インターフェイスの街角 (64) — バーコード Wiki

増井 俊之

Wiki (Wiki Wiki Web) や全文検索システムなどの導入により、私の計算機上のファイルは徐々にうまく整理できるようになってきました。しかし、オフィスや自宅は相変わらずひどく散らかったままです。日頃から書類や機器などもなんとか整理しなければと思ってはいるのですが、〝これならなんとか実行できそうだ〟といった整理法がみつけられず、じつのところ途方に暮れていました。

考えるまでもなく、もともと上手に整理できる人はあれこれ工夫する必要などないでしょう。その意味では、私のように整理ができない人間ほど、計算機を利用した整理法の研究に向いているはずです。そこで、計算機上の情報整理技術や検索技術を応用して現実の世界の事物を整理するための仕組みを考えてみました。

散らかる原因と対策

私のオフィスや自宅には、書籍や雑誌、手紙、書類、計算機関連の小物などがいつも散らかっていて、なんとも見苦しい状況です。不要ならさっさと捨てればいいのに、いつか必要になるかもしれないと思ってとりあえずとっておくことが多く、しかも分類が下手だったり収納を面倒がったりするために、いろいろなものがしかるべき場所に収まっていないのが最大の原因です。

私自身が片付けを苦手としているのは事実ですが、整理法についての本が数多く出版されているところをみると、整理に頭を悩ませている人は世の中にたくさんいるようです。

多くのものを効果的に整理するには、それらを分類したうえで書棚や引出し、整理棚などに収納しなければなりません。これは、分類や場所の割当てさえうまくできれば、なんとかなりそうです。

しかし、ファイルの整理の場合と同様、本質的に分類しにくい書類や機器もあります。たとえば、観光ガイドブックは書籍に分類して書棚に収めるべきなのか、それともパンフレットと一緒に地域ごとにまとめて保存したほうがよいのかと迷ってしまいます。商品のカタログにしても、内容別もしくは会社別に分類するのか、あるいは発行順に保存するだけでよいのかが分かりません。このように、分類がすこしでも難しいと、なかなか〝片付け〟という行動までいきつかないように思います。

最近出版された『わたしの整理術』[1]という本で、モバイル・コンピューティングなどで有名な大阪大学の塚本昌彦先生が〝一掃による片付け術〟を提案していました。塚本先生は片付けが得意中の得意で、大学から退出するときは机の上に電話とモニター以外のものは載っていないという状態をずっと保っているそうです。これを実践するには、勇気をもって〝一掃〟することがなによりも重要だと述べられていました。

とはいえ、なかには捨ててはいけないものもあるでしょうし、なんらかの方法による整理や検索は必要なはずです。私は〝一掃〟する勇気を出せそうになく、かといって、どこに何を置いたかもすぐ忘れてしまいます。そんな人間でも、苦労せずに〝一掃方式〟と同様な結果が得られる整理法はないものでしょうか。

タグによる管理

ここまでみてきたように、ものが片付かない最大の理由は、正しく分類、収納するのが難しいからだと思います。

そもそも、ものを分類して収納するのは、あとで簡単に探しだせるようにするためでしょう。逆にいえば、いつで

も苦労せずにみつけられるのであれば、わざわざ分類するまでもないということになります。正しく分類しなくてもよいのなら、とりあえずその辺の箱に放り込んで〝一掃〟できます。つまり、**真剣に分類しなくても、ものがみつかる**環境を作ればよいことになります。どこに何があるかが手軽に検索できるのなら、分類や収納に手間をかける必要はありません。

とはいっても、何も考えずにいろいろなものを箱に放り込んでいくと、すぐに何がどこに入っているのか分からなくなってしまいます。そこで、計算機で読み取って検索に使える特殊な〝タグ〟が考案されました。これを利用すれば、収納場所を忘れても計算機で検索できるようになります。

最近は、ものの種別を読み取ったり所在を追跡したりするために、無線技術を用いた RFID (Radio Frequency IDentification) タグがひろく使われるようになりました。RFID タグの利用は、これまでは流通など一部の分野に限定されていましたが、JR 東日本の〝Suica (スイカ)〟やビットワレットの〝Edy カード〟など、一般向けの製品も徐々に増えてきています。

