インターフェイスの街角 (29) — 個人用情報管理システム Q-Pocket

増井俊之

個人情報管理の悩み

世の中では Zaurus や PalmPilot、Palm サイズ PC など、各種の携帯情報端末がひろく使われています。最近は超小型の PC でも UNIX が動きますし、PocketBSD のように電池で動かせるものも多くなってきたので、名簿やメモなどの個人情報の管理に UNIX を利用している人も多いのではないでしょうか。しかし、そのわりには UNIX 上で雑多な個人情報を管理するためのツールはあまり使われていないようです。

UNIX のファイルシステムでは、プログラム・ファイルなどの性質や形式が定型的な情報は比較的簡単に扱えます。しかし、アイデアやメモ、手紙のように形式も内容も雑多な情報をうまく管理するのは苦手なようです。

階層型ファイルシステムをもつ UNIX や Windows などのシステムでこの種の雑多な個人情報を管理する場合、私は以下のような点に問題を感じています。

- ファイル名やディレクトリ構造に悩まなければならない

  UNIX や Windows などでは、データは名前を付けて保存するのが当り前と考えられていますが、性格がはっきりしない雑多なデータの場合は、名前や格納場所を決めるのが難しい場面がよくあります。内容に即したファイル名を考えるのは面倒ですし、あとで編集して内容を変更すると、ファイル名が中身にそぐわなくなることもあるでしょう。携帯情報端末でメモを書く場合も、Windows CE ではファイルに名前を付けなければなりませんが、PalmPilot の〝メモ帳〟なら名前を付けなくてもよいので気が楽です。

  ファイルを格納するディレクトリ構成も悩みの種です。現在、UNIX でも Windows でも木構造のディレクトリでデータを管理する手法が主流です。これを使いこなすには、物事をうまく階層的に分類する能力が必要ですが、これは誰にでも簡単にできることではありません。たとえば、手紙や FAX などを管理する場合、Letter/Yamada/、Fax/Tanaka/といったディレクトリ構成にすべきか、それとも Yamada/Letter/、Tanaka/Fax/といったふうにしたほうがいいのかと迷ってしまいます。あるいは、街を歩いていておもしろいものを売っている店をみつけたとき、その情報をどこに書いておけばいいのでしょうか。再度訪れるときのために TODO リストに書いておくべきか、店情報データベースに入れるべきか、〝おもしろいものリスト〟のデータベースを新たに作るべきか、アイデア・データベースに入れればいいのか …… などと悩んでしまいます。それでも頻繁に使うファイルならばいいのですが、ごくたまにしか参照しないものは、どういう方針でディレクトリを作ったかを思い出せずに頭を抱えてしまうことがよくあります（すくなくとも私はそうです）。

  エディタでデータ内容を簡単に修正できる場合も、いったんデータの置き場所の階層構造や名前を決めてしまうと、大幅に変更するのは難しくなります。

- アプリケーションの選択

  データをどのようなアプリケーションで扱える形式にすべきかも考えなければなりません。

  メッセージを手紙、電子メール、FAX のいずれで送るかは大きな違いではありませんが、アプリケーションはそれぞれ異なるのが普通です。たまたまそのときの気分で特定のアプリケーション形式のデータを作ってしまうと、データそのものが捜せなくなることもあります

し、あとでほかの形式に変換するのにも手間がかかります。最初からデータのカテゴリーを決めなければならないのは、やはり不便です。

- 検索が簡単におこなえない

  多くの計算機システムやPDAでは、検索機能は補助的なものと位置づけられているようです。これまでに何回か紹介した各種の検索システムを使えば、名前や内容による検索はできます。しかし、それでも特別なシステムを用意しなければなりません。データの検索機能は計算機を使ううえでもっとも重要な機能の1つであるにもかかわらず、いろいろと用意しなければならないのでは面倒です。

- 異なるシステムとのデータ同期が難しい

  PalmPilotやZaurusなどの携帯情報端末内のデータとデスクトップ計算機のデータとの整合をとるために、いろいろな〝データ同期(シンクロ)機構〟が開発され、なかには携帯情報端末のボタンを押すだけで両者のデータの整合性がとれるようになっているものもあります。一見便利な機構ですが、現実に使ってみるとさまざまな問題が発生します。

  – あらゆるアプリケーション間でデータを同期させるには、携帯情報端末のアプリケーションの種類にデスクトップ計算機のアプリケーションの種類を乗じた数のツールが必要になりますが、そのすべてを開発するのは非現実的です。事実、UNIXのアプリケーションと携帯情報端末との同期をとるツールはほとんどないので苦労します。
  – 携帯情報端末とデスクトップ計算機のアプリケーションとではデータ形式が大きく異なることが多いため、どうしても変換しきれない部分が発生します。たんなる名簿データであっても、片方の機械では電話番号を2つ以上登録できるのに、もう一方のシステムでは1つしか登録できないのであれば、完全にデータを一致させることはできません。
  – ユーザーにとって、データの同期や変換の手順はかならずしも分かりやすいものではありません。したがって、ちょっと変わった処理が必要になるとお手上げになってしまいます。たとえば、3種類以上の計算機間でデータの同期をとる方法はそう簡単には分からないでしょう。

