2008 年 01 月 24 日

更新された WWW ページの参照スクリプト

新潟工科大学 情報電子工学科 竹野茂治

1 はじめに

[3] では、あらかじめ作成しておいた URL リストファイルを参照し、その URL の指す HTML ファイルを一度に取得する、という csh スクリプトを紹介しました。

本稿では、そのスクリプトで取得した HTML ファイルを調べて、

- 前に取得したものから更新されているかどうかを調べる
- 更新されているものだけをブラウザで参照する

ということを行う csh スクリプトを作成することにします。ただしそのためには、[3] の csh スクリプトも修正する必要がありますので、それについても適宜説明していきます。

2 設計

今回の目標を達成するには、基本的には [3] で作成した csh スクリプト wwwcheck1.csh を実行するときに、以前に持ってきてあったファイルを別な場所に保存した上で、同じ WWW ページを新たに取得して、それらを比較すればいいので、wwwcheck1.csh の修正と、参照用のスクリプト wwwcheck2.csh は、以下のように考えればいいだろうと思います:

- wwwcheck1.csh の修正 :
  1. 前に取得してあったファイルを別名、または別のディレクトリへ移動してから新しいものを取得するようにする
  2. ブラウザの立ち上げ部分は不要なので削除する
- wwwcheck2.csh :
  3. 古いものと新しいものの違いを調べ、違いがある場合はそれを記録するか、違うものを別のディレクトリにコピーする

4. その記録に基づいて (または別なディレクトリにコピーしたものを) ブラウザで参照する

ファイルの違いを調べるには、Unix に標準的に用意されている diff というコマンドで行えます。

- `diff` *file1 file2* : *file1* と *file2* の違いを出力する

diff の出力を見れば、それらのファイルのどこがどのように違っているかも知ることができますが、ここでは単純に「違いがあるかどうか」だけを知るために使用することにします。それには、csh スクリプトのファイル検査式 `-z` を利用すればいいでしょう:

```
diff file1 file2 > tmpf
if ( ! -z tmpf ) then
    # 違いあり
else
    # 違いなし
endif
```

まず、リダイレクトを用いて diff の出力を一時ファイル tmpf に落としていますが、diff は違いがない場合は何も出力しませんので、その一時ファイルのサイズが 0 でないかどうかで違いがあるかないかを判別できることになります。

"`-z` *file*" は、*file* のファイルサイズが 0 のとき真となる式ですが、その前に `!` (否定) がついていますので、この if 文の条件文は「tmpf のサイズが 0 でないとき真」となります。つまり違いがある場合に if ブロックが、違いがない場合に else ブロックが実行されることになります。

3 wwwcheck1.csh の改良

まずは、2 節の 1., 2. にあるように、[3] の wwwcheck1.csh を改良してみます。ブラウザの立ち上げが不要なので、コマンドラインオプションは `-q`, `-v`, `-h` のみをサポートすることにします。これらは同時に指定することはないので、ひとつのオプションだけ考えるようにすればいいでしょう。

```
#! /bin/csh -f
#### 初期設定 ####
set workd = ~/wwwcheck      # 作業ディレクトリ
```

```
set urllistf = $workd/urllist     # URL リストファイル
set datad = $workd/data     # 取得したファイルの置き場所
set oldd = $workd/old     # 以前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 myverbose = 0     # 冗長出力をするかどうか

##### コマンドライン解析 ####
if ( $#argv > 0 ) then
    switch ( $argv[1] )
    case -v:
        set myverbose = 1
        breaksw
    case -q:
        set myverbose = 0
        breaksw
    case -h:
    default:
        echo "csh -f $0 ([option])"
        echo " [option]:"
        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
endif

##### 実行部分 ####
mv $datad/*.html $oldd     # 前回取得したファイルを移動
set j=1
foreach url ( `grep "^#" $urllistf` )
    if ( $myverbose ) then
        echo "[$j] $url ==> $j.html"
    endif
    $wget "$url" > $datad/$j.html
    @ j ++
end
```

