2006 年 9 月 29 日

# AWK によるシェル作業

新潟工科大学 情報電子工学科 竹野茂治

## 1 はじめに

以前、HTML ファイルを手元に保存して、それを AWK で加工する例を紹介しました ([4],[5],[6])。

私は、それに必要な複数の HTML ファイルを、ブラウザを使わずにまとめて取得するようなシェルスクリプト[1]を書いてそれを使っていて、今度はそれを紹介しようと思ったのですが、以下のような問題があることに気がつきました。

- これまでは Unix でも MS–Windows でも共通に動く物を書いてきたのに[2]シェルスクリプトの話にすると、ほとんどの MS–Windows ユーザには使えないし意味がない話になってしまう
- MS–Windows 上で、Unix 同様のシェルスクリプトを使うようにするためには、cygwin[3]を使うか、その他の Unix シェル互換のソフトを使う必要があるが、どちらにしてもやや面倒
- MS–Windows 上でそれを行う場合、MS–Windows のバッチファイルではやや不十分
- MS–Windows 上では、バッチファイルより少し高級なスクリプティング環境として WSH があるが、それを勉強するのは私の能力が不十分[4]

考えてみれば、AWK 内部でも system() 関数を使えばコマンドを実行できるので、いっそついでにこの話も AWK で書いてみることにしました。AWK 上ならば、プログラミングもできますし、関数も使えますし、Unix でも MS–Windows でもどちらでも動くものを作ることもできますし、シェルプログラムよりもいい点があります。

よって今回は、[5],[6] で紹介した作業、および HTML ファイルを持ってくる作業を題材にして、AWK を使ったシェルプログラミング (のようなもの) について考えてみたいと思います。

[1] シェルスクリプト とは MS–Windows でいうバッチファイルのようなもので、簡単なプログラミング構造を利用してコマンドの実行ができます。

[2] 厳密に言うと、これまでの話でも色々問題があるようです。それについては 8 節で述べます。

[3] **cygwin** は MS–Windows 上で Unix のコマンドを動作させる環境を作るフリーソフトです (`http://cygwin.com`)。

[4] 能力、および時間とやる気が不十分です。どなたか私の代わりに書きませんか ?

# 2 今回行う作業

今回行う作業は以下の通りです。

1. Yahoo! ニュース
   - `http://headlines.yahoo.co.jp/hl`

   の記事の一覧 HTML ファイル (複数ページに分かれている場合がある) を手元に持ってくる
2. それを、[5], [6] で示した方法で加工し、あらたな HTML ファイルを作成する

2. の方はそう面倒ではありませんので、主に 1. に関する話を紹介します。

# 3 wget

WWW ページの HTML ファイルを取得するには、ブラウザを使ってその WWW ページにアクセスし、そのブラウザの機能を使ってファイルとして保存するという方法がありますが、そのやり方だと自動化はできませんし、複数のファイルを取得するのは面倒です。

wget はコマンドライン[5]で URL を指定して、その URL の HTML ファイルを取得し手元に保存してくれるフリーソフトです。これは非常に高機能で、一度にたくさんのファイルをダウンロードしたりすることもできるのですが、今回はこの wget を使ってひとつひとつ必要なファイルをダウンロードすることにします。

wget は、最近の Unix では既にインストールされているかもしれませんが、以下に Unix 用のソースファイル、MS–Windows 用の実行バイナリファイルなどがあります。小さいソフトですし便利なものなので、入れておいて損はないでしょう。

- wget 公式サイト: `http://www.gnu.org/software/wget/wget.html`
- wget のソースの置き場所: `http://ftp.gnu.org/pub/gnu/wget/`
- MS–Windows 用バイナリ等: `http://xoomer.alice.it/hherold/`
- MS–Windows 用バイナリ等: `http://users.ugent.be/~bpuype/wget/`

[5]コマンドライン とは、MS–Windows でいう DOS プロンプトやコマンドプロンプト上で、コマンド名やそれに与えるパラメータを文字列として入力して実行する際の、その命令行のことを言います。Unix ではソフトの実行の標準的な方法です。

インストールや使い方については詳しくは説明しませんが、ひとつだけ注意をしておきます。WWW ページへのアクセスでプロクシサーバを利用している場合は、環境変数 http_proxy にそれを設定しておくと、wget は自動的にそれを見るようになります。MS–Windows の場合、例えばプロクシサーバが 192.168.0.1 で、ポートが 8080 の場合は

```
set http_proxy=http://192.168.0.1:8080
```