RFID はたしかに便利ですが、現時点ではタグ 1 個で数百円と高価です。ですから、個人の情報整理のためだけに、あらゆるものに RFID を貼るのは非現実的でしょう。

扱いやすさの点では RFID に劣りますが、スーパーや流通業界では安価なタグとしてバーコードがひろく使われています。オフィスや書斎でも、整理するものにバーコードを付け、スキャナを用いてその収納場所を管理すれば、何がどこにあるかを憶えておく必要がなくなるかもしれません。

さまざまな製品に使われている JAN (Japanese Article Number) コードでは、メーカーと製品番号が数字で表現されています。書籍の裏表紙に印刷されているバーコードからは、出版社番号や書籍番号などを数字で表した ISBN (International Standard Book Number) も読み取れます。

バーコード自体は数字や文字列を白と黒の線列で表現したもので、それが貼付されているものの内容を具体的に示すものではありません。数字や文字列が何を表現しているかを知るには、コードと内容との照合が可能なデータベースが必要です。

バーコードで表現される数字や文字列と Wiki ページのタイトルを対応づけた個人用データベースを用意しておけば、バーコードの文字列を Wiki ページのタイトルに変換し、Wiki ページを参照することができます。1 月号で、コマンドラインから Wiki ページを呼び出す WikiHelp システムを紹介しました。これと同様の手法を利用することで、バーコードをスキャンして Wiki ページを直接呼び出せるようになります。

Wiki ページの呼出しは、次のようにおこなわれます。

1. 登録されたバーコードをスキャンしたときは、そのバーコードに対応した Wiki ページを表示する。
2. 未登録のバーコードをスキャンしたときは、バーコードと Wiki タイトルを対応づけるための登録画面を表示する。

このような基本操作に加え、複数のバーコードを連続してスキャンしたときは、それらを表現する Wiki ページ間にリンクを定義することにします。

たとえば、〝123〟というバーコードの付いた PDA をバーコード〝456〟が貼られた箱に格納するときは、以下のような処理をおこないます。

1. 〝123〟をスキャンする。
2. 初めてスキャンすると登録画面が表示されるので、123 には PDA が対応することを登録する。
3. もう一度 123 をスキャンすると、PDA の Wiki ページが表示される。
4. 今度は〝456〟をスキャンする。
5. 登録画面が表示されるので、456 には箱が対応することを登録する。
6. 123 と 456 を連続してスキャンすることにより、PDA の Wiki ページと箱の Wiki ページのあいだにリンクが生成される。
7. PDA を箱に入れる。

これらの処理をおこなったあと、箱に貼られた〝456〟というバーコードをスキャンすると、箱を表す Wiki ページが現れ、そのなかに PDA へのリンクが表示されます。これで、その箱のなかに PDA が入っていることが分かります。また、PDA がどこにあるかを検索したい場合は、

Wiki ページ内のテキストの全文検索をおこなえば、PDA と箱を表現する Wiki ページがみつかります。

身の回りにあるすべてのものにバーコードを貼る、と聞くと終りのない大変な作業のように思えるかもしれません。しかし、多くの製品や書籍にはすでにバーコードが付いているので、それを流用できる場合も少なくないはずです。また、普通の付箋紙の代わりにバーコードを印刷したものを使えば、従来と同じような感覚でバーコードを活用できるでしょう。

あらゆるものについて Wiki ページで情報を登録しようとすると、これまた大仕事になってしまいます。しかし、そもそも整理の対象になるものは無限にあるわけではなく、多くても 1 日に 10 個増える程度でしょうから、登録しやすい Wiki ページを使えばそれほど負担にはならないと思います。

このように運用することにより、あらゆる対象が Wiki ページになりますから、実物の置き場所などを工夫して整理をおこなう代わりに計算機上での整理手法をそのまま使えることになります。

バーコードを付加する対象

本や雑誌、CD、コンビニエンス・ストアなどで販売されている多くの製品には、たいていバーコードが印刷されています。これらについては、そのバーコードをそのまま使うことができます。

