インターフェイスの街角 (16)
地図データベースの活用

増井俊之

このところ、計算機上で地図や位置情報を活用するシステムが急速に増えてきました。この種のシステムは、従来から **GIS**（Geographical Information System：地理情報システム）と呼ばれ、研究開発がおこなわれています。GIS というと高価で特殊なシステムというイメージがありますが、最近はカーナビや電子地図などが一般にも容易に手に入るようになり、誰でも手軽に地図データを利用できる環境が整ってきました。パーソナル・コンピュータで使える安価な地図ソフトが市販されていますし、国土地理院の各種の〝数値地図〟[1]も CD-ROM で比較的安価に入手できるようになりました。これらの地図はたんに眺めるだけでなく、各種の応用が可能です。

今回は、これらの地図情報の活用例を紹介します。

地図ソフトと Web 上の地図サービス

カーナビが順調に普及したのにくらべ、PC 上で使える地図システムはすこし前までほとんどありませんでした。しかし、3 年ほど前から PC 用の地図ソフトが市販されるようになり、最近はきわめて機能の豊富な製品が安価に手に入ります。地図関連専門のコーナーを設置するソフト販売店も増えているようです。なかでもアルプス社の ProAtlas やインクリメント P[2]の MapFan、昭文社の MappleLife、ゼンリンの Z[zi:]、ソニーの Navin' You などが代表的で、機能や使いやすさを競っています。

表示の工夫や検索機能、GPS（Global Positioning System）や PHS との連携機能などに加え、多くのソフトでは Web 上の情報やブラウザとの連携機能が強化されています。たとえば、ProAtlas では地図上に URL のリンクを貼り付けたり、インターネット上の交通情報や地価などの情報を地図上に表示したり、あるいは同社が運営している Web ページ〝クチコミ地図情報〟から地図を呼び出すこともできます。ゼンリンも〝Zi-Land〟という同様なページを用意して、同社の地図を利用した情報交換がおこなえるようになっています。

Web 上でも、いろいろな会社が地図サービスを提供しています。Mapion やインクリメント P の MapFan-Web、MAPOO Corp の Mapoo、富士通/ゼンリンの WildBird などが代表的です。地図にかぎらず、駅周辺の情報を掲載している〝駅前探検クラブ〟や、細かく分けた地域別のレストラン情報を提供する〝東京レストランガイド〟のようなサービスも増えています。

地図へのリンク

地図に関するソフトウェアや Web ページを単体で使うだけでなく、別の Web ページやプログラムから呼び出して利用できれば、さらに便利です[3]。アルプス社は、スクリプトを介して、ProAtlas と Web ブラウザなどのアプリケーションとを連携させるための**マップサーバ**という技術を提唱しています。スクリプトに表示したい場所の緯度・経度情報などの属性を記述しておき、サーバーとして機能する ProAtlas にこの情報を渡して地図を表示させます。

たとえば、私の勤務するソニー CSL の位置を示すマッ

1 国土地理院発行のデジタル地図データです。1/25,000 の地図などのデジタルデータは、日本地図センター (http://www.jmc.or.jp/) で購入できます。数年前まではフロッピーしか販売されていなかったため、扱いが面倒でしたが、現在は CD-ROM で入手できます。

2 パイオニアの子会社なので、こういう社名になったのだそうです。

3 Web ページから地図関連のソフトウェアや別の Web ページを呼び出す方法については、〝Sakura Garden〟(http://member.nifty.ne.jp/SAKURA_GARDEN/) に丁寧な説明があります。

図 1　マップサーバ・スクリプトの例 (csl.mps)

```
//{{%HEADER%
//Header:Prog:"1.5.5.2"' Templ:"1.0" ⇒
  Date:"1999/1/24";
//}}%HEADER%
//{{%SCRIPT%
mapserver script ver="1.0";
    //{{%BODY%
        //{{%MESSAGE%
        message    str="";
        //}}%MESSAGE%
        //{{%THEME%
        theme    str="";
        //}}%THEME%
        //{{%GROUP%
        group    gid="0"
                gname="Default"
                textrgb="255,0,0"
                linergb="0,0,255"
                areargb="0,0,255"
                bodrrgb="0,0,255"
                textstyle="frame#2"
                linestyle="line#0"
                areastyle="hatch#3"
                pointstyle="none"
                bodrstyle="line#0"
                iconsize="32"
                linewidth="2"
                bodrwidth="2";
        //}}%GROUP%
        //{{%POINT%
        point    oid="1"
                gid="0"
                pos="35/37/21.0,139/44/00.6"
                str="ソニーCSL"
                strformat="right"
                url="";
        //}}%POINT%
        //{{%DISP%
        disp    oid="1";
        //}}%DISP%
    //}}%BODY%
mapserver script end;
//}}%SCRIPT%
```

