インターフェイスの街角 (76) — 短い名前の利用

増井俊之

通常、ファイル名や URL は階層的なデータ構造や中身を反映するため、どうしても長くなります。たとえば、私が書いている原稿のフルパス名は/home/masui/DOC/UnixMagazine/0406/if0406.tex、この連載のサポートページの URL は http://pitecan.com/articles/UnixMagazine/と、いずれもかなり長くなっています。

シェルでこのようなファイルにアクセスする場合、フルパス名をつねに間違いなく入力するのは大変です。たいていは、シェルの補完機能などを使いながらディレクトリをたどることになりますが、それでもけっこうな手間がかかります。また、このような長いファイル名や URL を口頭で正しく他人に伝えるのはそう簡単ではありません。

最近の日記や Wiki のページでは CGI などを使っているため、URL が長くなりがちです。Amazon の商品紹介ページも、http://www.amazon.co.jp/exec/obidos/ASIN/4105413015/とたいへん長いものです。このように、URL の多くは人間がすぐには記憶できないほど長くなっています。

TinyURL

扱いにくい URL を他人に紹介する場合などのために、長い URL を短くしてくれる TinyURL[1]というサービスがあります (図 1)。これは、例に挙げたような長い URL を ``http://tinyurl.com/abcde''といった ``短い名前''に変換してくれるものです。

たとえば、http://pitecan.com/articles/UnixMagazine/という URL を TinyURL に登録すると、http://tinyurl.com/2s3cc という短い URL が返されます。これなら、もとの長い URL の代わりに ``2s3cc''だけを記憶したり伝達するだけですみます。この程度の長さであれば、短期的に憶えておけますし、電話などでも間違えずに伝えることができるでしょう。

1 http://TinyURL.com/

図 1 TinyURL.com

TinyURL.com のサービスは長い URL を短いユニークな名前にマッピングしているだけであり、バカバカしいほど単純な仕組みと思うかもしれません。しかし、TinyURL で割り当てられる短い名前は簡単に憶えられるのに、一般的な URL ではそれが無理であることを考えると、用途によってはけっこう役に立ちそうです。

短い名前を個人的に活用する

TinyURL はたしかに便利ですが、長い名前を短い名前に対応づけるデータベースは TinyURL.com のサーバー内にあります。したがって、なんらかの事故でサービスが停止したり、あるいはネットワークにアクセスできない場

著者校正 [2004 年 4 月 29 日]

表 1　用意できる短い名前

1 文字で表現できる名前		=	26 個 ('a' 〜 'z')
2 文字で表現できる名前	$26 * (10 + 26)$	=	936 個 ('a0' 〜 'zz')
3 文字で表現できる名前	$26 * (10 + 26)^2$	=	33,696 個 ('a00' 〜 'zzz')
4 文字で表現できる名前	$26 * (10 + 26)^3$	=	1,213,056 個 ('a000' 〜 'zzzz')

合には使えません。さらに、永遠に短い名前が使える保証もないので、どうしても間に合わせの用途にしか使えないことになります。また、TinyURL は URL しか扱えませんが、ファイル名やメールのメッセージなどにも同様の手法を適用し、あらゆるデータに短い名前でアクセスできたほうが、もっと便利でしょう。

考えてみると、コマンドにはたいてい〝ショートカット〟が用意されているのに、データについてはそういった工夫はあまりなされていません。もちろん、

```
% url=http://pitecan.com/articles/UnixMagazine/
```

とシェル変数を明示的に指定すれば、

```
% wget $url
```

のように短い名前をシェルで使うことは可能です。しかし、スクリプト内での利用はともかく、コマンド行でこういった方法を実践している人はあまりみかけません。

シェルにかぎらず、いろいろなアプリケーションからファイルや Web ページ、メールメッセージなどに共通の短い名前でアクセスできれば活用する場面が増えそうです。

必要な桁数

まず、情報の特定にどのくらいの長さの文字列が必要かを考えてみましょう。文字列を変数名として扱いやすくするには、文字は英数字だけとし、1 文字目を英字にするのがよさそうです。この場合には、短い名前を表 1 の数だけ用意できることになります。

文字列を 3 文字とすれば、毎日 10 個使ったとしても名前空間を使いきるのに 10 年はかかります。つまり、実用上は 3 桁で十分ということになります。自分が扱うあらゆるデータを、わずか 3 文字で表現できるのは興味深いところです[2]。万一、3 桁で足りないようなら 4 桁にすればいいわけですから、とりあえず文字列の長さについては心配しなくてよいでしょう。