のように設定します。Unix の場合でも環境変数名は同じです。これは初期設定ファイルで設定することもできるようですが、それについては wget のマニュアルを参照してください。

今回は、wget の使い方としては、単純に以下のような使い方のみを利用します。

```
wget -O [ファイル名] "[URL 名]"
```

このようにすると、[URL 名] として指定した URL の WWW ページの HTML ファイルを、[ファイル名] として指定したファイルとして保存してくれます。この場合、そのページ内に貼られている画像ファイルなどはダウンロードしません[6]。

# 4 複数ページの記事一覧ファイル

Yahoo! ニュースの一覧は、記事の件数が多いと複数の WWW ページに分かれています。例えば Yahoo! の社会ニュースの場合、記事の一覧のページの上端と下端に

1/6 ページを表示 (合計 123 件) ...

のように書かれていて、その次のページへのリンクが貼られています。

一覧の先頭のページの URL は、例えば社会ニュースの場合は

```
http://headlines.yahoo.co.jp/hl?c=soci&t=l
```

ですが、2 ページ目、3 ページ目の URL は、その後ろに追加ページ番号がついて

```
http://headlines.yahoo.co.jp/hl?c=soci&t=l&p=1   (2 ページ目)
http://headlines.yahoo.co.jp/hl?c=soci&t=l&p=2   (3 ページ目)
```

[6] wget のオプションを変えればそのページに貼られている画像ファイルなどもダウンロードさせることはできますが、今回は不要なので説明は省略します。

のようになっています。“soci” の部分がニュースの分野をあらわしていて、例えばコンピュータニュースならばここが “sci” に、スポーツニュースならば “spo” のように変わります。

今回は、

1. まず一覧の先頭ページをダウンロード (取得)
2. そのページから全部で何ページあるのかを読み取る
3. それに合わせて URL を指定してすべてのページをダウンロードする
4. それを [5], [6] に従って加工する

とすればいいわけです。

AWK には system() という関数があり、これを使って外部のコマンド呼び出すコマンドラインを実行することができます。また、そのコマンドラインの生成やファイル名、URL 文字列の作成は、文字列を地道に連結してもできますが、sprintf() 関数を使用すると便利です。

- system(s) = 文字列 $s$ をコマンドラインとして OS 上で実行し、その終了コードを返す
- sprintf(f,...) = printf(f,...) で出力されるような書式化された文字列を返す

# 5 全体のおおまかな構造

今回作成する AWK スクリプトは、入力は特になくてシェル作業を行うだけなので BEGIN ブロックだけで足ります。4 節の作業を疑似コードにしてみると以下のようになります。

```
##### プログラム本体 #####
  BEGIN{
      # (1) 先頭のページを soci-001.html として取得
      # (2) そこから全部で何ページあるかを読みとる (= pages とする)
      for(j=1;j<pages;j++){
         # (3) 先頭ページの URL の最後に "$p=j" をつけて HTML ファイルを
         #     取得し soci-[j+1].html として保存する
      }
      # (4) awk -f yahoo2.awk soci-001.html soci-002.html ... \
```

```
    #       > soci-lst1.html
    #     を system() で実行する
    # (5) awk -f yahoo3-soci.awk -f yahoo3.awk soci-lst1.html \
    #       > soci-lst2.html
    #     を system() で実行する
}
```

なお、yahoo2.awk は、[5] で紹介した、複数の一覧ファイルから余分なものを除いて一つの HTML ファイルに連結するための AWK スクリプトで、yahoo3-soci.awk, yahoo3.awk は、それぞれ [6] で紹介した、連結されたリストからジャンル分けする際のパターン定義部分の AWK スクリプトと共有部分の AWK スクリプトであるとします。

(4),(5) はコマンドラインを生成して system() に渡して実行するだけですし、(1),(3) は

```
wget -O soci-001.html "http://.../hl?c=soci&t=l"
```

(URL は長いので途中省略) 等を system() で実行するだけなので、特に考える必要があるのは (2) ということになります。しかし、(2) も getline, match(), substr() を使う程度で、それ程難しくはありません。

# 6 各部分の作成

## 6.1 ページ数の取得

この節では各部分を実際に作成していきますが、まずは (2) から考えます。これは、HTML のページ数が書かれている行を抜き出してその情報を取得すればよいのですが、今回はすべて BEGIN ブロック内だけで入力は仮定していませんから、明示的に getline を使ってファイルを読みこんでそこから取得することになります。

