2008 年 01 月 16 日
(2008 年 01 月 18 日 改訂)

# WWW ページの一括取得 csh スクリプト

新潟工科大学 情報電子工学科 竹野茂治

## 1 はじめに

本稿では、csh スクリプトの入門用のサンプルプログラムとして、指定したいくつかの URL の WWW ページをまとめて取得するものを紹介します。

定期的によく参照する WWW ページは、ブラウザ側でブックマークに登録して、それをブラウザ画面でマウスで選択しながらアクセスする、という方法が一般的だと思いますが、それだとそのページが更新されているかどうかの確認や、各ページへのネットワークアクセスに時間がかかったりします。

それを、WWW ページのソースファイル (HTML ファイル) だけ時間に余裕のあるときにまとめて持ってくるようにすれば、それを見るときはネットワークに接続せずに速く参照できますし、以前取得しておいたものと比較することで、更新されているもののみを参照するといったことも可能になります。

今回は、そのうち簡単なことを行う csh スクリプトをサンプルとして、それに必要となる csh スクリプトの構文や書き方について説明します。csh スクリプトの構文の概要については、[2] を参照してください。

## 2 外部コマンド

今回作成する csh スクリプトでは、Unix に標準的にインストールされているコマンドだけではなく、以下の (フリー) ソフトも利用します (OS や環境によっては既にインストールされているかもしれません)。

- **wget** : http (または ftp) アクセスできるサイトからデータをひとつ、あるいは複数まとめて取得するフリーソフト。
  - GNU Wget : `http://www.gnu.org/software/wget/`
- **WWW ブラウザ** : 本稿では、w3m か firefox を想定していますが、他の同等のものでも構いません。

- w3m : `http://w3m.sourceforge.net/index.ja.html`
- firefox : `http://www.mozilla-japan.org/products/firefox/`

wget は豊富なオプションを持つ非常に高機能なツールで、よってオプションを駆使すれば実は今回目標とするものは wget 単独でもそれに近いことを実現できるのですが、今回は単に指定した URL の HTML ファイルひとつを取得するものとして利用します。

# 3 最も単純な形のスクリプト

まずは今回目標とするスクリプトの最も単純な形のものを考えてみます。指定する URL はその一覧ファイル (以後 **URL リストファイル** と呼びます) を用意しておいて、その各行にひとつずつ書くことにます。そのファイルを仮に `~/wwwcheck/urllist` としておきます (`~` は、csh スクリプトではホームディレクトリを指します)。

`~/wwwcheck/urllist` の内容:

```
http://www.niit.ac.jp/
http://www.niit.ac.jp/ieehtml/iee/
...
```

wget で URL の HTML ファイルを取得するには、以下のようにします:

```
% wget -q -O - URL > HTMLFILE
```

ここで、指定している wget のオプションの意味はそれぞれ以下通りです。

- `-q` : wget の動作中の画面出力を抑制する。
- `-O` *file* : 取得した HTML ファイルを *file* に出力。ただし *file* が '`-`' である場合 (上の場合) は、標準出力に出力する。

よって、このようにすると取得した HTML ファイルは標準出力に出力されるので、上のようにリダイレクションを利用してファイルに落とします。

今回の目標は、URL リストファイルの各行に書かれている URL をひとつずつ wget に指定して HTML ファイルを取得すればいいので、以下のような csh スクリプトでそれが実現できます。

wwwcheck1.csh の内容:

```
#! /bin/csh -f
##### 初期設定 ####
set urllistf = ~/wwwcheck/urllist # URL リストファイル
set datad = ~/wwwcheck/data # 取得したファイルの置き場所
set wget = "wget -q -O -"     # wget のコマンドライン
##### 実行部分 ####
set j=1
foreach url ( `cat $urllistf` )
    $wget "$url" > $datad/$j.html
    @ j ++
end
```