活用する場面

短い名前は、以下のような場面で使えそうです。

- コマンド行での利用

```
% wget $abc
% cd $d0a
```

- Wiki ページでの参照

  Wiki ページで URL やメールなどを参照するとき、長い URL の代わりに短い名前で指定できます。たとえば、A 社を訪問する予定があるときは、A 社の URL の代わりに短い名前を記述しておけばよいでしょう。

```
13:00 [[ha9 A社]]訪問
```

- TinyURL.com と同様な使い方

  TinyURL.com と同じようなサービスを自分で用意することもできます。

- 携帯電話からのアクセス

  3 月号で、携帯電話などからサーバーにメールを送り、各種のコマンドを実行させる〝コマンドメール〟を紹介しました。短い名前は、ファイルやメールを直接指定して取得したりするときも便利そうです。

実装

以上のようなアイデアにもとづき、短い名前を登録、検索するライブラリ simplename.rb と、それを利用するためのコマンドなどを作ってみました。

データベースの構造

短い名前と長い名前を対応づけるデータベースは、いろいろな場所に置く必要があります。URL に関する情報は Web サーバーに置くと便利でしょうし、ファイル名に関する情報はローカルマシンに保存しておかなければ意味がありません。登録やアクセス、同期を簡単におこなうために、短い名前をファイル名としたファイルに長い名前を書くと

2 世界中のあらゆる情報は 13 バイトに圧縮できるという冗談があります。4 バイトで IP アドレスを特定し、ファイルのオフセットとサイズを 9 バイトで表現すれば 13 バイトに収まるからです。IPv6/TB 時代には、18 バイトくらいになるかもしれませんが …… 。

著者校正 [2004 年 4 月 29 日]

いう単純な形式でデータベースを表現することにします。

たとえば〝abc〟という短い名前で http://pitecan.com/を表現する場合には、~/.shortname/a/bc などのファイルに http://pitecan.com/というテキストを書いておくことにします。

■ abc ではなく、a/bc にする理由は?

ファイルなので同期は簡単ですが、複数の場所でデータベースを扱う場合は、データをつねに同期させておかないと二重登録してしまうおそれがあります。

短い名前ライブラリ

以下のリスト 1 に示す shortname.rb は、短い名前と長い名前をハッシュのように使うためのライブラリです。

リスト 1 shortname.rb

```
# ShortName.new(shortname,longname)

class ShortName
  include Enumerable

  ALPHA = ('a'..'z').to_a
  NUMERIC = ('0'..'9').to_a
  ALNUM = NUMERIC + ALPHA

  def initialize(dir=nil)
    @dir = dir
    if dir.nil? then
      @dir = File.expand_path('~/.shortname')
    end
  end

  attr_reader :dir

  def path(shortname)
    dir = shortname[0,1]
    file = shortname[1,2]
    return nil if dir.nil? || dir == ''
    return nil if file.nil? || file == ''
    "#{@dir}/#{dir}/#{file}"
  end

  def [](shortname)
    p = path(shortname)
    return nil if p.nil?
    longname = nil
    if File.exist?(p) then
      File.open(p){ |f|
        longname = f.gets.chomp
      }
    end
    longname
  end

  def []=(shortname,longname)
    p = path(shortname)
    return if p.nil?
    File.open(p,"w"){ |f|
      f.puts longname
    }
  end
```


著者校正 [2004 年 4 月 29 日]

```
  def newname(s)
    name = nil
    ALPHA.each { |c1|
      if s[0,1] == c1 || s[0,1] == "" || s[0,1] == nil then
        ALNUM.each { |c2|
          if s[1,1] == c2 || s[1,1] == "" || s[1,1] == nil then
            ALNUM.each { |c3|
              if s[2,1] == c3 || s[2,1] == "" || s[2,1] == nil then
                path = "#{@dir}/#{c1}/#{c2}#{c3}"
                if !File.exist?(path) then
                  name = "#{c1}#{c2}#{c3}"
                end
                break if name
              end
            }
          end
          break if name
        }
      end
      break if name
    }
    name
  end

  def each
    File.open("#{@dir}/list","w"){ |out|
      Dir.open(@dir).each { |dir|
        if dir =~ /^[a-z]$/ then
          Dir.open("#{@dir}/#{dir}").each { |file|
            if file =~ /^[a-z0-9][a-z0-9]$/ then
              shortname = "#{dir}#{file}"
              path = "#{@dir}/#{dir}/#{file}"
              File.open(path){ |f|
                longname = f.gets.chomp
                yield shortname, longname
              }
            end
          }
        end
      }
    }
  end

end
```