- getline < [file]: [file] として指定したファイルから現在行の次の行を読みこんでそれを新たな現在行とし $0 等に代入する。読み込みに成功すると 1, ファイルの最後に達した場合は 0, 読み込みに失敗したら $-1$ を返す。
- close([file]): 開かれている [file] を閉じ、[file] に対する作業を終了する。

これを利用すれば、

```
fname = "soci-001.html"
while(getline < fname){
```

```
    # $0 に各行が代入される
}
close(fname)
```

のようにして指定したファイルの各行の処理を、AWK の通常の入力行に対する処理と同じように行うことができます。

今回は、その行の中から

1/6 ページを表示 (合計 123 件) ...

のような行を取得すればいいのですが、これを例えば日本語のパターンを使って

```
$0 ~ /ページを表示/
```

のようにすることもできなくはありませんが、このようにすると漢字コード (スクリプトの漢字コード、入力ファイルの漢字コード、および使用している AWK が対応している漢字コード) によってうまくマッチできないことがありますので注意が必要です[7]。

調べてみると、この一覧の HTML ファイルでは "1/6" のような文字列が行頭にあるのはこの部分だけのようなので以下のようなパターンを使って取り出すことにします。

```
$0 ~ /^[1-9][0-9]*\/[1-9]/
```

この正規表現は、

`[0-9]`: 0 から 9 までの数字 1 文字
`[0-9]*`: 0 から 9 までの数字 0 文字以上の繰り返し
`[1-9][0-9]*`: 0 以外で始まる 1 文字以上の数字文字列
`^[1-9][0-9]*\/[1-9]`: 行頭が 0 以外で始まる数字列 + '/' + 0 以外の数字

を意味します。

この行の先頭部分から、'/' 以降の分母の部分を `match()` と `substr()` を使って取り出します。

- `match(s,r)` = 文字列 $s$ で正規表現 $r$ にマッチした部分文字列の先頭位置を返す (予約済み変数 RSTART に先頭位置を、RLENGTH に部分文字列の長さをセットする)

[7] 元々 Yahoo! ニュースの HTML ファイルの漢字コードは EUC–JP のようなので MS–Windows 環境では多分うまくいきません。

- substr(s,n,len) = 文字列 $s$ の $n$ 番目の文字から $len$ 文字分の文字列 ($len$ を省略した場合は最後まで)

この行内には「'/' + 数字」の部分はここにしかないようですので、これを目当てにして以下のようにして取り出します。

```
match($0,"/\/[1-9][0-9]*/)
pages = substr($0,RSTART+1,RLENGTH-1)+0
```

この substr() の引数ですが、RSTART+1 は '/' に続く数字列の先頭部分を意味しますし、'/' を取り除くためにマッチした長さから 1 を引いた RLENGTH-1 を切り出しています。

また、最後の "+0" は、変数 pages を数字化するためにつけています。通常、substr() で切り取ったものは文字列として返されますから、そのままだと文字列として変数に代入されますが、"+0" という数字演算をつけることで、その文字列が数字として評価されて代入されることになります。

AWK では、文字列と数字の明示的な区別がなく、評価されるところで適宜自動的な変換が行われるので、この変数を次に使用するところが数字演算の中などであればその時に自然に数字に変換されて使われるので問題はないのですが、例えば次に使用するところが "==" や ">" などの比較演算だとしたら、これらの演算は文字列も数字も受けつけて、それぞれで別の評価をしてしまいますので文字列のままと見なされると誤動作の原因となりえます。

よって、文字列を確実に数字にしたい場合は "+0" をつけるのが AWK の常套手段となっています。逆に数字を確実に文字列にしたい場合は

```
n = n ""
```

のように空文字列を連結します[8]。

以上をまとめると、以下のようにすれば pages を取得できることになります。

```
##### pages の取得 #####
pages = 0
fname = "soci-001.html"
while(getline < fname){
    if($0 ~ /^[1-9][0-9]*\/[1-9]/){
        match($0,/\/[1-9][0-9]*/)
```

[8] これらの明示的な変換手法は、AWK の原典である [7] の 2–2 節にも書かれています。

```
            pages = substr($0,RSTART+1,RLENGTH-1)+0
            break
        }
    }
    close(fname)
```

## 6.2 コマンドラインの作成

wget のコマンドラインを作るときは、URL の共通する部分をあらかじめ

```
topurl = "http://headlines.yahoo.co.jp/hl"
```