# で始まる行、また行の途中の # 以降はコメント部分で、csh スクリプトでは無視されます。

先頭部分でいくつか初期設定をしていますが、これらはもちろん右辺の値自体を直接実行部分の方で使用するようにしてもよいのですが、このようにしておけば、後で修正をする場合、あるいは繰り返し利用する場合などに便利です。set は、変数値の設定を行うコマンドですが、2 つ目の wget のコマンドラインはスペースを含む文字列なので、" " で囲まないと変数には文字列全体が保存されません。それらの変数の値は $ をつけて参照します。

foreach はリストベースのループ構文で、

```
foreach var ( str1 ... strn )
    command1
    ...
end
```

の形式で使用し、これは変数 *var* に文字列 *str1* を代入して end までのコマンド列を実行し、変数 *var* に次の文字列を代入して end までのコマンド列を実行し、それを *strn* まで繰り返します。wwwcheck1.csh の場合には変数 url に文字列が代入され、その各 url に対し $wget と @ で始まる 2 行が順次実行されることになります。

@ で始まる行は @ 文による計算代入式で、"@ j ++" は変数 $j$ の値をひとつずつ増やしますので、ループの最初では $j$ は 1 で、よって $wget 文により $datad/1.html が生成され、その後 @ 文により $j$ は 2 になりますので、ループの次の処理では $datad/2.html が生成される、ということが繰り返されます。

ところで、url に代入される文字列ですが、これはリスト変数

```
( `cat $urllistf` )
```

の文字列が順に代入されています。これがどのような文字列のリスト変数になっているのかを説明します。

まず、cat は Unix に標準的に用意されているコマンドで、

% cat *file1 file2 ... filen*

とすると、*file1* の内容を標準出力に書き出し、次に *file2* の内容を書き出し、これを *filen* まで繰り返すコマンドです。

一方、csh スクリプトで逆クォート ‘ ‘ で囲まれた部分は、それをコマンドと見なして実行し、その標準出力の、改行がスペースに置き換えられた文字列に展開されます。

例えばファイル *file* の中身が

```
str1 str2
str3
str4
```

であるファイルに対して、

```
( ‘cat file‘ )
```

とすると、これは

```
( str1 str2 str3 str4 )
```

と展開されます。

結局 wwwcheck1.csh では、url には URL リストファイル `$urllistf` の各行の内容、すなわち設定した URL がひとつずつ渡されて、そのそれぞれに対して `$wget` の行が実行されることになります。

つまり、このスクリプトにより、URL リストファイルの 1 行目の URL の HTML ファイルが 1.html に、2 行目の URL の HTML ファイルが 2.html に、次々と保存されていくことになります。

なお、`$wget` の行では変数 url の値を引用符で囲んでいますが、これは、URL には特別な文字 ‘?’ を含む場合があるからです (CGI によるページを参照する場合など)。そのような URL の場合は、これを引用符 (" ") で囲んでおかないと、その文字列が csh スクリプトのファイル名パターンと見なされて、そのようなファイルが存在しない、というエラーが起きてしまいます。単一引用符内 (’ ’) では変数の値は展開されませんが、二重引用符内 (" ") ならば展開されます。

# 4 コメント行のサポートとブラウザの起動

この節では、3 節のスクリプト wwwcheck1.csh の 2 つの簡単な拡張を考えてみます。

まずは、URL リストファイルでコメント行を許すことを考えます。例えば URL リストファイル内のいくつかの URL の取得をやめたい場合、その行を本当に消してしまうよりもコメント行として無視されるようにしておけば、後でその行を復帰させる場合やテストをする場合などに有用です。また、各 URL にその URL の説明をつけたり、URL リストファイルの更新状況などをメモしておくのにも、コメント行があると便利です。