これを利用すると、

```
sn = ShortName.new
sn['abc'] = 'http://example.com/'
```

のようにデータベースに登録したり、

```
sn = ShortName.new
print sn['abc']
```

として、短い名前で長い名前の実体にアクセスできるようになります。

登録プログラム

長い名前に短い名前を割り当てるには、**sn** コマンドを使います（リスト 2）。

以下のように、sn コマンドの引数に長い名前を指定すると、短い名前が新たに割り当てられてデータベースに登録されます。

```
% sn http://pitecan.com/
a02
%
```

また、短い名前の一部を指定して長い名前を登録することもできます。以下の例では、http://pitecan.com/に``h''で始まる短い名前を割り当てています。

```
% sn http://pitecan.com/ h
h05
```


著者校正 [2004 年 4 月 29 日]

リスト 2 sn

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

require 'shortname'

def usage
  STDERR.puts "sn - register a 3-letter keyword for long URL/path"
  STDERR.puts "Usage: % sn http://pitecan.com/ h00"
  STDERR.puts "       % sn http://pitecan.com/ a"
  exit
end

longname = ARGV[0].to_s
namehint = ARGV[1].to_s
usage if longname == ''

sn = ShortName.new
shortname = sn.newname(namehint)
sn[shortname] = longname
STDERR.puts shortname

File.open("#{sn.dir}/list","w"){ |out|
  sn.each { |shortname,longname|
    out.puts "export #{shortname}='#{longname}'"
  }
}
```

ShortName はシェル変数としても使いたいので、``abc''という名前が登録されているときは$abc というシェル変数が使えるようにします。

また、シェル変数として、

```
% echo $h05
http://pitecan.com/
%
```

というふうにも使えるようにするため、以下のシェル関数を定義しておきます。sn コマンドは登録結果を list というファイルに出力し、これを読み込むことでシェル変数が登録されます。

```
sn()
{
  /home/masui/bin/sn "$1" $2
  source /home/masui/.shortname/list
}
```

コマンド列を短い名前に登録しておくこともできます。

```
% sn 'tar czvf /tmp/junk.tar .' tzj
% $tzj
./
./.bash_history
./.bash_profile
./.emacs
./.ssh/
  ......
%
```

参照プログラム

短い名前で、ファイルや Web ページを簡単に参照できるプログラムがあると便利です。産業技術総合研究所の田中 哲氏が作成した open-uri.rb という Ruby のライブラリ[3]を利用すると、普通のファイルもネットワーク上の Web ページも同じように扱うことができます。これを使って、短い名前からファイルや Web ページにアクセスする **sncat** というプログラム（リスト 3）を作成しました。

sncat は、次のように実行します（誌面の都合上、⇒ で折り返しています）。

```
% sncat h05
<!DOCTYPE html PUBLIC "-//W3C//DTD ⇒
HTML 4.01 Transitional//EN">
<HTML>
<HEAD>
<META http-equiv="Content-Type" ⇒
content="text/html; charset=EUC-JP">
  ......
%
```

Web からの利用

Apache の mod_rewrite モジュールを導入すると、RewriteRule というディレクティブが使えるようになりま

3 http://www.ruby-lang.org/en/raa-list.rhtml?id=758

著者校正 [2004 年 4 月 29 日]

リスト 3 sncat

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

require 'open-uri'
require 'shortname'

def usage
  STDERR.puts "sncat - output the contents specified by a short name"
  STDERR.puts "Usage: % sncat h00"
  exit
end

shortname = ARGV[0].to_s
usage if shortname.length != 3
sn = ShortName.new
longname = sn[shortname]
usage if longname.nil?

open(longname){ |f|
  f.each { |line|
    puts line
  }
}
```

す。これを利用すれば任意の URL を別の URL に変換できるので、TinyURL と同様な機能が簡単に実現できます。

たとえば ``xxx''という短い名前を http://www.asahi.com/に割り当てておくと、http://example.com/xxx などの URL から http://www.asahi.com/にジャンプできるようになります。これは、次のようにします。

