インターフェイスの街角 (90) – TV 番組の検索と録画予約システム

増井 俊之

■ 括弧内：PK-AX20?→も出荷停止のようですが

TV 番組をハードディスクに録画する〝ハードディスク・レコーダ〟の売行きが好調です。全体的には録画データを DVD に書き込める製品が主流のようですが、PC ユーザーのあいだでは、録画した番組を PC で視聴したり、PC へのデータ転送が可能な製品に人気があります。

私自身も、しばらく日本電気の「PK-AX10」という製品を利用していました (現在は、後継機の PK-AX30 が販売されています)。PK-AX10 は x86 ベースの Linux で動くハードディスク・レコーダで、普通のビデオレコーダのようにリモコンで録画予約できるだけでなく、Ethernet で接続した PC で番組を視聴したり、録画された MPEG2 データに Samba 経由でアクセスできるのが特徴でした。最近は、同様の機能をもつ東芝の RD-H1 なども人気があるようです。

PK-AX10 は Linux で動いているので、sshd をインストールしてログイン可能にしたり、cron による各種の自動処理も比較的簡単でした。ただし、外部からの録画予約方法は公開されていなかったため、Windows 用の専用アプリケーションを起動し、そのなかに表示される番組表を利用する必要がありました[1]。

一方、PC に接続するタイプの TV キャプチャー機器では、PC からの録画データへのアクセスはもちろん、たいていは、Web の番組表サイト[2]を利用した録画予約も可能です。家電製品としてみれば、一般消費者向けの機器よりも扱いにくいのですが、番組表サイトが使えたり、録画予約の自動化が容易という点で、PC のユーザーにとっては

1 AX10 でインターネット上の番組表を利用するには、この専用ソフトでダウンロードする必要があります。

2 番組表サイト情報は、http://zxz.gozaru.jp/tvg/tvlink.htm によくまとめられています。

図 1 iEPG データの例

```
Content-type: application/x-tv-program-info;⇒
              charset=shift_jis
version: 1
station: NHK教育
year: 2005
month: 06
date: 26
start: 16:35
end: 17:00
program-title: 学校デジタル羅針盤[再]
```

(誌面の都合上、⇒ で折り返しています。以下同様)

このような製品のほうがおもしろいでしょう。

録画システムの現状

TV キャプチャー製品はいろいろありますが、Web 上で録画情報を提供する際のデータ形式や、EPG (Electronic Program Guide：TV 番組表) のデータ形式は事実上の標準があり、ほぼすべての製品で同じ情報が使えます。

iEPG

現在、番組の録画予約でひろく使われているのは、ソニーが 2000 年に提唱した「iEPG (Internet Electronic Program Guide)」というフォーマットです。ブラウザで Web サーバー上の iEPG 形式のデータにアクセスすると、iEPG 形式のデータがダウンロードされ、関連づけられたアプリケーションが起動します。その後は、キャプチャー機器ごとに異なる方法で録画予約がおこなわれます。