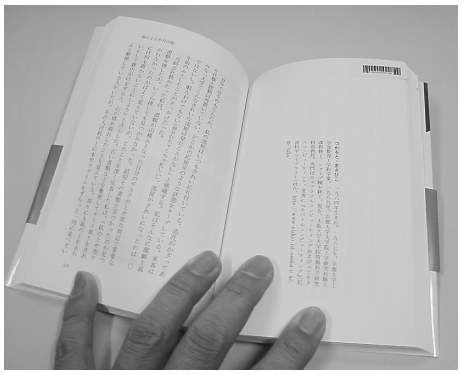
一方、通常のメモや手紙、書類などにはバーコードは付いていないため、自分で用意したバーコードを貼り付けなければなりません。必要になったときに印刷するのではなく、あらかじめバーコードを印刷したシールをたくさん用意しておくとよいでしょう。

バーコードを洋服や食器、食品のようなものに貼り付けるのは難しいかもしれません。しかし、一般的なオフィスや書斎ではこれらは整理の対象にならないといってもよいでしょう。細かな部品に 1 つずつバーコードを付けるのも難しそうですが、これらの部品はある程度まとめて袋や小箱に入れ、それにバーコードを貼ればよいと思います。

時刻を利用した管理

バーコードを貼った整理対象を Wiki ページに登録するときに、現物を撮った写真も Wiki ページに貼っておくと便利です。とはいえ、デジタルカメラなどで撮影し、計算機に保存した写真を Wiki から参照するのは面倒です。

図 1　バーコード付き付箋紙を貼り付けた本



これも、写真の撮影時刻と Wiki ページの作成時刻が近い場合は、写真が自動的に Wiki ページと関連づけられるようにすればよいでしょう。これなら、わざわざ Wiki ページから写真へのリンクを設定しなくても、ほぼ自動的に写真を Wiki ページに関連づけることができます。

時刻による関連づけだけでなく、2002 年 11 月号で紹介した〝近傍検索〟の考えを応用すれば、関連したファイルやページを自動的にリンクとして加えることができます。

バーコード Wiki の利点

バーコード Wiki には、以下のような利点があります。

- 書類や機器など、散らかりやすいものを分類せずに箱に隠しておくことができる。
- Wiki の利用により、計算機内やネットワーク上の情報と実物とのリンクが定義できる。
- Wiki ページはどこからでもアクセスできるので、どこで検索しても同じ情報が得られる。
- 本などに付箋紙を貼ることは多いので、バーコード付きの付箋を貼る作業もそれほど負担にはならない。
- ファイルの場合と同じような感覚で、実物間での近傍検索ができる。

あらゆるものにバーコードを貼付したり、それぞれについての情報を Wiki ページに登録したりするのは、それなりの手間がかかります。しかし、上記の利点が得られるの

図 2　バーコードを印刷したシート

なら手間をかける価値はありますし、結果としてバーコードを用いた整理法は長続きするかもしれません。

実装

運用中の Wiki Wiki Web をバーコード対応にするのは比較的簡単です。まず、バーコードを読み取ったら、バーコードの数字と Wiki ページのタイトルとの対応を調べます。そして、WikiHelp の手法を用いて、そのタイトルをもつ Wiki ページにジャンプします。また、対応するタイトルがない場合は登録ページにジャンプします。

バーコード生成

末尾のリスト 1 は、JAN コード形式のバーコードを生成するライブラリです。PBM と EPS 形式のバーコードを出力できます。

リスト 2 は、このライブラリを使い、A4 判サイズいっぱいにバーコードを表示するプログラムです。

これをシール用紙に印刷すれば（図 2）、バーコード付きの付箋紙が作れます。

シリアル接続のバーコード・リーダー

図 3 は、Symbol Technologies[1]が Palm III をベースに開発したバーコード・リーダー「SPT1500」です。このようなシリアル接続のバーコード・リーダーでデータを読み取り、これに対して barcodewiki プログラム（リスト 3）を実行すると、該当する Wiki ページにジャンプすることができます。

