2016 年 04 月 15 日

計算機実習 III (2016 年度)
第 1 回: コマンドプロンプトとバッチファイル その 1

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

目 次

1 コマンドプロンプトの使い方 1
2 otbedit の使い方 4
3 バッチファイルの実行 5
コラム: MS–Windows のスクリプト環境 7

1 コマンドプロンプトの使い方

コマンドプロンプト は主に文字をキーボードで入力することでコンピュータを操作する形式のコマンド実行環境 (「コマンド」= 命令、「プロンプト」= 入力待ち)。

コマンドプロンプトを起動するには、この計算機実習 III では Windows 7 の画面の「計算機実習 IV」のアイコン[1]をクリックする。それ以外の方法でコマンドプロンプトを起動した場合は、この実習で必要なソフトが使えない。

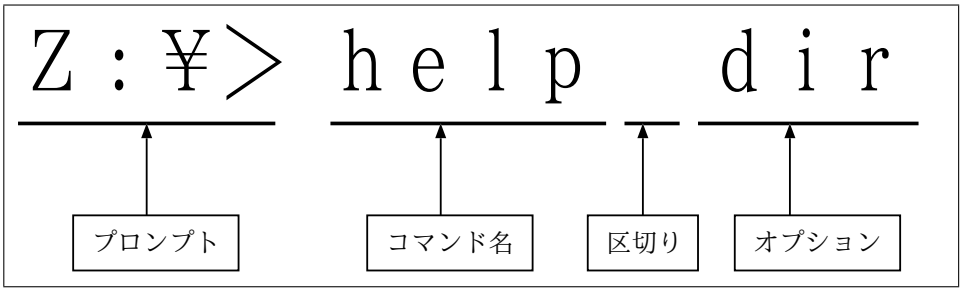
コマンドプロンプト上に表示される「Z:¥>」がプロンプトで、その右側にコンピュータへのコマンド (命令) をキーボードで入力し、Enter を押すことでそのコマンドを実行する。例[2]:

```
Z:¥> help dir
Z:¥> dir /w "C:¥Program Files"
```

補足: 「Z:¥>」のうち 「>」は **プロンプト記号**、その左の「Z:¥」は **カレントディレクトリ** (= 今自分がいることになっているディレクトリ)。ディレクトリ等については第 2 回で説明する。

[1]「計算機実習 III」でないのは、このソフトを入れたときはこの科目が「実習 IV」だったため。
[2]以後、このように二重枠で示すのは、コマンドプロンプトでの実行例であるとする。

この例のように、コマンドの後ろにスペースを開けて書くコマンド用の追加指示を**オプション**、**オプションパラメータ** などと呼ぶ。オプションは、原則スペースで区切って書き並べる。



- 「コマンド名」= ソフトウェアの実体
- 「オプション」= そのソフトウェアへの追加指示

コマンドプロンプトで使える簡単なコマンド (+ オプション) の例を表 1 にあげる。

コマンド	説明
`dir`	ディレクトリの一覧の表示
`cls`	コマンドプロンプト画面のクリア
`help [コマンド名]`	コマンドの説明の表示
`date /t`	今日の日付の表示
`time /t`	現在の時刻の表示
`echo [文字列]`	[文字列] をコマンドプロンプト画面に表示
`color [色指定]`	コマンドプロンプト画面の色指定
`start [コマンド]`	[コマンド] を別ウィンドウで実行
`pause`	Enter キー入力待ち
`rem [文字列]`	何もしない

表 1: 基本コマンド

注意:

- 表 1 のように、ユーザが変更できるオプション部分は、[文字列] のようにその種類を [ ] を囲んで記すこととするが、これは説明用の記法であり、実際に実行する際はオプションに [ ] はつけてはいけない。
- `echo` コマンドは、通常はオプションとして指定した文字列を表示するが、その [文字列] が ON か OFF (小文字でもよい) という文字列の場合はエコーモードの切り替えを行い、それ以降のコマンド行自体の表示設定を ON または OFF にする。この `echo on`、`echo off` は主にバッチファイルで使用する。