[3] の 7 節のスクリプトと比べると、先頭に workd, oldd の変数が追加されていますが、workd は、“~/wwwcheck” が繰り返し使われるのでそれを置き換えたもので、oldd は前に取得したファイルを保存するディレクトリです。

繰り返し現れる名前は、変数値にしておいてその変数を使用するようにしておくと、それを変更する必要がある場合にその一箇所だけを変更すればよくなります。

コマンドラインオプションのチェックは foreach 文から if 文に変えて、最初のオプション `$argv[1]` のみを調べるようにしてあり、さらに `-b`, `-nb` のオプションを削除していますが、最も大きな変更は実行部分の最初の行にある mv の行です。mv はファイルの移動やファイル名の変更に使われる Unix 標準コマンドで、以下のように使用します。

- `mv` *file1 file2* : ファイル名 *file1* を *file2* に変更 (または移動)
- `mv` *file1 ... filen dir* : ファイル *file1 ... filen* をディレクトリ *dir* へ移動

よって、このスクリプトの mv の行により、`$datad` にあった`*.html` (`.html` で終わる名前を持つすべてのファイル) が `$oldd` へ移動されることになります。

なお、[3] で説明したように、`*.html` は、mv の実行前に実際のファイル名に展開されて実行されるので、実際には

```
mv $datad/1.html $datad/2.html $oldd
```

のようなものが実行されることになります (もちろん実際は `$datad` や `$oldd` もその値に展開されてから実行されます)。ところがその展開の際に、`$datad` 内にそのパターンにマッチするファイルが一つもない場合には、そのファイル名パターンの展開に失敗し csh がエラーを出してしまいますので、本当はそのようなファイルがあるかどうかチェックしてから実行するようにした方が安全なのですが、とりあえずここではそのチェックは省略することにします[1]。

4 wwwcheck2.csh スクリプト

wwwcheck2.csh は、2 節の 3., 4. にあるように、2 つのディレクトリ `$oldd` と `$datad` のファイルを比較すればいいのですが、ここでは diff で一つ一つ対応するファイルを比較する必要があるので、ファイルパスからファイル名部分を取り出す必要があります。以下に、簡単な疑似コードを上げて説明します。

```
foreach newf ( $datad/*.html )
    # $datad 内のファイル $newf と、これと同じファイル名の
    # $oldd 内のファイルとを diff で比較し、その結果を
```

[1] 多分 `if ( ‘ls $datad | grep '.html$' | wc -l‘ ) then` という if ブロックに入れればいいだろうと思います。

```
        # $tmpf に保存する
        if ( ! -z $tmpf ) then
            # 違いあり
        else
            # 違いなし
        endif
    end
```

この疑似コードの最初のコメント部分に書いたように、変数 newf には、

```
"~/wwwcheck/data/2.html"
```

のような値 (ファイルパス) が入っていますから、ここからディレクトリ名を抜きとった “2.html” というファイル名のみを取り出して、`$oldd/2.html` と `$newf` を diff で比較しなければいけません。

このように、ファイルパスからファイル名のみを取り出したり、ディレクトリ部分のみを取り出すには、basename, dirname という Unix 標準コマンドを使用します:

- `basename` *path* : ファイルパス *path* のファイル名部分を出力
- `dirname` *path* : ファイルパス *path* のディレクトリ部分を出力

これらの出力は標準出力なので、csh スクリプトでは ‘ ‘ (バッククォート) を使ってその出力を取得できます:

```
set fname = ‘basename $newf‘
set oldf = $oldd/$fname
```

ファイルに違いがあった場合の実際の作業は、それらのファイル名 (`$newf`) をリストの要素とするリスト変数を作っておきます。最後にそれを引数としてブラウザに渡して実行するわけです。このリスト変数の作成は、foreach の前に、

```
set newflist = ( )      # 空リスト
```

