情報通信コース実験 II
スクリプトプログラミング (担当 竹野、2020 年度)

(http://takeno.iee.niit.ac.jp/%7Eshige/
math/lecture/jikken1/jikken1.html)

第 1 回 コマンドプロンプトとバッチファイル

以後、枠で囲んだものは、原則以下を示す。

- 二重枠の囲みは、コマンドプロンプトでの実行例。例:

```
E:¥work> help dir
```

- 影付きの枠の囲みは、テキストファイルの内容。例:

```
@echo off
```

- 一重枠の囲みは、単なる説明。例:

```
1 行目の「@echo off」= コマンド行自体の表示を消すためのもの
```

1 コマンドプロンプト

コマンドプロンプト は、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。この実習では、このコマンドプロンプト上で作業を行う。

コマンドプロンプトは、この実習では以下のようにして USB メモリ上で起動する。

1. USB メモリをパソコンに刺し
2. パソコンでエクスプローラーを起動し
3. USB メモリドライブを選択して (通常は E:)
4. run1.bat を実行 (ダブルクリック)

これで図 1 のようなコマンドプロンプトが起動する。バッチファイルの実習も、AWK の実習も、このコマンドプロンプトで作業を行う。

コマンドプロンプト上に表示される「E:¥work> 」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が実行される。

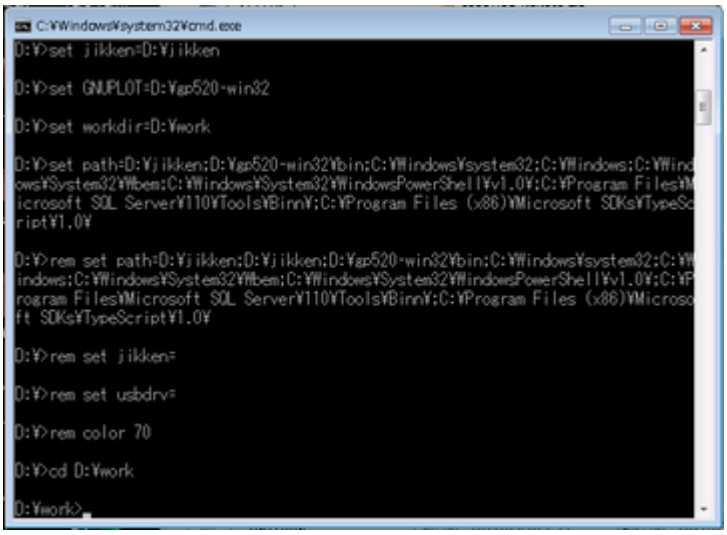

図 1: コマンドプロンプトの起動画面

なお、**プロンプト記号**「>」の左の部分は、今自分がいることになっているディレクトリ (**カレントディレクトリ**) を意味するが、実行例で毎回示すのは無駄なので、プロンプト部分は以下のように「>」のみ示すことにする。

```
> help dir
```

この例の場合、help がコマンド名 (プログラム名)、dir はそのコマンドに対する **オプション** (追加指定)。オプションは、コマンドの後ろにスペースで区切って書き並べる。

コマンドプロンプトで使えるコマンド (とオプション) には、例えば表 1 のようなものがある (実行する際に [ ] をつけないように)。

例えば、

```
> cls
> date /t
> time /t
> color 70
```

とすると (入力するのは > の右の部分)、まず画面がクリアされ、次に今日の日付、現在の時刻が順に表示され、最後に背景が灰色になって文字が黒になる (実際には、各行で Enter を打った際にそれぞれが実行され、画面に結果が表示される)。

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

コマンド	説明
help [コマンド名]	コマンドの説明の表示
dir	ディレクトリ内の一覧表示
more [ファイル]	[ファイル] の内容を画面に表示
copy [ファイル 1] [ファイル 2]	[ファイル 1] を [ファイル 2] にコピー
del [ファイル]	[ファイル] を削除
cls	コマンドプロンプト画面のクリア
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
rem [文字列]	何もしない