（誌面の都合上、⇒ で折り返しています。以下同様）

プサーバ・スクリプトは図 1 のようになります。これに拡張子 `.mps´を付けて Web ページ上に置いておけば、ブラウザがスクリプトをロードして ProAtlas を起動することによって、PC 上に該当箇所の地図が表示されます[4]。

4 マップサーバ・スクリプトを解釈できればほかのプログラムの地図情報も使えるはずですが、現時点ではアルプス社の ProAtlas 以外では利用できません。

図 2　Navin' You スクリプトの例 (csl.nvi)

```
NPOIs100@NRA/R1;
 1999/01/24_21:22:14_JST;
 254;
 $16873371ED17396E;
 TTL = {ソニーCSL};
 CHAR = SJIS;
 ADDR = {東京都品川区東五反田３丁目};
 HPGL = {http://www.csl.sony.co.jp/};
 MAIL = {masui@csl.sony.co.jp};
 COMM = {};
eoNPOIs@NRA
```

マップサーバほど汎用的ではありませんが、ソニーの Navin' You でも図 2 のようなスクリプトを使って同様な動作をさせることができます。

このような方法でローカルマシン上の地図を表示させる場合、スクリプトのサイズが小さいので、情報の取得から地図表示まで高速におこなえます。しかし、地図を表示させるには ProAtlas や Navin' You が動く Windows マシン、CD-ROM ドライブが必要です。

これに対し、Web 上の地図サービスを利用する場合は、ブラウザで地図画像を表示するので時間はかかりますが、任意のブラウザで地図が参照できます。

Web 上の多くの地図サービスでは、緯度と経度を指定して直接地図を表示させる CGI を用意しています。たとえば、MapFan Web では、

```
http://meringue.mapfan.com/link.cgi?MAP= ⇒
E139.44.00.6N35.37.21.0&ZM=10
```

Mapion では、

```
http://www.mapion.co.jp/cgi-bin/nttp?NL=  ⇒
35.37.21.0&EL=139.44.00.6
```

Mapoo では、

```
http://www.mapoo.or.jp/cgi-bin/gis.cgi?N= ⇒
35.37.21.0&E=139.44.00.6
```

といった記述で同じ場所の地図を表示させることができます。

このように場所の緯度と経度を指定した URL を使用すれば、外部プログラムや別のページからも Web 上の地図サービスに簡単にアクセスできるようになります。

実世界での位置情報取得

現実の世界で現在位置を知ることができれば、前述のようなアプリケーションや Web 上のサービスを用いてその場所の地図を利用できるでしょう。現在位置を知る方法としては、これまでカーナビなどでおもに GPS が利用されてきましたが、最近は PHS による方法も使われ始めています。

GPS による位置情報取得

このところ、GPS の受信機がかなり安く入手できるようになったので、PC に GPS 受信機を接続してカーナビのように使ういわゆる〝PC カーナビ〟が流行しているようです。PC 用の地図ソフトの多くは、GPS 受信機を接続することでカーナビのように利用できます。特定の GPS 受信機にしか対応していないものもありますが、ユーザーの手によって多数の GPS 受信機が使えるようになっています。

たとえば、津守美弘氏らが作成した地図表示ソフト Dmapwin と〝ひろくん〟作の GPS 制御ソフト GPS Player とを組み合わせれば、市販のカーナビ用地図 CD や Windows 用の地図ソフトを、入手可能なほとんどの GPS 受信機と連携させることができます。

多くの GPS 受信機は、NMEA-0183 形式の緯度・経度情報を 4,800bps でシリアル出力します。NMEA-0183 では緯度・経度情報が ASCII テキストとして出力されるため、データの扱いも比較的簡単です。GPS 受信機とその応用については、NIFTY-Serve の GPS フォーラム FGPS で活発な論議がおこなわれているようです。

PHS による位置情報取得

最近は、PHS 各社がそれぞれ PHS 端末を用いた位置情報サービスを始めています。

たとえば、NTT DoCoMo は〝いまどこサービス〟を利用した〝ここ Navi〟というシステムを提供しています。これは、PC 上で現在位置を取得し、Mapion や WildBird、MapFanWeb などにアクセスできるようにするものです。DDI ポケットも同様の位置情報サービスを提供しており、端末の位置情報を取得して駅前探検クラブにアクセスできるようになっています。アステルの位置情報サービス〝P ナビ/データ〟では、Mapion、MapFan-Web、駅前探検クラブに加え、たべあるき東京、東京レストランガイドなどにもアクセスできます。