のように定義しておいて、使うところでそれに必要な文字列をつけたすようにすれば、長い URL をたくさん書かなくて済みますし、後で URL を変更する場合も楽です。また、ニュース分野も容易に変えられるように

```
field = "soci"
```

のように定義しておくことにします。

作成するファイル名は、この分野名とページ番号を使って

```
soci-001.html, soci-002.html, ...
```

のような名前にしますが、このような文字列は、

```
fname = sprintf("%s-%03d.html",field,j)
```

のようにして生成できます。sprintf() は C 言語の printf() と同様の構文で書式化された文字列を生成でき、

- %s $= field$ の値が文字列としてそれに置き換わる
- %d $= j$ を数字とみてそれを文字列に直したものがそれに置き換わる

のようになります。なお、%03d は、%d にさらに桁指定等がついているのですが、'3' は 3 桁で表示、'0' は $j$ の値が 3 桁より少ない場合は上位の桁は 0 で埋めることを意味します。よって、$j = 1$ の場合は "001", $j = 12$ の場合は "012", $j = 123$ の場合は "123" のような文字列に置き換わることになります。

これにより、wget のコマンドラインは、

```
topurl = "http://headlines.yahoo.co.jp/hl"
field = "soci"
fname = sprintf("%s-%03d.html",field,1)
url = sprintf("%s?c=%s&t=l",topurl,field)
cmd = sprintf("wget -O %s \"%s\"",fname,url)
```

とすれば $cmd$ が

```
wget -O soci-001.html "http://.../hl?c=soci&t=l"
```

(URL は長いので途中省略) という文字列になります。なお、sprintf() 内の "" の中で " という文字自体を使うときは \" と書きます。2 ページ目以降の場合は、これに "&p=1" などをつけていけばいいわけです。

そしてこれらを

```
system(cmd)
```

として実行すれば wget が AWK の外で実行されます。

また、[5] にある加工を行う場合は、pages 個数の引数をつけて AWK を実行する必要がありますので、

```
awk = "awk"
cmd = sprintf("%s -f yahoo2.awk",awk)
for(j=1;j<=pages;j++)
  cmd = sprintf("%s %s-%03d.html",cmd,field,j)
cmd = sprintf("%s > %s-lst1.html",cmd,field)
```

のようにしてコマンドラインを追加しながら作成すればいいでしょう。

なお、awk="awk" としているのは、AWK コマンド名が "gawk" などである場合に容易に修正できるように、と考えてのことです。

# 7 全体のソースコード

以上を踏まえて全体のソースコードを紹介すると以下のようになります。

```
BEGIN{
```

```
    topurl = "http://headlines.yahoo.co.jp/hl"
    if(field == "") field = "soci"
    awk="awk"

    ### (1) 先頭のページを取得
    fname = sprintf("%s-%03d.html",field,1)
    url = sprintf("%s?c=%s&t=l",topurl,field)
    cmd = sprintf("wget -O %s \"%s\"",fname,url)
    system(cmd)

    ### (2) pages を取得
    pages = getpages(fname)

    for(j=1;j<pages;j++){
        ### (3) 2 ページ目以降を取得
        fname = sprintf("%s-%03d.html",field,j+1)
        url = sprintf("%s?c=%s&t=l&p=%d",topurl,field,j)
        cmd = sprintf("wget -O %s \"%s\"",fname,url)
        system(cmd)
    }

    ### (4) [field]-lst1.html を作成
    cmd = sprintf("%s -f yahoo2.awk",awk)
    for(j=1;j<=pages;j++)
      cmd = sprintf("%s %s-%03d.html",cmd,field,j)
    cmd = sprintf("%s > %s-lst1.html",cmd,field)
    system(cmd)

    ### (5) [field]-lst2.html を作成
    cmd = sprintf("%s -f yahoo3-%s.awk -f yahoo3.awk",awk,field)
    cmd = sprintf("%s %s-lst1.html > %s-lst2.html",cmd,field,field)
    system(cmd)
}

##### pages の取得 #####
function getpages(fname,        pages)
{
    pages = 0
    while (getline < fname){
        if($0 ~ /^[1-9][0-9]*\/[1-9]/){
            match($0,/\/[1-9][0-9]*/)
            pages = substr($0,RSTART+1,RLENGTH-1)+0
```

```
            break
        }
    }
    close(fname)
    return pages
}
```