- echo コマンドに指定する文字列は、C 言語とは違い引用符では囲まない。途中に空白が含まれていても引用符はつけない。逆に引用符をつけると、引用符も一緒に表示される。

- echo コマンドでオプションをつけないと、現在のエコーモードの状態を表示する。単なる空行を表示させるには、echo. とする (スペースを空けずにピリオドをつける。主にバッチファイルで使用)。

- start コマンドのオプションとして URL や mailto:[メールアドレス] やファイル名を指定すると、それぞれ規定のブラウザ、規定のメーラ、ファイルに関連づけられたアプリケーションを別ウィンドウで起動してそれらを開く。例えば、

```
Z:¥> start time /t
Z:¥> start http://www.niit.ac.jp
```

  のようにすると、前者は別のコマンドプロンプトウィンドウを開いて時刻を表示し、後者は工科大のホームページを開いたブラウザが立ち上がる。

- color コマンドの [色指定] は 2 桁の 16 進数で、上の桁が背景色、下の桁が文字色。各桁の数字と色の関係は、0〜7 は、暗色の黒、青、緑、水色、赤、紫、黄、白、8〜f は、明色の黒、青、緑、水色、赤、紫、黄、白、をそれぞれ意味する。オプションをつけずに実行するとデフォルトに戻る。例えば

```
Z:¥> color 6a
```

  とすると、コマンドプロンプトの背景を暗色の黄色に、文字色を明色の緑にする。

- pause は、入力をうながすメッセージを表示して Enter キーの入力待ちになる。このコマンドは主にバッチファイルで使用する。

コマンドの実行方法について、注意すべき点をいくつかあげる。

1. コマンド名は、大文字でも小文字でもよい。オプションは大文字小文字の区別がないものも多いが、例外もある。

2. 以下の形式のファイルは、コマンドプロンプトで「コマンド」として使うことができる。

   - **実行ファイル** (拡張子 .exe)。
     C 言語等で作成したプログラムファイル。単独で実行でき、.exe 部分は指定しなくてもよい。例えば、「メモ帳」のプログラムファイル名は notepad.exe だが、notepad で起動できる。
   - **バッチファイル** (拡張子 .bat)。
     詳細は後で紹介する。.bat 部分は指定しなくてもよい。

- ワープロ文書ファイル (拡張子 .doc) など、特定の拡張子を持ち、それに関連付けされたアプリケーションがあるファイル。
  そのファイル名をコマンドとして実行すると、関連付けられたアプリケーションが実行され、それがそのファイルを開いてくれる。

  [**用語**: 「拡張子」= 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。詳しくは第 2 回で説明する。]

3. ヘルプ
   コマンドの説明 (ヘルプメッセージ) は、MS–Windows に用意されている標準的なコマンドの場合は、例えば dir というコマンドであれば

   ```
   Z:¥> help dir
   Z:¥> dir /?
   ```

   のように help コマンドを使うか、そのコマンド自体に /? オプションを指定すれば、そのコマンドの意味、そのコマンドで使えるオプションなどの詳しい説明が表示される。

4. 履歴
   コマンドプロンプトに過去に入力したコマンドは、上下の矢印キーで **履歴** を呼び出すことができる。左右の矢印キーでカーソルを動かして修正もできるが、その際、カーソルが行の右端でなくても、Enter を入力すれば行全体がコマンドとして実行される。

5. TAB 補完
   コマンドプロンプトで、コマンドやオプションとしてファイル名を入力する場合、先頭の何文字かを入力した後に TAB キーを入力すれば、それで始まるファイル名を補完してくれる。該当するファイルが複数ある場合は、そのまま TAB キーを繰り返し打つことで該当するファイル名を巡回してくれる。入力を楽にするとともに、打ち間違いを防ぐのに役立つ。