PHS を使って正確な位置情報を取得するには、PHS 各社のサービスを利用する必要があります。ただし、PHS 端末だけでアンテナの ID や電界強度を調べられる場合もあるので、うまく使えばおおよその現在位置を自動的に把握することができます。

携帯電話で位置情報を利用する技術の開発も進められています。NTT DoCoMo では米国 SnapTrack 社の SnapTrack 技術[5]を用いて、GPS を内蔵した携帯電話で正確な位置情報を得る技術の実験をおこなっています。あらゆる携帯電話で位置情報が使えるようにすることを目指しているそうなので、近い将来、PHS と同様な位置情報サービスが登場するかもしれません。

住所・電話番号を使う方法

住所が分かっていて、データベースからその緯度・経度を調べられれば、Web 上の地図を参照できます。電話番号から名前や住所が検索できるシステムも市販されているので、電話番号から住所を検索し、さらに緯度・経度を調べて該当箇所の地図を見ることもできるでしょう。街を歩いているときに地図を調べるのなら、住居表示や電話番号をもとにしたほうがてっとり早いかもしれません。

現在のところ、住所をもとに地図を検索するための一般に公開されているデータベースはありません。しかし、あとで紹介する、位置をもとに情報を検索するシステムを利用すれば、住所から緯度・経度情報が得られます。

その他の手法

街なかで位置を特定するには、住所や電話番号の代わりに特殊な ID をもつ電柱番号などが使えますし、ビル名から住所や緯度・経度情報を知ることも可能です。

手掛かりとなる建造物などが何もない場所で、山の稜線の形を地形データベースと照合して位置を調べる手法も研究されています。

5 移動端末で GPS 信号の数秒ぶんのスナップショットを計測して電話経由でサーバーに送り、サーバーで計算した位置情報を端末に送り返す仕組みです。端末側で独自に位置を計測する必要がなく、また受信可能なすべての GPS 信号を利用できるため、ビルのなかなどの電波の届きにくい場所でも数秒以内で正確に位置を測定できるそうです。

位置にもとづく情報検索

ここまでに紹介したように、インターネットでは地図をはじめとする各種の情報が得られます。位置情報を取得するハードウェアもいろいろあるので、位置をもとにした検索はかなり実用に近づいてきたといえるでしょう。

NTT ソフトウェア研究所では、位置情報をもとに Web 上の情報を検索する〝モーバイルインフォサーチ〟(別名〝ここのサーチ〟)の実験をおこなっています [1, 3]。Web 上の情報には、どこからでもアクセスできるという利点があります。しかし、近くにあるレストランを探したいといったように場所をもとに情報を検索したい場合は、普通の検索エンジンでは役に立ちません。〝ここのサーチ〟は、路線バスの宣伝アナウンスのように、いま〝ここ〟で役に立つ情報を Web 上で検索する仕組みがあればと考えて開発されたそうです [1]。

〝ここのサーチ〟では、緯度・経度、住所、最寄り駅、郵便番号、GPS 受信機、前述の〝ここ Navi〟などを使って位置情報を指定します。たとえば郵便番号で位置を指定する場合は、位置入力ページ (http://www.kokono.net/move.html) で〝1410022〟のように入力します。すると、

東京都品川区東五反田付近
〒141-0022
(北緯 35.37.27.311 東経139.43.51.175 付近)
最寄駅 五反田,高輪台

というデータが検索条件として指定され、この条件から地図や電話帳、天気予報などの位置に関連した検索サービスにアクセスできます。あるいは、NTT の〝インターネットタウンページ〟を検索すると、その場所の近くにある会社などのリストが得られます。

NTT のタウンページには、もともと郵便番号や緯度・経度情報は掲載されていません。〝ここのサーチ〟では、タウンページに記載されている住所をあらかじめ位置情報に変換して検索対象に含めています。

〝ここのサーチ〟では、個人情報にもとづくデータは検索の対象になっていません。プライバシー保護の問題などもありますが、位置をもとにした個人的な情報も手軽に検索できればさらに有用でしょう。店舗などの情報には、たいてい住所や電話番号が付記されています。したがって、興味を覚えた店の住所などを位置情報に変換して保存しておけば、現在地付近にあるおもしろい店を探したり、たまたま近くに来たときにその店のことを思い出させてくれるようなシステムも作れるでしょう。あるいは、各人の現在位置情報を友人のあいだで共有するようにしておけば、友人が近くに来たときに連絡をとるといったシステムも実現できるかもしれません。