表 1: 基本コマンド

けずに color を実行するとデフォルトに戻る。

コマンドの実行に関する注意:

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 (注: デフォルトの設定ではエクスプローラで拡張子は表示されない)。

   - **実行ファイル** (拡張子 .exe)。
     C 言語などで作成したプログラムファイル。拡張子 (.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。拡張子は指定しなくてもよい。
   - ワープロ文書ファイルなど、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ソフトが設定されているファイル。
     そのファイル名 (拡張子も必要) 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ソフトが起動され、そのソフトがそのファイルを開く。

3. コマンド履歴
   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ので、前と同じコマンドを打ち直さずに済むし、前のコマンドを少し修正して実行することもできる。

2 otbedit の使い方

otbedit は、「メモ帳」同様のテキストファイル編集ソフト (**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 (USB メモリにインストール済み)。

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

otbedit には「プログラムソースのキーワードの色付け」「改行文字や全角空白文字の表示」「複数ファイルのタブ切替」などの機能がついているので、プログラミングには「メモ帳」より便利。

otbedit のコマンド名は `otbedit`。コマンドプロンプトで実行すると図 2 のようなウィンドウが開く。

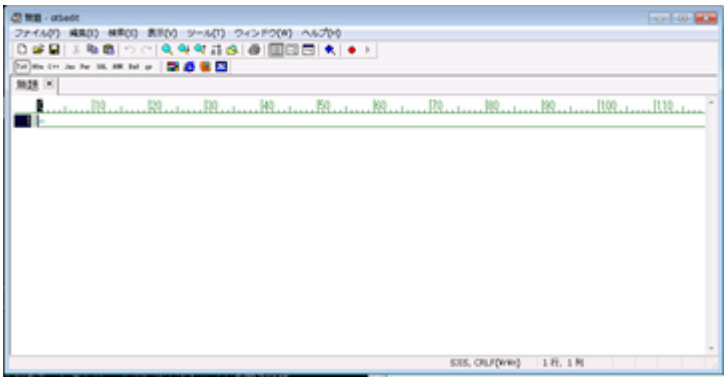

図 2: otbedit の起動画面

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。そこをクリックすればその編集モードになる。

3 バッチファイル

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`)。バッチファイルはコマンドとして実行させることができ、その場合原則その中身が 1 行ずつ順に (間を置かずに) 実行される。

例えば、otbedit で以下のような 5 行のファイルを作成し、`test1.bat` という名前で保存する (`E:¥work` 内に)。

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

otbedit では、Bat アイコンをクリックするか、または .bat の拡張子のファイルを開いた場合にバッチファイル編集モードになる。

このバッチファイルをコマンドプロンプトで

```
> test1.bat
```

と実行すると、これら 5 行のコマンドが上の行から順番に実行される。各行の意味は以下の通り。

```
1 行目の「@echo off」= コマンド行自体の表示をオフにする
2 行目の rem で始まる行 = 何もしない（コメント行）
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 = 現在の日付を表示
5 行目の time /t = 現在の時刻を表示
```

以後、バッチファイルの 1 行目は常に「@echo off」とする。ただし、デバッグの際は rem でその行をコメントアウトすると便利な場合もある。

4 変数

コマンドプロンプトや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 2: 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ると、変数名や値文字列にスペースが含まれてしまう。
- 変数名の大文字小文字は区別されず、例えば str も STR も sTr も同じ変数。
- 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、既に定義されている変数は使わないようにすること。
- 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。なお、「start [URL]」は、デフォルトのブラウザを立ち上げて指定した URL を表示する。

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 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、展開する方 (4 行目) で %n% を使って「echo %s%n%%」のようにしてもうまくはいかない。