1 http://www.symbol.com/

図 3　Symbol SPT1500

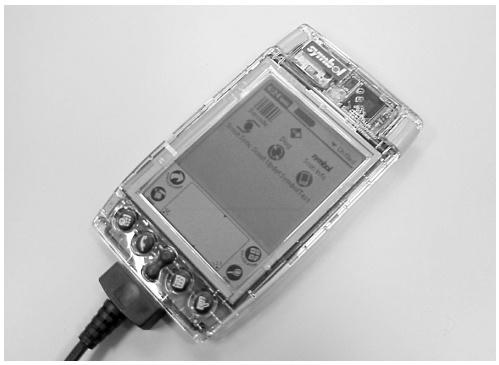


図 4　バーコードを貼った変換ケーブル

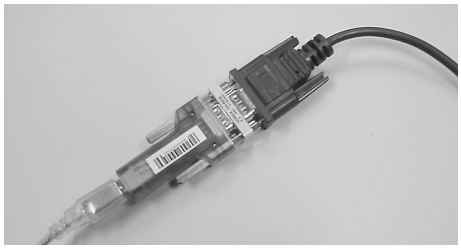


図 5　バーコードのスキャンにより呼び出されたページ

バーコード Wiki の使用例

私が使っているアイ・オー・データ機器製の USB/シリアル変換ケーブル USB-RSAQ2（図 4）に貼ったバーコードをスキャンすると、barcodewiki プログラムのなかで BARCODECGI に設定した CGI が起動し、図 5 のような Wiki ページが表示されます。

この画面の右上にある〝一覧〟をクリックすると、バーコード Wiki ページのリストが表示されます（図 6）。

図 6　バーコードに関連づけられた Wiki ページのリスト

使ってみた感想

バーコード自体は長い歴史をもっています。そのため、かなり以前から、いろいろなものにバーコードを貼って現実の世界と計算機を関連づける研究が数多くおこなわれています。

今回のシステムは、いろいろなものを Wiki ページに関連づけるところに特徴があるわけですが、たとえば以下のような場面でとくに重宝しそうです。

- 複雑な機器の情報参照

  先日、ADSL ルータの調子が悪くなったので、Web ブラウザから診断画面を参照しようとしたのですが、管理画面のユーザー名やパスワードを忘れてしまい、それが記入してあるマニュアルを探すのにずいぶん苦労しました。ルータにバーコードを貼っておき、Wiki ページにこれらの重要な情報を書いておけば、これほど苦労はしなかったでしょう。

  ルータにかぎらず、最近は複雑な機器が増えているので、マニュアルやヘルプなどを Wiki ページに書いておくと便利でしょう。

- 資料の関連づけ

  FAX で送ってもらった資料にバーコードシールを貼り、それに関連した情報を Wiki ページに書いておくことで、死蔵されがちだった資料の有効活用ができます。

さきほども述べたように、バーコードを用いたインターフェイス・システムはいろいろと提案されています。しかし、すくなくとも私自身は、オフィスなどで実際に活用している例をみたことがありません。

一般に、特殊なシステムを新たに作成、導入する場合は、それがいくら便利なものであっても、従来の作業形態から新しいシステムへの移行にコストがかかります。さらに、運用に慣れるまでにはそれなりの時間が必要です。Wiki Wiki Web 自体が便利ですし、今回のシステムはそれにバーコードによる機能拡張をおこなっただけなので、移行や運用についてはとくに支障はありませんでした。なんらかの事情でバーコードによる管理ができなくなった場合も、Wiki ページは残るので、作業が無駄になることはないでしょう。

おわりに

今回は、オフィスや書斎の整理にバーコードを活用する手法を紹介しました。世の中には、バーコード付きの商品がたくさんあります。ところが、これが使われるのは流通段階だけで、消費者の手許ではほとんど活用されていないのは、考えてみれば不思議な話です。せっかく有用な情報が付いているのですから、積極的に利用してみてはいかがでしょうか。

