一部として組み込まれているが、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専用形式のバイナリファイルなので、MS–Word を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最大の長所であるが、文章を書くといったような最も仕事量の多い作業には使い慣れたソフトを使うのが正しいだろう。例えば、メーラソフトやプログラムの統合開発環境を使う人は、メールを書くときとプログラムソースを書くときで別なエディタを使うことになるが、もしその両者でショートカットキーのキー割り当てが違っていたら、使いやすさや作業効率は著しく落ちるだろう。例えば、ファイルを保存するためのショートカットキーとファイルを破棄するためのショートカットキーがその両者で丁度逆に割り当てられていたりしたらどんなに悲惨か想像することは難しくない。

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

といった機能を備えているとよい。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。

このような機能の多くを備えているフリーのエディタに Notepad++ や SciTE などがあるそうであるが、用意されていないプログラム言語に対する 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、本講義ではフリーの otbedit を選択した。otbedit は上記のすべてを完全に備えているとはいいがたいが、ある程度は持っているし、本講義で使用するバッチ、AWK、gnuplot 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。ちなみに私は普段は emacs (MS–Windows 用のものは Meadow) という Unix 上最も有名でとても便利なエディタを使っているが、かなり癖もあるし、MS–Windows 上ではその良さがわかるようになるとはあまり思えないので、決しておすすめのソフトではない。しかし、プログラムなどを書く機会のある人は、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。