図 3 モーバイルインフォサーチのページ

住所や電話番号を位置情報に変換することによって、ネットワーク上の膨大な情報を地図にリンクできるだけでなく、現実世界の場所からネットワーク上の情報にも膨大なリンクが張れるようになるので、さまざまな応用が考えられるでしょう。

地名サーバーとその応用

昨年、郵便番号が 7 桁になったために住所録の更新などに手間どった人も多かったのではないでしょうか。ただし、今回の変更にともなって郵政省が住所/郵便番号データベース CD-ROM を配布したため、地名/郵便番号データベースが手軽に使えるようになった点では意義があったかもしれません。

このデータを用いて、曖昧検索が可能な〝地名サーバー〟を作っておけば、住所録の作成や、曖昧な記憶からの正確な住所の検索などが容易になります。さらに、検索された郵便番号をもとに前述の〝ここのサーチ〟を利用すれば、地図や店舗情報などもより簡単に使えるようになります。

図 4　ローマ字表記の住所データベース

```
      ……
東京都,品川区,西中延 toukyouto,shinagawaku,nishinakanobu 1420054
東京都,品川区,旗の台 toukyouto,shinagawaku,hatanodai 1420064
東京都,品川区,東大井 toukyouto,shinagawaku,higashiooi 1400011
東京都,品川区,東五反田 toukyouto,shinagawaku,higashigotanda 1410022
東京都,品川区,東品川 toukyouto,shinagawaku,higashishinagawa 1400002
東京都,品川区,東中延 toukyouto,shinagawaku,higashinakanobu 1420052
東京都,品川区,東八潮 toukyouto,shinagawaku,higashiyashio 1350092
      ……
```

地名データベース

郵政省が Web ページで公開している 7 桁郵便番号のデータベースを使って、図 4 のようなローマ字表記の住所データベースが作れます。

地名/郵便番号検索サーバー

上記のようなデータをもとにして、ローマ字の読みまたは郵便番号から住所データを検索する地名検索サーバーを作ってみましょう（ソースコードは末尾に掲載します）。

このサーバーは、2 種類のコマンドを受け付けます。

S（検索コマンド）

形式：S　**曖昧度　検索文字列**

曖昧度を指定し、地名または郵便番号を検索します。曖昧度は〝0〜9〞の 1 桁の数字で、0 が完全一致、9 がもっとも曖昧になります。サーバーは、検索された項目数を 10 進数で返します。

C（検索結果取得コマンド）

形式：C　**取得する最初のデータのインデックス**
[, 最後のデータのインデックス]

直前の検索結果を取得します。サーバーは、直前の検索でマッチした項目のうち、指定されたインデックスに該当するものを返します。

図 5 は、telnet コマンドで住所検索サーバーに接続して検索しているところです（下線部がユーザーの入力です）。

最初に、〝emerald〞という名前を含む地名を探そうとしています。曖昧度 0（完全一致）では該当するデータはなく、曖昧度を 1 にして再検索してもみつかりません。しかし、曖昧度を 2 にして検索すると 4 件のデータがみつかり、〝エメラルドタウン〞という地名が表示されています。

2 番目の例では、〝uchide koduchi〞を含む地名を検索しています。

図 5　地名サーバーの検索

```
% telnet localhost 5560
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
S0 emerald
0
S1 emerald
0
S2 emerald
4
C0,3
青森県,南津軽郡,田舎館村,堂野前 0381102
石川県,石川郡,鳥越村,出合 9202368
静岡県,田方郡,韮山町,エメラルド タウン 4102103
奈良県,山辺郡,山添村,堂前 6302222
S0 uchide koduchi
1
C0
兵庫県,芦屋市,打出小槌町 6590028
Q
Connection closed by foreign host.
%
```

最初の例のように曖昧度を試行錯誤しながら入力するのは面倒ですが、検索結果が得られるまで曖昧度を自動的に増やす〝動的曖昧検索〞方式を利用すれば、指定したパターンにもっとも近い読みの住所を簡単にみつけることができます。

Web からのアクセス

CGI で上記の地名検索サーバーにアクセスして地名を検索する Web ページも簡単に作れます。まず、図 6 の HTML 文を作成して郵便番号検索用のページ（図 7）を作ります。