今回紹介したシステムは、バーコードなどなくても書類や小物をきちんと整理できる人にはほとんど意味がないでしょう。しかし、私のようにどうしても整理ができない人間にとっては救世主となるかもしれません。このシステムを使えば、分類・整理のしやすさよりもデザインのよさを基準に箱や引出しが選べますから、美しい書斎やオフィスを作ることもできそうです。

（ますい・としゆき　ソニー CSL）

[参考文献]

[1] 岩波アクティブ新書編集部編『わたしの整理術』、岩波書店、2003 年 (http://www.iwanami.co.jp/.BOOKS/70/X/7000620.html)

リスト 1　JAN バーコード生成ライブラリ (barcode.rb)

```
class JAN
  PARITY = [
    'OOOOOO',  'OOEOEE',
    'OOEEOE',  'OOEEEO',
    'OEOOEE',  'OEEOOE',
    'OEEEOO',  'OEOEOE',
    'OEOEEO',  'OEEOEO',
  ]
  DATA = {
    'LO0' => '0001101',  'LO1' => '0011001',
    'LO2' => '0010011',  'LO3' => '0111101',
    'LO4' => '0100011',  'LO5' => '0110001',
    'LO6' => '0101111',  'LO7' => '0111011',
    'LO8' => '0110111',  'LO9' => '0001011',
    'LE0' => '0100111',  'LE1' => '0110011',
    'LE2' => '0011011',  'LE3' => '0100001',
    'LE4' => '0011101',  'LE5' => '0111001',
    'LE6' => '0000101',  'LE7' => '0010001',
    'LE8' => '0001001',  'LE9' => '0010111',
    'RE0' => '1110010',  'RE1' => '1100110',
    'RE2' => '1101100',  'RE3' => '1000010',
    'RE4' => '1011100',  'RE5' => '1001110',
    'RE6' => '1010000',  'RE7' => '1000100',
    'RE8' => '1001000',  'RE9' => '1110100',
  }

  def checksum(id) # 13桁目を計算
    sum = 0
    (0..11).each { |i|
      d = id[i,1].to_i
      sum += (i % 2 == 0 ? d : d * 3)
    }
    d = (10 - sum % 10) % 10
  end

  def to_s(recalc = false)
    if recalc then
      @s = ''
      @s += '101'
      parity = PARITY[@id[0,1].to_i]
      (0..5).each { |i|
        d = @id[i+1,1]
        p = parity[i,1]
        @s += DATA["L#{p}#{d}"]
      }
      @s += '01010';
      (0..5).each { |i|
        d = @id[i+7,1]
        @s += DATA["RE#{d}"]
      }
      @s += '101';
    end
    @s
  end

  def initialize(id)
    @id = id
    @id += checksum(@id).to_s
           if @id.length == 12
    if @id.length != 13 then
      puts "Invalid ID: #{id}"
      exit
    end
    to_s(true)
  end

  attr_reader :id

  def pbm
    "P1\n95 1\n#{to_s}\n"
  end

  def eps(width=1, height=20, print_digit=false)
    barcode_width = width * 92
    digit_height = barcode_width / 14
    digit_width = barcode_width * 0.8
    digit_eps = ""
    if print_digit then
      digit_eps = <<EOF
/id (#{id}) def
/barcodewidth #{barcode_width} def
/digitheight #{digit_height} def
/Courier findfont digitheight scalefont setfont
/idwidth id stringwidth pop def
/bgwidth idwidth 1.2 mul def
/bgheight digitheight 1.0 mul def
barcodewidth bgwidth sub 2 div 0 moveto
bgwidth 0 rlineto 0 bgheight rlineto
0 bgwidth sub 0 rlineto closepath
1 setgray fill
barcodewidth idwidth sub 2 div
digitheight 5 div moveto
0 setgray id show
EOF
    end
    <<EOF
%!PS-Adobe-2.0 EPSF-2.0
%%Creator: barcode.rb
%%Title: #{id}
%%BoundingBox: 0 0 #{width*92} #{barcode_width}
%%EndComments
/width #{width} def
/height #{height} def
/bar {
  gsave
  width 0 rlineto
  0 height rlineto
  0 width sub 0 rlineto
  closepath
  49 eq { fill } if
  grestore
  width 0 rmoveto
} def
0 0 moveto
(#{to_s})
{ bar } forall
#{digit_eps}
EOF
  end
end
```