ここでは、行頭に '#' がついている行を URL リストファイルのコメント行と見なし、wwwcheck1.csh に無視させることにします。こうするには、foreach 文の（ ）内で、cat で URL リストファイルのすべての行を出力させるのではなく、行頭が '#' でない行のみを出力させるようにすればいいわけですから、cat の代わりに以下のようにすればうまくいきます:

```
( `grep -v "^#" $urllistf` )
```

grep は Unix の標準コマンドで、指定したパターンにマッチする (またはマッチしない) 行のみを出力させるコマンドです。grep で与えるパターンは 正規表現 と呼ばれ、正規表現では '^' は行頭を意味しますので、"^#" で「'#' から始まる行」を意味することになります。grep のオプション -v は、「そのパターンにマッチしない行」を出力します。これを指定しないと、逆にコメント行のみを出力することになってしまいます。

次に、wget で HTML ファイルを保存した後で、最後にブラウザを立ち上げてそれらのファイルを見るようにすることを考えてみます。ブラウザが w3m や firefox の場合には、それを行うにはそのコマンド名の後ろにローカルの HTML ファイル名をスペースで区切って並べて書いていけばいいだけのようですので、スクリプトの先頭で

```
set browser = "w3m -N"
```

のように設定しておいて、スクリプトの最後に

```
$browser $datad/*.html
```

という行を追加すればいいだけです。'*' は、csh スクリプトでは '?' と同様、ファイル名パターン用の特殊文字で、任意の文字列に一致し、そしてこれが含まれている単語が、それにマッチする「複数のファイル名」に展開されます。つまり、`$datad/*.html` の部分は、単にひとつの `$datad/1.html` のように置き換えられるのではなく、存在するファイルによる複数の単語に置き換えられるので、上の行は実際には

```
$browser $datad/1.html $datad/2.html ...
```

のような文字列に展開されて実行されます (もちろん実際には $browser や $datad の部分も設定された文字列に展開されます)。

なお、w3m の -N オプションは、指定した各 HTML ファイルをタブとして開くオプション (w3m バージョン 0.5 以降) です。

# 5 コマンドラインオプションのサポート

4 節の最後で、取得したものを見るためにブラウザを立ち上げるような改良を考えましたが、取得した直後にはブラウザを立ち上げたいとは思わないかもしれませんし、既にブラウザが立ち上がっている場合は、二重にブラウザを立ち上げないようにしている場合もあるでしょう[1]。

よって、そのブラウザの立ち上げを、スクリプトの実行時にコマンドラインオプションを与えることで切り替えられるようにすることを考えてみます。まずは、とりあえず以下のようなオプションをサポートしてみることにします:

```
% csh -f wwwcheck1.csh -b : ブラウザを立ち上げる
% csh -f wwwcheck1.csh -nb : ブラウザを立ち上げない
% csh -f wwwcheck1.csh -h : オプションの説明のヘルプの表示のみ
% csh -f wwwcheck1.csh : ブラウザは立ち上げない (default)
```

このために変数 browserOn を設定し、これが 1 のときはブラウザを立ち上げ、0 のときはブラウザを立ち上げないようにします。よってスクリプトの最初で、

```
set browserOn = 0 # 0 がデフォルト
```

とし、最後のところは

```
if ( $browserOn ) then
    $browser $datad/*.html
endif
```

のように if 文で条件分岐すればいいでしょう。この if 文は、( ) 内の条件式が真 (0 以外) のときに endif までの部分が実行されます。

[1] 二重にブラウザを立ち上げると、それらでブックマークを別に更新したりして問題が起こる可能性がありますし、そもそも二重には立ち上げられないかもしれません。

コマンドラインオプションの取得は、特別なリスト変数 argv を参照することでできます。オプションを調べるのには switch 文を利用します:

```
if ( $#argv > 0 ) then
    switch ( $argv[1] )
    case -b:
        set browserOn = 1
        breaksw
    case -nb:
        set browserOn = 0
        breaksw
    case -h:
    default:
        echo "csh -f $0 ([option])"
        echo " [option]:"
        echo "     -b :  ブラウザを立ち上げる"
        echo "     -nb:  ブラウザを立ち上げない (default)"
        echo "     -h :  このメッセージ"
        exit
    endsw
endif
```