ページの取得は getpages() という関数にしてあります。全体の流れはそう難しくはないと思います。

# 8 MS–Windows ユーザの場合

ここまで書いてきて、MS–Windows の場合はいくつか問題があることがわかりました。

- Yahoo! の HTML ファイルの漢字コードは基本的に EUC–JP であるようだが、それをそのコードのまま MS–Windows 上の Shift_JIS の AWK スクリプトで処理すると問題が起こる (スクリプト内部で漢字を出力する部分、および [6] の日本語をパターンとする部分)
- HTML ファイルを Shift_JIS に変換してから処理する場合は、[5], [6] のスクリプトは、加工して作成する HTML ファイルの漢字コードが EUC–JP であるとして出力している部分の修正が必要
- スクリプトで Shift_JIS の漢字パターンを使用する場合 ([6] のスクリプト)、日本語に対応している AWK を使用しないと Shift_JIS 2 バイト目が '\'[9]である場合に問題が起こる可能性がある

以上のような問題を解決するために、MS–Windows では以下のような対処を取る必要がありそうです。

1. nkf のような漢字コード変換フィルタを利用してダウンロードしたファイルを Shift_JIS に漢字コード変換する
   これは、nkf をインストールして、今回のスクリプトの、以下の部分 (2 箇所)

   ```
   cmd = sprintf("wget -O %s \"%s\"",fname,url)
   ```

   を

   ```
   cmd = sprintf("wget -O - \"%s\" | nkf --windows > %s",
         url,fname)
   ```

[9] '\' は日本語フォントでは円記号を意味します。

のように変えれば可能です。wget は出力ファイル名が - の場合は標準出力にそれを流しますので、このようにすれば nkf を通して Shift_JIS コードのファイルに変換されます。

MS–Windows 用の nkf は、現在 (2006 09/28) は以下でダウンロードできるようです。

```
http://www.vector.co.jp/soft/win95/util/se295331.html
```

2. [5], [6] のスクリプトの putheader() の中で

```
printf " content=\"text/html; charset=EUC-JP\">\n"
```

としているところを

```
printf " content=\"text/html; charset=Shift_JIS\">\n"
```

に変える

3. 今回のスクリプト内部で呼びだす AWK を、日本語対応 (マルチバイト対応) 化された Gnu Awk を使用する
マルチバイト対応の Gnu Awk の所在については、[1] を参照してください。

とりあえず、以上のような対処によって動作させることが可能だと思います[10]。

# 9 最後に

今回は、HTML ファイルの整形の話の最後として、本来はシェルスクリプトでやる作業を AWK を使ってやる方法を紹介しました。

通常は、このような目的の場合は Perl を使うことが多いと思いますが、AWK は軽いし、system() を使えば簡単な処理なら可能なので、MS–Windows 上ではコマンドプロンプトの作業、バッチファイルなどを補佐しうると思われますし、実際今回検討、実験してみて、その可能性を十分に感じました。そのような方面での利用法でおもしろいものがあれば今後も取り上げていきたいと思います。

# 参考文献

[1] 竹野茂治、「AWK に関する基礎知識」 (2006)

[10] うちの環境: MS–Windows XP, Gnu Awk 2.15 + mb1.3 (または Gnu Awk 3.0.6 + mb1.15), nkf 2.0.5 for Win32 ではとりあえず問題ないようです。

[2] 竹野茂治、「AWK による簡単なタイプ練習ソフト」(2006)

[3] 竹野茂治、「AWK による数合てゲームの作成」(2006)

[4] 竹野茂治、「AWK による HTML ファイルの整形」(2006)

[5] 竹野茂治、「AWK による HTML ファイルの整形 その 2」(2006)

[6] 竹野茂治、「AWK による HTML ファイルの整形 その 3」(2006)

[7] A.V. エイホ、B.W. カーニハン、P.J. ワインバーガー (足立高徳訳)、「プログラミング言語 AWK」、新紀元社 (2004) (元版は 1989)

[8] D.Dougherty、A.Robbins (福崎俊博訳)、「sed & awk プログラミング 改訂版」、オライリー・ジャパン (1997)

[9] 志村拓、鷲北賢、西村克信、「AWK を 256 倍使うための本」、アスキー出版 (1993)

[10] 磯野康孝、蔵守伸一、「HTML ハンドブック」、ナツメ社 (1996)

[11] 大藤幹、「詳解 HTML & XHTML & CSS 辞典」、秀和システム (2002)

[12] 「とほほの WWW 入門」、http://www.tohoho-web.com/www.htm