リスト 2　バーコードを A4 判サイズに表示するプログラム (idsheet)

```
#!/usr/bin/env ruby

require 'barcode.rb'

id = ARGV.shift
exit unless id =~ /^\d{12}\d?$/

puts <<EOF
%!PS-Adobe-2.0
%%Creator: idsheet
%%Title: Barcode Sheet
%%CreationDate: #{Time.new.asctime}
%%EndComments
%%EndProlog
%%BeginSetup
%%EndSetup
/BeginEPSF {
  /b4_Inc_state save def
  /dict_count countdictstack def
  /op_count count 1 sub def
  userdict begin
  /letter {} def /note {} def
  /legal {} def /11x17
  /b4 {} def /b5 {} def /a5 {} def
  /a4 {} def /a3 {} def
  /showpage {} def /copypage {} def
  /erasepage {} def
  0 setgray 0 setlinecap
  1 setlinewidth 0 setlinejoin
  10 setmiterlimit [] 0 setdash newpath
  /languagelevel where
  {pop languagelevel
  1 ne
    {false setstrokeadjust false setoverprint
    } if
  } if
} bind def
/EndEPSF {
  count op_count sub {pop} repeat
  countdictstack dict_count sub {end} repeat
  b4_Inc_state restore
} bind def

30 40 translate
EOF

(0..24).each { |i|
  y = 30 * i
  (0..4).each { |j|
    x = 120 * j
    barcode = JAN.new(id)
    puts <<EOF
BeginEPSF
%%BeginDocument:
gsave
#{x} #{y} translate
#{barcode.eps(0.7,16,true)}
grestore
%%EndDocument
EndEPSF
EOF
    id = id.succ
  }
}
```

リスト 3　バーコードから Wiki ページにアクセスするためのプログラム (barcodewiki)

```
#!/usr/bin/env ruby

require 'termios'
require 'fcntl'

SERIALDEV = '/dev/com4'
BROWSER = "/cygdrive/c/'Program Files'/Netscape/Netscape/Netscp.exe"
BARCODECGI = "http://www.csl.sony.co.jp/person/masui/WikiHelp/programs/barcode.cgi"
LINKCGI = "http://www.csl.sony.co.jp/person/masui/WikiHelp/programs/makelink.cgi"
```

```
def setserial(dev)
  line = open(dev, File::RDWR | File::NONBLOCK)
  mode = line.fcntl(Fcntl::F_GETFL, 0);
  line.fcntl(Fcntl::F_SETFL, mode & ~File::NONBLOCK);

  newtio = Termios::new_termios()
  newtio.c_iflag = Termios::IGNPAR
  newtio.c_oflag = 0
  newtio.c_cflag = (Termios::B9600 | Termios::CLOCAL |
     Termios::CS8 | Termios::CREAD | Termios::HUPCL)
  newtio.c_lflag = 0
  newtio.c_cc[Termios::VMIN]  = 1
  newtio.c_cc[Termios::VTIME] = 0

  newtio.ospeed = Termios::B9600
  newtio.ispeed = Termios::B9600

  Termios::flush(line, Termios::TCIOFLUSH)
  Termios::setattr(line, Termios::TCSANOW, newtio)
  line
end

def showpage(id)
  system "#{BROWSER} #{BARCODECGI}?jan=#{id}"
end

def makelink(lastid,thisid)
  system "#{BROWSER} '#{LINKCGI}?lastid=#{lastid}&thisid=#{thisid}'"
end

def main
  line = setserial(SERIALDEV)

  lasttime = 0
  lastid = nil
  while true
    s = line.gets
    if s =~ /(\d{13})/ then
      thisid = $1
      thistime = Time.new
      if thistime - lasttime > 5 then
 showpage(thisid)
      else
 makelink(lastid,thisid)
      end
      lastid = thisid
      lasttime = thistime
    end
  end
end

main
```