を入れて、foreach の if ブロックに

```
set newflist = ( $newflist $newf )
```

とすればいいでしょう。右辺のカッコ内の最初の要素は、このリスト変数自体の名前になっていますが、これは直前に設定されているこのリスト変数の値 (各要素をスペース区切りで並べた文字列) で、その最後に `$newf` を追加したことになりますので、これで順次違いのあった `$newf` が要素として追加されていくことになります。

また、今回は違いがない場合は特に何もする必要がないので、疑似コードの else ブロックは不要です。

なお、上の疑似コードでは、`$oldd` 内に `$fname` という名前のファイルが存在するとして書いていますが、実際にはそうではない場合もありえます。例えば URL リストファイルに新たな URL を追加した後に wwwcheck1.csh を実行すると、前回は取得してないのに今回は取得した、というファイルができることになりますから、この場合は `$oldd` 内にはそれに対応するファイルはありません。この場合も `$newf` は一応更新されていると見て、これも newflist に追加することにします。

以上をまとめると、ほぼ以下のようになります:

```
set newflist = ( )
foreach newf ( $datad/*.html )
    set fname = `basename $newf`
    set oldf = $oldd/$fname
    if ( -f $oldf ) then
        diff $oldf $newf > $tmpf
        if ( ! -z $tmpf ) then
            set newflist = ( $newflist $newf )
        endif
    else
        set newflist = ( $newflist $newf )
    endif
end
```

diff の外側に、"`-f $oldf`" による if 文を追加していますが、"`-f` *file*" は、*file* が存在するときに真となる式なので、`$oldf` が存在すれば diff で違いを調べ、存在しなければ else ブロックで newflist に `$newf` を追加することになっています。

これに初期設定を追加すればほぼ wwwcheck2.csh ができあがります。

5 URL リストファイルの書式の変更

ここでは、wwwcheck2.csh にともなう URL リストファイルの書式の変更の必要性と、それにともなう wwwcheck1.csh の修正について説明します。

[3] では、URL リストファイルにコメントの記述を許すようにし、それにより URL リストファイルから各 URL エントリを削除、追加することを容易にしました。ところが、wwwcheck2.csh を行うことを考えると、現在の URL リストファイルには問題があります。3 節の wwwcheck1.csh では、取得した HTML ファイルを [n].html という名前で保存しますが、この n は、「URL リストファイルの先頭から何番目」であるかという数字になっています。

しかしそれだと、上の方の URL エントリをコメントアウトして削除した場合、それより下のエントリに対する n の数字が、次回の取得ではずれてしまうことになります。例えば、URL リストファイルが、

```
http://www.iee.niit.ac.jp/
http://www.yahoo.co.jp/
```

である場合は、`http://www.yahoo.co.jp/` は 2.html として保存されますが、それを

```
# http://www.iee.niit.ac.jp/
http://www.yahoo.co.jp/
```

のように 1 行目をコメントアウトして削除した場合は、3 節の wwwcheck1.csh では、`http://www.yahoo.co.jp/` は 1.html として保存されてしまいます。wwwcheck1.csh では、これ自体は問題はありませんが、wwwcheck2.csh で古いものと比較する場合にファイル名のずれは問題となり、正しく違いを調べることができなくなります。

よって、各 URL エントリと保存するファイル名の間に、URL リストファイルの更新 (URL エントリの追加や削除) に左右されないしっかりした対応をつける必要があります。これを行うには、例えば次のような 2 つの方法が考えられます。

- 削除するエントリは、コメントアウトするのではなく別な方法で無視することにし、その無視したものも数えることで URL エントリと番号との対応を固定化する。そして、追加するエントリは必ず末尾に追加することにする。
- URL リストの書式を変更し、各行を

  url *name*

  の形式で書き、その *url* の HTML を *name*.html という名前で保存することとする。そしてこの *name* の部分は重複しないようにする。この場合は、URL エントリの削除や追加、並べ換えなども任意に行える。

この 2 つの方法では、明らかに後者の方が柔軟性は高いのでそちらを採用して、URL リストファイルの書式をそのように変更することにします。*name* は、とりあえずは 1,2,... のようなもので構いません。