$#argv はリスト変数 argv の要素の個数を意味し、コマンドラインオプションを指定しなければ 0、指定すればその個数になり、また $argv[1] はその最初のオプション文字列になります。

switch 文は C 言語の switch 文とほぼ同様ですが、文字列ベースで比較されますので、case ラベルには比較対象となる各文字列を置きます。C 言語では break で各 case 処理を中断して抜けますが、csh スクリプトの switch 文では breaksw で抜けます。endsw は switch 文の終わりを意味します。

switch 文の代わりに if 文を使って、

```
if ( $argv[1] == "-b" ) then
    set browserOn = 1
else if ( $argv[1] == "-nb" ) then
    set browserOn = 0
else if ( $argv[1] == "-h" ) then
    ...
```

のようにしてもよさそうに見えるかもしれませんが、csh スクリプトでは if などの構文の解析の前に変数置換が行われ、よって最初の if 文が

```
if ( -b == -b ) then
```

のような行になってしまい、if 文のカッコ内では数式評価が行われるため、それが - から始まるとファイル検査式であると見なされて、エラーになってしまいます (詳しくは [2] 参照)。これは、

```
if ( "X$argv[1]" == "X-b" ) then
```

のようにする、という対処もありますが、数式評価を行わない switch 文を使用する方が無難だし、楽だと思います。

ところで、この switch 文では、オプションに従って変数 browserOn の値を切り替えてセットしていますが、オプションが "-h" の場合と、いずれのオプションにもマッチしないものが指定された場合には (default: ラベル)、ヘルプメッセージを表示して exit でスクリプトを強制終了するようにしています。

また、このヘルプ出力では、以下のものが使用されています。

- echo *string* : 文字列 *string* を画面表示
- $0 : このスクリプト自身の名前 (C 言語の argv[0] のようなもの)

これをさらに拡張して、立ち上げるブラウザの種類をオプションで切り替えたりすることもできるでしょう。特に firefox では、既に firefox が立ち上がっている場合はそれを操作して別ウィンドウのみを開くオプション -remote がありますので、firefox が立ち上がっているかどうかを調べて、立ち上がっている場合にはそれを利用する、といった改良も考えられるでしょう。

# 6 冗長出力

もうひとつの改良として、この節では冗長出力を考えてみます。

Unix のコマンドは、元々必要最小限の情報しか画面表示しない寡黙なものが多く、このスクリプト wwwcheck1.csh もそのようになっているのですが、コマンドラインオプションにより、途中経過を表示するようにしてみます。これは、デバッグにも役立ちます。

具体的には、以下の 2 つを行わせるようにします。

- foreach ループの各実行状況の表示

- 認識したオプション (とそれによる変数の値) の表示

そのために、この csh スクリプトに -v, -q というオプションを追加し、-v を指定した場合に冗長出力、-q とした場合は今まで通りとして、それを変数 myverbose の値で切り替えることにします。なお、“verbose” という名前は、特別な意味を持ちますので (予約済み)、変数名として使用することはできません。

これは、-b などとは独立なオプションですから、今度はオプションは一つとは限りませんので、オプションのチェックもリスト変数 argv の要素がある間行わなければいけません。よって、この場合は 5 節のコマンドライン解析部分の switch 文全体を囲んでいる if 文を foreach 文にすればいいわけです:

```
set browserOn = 0
set myverbose = 0
foreach option ( $argv )
    switch ( $option )
    case -v:
        set myverbose = 1
        breaksw
    case -q:
        set myverbose = 0
        breaksw
    case -b:
        ...
    endsw
end
```

$argv は、リスト変数 argv の要素すべてをスペース区切りで書き並べたものに対応しますので、「( $argv )」で argv のリストを再現できます。

これは、もう一つのループである while 文で書くこともできます。

```
set browserOn = 0
set myverbose = 0
while ( $#argv )
    switch ( $argv[1] )
    case -v:
        set myverbose = 1
        breaksw
    case -q:
        set myverbose = 0
        breaksw
```

```
        case -b:
            ...
        endsw
        shift
    end
```

while 文は C 言語の while 文と同様で、( ) 内の条件が真 (0 以外) の間 end までのコマンド列が実行されます。この場合、switch 文の endsw の次についているコマンド shift が必要で、shift は、