2 otbedit の使い方

otbedit はタブ機能付きの小さなテキストエディタで、以下で公開されているフリーソフト (実習室には既にインストールされている)。

- OTBEdit (A.Ogawa)
  http://www.hi-ho.ne.jp/a_ogawa/otbedit/index.htm

用語: 「テキストエディタ」= テキストファイルを作成、編集するソフトウェア。「メモ帳」もその一つ。

メモ帳でもバッチファイルは作成できるが、otbedit には「プログラムソースのキーワードの色付け/カスタマイズ機能」「改行文字や全角空白文字の表示」「フォントの設定」などの機能がついているので、こちらの方が便利。

otbedit のコマンド名は `otbedit`。後ろにファイル名をオプションとしてつけて実行すると、そのファイルを開いて立ち上がるが、もちろんオプションなしで otbedit を起動してからメニューからフォイルを開くこともできる。

otbedit の使い方は、起動して試してみればすぐにわかると思うので省略するが、`Ctrl-N` (新規ファイルのオープン (New))、`Ctrl-O` (ファイルを開く (Open))、`Ctrl-S` (ファイルを保存 (Save)) のショートカットは覚えておくとよい。

otbedit の左上にある Txt, Htm 等は、そのファイル形式専用の編集モード。この実習では主に Bat (バッチファイル編集モード) と AWK (AWK スクリプト編集モード) を使用する。

3 バッチファイルの実行

バッチファイル は、基本的にはコマンドプロンプトで実行できるコマンドを、一行ずつ並べて書いたファイル (拡張子 .bat)。このファイルはコマンドプロンプト等で直接実行させることができ、複数のコマンドをまとめて実行させるのに用いる。分岐、ジャンプ、ループなどの簡単なプログラム構文も用意されている。

例えば、otbedit で以下のような 5 行のファイルを作成する[3] (ファイル名は `test1.bat` とする)。otbedit では、`.bat` の拡張子のファイルを開くか、画面の左上にある Bat というアイコンをクリックするとバッチファイル専用の編集モードになる。

```
@echo off
rem バッチファイルの簡単なサンプル
dir
date /t
time /t
```

このバッチファイル (ひとつ) を実行すると、これら 5 行のコマンドが上の行から順番

[3]以後、ファイルの内容を示すときは、このように影付きの枠で囲むことにする。

に実行される。それぞれの意味は以下の通り[4]。

```
1 行目の「@echo off」= コマンド行自体の表示を消すためのもの
2 行目の rem で始まる行 = コメント行 (rem は何もしないコマンド)
3 行目の dir = カレントディレクトリの一覧を表示
4 行目の date /t = 日付を表示
5 行目の time /t = 現在時刻を表示
```

バッチファイルの実行方法は、

1. エクスプローラ上でバッチファイルのアイコンをダブルクリック (マウス)
2. コマンドプロンプト上でバッチファイル名を入力 (キーボード)

のいずれかがあるが、1. は新たなコマンドプロンプトウィンドウが開くが、バッチファイルが終了するとすぐにそのウィンドウが閉じられてしまい、何が行われたかわからないので (pause を最後に入れれば回避できるが)、この実習では基本的に 2. の方法で実行する。

例えば、上の test1.bat というバッチファイルを実行するには、

```
Z:¥> test1.bat
```

とする。ただし、このように実行するには、バッチファイル test1.bat をカレントディレクトリの Z:¥ に保存する必要がある。別のフォルダに保存したバッチファイルを実行する方法については、第 2 回で紹介する。

「@echo off」に関する補足:

- 「echo off」は、そのコマンド以後のコマンド行の表示を消すので、これでバッチファイルの 2 行目以降のコマンド行自体の表示が消える。そして、1 行目の先頭の @ は、その行のコマンド行の表示を消すので、これですべてのコマンド行自体の表示が消えることになる。
- バッチファイルでは通常コマンド行自体の表示は必要ないので、以後バッチファイルの 1 行目はすべて「@echo off」で書き始めることとする。
- コマンドプロンプトで「echo off」を実行すると、プロンプトも入力も表示されなくなるので、コマンドプロンプトでは実行しないこと (した場合は echo on で復帰できる)。
- バッチファイル内の「echo off」「echo on」は、バッチファイル実行中の echo の状態にしか影響を与えない。例えばコマンドプロンプトで echo off としても、