URL リストファイルをこのようにすると、wwwcheck1.csh ではそこから各行の 1 列目の要素と 2 列目の要素を別々に読み出す、という作業が必要になります。これを行うのにも、例えばリスト変数を利用する方法と、AWK を利用する方法があります。

リスト変数を利用して 1 列目、2 列目を取得する方法とは、以下のようにするやり方です:

```
set urldata = ( `grep "^#" $urllistf` )
while ( $#urldata > 1 )
    set url = "$urldata[1]"
    set name = "$urldata[2]"
    $wget "$url" > $datad/$name.html
    shift urldata
    shift urldata
end
```

最初のリスト変数 urldata には、URL リストファイルのコメント行以外の各行の要素が入るので、

([1 行目の *url*] [1 行目の *name*] [2 行目の *url*] [2 行目の *name*] ...)

のようになります。よってそれを先頭から 2 つずつ (`$urldata[1]` と `$urldata[2]`) 取り出していけばいいことになります。上のコードでは最後に shift の行が 2 行ありますが、shift はリスト変数の先頭の要素を削除するので、これを 2 回行うことで先頭の要素を 2 つ取り除くことになります。

しかしこのリスト変数を用いる方法では、ある行の URL エントリで *name* 部分を書き忘れてしまった場合は、そこから先はずれてしまって問題が起きてしまいます。例えば、URL リストファイルが、

```
http://www.iee.niit.ac.jp
http://www.yahoo.co.jp      yahoo
```

のように 1 行目に *name* 部分を忘れてしまった場合、リスト変数 urldata の内容は

```
( "http://www.iee.niit.ac.jp" "http://www.yahoo.co.jp" yahoo )
```

となるので、最初の実行で

url = “`http://www.iee.niit.ac.jp`”
name = “`http://www.yahoo.co.jp`”

となってしまうことになります。これを防ぐには、各行に 2 列ずつのエントリが書かれていることを確認するといいのですが、それにはやはり AWK などが必要になります。

今度は、その AWK を使う方法を紹介しましょう。AWK は、非常に高機能なフィルタ (プログラム言語) ですが、ここで使用するのは、

- コメント行を無視する
- 各行の 1 列目、2 列目のみを取り出す

という機能ですのでそれだけを紹介します。AWK の詳しい機能については、[4] などを参照してください。

AWK でファイルのコメント行以外の行の 1 列目、2 列目のみを出力させるには、それぞれ以下のようにすればできます:

1. `awk '!/^#/{print $1}'` *file*
2. `awk '!/^#/{print $2}'` *file*

中カッコの前の / / 内に書かれているのは、[3] の grep のところでも紹介した正規表現で、コメント記号から始まる行を意味しています。その前に否定を意味する ! がついているので、いずれも「コメント行以外の行に対して中カッコ内を実行」という意味になります。AWK では、$1 が各行の 1 列目の要素 (スペースまたはタブ区切りの最初のフィールド要素)、$2 が 2 列目の要素を意味しますので、それを AWK の print 命令で出力することで、コメント行以外の行の 1 列目、2 列目を取り出すことができます。

もしコメント以外の行で 1 列しかデータが含まれない行があれば、上の 1. と 2. では、必ず 1. の方の出力する要素の個数が多くなります。3 列以上の列を持つ行や、列を持たない行 (空行) を不正な行と見なさないことにするなら、上の出力の要素の個数を比較することで、不正な行があるかないかを判別できます。1., 2. の出力をそれぞれリスト変数にすればそのような判別が行えます:

```
set urldata = ( `awk '!/^#/{print $1}' $urllistf` )
set namedata = ( `awk '!/^#/{print $2}' $urllistf` )
if ( $#urldata == 0 || $#urldata != $#namedata ) then
    echo "$urllistf に不正な行が存在するようです。"
    exit
endif
```