このように、既存の方法では個人情報をうまく管理できないのが実情です。

そこで、今回はこれらの問題を解決するために私が開発した情報管理システム「Q-Pocket」を紹介します。

Q-Pocket

Q-Pocketでは、**あらゆる情報をIDだけで管理**することによって前述の問題点の解決を試みています。

すべての情報を単純なテキストで表現し、ディレクトリ名やファイル名の代わりに情報の作成/更新時間を示すIDを用いて管理します。この手法には、以下のような利点があります。

- 情報をカテゴリーに応じて分類する必要がなく、ディレクトリ名やファイル名を気にせずにすむ。
- IDで情報をソートしておけば、〝超整理法〟と同じように、最近扱った情報がつねに先頭に位置するように並べることができる。
- データ構造が単純なので、どのような計算機/PDAでも同じデータを共通に扱うことができ、データの整合性を簡単に保てる。

  UNIXなどではIDをファイル名に対応づければよく、ファイルシステムをもたないPalmPilotのような機器でも、データとIDとの対応づけは容易です。

  複数のシステムで異なる情報が作成された場合でも、データをマージするだけで整合性が保てます。同一内容のデータが複数できることもあるとは思いますが、たいていはIDがもっとも新しいものを使うと割り切ればよいでしょう。

  データ内容の変換を含まない、たんなるコピー操作を主体としているため、相互変換にかかわる問題が減り、結果として異なるシステム間でのデータ変換も容易になります。
- バージョン管理が不要である。

  編集前のデータを破棄しなければ、異なるIDをもつ複数のデータとして保存されるため、旧い情報を保存しておきたい場合にも便利です。

Q-Pocketのもう1つの特徴は、**既存の情報の検索/更新/情報追加を基本操作とする**点にあります。

Q-Pocket で扱う情報には名前が付いていないので、情報にはキーワード検索のみによってアクセスします。ディレクトリ名やファイル名がないとデータを捜しにくいと思われるかもしれませんが、強力な検索機能があればこれらの名前は不要です。個人の扱う情報の数は多くても数万件程度でしょうから、検索時間が問題になることもありません。

Q-Pocket に情報を追加するときは、既存のデータをテンプレートとして使います。たとえば、同じ組織に所属する複数の人に関する情報を入力したり、似たような手紙を何通も出すときなどは、作成済みの情報をもとにすると便利なことがよくあります。このような場合、普通のテキストファイルやエディタを使うのであれば、次のような手順で作業を進めることが多いのではないでしょうか。

1. テンプレートとなるデータを検索する。
2. それをコピーする。
3. 新たに空のデータを作成する。
4. コピーしたテンプレート・データをペーストする。
5. 情報の追加や修正をおこなう。
6. 別の名前を付けて保存する。

これに対し、Q-Pocket では以下の手順で作業を進めます。

1. テンプレートとなるデータを検索する。
2. 情報の追加や修正をおこなう。
3. 新しい ID で追加保存する。

このように、手間を大幅に減らすことができます。

Q-Pocket の使用例

図 1 は、Mule (Emacs) 上で Q-Pocket を動かしているところです。アイデアやレストラン情報などの雑多な情報のタイトルが新しい順に並んでおり、カーソル位置の情報の内容が下のバッファに表示されています。各行の先頭にある 14 桁の数字が情報の作成時刻を示す ID で、ファイル名にもこの ID を使います。カーソルを移動させると、カーソル位置の ID をもつデータが下のバッファに表示されます。

上のバッファでのキー入力は検索パターン指定と解釈され、そのパターンにマッチした情報だけが動的に検索されて一覧表示されます。たとえば `rest´という文字列をパターンとして入力すると、`rest´を含む情報が検索、表示されます (図 2)。

このように、内容の異なる雑多な情報を一緒くたにしている場合も、キーワードで簡単にフィルタリングできれば、`レストラン・データベース´のような目的別データベースを作った場合と同じ効果が得られます。レストラン情報を書いたメールや、食事をしたレストランの感想を書いた日記など、どのようなデータでも検索できるので、定型的なデータベースよりも柔軟で役に立ちます。

図 1　Q-Pocket 画面

図 2　`rest´で検索した例

図 3　テンプレート・データの検索

下のバッファに移動してデータを修正してから引数付きで `(save-buffer)´を呼び出す[1]と、修正後のデータに新しい ID が割り当てられ、その ID に対応するファイルを引数として `(write-file)´が実行されます。したがって、旧い ID に対応するデータはそのまま残り、新しいデータが追加されるかたちになります。このようにしておくと、検索したデータをテンプレートとして新しいデータを作ったり、いろいろなバージョンのデータを保存しておくときに便利です。