1. mod_rewrite の設定

   Apache の設定ファイルに以下の 1 行を加えて、mod_rewrite を有効にする。

   ```
   LoadModule rewrite_module modules/mod_rewrite.so
   ```

2. RewriteRule の設定

   http://example.com/の .htaccess に以下のような指定を加えれば、任意の 3 文字の名前に対して shortname.cgi を呼び出すことができます。

   ```
   RewriteEngine on
   RewriteRule ^(...)$ shortname.cgi?shortname=$1
   ```

   CGI プログラム shortname.cgi（リスト 4）を使うと、データベースに登録された URL に自動的にジャンプさせることもできます。

このようにしておくと、短い名前をブックマークと同様に利用することができます。Web ブラウザのブックマークはコピーしなければ別のブラウザでは使えませんが、サーバーに短い名前が記憶されていれば、どこにいても同じように使えます。

おもしろい記事やページをみつけたとき、ブックマークに書いておく価値があるだろうかなどと悩まずに、どんどん記録していけるでしょう。

メールへのアクセス

メールメッセージについても、短い名前が使えると便利でしょう。メールをファイルや Web ページとして保存するか、Message-ID などを URL に変換してアクセスできるようにしておけば、今回紹介した方法が使えます。

おわりに

短い名前はコマンドのショートカットと似ているので、本当に役立つかは微妙なところです。たしかに便利そうですが、いくら短い名前といってもすべてを記憶するのは無理なので、つねに有用かどうかは分かりません。

計算機上のデータの特定は、きわめて重要な操作です。内容からデータを特定するのが ``検索''であり、ディレクトリやファイルはデータを特定する際の手掛かりの 1 つといえます。GUI アプリケーションでは、アイコンやメニューのように、場所や外観によってデータを特定する手法がひろく使われています。``超整理法''は、時間を用いたデータの特定手法といえるかもしれません。検索に適したものからそうでないものまで、データの特定方法は連続的にひろ

著者校正 [2004 年 4 月 29 日]

リスト 4 shortname.cgi

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

require 'cgi'
require 'shortname'

cgi = CGI.new('html3')

shortname = cgi['shortname'].to_s
longname = cgi['longname'].to_s
sn = ShortName.new('/home/masui/.shortname')

if longname == '' then
  longname = sn[shortname].to_s
else
  sn[shortname] = longname
end

cgi.out {
  cgi.html {
    cgi.head {
      longname =~ /^http/ ?
        "<META HTTP-EQUIV=\"Refresh\" CONTENT=\"0;URL=#{longname}\">" : ""
    } +
    cgi.body {
      longname == '' ?
        cgi.form('NAME' => 'postform', 'METHOD' => 'post', 'ACTION' => 'shortname.cgi'){
         'short: ' +
         cgi.input('TYPE' => 'text', 'NAME' => 'shortname', 'SIZE' => '5', 'value' => shortname) +
         cgi.br +
         'long: ' +
         cgi.input('TYPE' => 'text', 'NAME' => 'longname', 'SIZE' => '100') +
         cgi.br +
         cgi.input('TYPE' => 'submit', 'VALUE' => ' Register   ')
        } : longname
    }
  }
}
```

リスト 5 さまざまなデータの特定方法

```
fopen('locate', '特定', 'fopen');          ←全文検索による特定
fopen('2004/4','データの特定');            ←内容と日付から特定
fopen('h04');                              ←短い名前
fopen('20040407012345');                   ←作成日付
fopen('/home/masui/PIM','datalocate');     ←ディレクトリとファイル名
```

い範囲に分布しているといえるでしょう。

考えてみれば、データの特定は日常の仕事の 9 割くらいを占めているといっても過言ではありません。UNIX でも、grep や ls、cd、locate、find などのコマンドがデータの特定に使われています。今回紹介したような、短い名前による手法を既存のいろいろな方法と組み合わせれば応用範囲がひろがりそうです。

多くのプログラムではデータの特定にファイル名が使われますが、今回のような手法を含め、リスト 5 に示したような多様な方法でデータが指定できるようにすると便利かもしれません。

ファイルのパス名は、入力する際に厳密さを求められるうえに、長くて扱いにくいものです。内容や短い名前をデータの特定に活用する方法を、もうすこし探ってみようと考えています。

（ますい・としゆき　産業技術総合研究所）

著者校正 [2004 年 4 月 29 日]