後は、この 2 つのリスト変数の要素をひとつずつ使っていけばいいわけです。

```
while ( $#urldata > 0 )
    set url = "$urldata[1]"
    set name = "$namedata[1]"
    $wget "$url" > $datad/$name.html
    shift urldata
    shift namedata
end
```

なお、shift を利用すると、この while 文が終わったときにリスト変数の要素がなくなってしまいますから、もしその後で同じリスト変数を使用する場合は、shift を使わずに行う必要があります (今回は必要ありません):

```
set j = 1
while ( $j <= $#urldata )
    set url = "$urldata[$j]"
    set name = "$namedata[$j]"
    $wget "$url" > $datad/$name.html
    @ j ++
end
```

これだと、参照する添え字変数 j の値が増えるだけで、リスト変数の内容は変化しません。なお、この while 文は、最初に j を 1 とし、while ブロックの最後で j の値を一つ増やしていて、j が $#urldata に達するまでのループなので、丁度 C 言語での for 文に相当します。

6 全ソースファイル

ここまでの、wwwcheck1.csh と wwwcheck2.csh の全ソースファイルを紹介します。

wwwcheck1.csh のソースファイル:

```
#! /bin/csh -f
# wwwcheck1.csh
# shige 01/18 2008
#
#### 初期設定 ####
set workd = ~/wwwcheck      # 作業ディレクトリ
set urllistf = $workd/urllist      # URL リストファイル
set datad = $workd/data      # 取得したファイルの置き場所
```

```
set oldd = $workd/old     # 以前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 myverbose = 0     # 冗長出力をするかどうか

##### コマンドライン解析 ####
if ( $#argv > 0 ) then
    switch ( $argv[1] )
    case -v:
        set myverbose = 1
        breaksw
    case -q:
        set myverbose = 0
        breaksw
    case -h:
    default:
        echo "csh -f $0 ([option])"
        echo " [option]:"
        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
endif

##### 実行部分 ####
mv $datad/*.html $oldd     # 前回取得したファイルを移動
set urldata = ( `awk '!/^#/{print $1}' $urllistf` )
set namedata = ( `awk '!/^#/{print $2}' $urllistf` )
if ( $#urldata == 0 || $#urldata != $#namedata ) then
    echo "$urllistf に不正な行が存在するようです。"
    exit
endif

while ( $#urldata > 0 )
    set url = "$urldata[1]"
    set name = "$namedata[1]"
    if ( $myverbose ) then
        echo "$url ==> $name.html"
    endif
    $wget "$url" > $datad/$name.html
    shift urldata
    shift namedata
```

```
end
```

wwwcheck2.csh のソースファイル:

```
#! /bin/csh -f
# wwwcheck2.csh
# shige 01/18 2008
#
#### 初期設定 ####
set workd = ~/wwwcheck      # 作業ディレクトリ
set urllistf = $workd/urllist      # URL リストファイル
set datad = $workd/data      # 新しいファイルの置き場所
set oldd = $workd/old      # 古いファイルの置き場所
set tmpf = $workd/tmpf-$$      # 一時ファイル
set browser = "w3m -N"      # ブラウザ定義
#set browser = firefox

##### 実行部分 ####
set newflist = ( )
foreach newf ( $datad/*.html )
    set fname = `basename $newf`
    set oldf = $oldd/$fname
    if ( -f $oldf ) then
        diff $oldf $newf > $tmpf
        if ( ! -z $tmpf ) then
            set newflist = ( $newflist $newf )
        endif
    else
        set newflist = ( $newflist $newf )
    endif
end
rm $tmpf

if ( $#newflist > 0 ) then
    $browser $newflist
endif
```

一つだけ説明を追加します。wwwcheck2.csh では一時ファイルの名前が “tmpf-$$” のように設定されていますが、$$ は csh スクリプトではその csh のプロセス番号に展開され、よって “tmpf-2536” のようなものになります。これは一時ファイルを作るときによく使われる手法で、こうしておけばそのようなスクリプトを (一人あるいは複数