また、環境変数には、表 3 に示す、値が自動的に設定される **動的環境変数** もある。これは、単なる set コマンドでの一覧には表示されず、ユーザが設定することもできない。これらは、参照する際に値が自動的に設定され、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 3: 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、バッチファイルを実行するときにオプションを与えて実行すると、それをバッチファイル内部では、以下のような変数として使うことができる。ただし、これらは参照のみで代入はできない。

名前	意味
%1,%2,. . . ,%9	指定した順の個々のオプション文字列
%0	そのバッチファイル自身の名前
%*	%1 以降のオプション文字列全体

表 4: バッチファイルのオプション値の参照

例えば

```
@echo off
echo %%2 = %2
echo %%0 = %0
echo %%* = %*
```

というバッチファイル test1.bat を

```
> test1.bat 123 竹の 茂治
```

のように実行すると、

```
%2 = 竹の
%0 = test1.bat
%* = 123 竹の 茂治
```

と表示される。また、この場合は %4 以降は空文字列となる。

5 if 文

コマンドプロンプトやバッチファイルでも if 文 (正確には if コマンド) を用いることで条件分岐を行うことができる。if は、以下の 2 種類の形式が使用できる ([ ] はつけないこと)。

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

この [コマンド群] の部分には、コマンド 1 つ、あるいは複数行のコマンドを書くことができる。例:

```
@echo off
if %1==0 ( echo %1 は偽です。) else echo %1 は真です。
```

この最後の部分のように指定するコマンドが一つで 文の途中でない場合は ( ) は省略できる。

```
@echo off
if exist %1 (
    echo ファイル %1 が存在します。
    dir %1
) else (
    echo ファイル %1 は存在しません。
)
```

なお、「dir [ファイル名]」は、そのファイルの情報のみを表示する。

if, else を複数つなげたい場合、C 言語同様以下のように書くこともできる。

```
if [条件 1] ( [コマンド 1]
) else if [条件 2] ( [コマンド 2]
) else [コマンド 3]
```

注意:

- else を行頭に書くとエラーになるので、else は [コマンド群 1] の直後、または閉じかっこ「)」の後ろに続けて書かなければいけない。例えば、以下はエラーになる。

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

- if の [条件式] の後ろや else の後ろに「(」を続けて書くときは、その間に空白を入れないとエラーになりうる。

- ( ) の中に変数参照を書くと、その中のすべての変数が先に置換されてから実行されることに注意。

if 文の条件部分には、表 5 のいずれかの形式を使う。

形式	意味
[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

表 5: if 文の条件で使える形式

表 5 の 2 番目の形式の比較演算子は、表 6 の 3 文字の演算子である。

演算子	意味	演算子	意味
equ	等しい (=)	neq	等しくない (≠)
leq	以下 (≤)	lss	より小さい (<)
geq	以上 (≥)	gtr	より大きい (>)

表 6: 比較演算子

表 5 の条件式の前に、例えば「not %x%==0」のように not をつけるといずれの場合も真偽が反転される。

表 5 の 1 つ目、2 つ目の形式では、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。

また、表 5 の 1 つ目、2 つ目の形式では、if と条件の間に 「/i」を置くとアルファベットの大文字、小文字を区別しない。例えば「if A==a」は偽で「if /i A==a」は真となる。

if の条件式の注意:

- C 言語とは違い、条件式の値の部分に「%x%+1 gtr 3」のように数式を書くことはできないし、複数の条件を AND (&&) や OR (||) で結ぶこともできない。
- 表 5 の 1 つ目、2 つ目の形式では、両辺の一方の値が空文字になるとエラーとなる。よって、両辺の一方が空文字になる可能性がある場合 (特に環境変数やオプション変数などを使う場合) は、両辺をそれぞれ " " で囲んで「if "%1"=="1"」のようにするとよい (" " つきの文字列として比較される)。
- C 言語とは違い、条件部分を「if (%3==1)」のようにかっこつきで書くとエラーになる。

バッチファイルのオプション %1, %2, . . . は、指定されてないもの以降は空文字になるので、オプションがいくつあるかを if 文で判定することができる。例:

```
@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]
```