iEPG は、図 1 のようなテキスト形式のデータです[3]。

3 フォーマットは、http://350ml.net/format/iepg.html などで公開されています。

図 2 xmltv 形式のデータ例

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE tv SYSTEM "xmltv.dtd">

<tv source-info-url="http://www.ontvjapan.com/" ⇒
    source-data-url="http://www.ontvjapan.com/program/"
    generator-info-name="XMLTV" generator-info-url="http://membled.com/work/apps/xmltv/">
  <channel id="0031.ontvjapan.com">
    <display-name lang="ja_JP">ＮＨＫ総合</display-name>
    <display-name lang="en">NHK</display-name>
  </channel>
  <channel id="0041.ontvjapan.com">
    <display-name lang="ja_JP">ＮＨＫ教育</display-name>
    <display-name lang="en">ETV</display-name>
  </channel>
    ……
  <channel id="0016.ontvjapan.com">
    <display-name lang="ja_JP">東京ＭＸテレビ</display-name>
    <display-name lang="en">MX</display-name>
  </channel>
  <programme start="20050719081500 +0900" stop="20050719083000 +0900" ⇒
             channel="0031.ontvjapan.com">
    <title lang="ja_JP">ファイト</title>
    <desc lang="ja_JP">本仮屋ユイカ　緒形直人</desc>
    <category lang="ja_JP">ドラマ</category>
    <category lang="en">Drama</category>
  </programme>
    ……
  <programme start="20050802030000 +0900" stop="20050802060000 +0900" ⇒
             channel="0016.ontvjapan.com">
    <title lang="ja_JP">ヒーリングタイム</title>
    <category lang="ja_JP">情報</category>
    <category lang="en">Home/How-to</category>
  </programme>
</tv>
```

このような情報を Web サーバーで公開すれば、データをクリックするだけで簡単に録画予約がおこなえます。すでに多くの番組表サイトがありますが、個人的にデータを作成して提供したり、自動的に iEPG データを作成するのも簡単なので、TV 番組に関する情報交換の形式として便利です。

xmltv による EPG データ取得

iEPG はなかなか便利ですが、番組表のオリジナルデータがあれば、自前の番組表を作ったり、キーワードにもとづいて iEPG データを自動生成するといった高度な処理も可能です。

番組表データは世界各地の Web サイトで提供されていますが、「xmltv」[4] というシステムを利用すれば、世界中の TV 番組情報を統一された XML 形式で取得できます。

これは Edward Avis 氏らが開発中のシステムで、基本的に xmltv パッケージに含まれる tv_grab コマンドを用いて情報を取得します。ただし、国ごとに方法が異なるため、日本の番組表は tv_grab_jp、スペインの番組表は tv_grab_es といったふうに、国ごとに別のコマンドが用意されています。

xmltv で扱う番組データは、図 2 のような形式になっています。これを検索可能な状態にし、Web サーバーで iEPG 形式データとして提供すれば、検索から録画予約までの作業が簡単におこなえるようになります。

xmltv と iEPG による録画予約

xmltv と iEPG を組み合わせれば、自前の番組表や予約システムが作れます。私がみたかぎりでは、既存の TV 番組表サイトは、一覧性や検索のしやすさなどに問題がありそうなので、新しいものを作ってみるのも意義があると思います。

4 http://membled.com/work/apps/xmltv/

写真 1 PC-MV7DX

キャプチャー環境の設定

最近、私はバッファローの「PC-MV7DX」（写真 1）という TV キャプチャー機器[5]を利用して TV 番組を録画しています。Windows PC に USB で接続するのが一般的な使い方ですが、同社のネットワーク接続ハードディスク (NAS) である「LinkStation」[6]に USB で接続し、「Link de 録!」[7]というシステムとして利用することもできます。この場合は、LinkStation をサーバーとし、PC を介さない TV 番組の録画や予約も可能です。LinkStation 内蔵の Web サーバーを用いてブラウザ経由で録画予約をしたり、Windows 用の PCastLink という専用アプリケーションで録画予約や視聴などもおこなえます。

「Link de 録!」は PK-AX10 などのハードディスク・レコーダと同等の機能をもっていることになりますが、私は以前から LinkStation を利用しているため、導入コストはキャプチャー機器のハードウェア価格（2 万円弱）だけでした。LinkStation は FTP サーバーにもなるので、予約だけでなく、録画した MPEG ファイルを外部から取得するのも簡単です。

Web ブラウザで TV 番組サイトの iEPG データにアクセスすると、iEPG データがダウンロードされ、PCastLink 経由で録画予約ができます（図 3）。もちろん、自作の Linux サーバーなどでも TV 録画システムは作れますが、「Link de 録!」の場合は必要なものがほとんど揃っているぶん、取扱いが楽です。HTTP や FTP を通すようにルータを設定すれば、外部から録画予約をしたり自宅の TV を視聴することも可能です。

5 http://buffalo.melcoinc.co.jp/products/catalog/item/p/pc-mv7dx_u2/

6 http://buffalo.melcoinc.co.jp/products/catalog/item/h/hd-lan/

7 http://buffalo.melcoinc.co.jp/products/catalog/multimedia/pcasttv/ldr/

図 3 録画予約画面

番組検索/録画予約システム

xmltv で取得した番組情報をブラウザ上に表示して検索可能にしておき、録画したい番組の情報を iEPG 形式で提供すれば、番組検索/録画予約システムができあがります。具体的には、以下のような手順になります。

- xmltv の tv_grab_jp コマンドを使って番組情報を XML 形式で取得する。
- XML 形式の番組情報を JavaScript データに変換する。
- 番組検索プログラムを JavaScript で記述し、選択した番組を iEPG 形式で読み出せるようにする。

XML 形式のデータの解析には、Ruby の REXML ライブラリ[8]が便利です。

番組予約 CGI

まず、番組名や放映時間などを指定して iEPG データに変換する CGI プログラム（リスト 1）を用意します。これに引数を指定して呼び出すと、「Link de 録!」の場合は番組予約画面（図 3）が表示されて予約可能になります。

XML から JavaScript への変換

tv_grab で取得した XML 形式の番組情報は、tvprog.rb（リスト 2）で JavaScript の配列データに変換します。

8 http://www.germane-software.com/software/rexml/

リスト 1　番組予約用 CGI (reserve.cgi)