人が) 同時に立ち上げても、それらの一時ファイル名はバラバラになり、同じ一時ファイルを使ってしまうことはなくなります。

7 2 つのスクリプトの統合について

今回は 2 つのスクリプトファイルの形で紹介しましたが、これらには共通の設定も多く、またこの 2 つを連続して実行することも多いだろうと思いますので、いっそこの 2 つを統合して一つのスクリプトにするということも考えられます。

その場合は、[3] の wwwcheck1.csh のオプション -b, -nb をサポートして、-nb なら後半の wwwcheck2.csh の部分を実行しないとか、また逆に wwwcheck2.csh の部分のみを実行するオプション (例えば -bo) をサポートすることなども考えてみるといいでしょう。

ところで、そのような csh スクリプトの統合には別の形もあり、別の csh スクリプト wwwcheck3.csh を用意して、その中で

```
#! /bin/csh -f
csh -f wwwcheck1.csh
csh -f wwwcheck2.csh
```

のようにする、という手もあります。もちろん、この wwwcheck3.csh でコマンドライン解析をして、上に述べたオプションを実現したり、あるいは wwwcheck1.csh 用のオプションを受けとった場合はそれを wwwcheck1.csh にオプション引数として渡す、ということもできます。例:

```
set check1op = ""     # wwwcheck1.csh に渡すオプション
set check1On = 1     # 1 なら wwwcheck1.csh を実行する
set check2On = 1     # 1 なら wwwcheck2.csh を実行する
foreach option ( $argv )
    switch ( $option )
    case -v:
    case -q:
        # この 2 つのオプションは wwwcheck1.csh に渡す
        # set check1op = "$check1op $option"
        # 2 つ以上のオプションを渡すときは上のように書く
        set check1op = "$option"
        breaksw
    case -b:
        # wwwchech2.csh を実行する
```

```
            set check2On = 1
            breaksw
        case -nb:
            # wwwchech2.csh を実行しない
            set check2On = 0
            breaksw
        case -bo:
            # wwwchech2.csh のみ実行する
            set check1On = 0
            set check2On = 1
            breaksw
        endsw
    if ( $check1On ) then
        csh -f wwwcheck1.csh $check1op
    endif
    if ( $check2On ) then
        csh -f wwwcheck2.csh
    endif
```

このような統合を行う場合は、wwwcheck1.csh, wwwcheck2.csh も残っているので、あいかわらずそれぞれを単独で使えることになるのですが、ファイルが別々になっていますので保守性は悪くなります。特に、wwwcheck1.csh, wwwcheck2.csh では共通の設定がありますので、それを変更する場合は両方のファイルを忘れずに修正しなければいけません。

共通の設定は、それらを wwwcheck1.csh, wwwcheck2.csh から分離し、別の一つのファイルにまとめて書いておいて、それを利用する csh スクリプト (wwwcheck1.csh, wwwcheck2.csh) でその共通の設定ファイルを source 内部コマンドで読み込む、という手もありますが、後で修正する場合は、結局複数のファイルの中身を見ないとわからないでしょうから、その方法でも保守性がよくないという点はあまり改善しないだろうと思います。

8 最後に

今回は、WWW ファイルの比較を行って参照するスクリプトを題材として、

> ファイルの違いのチェック、ファイル検査式 (`-f`, `-z`)、mv、一時ファイルの扱い、少し複雑なリスト変数の使い方、AWK との組み合わせによるファイルの列データの取得

などを紹介しました。特に、AWK との組み合わせは強力で、それにより csh スクリプト内部で乱数を発生させたり、表計算や実数計算などをさせることができるようになり、csh スクリプトの足りない機能を AWK で補うことで応用が広がります。

いずれ、例えば乱数を利用した簡単なゲームのようなものなども紹介していきたいと思います。

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)

[3] 竹野茂治、「WWW ページの一括取得 csh スクリプト」 (2008)

[4] 竹野茂治、「AWK に関する基礎知識」 (2006)