検索文字列として〝hatsudai〞を入力し、図 9 の CGI スクリプトを用いて検索すると、結果が図 8 のように表示されます。図 9 のスクリプトでは、検索結果が得られるま

図 6　郵便番号検索 HTML

```
<html>
<head><title>郵便番号検索</title></head>
<body bgcolor=white>
<form method="get" action="cgi-bin/zip.cgi">
郵便番号検索 <input name="word" size=20>
</form>
</body>
</html>
```

図 7　郵便番号検索ページ

図 8　郵便番号検索結果

で曖昧度$mismatchを増やしていくことにより、動的曖昧検索を実現しています。

この検索結果の郵便番号をモーバイルインフォサーチのページで入力すれば、その場所に関する地図などの情報が得られます。

新しい応用

地図や位置情報が手軽に扱えるようになると、いままでにない新しい応用が可能になります。

本格的地理情報システムへの進化

地図情報とそれ以外の情報とを融合させれば、高度な GIS として使えるようになります。たとえば、ProAtlas のような地図ソフトが地価や天気の情報とリンクしているのは、本格的な GIS への進化の第 1 段階と捉えることもできます。思いつくままに挙げただけでも、次のような例が考えられます。

- grep コマンドを使って NetNews から地名を検索して地図上にプロットし、〝話題になっている地域〟を視覚化する
- 住所データベースを用いて姓の分布を視覚化する
  〝望月〟は山梨県に多いとか、〝小野寺〟は東北地方に多いといった傾向が一目で分かります。
- 国勢調査の結果を地図上に視覚化する

図 9　郵便番号検索用 CGI スクリプト

```
#!/usr/local/bin/perl
require "chat2.pl";
$zipserver = "localhost";
$zipport = 5560;

# CGIの引数を取得
$ENV{'QUERY_STRING'} =~ /^word=(.*)/;
$pat = $1;
$pat =~ s/[\+\*]/ /g;

# サーバーに接続
&chat'open_port($zipserver,$zipport) ||
  die "ZIP server <$zipserver> not found";

# HTMLヘッダを出力
print <<EOF;
Content-type: text/html

<html>
<head><title>郵便番号検索結果</title></head>
<body bgcolor=white>
<font size=6>``$pat'' 検索結果</font><br>
<a href="http://www.kokono.net/move.html">
「ここの」位置入力</a>
<hr>
<table border>
<tr><th>住所</th><th>郵便番号</th></tr>
EOF

# 動的曖昧検索
for $mismatch (0..3){
  &chat'print("S$mismatch $pat \n");  # 検索要求
  $n = &chat'expect(2,'(.|\n)+','$&');
  for($i=0;$i<$n;$i++){
    &chat'print("C$i\n");              # 結果取得
    $_ = &chat'expect(2,'(.|\n)+','$&');
    s/,//g;
    ($addr,$zip) = split;
    print "<tr><td>$addr</td><td>$zip</td>\n";
  }
  last if $n > 0;
}

# HTMLフッタ出力
print <<EOF;
</table>
</body>
</html>
EOF
```

これらは簡単な例ですが、複雑なデータを地図情報と関連づけた本格的なシステムも手軽に利用できるようになるでしょう。これが実現すれば、一般用の地図ソフトの使いやすさと GIS の優れた機能とをあわせもつ、優れた地図利用システムとしてひろく普及すると思います。

図 10　WING

Web/テキスト検索の補助

現在の Web による検索の大部分はテキスト情報を中心としたものですが、地理的関係を組み合わせれば検索の効率が上がる可能性があります。

完全な位置や正確な名前が不明でも、おおよその場所と名前さえ分かれば、検索も容易です。たとえば、SnapTrack 社のページを検索するとき `San Jose の´ `なんとか Track という名前の´ `情報関連の会社´というふうに、場所と名前、カテゴリーを組み合わせて検索できれば、全世界から探すよりはるかに簡単で効率的です。

図 10 は、3 次元地図による検索とカテゴリー/名前による検索を組み合わせた観光案内システム「WING」[2] です。WING では、画面の左上に表示された地図の操作による地域指定と、左下のカテゴリー指定、右下の名前指定による絞込みを組み合わせて検索できるので、テキストだけ、あるいは地図だけを使う場合にくらべて、はるかに効率的な検索ができます。Web 上のデータについても、地図情報と組み合わせた検索が可能になれば、現状よりもはるかに簡単に目的の情報を探しだせるはずです。