[4]以後、単純な一重枠で囲んだものは、コマンドプロンプトでもなく、ファイルの内容でもなく、単なる説明を意味するものとする。

そのバッチファイルが終了すればコマンドプロンプトの `echo` 状態は ON に戻る (`color` などはバッチファイルを終了しても元には戻らない)。

- 1 行目の `@echo off` の前に `rem` を書いて「`rem @echo off`」とすると、その機能がコメントアウトされ、すべてのコマンド実行行が実行結果とともに表示される。これは、バッチファイルの実行中にエラーメッセージが表示される場合、そのエラーがどのコマンドから出ているのかを知るのに役に立つ (デバッグ用)。

コラム: MS–Windows のスクリプト環境

Unix には「小さいプログラムを使い回したり、組み合わせて使う」ための環境として、対話型シェル環境やシェルスクリプトがあるが、MS–Windows にもそれに近いものとして以下のものがある。

- コマンドプロンプト + バッチファイル (以下、バッチ環境)
- WSH (Windows Scripting Host)
- Windows PowerShell

これらの特徴や違いなどを表 2 に示す。

	バッチ環境	WSH	PowerShell
対話環境	コマンドプロンプト	なし	ある (専用のシェルウィンドウ)
スクリプト言語	バッチファイル (.bat)	VBScript (.vbs), JScript (.js) 等	専用のスクリプト言語 (.ps1)
構文や変数	一応使える	使える	使える
外部コマンドやモジュール	一応ある	ある (オートメーションオブジェクト)	一応ある
パイプ、リダイレクション	ある (ファイルベース)	少なくとも簡単なものはなさそう	ある (オブジェクトベース)
実行ファイル	cmd.exe	wscript.exe (cscript.exe)	PowerShell.exe
XP では	そのまま使える	そのまま使える	要インストール
歴史	MS–DOS から	Windows 98 から	2007 年頃から

表 2: バッチ環境、WSH、PowerShell の特徴

バッチ環境は、正確には MS–DOS の頃は command.com というインタープリタが使われていて、Windows NT 以降はその拡張版である cmd.exe に置きかわっている。現在は PowerShell への移行が推奨されているようである。

また、WSH も PowerShell への移行が推奨されていて、今後はメジャーバージョンアップもないという話であるが、PowerShell はまだ新しく、安定性や仕様の変更も含めて今後どうなるかはわからない。

よって、これらのうちでは、古くから使われていて、そして今後もなくなることはないと思われるバッチ環境を利用するのが一番ましな気がするが、バッチファイルは以下の点などで Unix のシェルスクリプトにはかなり劣る。

- 「引用符」がちゃんと使える形にはなっていないので、文字列変数、コマンドのパラメータ等に関する制限がある。
- while 文、switch 文がないので if, goto に頼りがちになり、スパゲッティ化しやすい。
- フィルタやバッチファイル用に用意されているコマンドが少ない。
- ヒアドキュメントが書けない。
- 長いコマンドを複数行に分けて書けない。

よって、本当に実用的なものを作るには、バッチファイルと AWK やフリーソフトなどを組み合わせる必要があるだろう。

しかし簡単な処理なら C のようなプログラミング言語を使わなくてもバッチファイルで行えるし、既存のプログラムを組み合わせて自分用の便利な仕組みを簡単に作れること、ファイルとフォルダに関する操作もひとつひとつエクスプローラでやるより for 文などで簡単に行えることが多いこと、および C 言語で外部プログラムを利用する際もバッチファイルの知識が役に立つことなどから、バッチファイルやコマンドプロンプトのことを知ることは意味があると思う。