```
#!/usr/bin/env ruby
require 'cgi'

cgi = CGI.new('html3')
y = cgi.params['year'].to_s
m = cgi.params['month'].to_s
d = cgi.params['day'].to_s
s = cgi.params['start'].to_s
e = cgi.params['end'].to_s
c = cgi.params['station'].to_s
p = cgi.params['program'].to_s

s = "version: 1\r\n" +
    "station: #{c}\r\n" +
    "year: #{y}\r\n" +
    "month: #{m}\r\n" +
    "date: #{d}\r\n" +
    "start: #{s}\r\n" +
    "end: #{e}\r\n" +
    "program-title: #{p}\r\n"

cgi.out('type' =>
        'application/x-tv-program-info',
        'charset' => 'shift_jis') {
  s
}
```

図 4　番組検索ページ

ローマ字でもタイトルや内容が検索できるように、Kakasi を用いてローマ字に変換したデータも用意しています。

JavaScript による番組検索

epg.cgi（リスト 3）は、番組検索/予約ページを生成する CGI プログラムです。矢印キーで番組を検索したり、あるいは入力フィールドにローマ字を入力してインクリメンタルに番組のフィルタリングができるようにしています。

リターンキーを押すと reserve.cgi が呼び出され、選択している番組の予約画面が表示されます。

図 4 は、epg.cgi で検索画面を呼び出して `fai´ と入力してみたところです。検索は JavaScript でおこなわれるため、ページ遷移が発生せず、番組を素早くみつけることができます。

今後の展望

私が使っている Windows マシンでは、iEPG ファイルのヘルパー・アプリケーションとして PCastLink が登録されているため、今回紹介した検索システムにアクセスすると、PCastLink 経由で録画予約をすることができます。ヘルパー・アプリケーションへの登録を変更することで、その他のキャプチャー・システムも使えます。

今回の検索/予約システムではインテリジェントな処理はまったくおこなっていませんが、キーワードや予約履歴などを利用すれば、好みの番組を自動的に抽出するシステムも比較的簡単に構築できるでしょう。

xmltv や iEPG のデータの交換は容易なので、自分がすでに観た番組や観ようと思っている番組に関する情報の共有や交換にも使えそうです。2004 年 10 月号で紹介した本棚システムや 2005 年 1 月号で紹介した地図帳システムと同様、TV 番組情報を共有、交換するシステムが流行しそうな予感がします。近い将来、チャンネルや番組の数がさらに増えるので、他人の評価やインテリジェントな検索システムはますます重要になってくるでしょう。

私は、以前はあまり TV を観なかったのですが、最近はこのような検索/予約システムで録画した映像を携帯端末に入れ、電車のなかなどで観ることが増えました。いつでも、どこでも好きな TV 番組が観られる時代は、すぐそこまできているようです。

（ますい・としゆき　産業技術総合研究所）

リスト 2　XML データの変換 (tvprog.rb)

```
class String
  require "iconv"
  require "kakasi"
  include Kakasi
```

```
  def roma
    kakasi("-oeuc -Ja -Ha -Ka",self).tr('^','-')
  end

  def u8toeuc
    iconv = Iconv.new("EUC-JP", "UTF-8")
    iconv.iconv(self)
  end
end

class TVProg
  require "rexml/document"
  include REXML

  def xmlfile
    "/tmp/tv#{Time.now.strftime('%Y%m%d')}.xml"
  end

  def initialize
    get_xml(xmlfile)
    @doc = Document.new(File.new(xmlfile))
    get_ch_info
  end

  def get_xml(xmlfile)
    system "tv_grab_jp --output #{xmlfile}" unless File.exist?(xmlfile)
  end

  def get_ch_info
    @chname = {}
    @doc.elements.each("tv/channel"){ |element|
      id = element.attributes["id"]
      element.elements.each { |el|
        if el.name == 'display-name' && el.attributes["lang"] == 'ja_JP' then
          @chname[id] = el.text
        end
      }
    }
  end