位置にもとづく情報交換

前述のアルプス社のクチコミ地図情報やゼンリンの ZiLand では、Web 上で特定の場所の情報が交換できます。位置が認識できる携帯端末を使えば、その場所に関心のある人たちのあいだでリアルタイムに情報交換ができるでしょう。たとえば、ある駅の周辺にいる人たちがその周辺の情報を話題にする `駅前チャット´が可能になりますし、自分のいる場所と目的とをネットワーク上で公開すれば、目的を共有する人との新たな出会いの場になるかもしれません。あるいは、野球場やコンサート会場などの特定の場所でチャットをしたり、店の評価などを仮想的に貼り付けたり読んだりしながら街を歩くといった楽しみ方もあるでしょう。

中古 PC を売りたい人が、PC を秋葉原に持参して、値段などの情報をその人の周辺に仮想的に貼りだすといった使い方も考えられます。この例の場合は、秋葉原周辺が仮想的な `市´として機能し、その場で PC が売れる可能性が高くなります。

時空間情報の利用

位置と時間の情報を組み合わせれば、応用範囲はさらにひろがります。

たとえば、その昔、遺跡でどのような生活が営まれていたかを学ぶ、今後のコンサートのスケジュールを調べる、ビデオの予約状況をブラウズする、1 年前に書棚にあった雑誌のリストを調べるといった各種の検索が可能になります。時空間情報を思いのままに操作、検索できれば、また新たな応用がひろがるのではないでしょうか。

おわりに

地図関連会社に勤務している人のなかには、名刺に緯度・経度情報を印刷している人もいるそうです。このような人はまだ少数ですが、地図データベースがさらに普及すれば位置情報の活用も珍しくなくなるでしょう。

米国では、各種の地図データベースがインターネットなどから無料で入手できますが、日本ではわずかしか公開されていません。各種データベースの公開が待たれます。

地理情報システムをはじめとする地図関連情報は、アルプス社の Web ページに数多くのリンクがまとめられています。今回紹介した住所検索サーバーのデータとソースコードは、http://www.csl.sony.co.jp/person/masui/UnixMagazine/で公開していますのでご利用ください。

（ますい・としゆき　ソニー CSL）

[参考文献]

[1] 高橋克巳、三浦信幸、坂本仁明、島 健一「位置指向の情報統合」、*Japan World Wide Web Conference '97 Proceedings*、1997 年 12 月 (http://www.kokono.net/w3c/)

関連 URL

国土地理院	http://www.gsi-mc.go.jp/
アルプス	http://www.alpsmap.co.jp/
インクリメント P	http://www.incrementp.co.jp/indexj.html
昭文社	http://www.mapple.co.jp/company/mapple.html
ゼンリン	http://www.zenrin.co.jp/
ソニー Navin' You	http://vaio.sony.co.jp/software/NavinYou/top.html
クチコミ地図情報	http://www.alpsmap.co.jp/forum/kutikomi/index.html
Zi-Land	http://z.zenrin.co.jp/
Mapion	http://www.mapion.co.jp/
MapFanWeb	http://www.mapfan.com/
Mapoo	http://www.mapoo.or.jp/
WildBird	http://www.wildbird.or.jp/
駅前探検クラブ	http://ekimae.toshiba.co.jp/
東京レストランガイド	http://www.nihon.net/tokyo/
Sakura Garden	http://member.nifty.ne.jp/SAKURA_GARDEN/
マップサーバ	http://www.alpsmap.co.jp/mapserv/
Dmapwin	http://member.nifty.ne.jp/Hirokun/map/dmapwin.htm
GPS Player	http://hp.vector.co.jp/authors/VA004314/
NMEA-0183	http://www4.coastalnet.com/nmea/0183.htm
NIFTY-Serve FGPS	http://www.niftyserve.or.jp/forum/fgps/
いまどこサービス	http://www.nttdocomo.co.jp/products/phs/service/ichi.html
ここ Navi	http://www.nttdocomo.co.jp/products/phs/mobile/a-kokonav.html
	http://www.lanworld.co.jp/phs.htm
DDI 位置情報サービス	http://www.j-plaza.or.jp/ddi-pocket/ichijoho/gaiyou.html
P ナビ /データ	http://www.astel.co.jp/tokyo/location/
たべあるき東京	http://www.mapple.co.jp/tabe/
SnapTrack	http://www.snaptrack.com/
NTT DoCoMo 位置情報サービス	http://www.nttdocomo.co.jp/new/contents/98/whatnew55.html
モーバイルインフォサーチ	http://www.kokono.net/
郵便番号データ	http://www.postal.mpt.go.jp/newnumber/search.htm
地図関連情報	http://www.alpsmap.co.jp/mapsite/