- shift *var* : リスト変数 *var* の最初の要素を取り除く (*var* を指定しなかった場合は *argv* が対象)

ということを行います。例えば argv が最初

```
( "-b" "-v" )   ($#argv = 2)
```

だった場合、一度 shift を行うとこれが

```
( "-v" )   ($#argv = 1)
```

と変わります。

よって上のソースは、while 文の条件文である argv の要素の数 ($#argv) が 0 でない間、最初の要素 $argv[1] を switch 文に与えていることになり、これでもちゃんと argv の要素がひとつひとつチェックされることになります。

このコマンドラインオプションの解析部分が済んだ後で、まず -v に対応する以下の処理を追加します:

```
if ( $myverbose ) then
    echo "(browserOn,myverbose) = ($browserOn,$myverbose)"
endif
```

そして、wget での取得を実行する foreach 文の部分を例えば以下のようにすればいいでしょう:

```
foreach url ( `grep "^#" $urllistf` )
    if ( $myverbose ) then
        echo "[$j] $url ==> $j.html"
```

```
    endif
    $wget "$url" > $datad/$j.html
    @ j ++
end
```

この場合、-v を指定すると、

```
(browserOn,myverbose) = (0,1)
 [1] http://www.niit.ac.jp/ ==> 1.html
 [2] http://www.niit.ac.jp/ieehtml/iee/ ==> 2.html
...
```

のようなものが表示されることになります。

# 7 全ソースファイル

ここまでの改良も含めた、全ソースファイルを最後に紹介しておきます。

```
#! /bin/csh -f
# wwwcheck1.csh
# shige 01/16 2008
#
#### 初期設定 ####
set urllistf = ~/wwwcheck/urllist # URL リストファイル
set datad = ~/wwwcheck/data # 取得したファイルの置き場所
set wget = "wget -q -O -"     # wget のコマンドライン
set browser = "w3m -N"     # 最後に立ち上げるブラウザ
#set browser = "firefox"
set browserOn = 0     # ブラウザを立ち上げるかどうか
set myverbose = 0     # 冗長出力をするかどうか

##### コマンドライン解析 ####
foreach option ( $argv )
    switch ( $option )
    case -v:
        set myverbose = 1
        breaksw
    case -q:
        set myverbose = 0
```

```
        breaksw
    case -b:
        set browserOn = 1
        breaksw
    case -nb:
        set browserOn = 0
        breaksw
    case -h:
    default:
        echo "csh -f $0 ([option])"
        echo " [option]:"
        echo "     -b :  ブラウザを立ち上げる"
        echo "     -nb:  ブラウザを立ち上げない (default)"
        echo "     -v :  冗長出力"
        echo "     -q :  画面出力を抑制 (default)"
        echo "     -h :  このメッセージ"
        exit
    endsw
end

##### 実行部分 ####
if ( $myverbose ) then
    echo "(browserOn,myverbose) = ($browserOn,$myverbose)"
endif

set j=1
foreach url ( `grep "^#" $urllistf` )
    if ( $myverbose ) then
        echo "[$j] $url ==> $j.html"
    endif
    $wget "$url" > $datad/$j.html
    @ j ++
end

if( $browserOn ) then
    $browser $datad/*.html
endif
```

# 8 最後に

今回は、foreach, switch, while, if などの構文や変数の使用法、およびコマンドラインオプションの簡単な検査方法などを紹介しました。

次回は、今回のスクリプトをさらに改良し、前に取得した WWW ページと今回取得したものを比較し、更新されているもののみを参照できるようにする予定です。

# 参考文献

[1] 山口和紀監修、「The UNIX Super Text 上」技術評論社 (1992)

[2] 竹野茂治、「csh スクリプトに関する基礎知識」 (2008)