新たなデータを作成するときは、テンプレートとなるデータを検索して修正してもいいですし、何もない状態から書き始めてもかまいません。たとえば `resuto temp´という検索パターンを入力すると、レストラン情報のテンプレート・データが表示されます（図 3）。ここで下のバッファに移動してデータを編集し、新しいレストラン情報を追加します。定型データの各項目をすべて入力するのが面倒なら、とりあえずレストランの名前だけメモしておけばいいでしょう。新たなデータを保存すると、そのデータがリストの先頭に表示されます（図 4）。

上のバッファで Ctrl-T を押すと、カーソル位置のデータに新しい ID が割り当てられて先頭に移動します。たとえば、図 4 の `Playstation2´のところで Ctrl-T を押すと画面は図 5 のように変化します。

1 通常のキー割当てでは Ctrl-U Ctrl-X Ctrl-S です。

図 4　情報を追加して保存

図 5　情報の `超整理´的更新

Q-Pocket の実装

Emacs や Tcl/Tk などからも使えるようにするために、Q-Pocket では検索をおこなうサーバー部分を分離して実装しました。ソースコードは長くなるので掲載しませんが、私の Web ページ[2]から入手することができます。

14 桁の ID をもつ多数のデータをそれぞれファイルとして扱うため、各 ID に対して、

2 http://www.csl.sony.co.jp/person/masui/UnixMagazine/

$*PIMDIR*/年/月/日/*ID*.pim

というディレクトリ構成にしています。

サーバー

サーバーは ID をもつデータのリストを保持し、クライアントからの検索要求に応じて検索を実行し、マッチした結果を返します。データを〝読み〟で検索できるようにするため、サーバーでは検索時に漢字を〝かな〟やローマ字に変換するプログラム KAKASI を用いて各データをローマ字に変換したものを使います（本来、KAKASI はフィルタ・プログラムですが、ライブラリとして呼び出せるように若干改造しています）。

サーバー部は、以下のファイルから構成されています。

`pimserver.c`：サーバのメインルーチン（クライアントからの要求を受け付けて、データのリストを管理する）
`readdata.c`：指定されたディレクトリ以下の、〝.pim〟という拡張子をもつすべてのデータファイルを取得する（サーバーの初期化時に使用）
`contents.c`：ローマ字変換/タイトル抽出ライブラリ
`search.c`：曖昧検索ライブラリ
`server.c`：汎用のサーバー・ライブラリ

クライアント

Emacs/Mule 用のクライアント pim.el は、検索パターンの入力を処理したり、データの編集やファイル操作などをおこないます。クライアント上でパターン文字列が入力されると、クライアントは文字列をサーバーに送って検索を依頼します。検索結果のリストはすぐにクライアントに返され、クライアント・アプリケーションで表示されます。

クライアント側でデータの追加/削除をおこなった場合は、それをサーバーに知らせてリストを更新します。図 5 のように ID を付けなおす場合は、旧い ID の削除と新たな ID の作成をサーバーに通知します。

サーバーもクライアントもファイルを直接扱うため、両方とも同じマシン上で動かす必要があります。サーバー部は C 言語で記述されており、通常の UNIX および Windows 上の Cygwin 環境で利用できます。現在のところ、クライアント部は UNIX および Windows 上の Emacs/Mule と Tcl/Tk に対応しています。

おわりに

どのようなアプリケーションであっても、使用する際の制約は少ないほうがよく、機能の数も限定したほうが使いやすいのではないでしょうか。たとえば、PalmPilot のメモ帳などのアプリケーションにはわずかな機能しかないので、最初は物足りなさを覚えるかもしれません。しかし、使い続けていると、むしろ必要最小限の機能がよく選択されていると感じるようになります。こんなところが、人気の一因かもしれません。その PalmPilot でも、メモ帳と予定表、電話帳は別になっているように、データはカテゴリーに応じて分類すべきだという考えはかなり根強いようです。そろそろ、このあたりから考えなおすべき時期にきているのではないでしょうか。

Q-Pocket の情報管理法は、メール用のアプリケーションである MH の手法に似ているところがあります。MH ではあらゆるメールを番号で管理し、新しいメールほど大きな番号が付くようになっています。

MH を含め、各種のメールツールには、メッセージの分類や宛先リストの管理、メッセージの検索など、多数のデータを管理するための機能がたくさん用意されています。これらのメール用ソフトウェアにはさまざまなデータ管理機能があるため、メール以外のアイデアやメモなどのデータもメールメッセージとして管理する人もいるようです。しかし、これは発想の順序が逆ではないでしょうか。まず、どんなデータでもうまく扱えるような使いやすいデータ管理ソフトウェアを用意し、その上でメールの読み書きもできるようにしたほうがより応用範囲がひろがると思います。

Q-Pocket では、メールメッセージだけでなく名簿もレストラン情報もすべて同じ方式で管理されるので、受け取ったメールに宛先を追加し、別の情報を加えて送信するといった作業も簡単におこなえます。Q-Pocket を拡張してメールツールとしても使えるようにすれば、メールも手紙も名簿もアイデアも区別せずに管理できるようになるでしょう。

（ますい・としゆき　ソニー CSL）