[2] 水口 充、増井俊之、ボーデン・ジョージ、柏木宏一「なめらかなユーザインタフェースによる地図情報検索システム」、田中二郎 (編)『インタラクティブシステムとソフトウェア III』、日本ソフトウェア科学会 WISS'95、pp.231–240、近代科学社、1995 年

[3] 三浦信幸、高橋克巳、横路誠司、島 健一「位置指向の情報統合—モーバイルインフォサーチ 2 実験」、情報処理学会 第 57 回全国大会 講演論文集 第 3 分冊、pp.637–638、1998 年 10 月

地名サーバーのソースコード

```
//
// 地名検索サーバー
//
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

main()
{
  readdata();
  server();
}
```

```
//
//  曖昧検索ルーチン（1998年2月号で紹介したASearchと同じもの）
//  makepat(パターン,曖昧度);
//  match(テキスト);
//
int mismatch;
unsigned int epsilon, acceptpat;
unsigned int shiftpat[0x100];

void makepat(unsigned char *pat, int m)
{
  int i;
  unsigned int mask = 0x4000;

  mismatch = (m <= 0 ? 0 : m <= 3 ? m : 3);
  epsilon = 0;

  for(i=0;i<0x100;i++) shiftpat[i] = 0;
  for(;*pat;pat++){
    if(*pat == ' '){ // ワイルドカード文字
      epsilon |= mask;
    }
    else {
      shiftpat[*pat] |= mask;
      mask >>= 1;
    }
  }
  acceptpat = mask;
}

int match(register unsigned char *text)
{
  unsigned int i0 = 0x4000, i1=0, i2=0, i3=0;
  unsigned int mask;
  unsigned int e = epsilon;

  for(;*text;text++){
    mask = shiftpat[*text];
    i3 = (i3 & e) | ((i3 & mask) >> 1) | (i2 >> 1) | i2;
    i2 = (i2 & e) | ((i2 & mask) >> 1) | (i1 >> 1) | i1;
    i1 = (i1 & e) | ((i1 & mask) >> 1) | (i0 >> 1) | i0;
    i0 = (i0 & e) | ((i0 & mask) >> 1);
    i1 |= (i0 >> 1);
    i2 |= (i1 >> 1);
    i3 |= (i2 >> 1);
  }
  switch(mismatch){
    case 0: return (i0 & acceptpat);
    case 1: return (i1 & acceptpat);
    case 2: return (i2 & acceptpat);
    case 3: return (i3 & acceptpat);
    default: return 0;
  }
}

//
//  郵便番号辞書の読込み
//
#define ZIPFILE "./loclist"

#define MAXWORDS 120000
unsigned char *address[MAXWORDS];
```

```
unsigned char *addressyomi[MAXWORDS];
unsigned char *zip[MAXWORDS];
int nlines = 0;

#define MAXLEN 1000

readdata()
{
  unsigned char buf[MAXLEN];
  unsigned char s1[MAXLEN],s2[MAXLEN],s3[8];
  FILE *f;

  if((f = fopen(ZIPFILE,"r")) == NULL){
    fprintf(stderr,"Can't open ZIP file %s\n",ZIPFILE);
    exit(0);
  }
  while(fgets(buf,1000,f)){
    int len = strlen(buf);
    if(buf[len-1] == '\n') buf[len-1] = '\0';
    sscanf(buf,"%s %s %s",s1,s2,s3);
    address[nlines] = (unsigned char*)strdup(s1);
    addressyomi[nlines] = (unsigned char*)strdup(s2);
    zip[nlines] = (unsigned char*)strdup(s3);
    nlines++;
  }
}

//
//  検索サーバーのメインルーチン
//

#include  <ctype.h>
#include  <sys/types.h>
#include  <sys/ioctl.h>
#include  <sys/socket.h>
#include  <netinet/in.h>
#include  <netdb.h>

#define MAXDTAB  20
#define MAXCLIENT 32  // 最大接続数
#define  BUFSIZE 512  // リクエストの最大サイズ

#define  MAXQUE 5

#define ZIPSERVERPORT  5560

int initsock;            // 要求待ちsocket
int clientfd[MAXCLIENT]; // 各クライアント用socket
int nclients = 0;

server()
{
  int i;
  int len;
  struct sockaddr_in sin;
  struct sockaddr_in from;
  struct servent *sp;

  fd_set readfds, writefds, exceptfds;
  fd_set getrfds();

  bzero((char*)&sin, sizeof(sin));
```

```
  sin.sin_family = AF_INET;
  sin.sin_addr.s_addr = htonl(INADDR_ANY);

  sin.sin_port = htons(ZIPSERVERPORT);

  if((initsock = socket(PF_INET,SOCK_STREAM,0))<0){