  def js
    list = []
    @doc.elements.each("tv/programme"){ |element|
      desc = ''; category = ''; title = ''
      start = element.attributes["start"]
      stop = element.attributes["stop"]
      cname = @chname[element.attributes["channel"]].u8toeuc
      element.elements.each { |el|
        desc = el.text.u8toeuc if el.name == 'desc'
        title = el.text.u8toeuc if el.name == 'title'
      }
      start =~ /^(....)(..)(..)(..)(..)/
      year = $1.to_i; month = $2.to_i; day = $3.to_i
      starttime = "#{$4}:#{$5}"
      stop  =~ /^........(..)(..)/
      stoptime = "#{$1}:#{$2}"
      list << "[\"#{title}\", \"#{desc}\", \"#{(title+desc).roma}\", " +
        "'#{cname}', #{year}, #{month}, #{day}, '#{starttime}', '#{stoptime}']"
    }
    "data = [\n" + list.join(",\n") + "\n]\n"
  end
end
```

リスト 3　番組検索 CGI (epg.cgi)

```
#!/usr/bin/env ruby
require 'cgi'
require 'tvprog'
cgi = CGI.new('html3')
tbl = (0..9).collect { |i|
  "<tr><td id='list#{i}' onkeyup='keyup(event)' onmousedown='mousedown(#{i})'>list#{i}</td></tr>"
}.join("\n")

body = <<EOF
検索: <input type="text" id="query" onkeyup="do_search(event)"
             autocomplete="off" style="font-size:10pt;"><p>
<table>#{tbl}</table><p>
<div id="desc"></div>

<script type="text/javascript">
#{TVProg.new.js}
oldquery = "-"
function do_search(event){
  query = document.getElementById("query").value
  if(oldquery != query){
    reg = new RegExp(query,"g");
    matched = []
    for(i=j=0;i<data.length;i++){
      res = reg.exec(data[i][2])
      if(res != null){
        matched[j] = i
        j += 1
      }
    }
    selected = firstline = 0
  }
  oldquery = query
  display()
}

function mousedown(i){
  d = data[matched[firstline+i]]
  register(d)
}

function register(d){
  prog =  d[0]; station = d[3];  year = d[4]
  month = d[5];     day = d[6]; start = d[7]; end = d[8]
  location.href = "reserve.cgi?start="+start+"&end="+end+"&program="+prog+"&station="+station+
                  "&year="+year+"&month="+month+"&day="+day
}

function keypress(event){
  k = keycode(event)
  if(k == 40 || k == 39 || k == 38 || k == 37){
    return false;
  }
}

document.onkeypress = keypress;
document.onkeyup = keyup;
function keycode(event)
{
  if(navigator.appName.indexOf("Microsoft") != -1)
    return window.event.keyCode;
  if(navigator.appName.indexOf("Netscape") != -1){
    a = event.keyCode
    if(a != 0){ return a }
    return event.which
```

```
  }
}

function keyup(event)
{
  i = keycode(event);
  if(i == 13){
    d = data[matched[firstline+selected]]
    register(d)
  }
  if(i == 38){ // up
    if(selected > 0){ selected -= 1 }
    else { if(firstline > 0){ firstline -= 1 } }
  }
  if(i == 40){ // down
    if(selected < 10-1){ selected += 1}
    else { firstline += 1 }
  }
  display();
}

matched = []
for(i=0;i<data.length;i++){ matched[i] = i }
var selected = 0
var firstline = 0

function display(){
  for(i=0;i<10;i++){
    document.getElementById("list" + i).innerHTML = ''
  }
  for(i=0;firstline+i < matched.length && i < 10;i++){
    t = document.getElementById("list" + i);
    d = data[matched[firstline+i]]
    t.innerHTML = d[4]+'/'+d[5]+'/'+d[6]+' '+d[7]+' '+d[3]+' '+d[0]
    t.style.width = 600
    if(i == selected){
      t.style.backgroundColor = '#ccccff'
      t = document.getElementById("desc")
      t.innerHTML = d[1]
    }
    else {
      t.style.backgroundColor = '#ffffc0'
    }
  }
  for(;i<10;i++){
    t = document.getElementById("list" + i);
    t.innerHTML = ' '
    t.style.backgroundColor = '#ffffc0'
  }
}
display();
</script>
EOF

cgi.out {
  cgi.html {
    cgi.head {
      cgi.title { 'EPG検索' } +
      cgi.meta('http-equiv' => "Content-Type", 'content' => "text/html; charset=euc-jp")
    } + cgi.body {
      body
    }
  }
}
```