    fprintf(stderr,"socket: can't create a socket");
    exit(0);
  }
  if(bind(initsock, struct sockaddr*)&sin,sizeof(sin))<0){
    fprintf(stderr,"bind: socket already used");
    exit(0);
  }
  if(listen(initsock,MAXQUE)<0){
    fprintf(stderr,"listen:");
    exit(0);
  }

  FD_ZERO(&writefds);
  FD_ZERO(&exceptfds);
  for(;;){
    FD_ZERO(&readfds);
    FD_SET(initsock,&readfds);
    for (i = 0; i < nclients; i ++)
      FD_SET(clientfd[i], &readfds);
    if (select(MAXDTAB,&readfds,&writefds,&exceptfds,NULL)<0){
      fprintf(stderr,"select:");
    }
    if (FD_ISSET(initsock, &readfds)) {
      len = sizeof(from);
      if((clientfd[nclients++] = accept(initsock,&from,&len)) < 0){
        fprintf(stderr,"accept:");
      }
      if(nclients >= MAXDTAB-3) {
        write(clientfd[--nclients],"F",1);
        close(clientfd[nclients]);
      }
    }
    for(i=0;i<nclients;i++){
      if (FD_ISSET(clientfd[i],&readfds)) {
        if (process(clientfd[i]) < 0) {
            // -1 means client closed the connection
          close(clientfd[i]);
          clientfd[i] = clientfd[nclients-1];
          nclients--;
        }
      }
    }
  }
}

//  クライアントからの要求と返答
//  C  "S0abc\n"  "abc"を曖昧度0で検索
//  S  "123\n"    123個マッチ
//  C  "C2"       2番目の項目内容を要求
//  S  "大きな"    項目内容を返す

#define MAXCANDS 100       // 最大候補数
int candindex[MAXCLIENT][MAXCANDS];
int ncands[MAXCLIENT];     // 検索結果の候補数

process(int fd)
{
```

```
  unsigned char buf[BUFSIZE];
  int n;
  int i;
  unsigned char retbuf[MAXLEN+10];

  if((n=read(fd,&buf[0],BUFSIZE))<=0){
    fprintf(stderr,"read error; transmission between processes\n");
    return -1;
  }
  else { // サーバーへの要求
    // 行末の改行文字の除去
    for(;n>0 && (buf[n-1]=='\n' || buf[n-1]=='\r');n--);
    buf[n] = '\0';
    if(buf[0]=='S'){              // 検索要求 "S1 emerald ", etc.
      makepat(&buf[2],buf[1]-'0');
      for(i=n=0;i<nlines && n<MAXCANDS;i++){
        if(match(addressyomi[i]) || match(zip[i])){
          candindex[fd][n] = i;
          n++;
        }
      }
      ncands[fd] = n;
      sprintf(retbuf,"%d\n",n);
      if(write(fd,retbuf,strlen(retbuf))<0){
        fprintf(stderr,"error in writing to socket\n");
        return -1;
      }
    }
    else if(buf[0]=='C'){         // "C1", "C2,4", "C2,10,2", etc.
      int start=0,end=-1,skip=1;
      n = sscanf(buf+1,"%d,%d,%d",&start,&end,&skip);
      switch(n){
        case 1:  end = start;
        case 2:  skip = 1;
        case 3:  break;
        default:
          start=0; end = -1;
      }
      for(i=start;i<=end && i<ncands[fd];i+=skip){
        strcpy(retbuf,address[candindex[fd][i]]);
        strcat(retbuf," ");
        strcat(retbuf,zip[candindex[fd][i]]);
        strcat(retbuf,"\n");
        if(write(fd, retbuf, strlen(retbuf)) < 0){
          fprintf(stderr,"error in writing to socket\n");
          return -1;
        }
      }
    }
    else if(buf[0] == 'Q'){       // クライアント切断要求
      return -1;
    }
    else if(buf[0] == 'X'){       // サーバー中止要求
      close(initsock);
      close(fd);
      exit(0);
    }
  }
  return 0;
}
```