インターフェイスの街角 (73) — コマンドメール

増井俊之

理想の PIM システム

Palm や Pocket PC に代表される PDA (Personal Digital Assistants) では、ちょっとしたメモや予定表、名簿、TODO など、小さく雑多ではあるが重要なデータを扱ういわゆる PIM (Personal Information Management) の機能が重視されています。また、PC や携帯電話などでも、個人的な情報を扱うためのいろいろな手法が使われています。

重要かつ多様な情報を扱うことから、以下のような要求を満たす PIM システムが求められています。

- いつでもどこでも使いたい。
- 必要になったときに情報を簡単に参照したい。
- 情報の追加や修正は手軽におこないたい。
- データは安全に確保したい (紛失したり破壊されたりするとひどく困る)。

これらの要望はごく自然なものといってよいと思いますが、以下にみていくように、これらをすべて満たすシステムの実現は容易ではありません。

PC でローカルに管理する

PC のファイルシステムや検索システムを利用したり、いわゆるカード型データベースを使えば、PC 上での PIM データの編集や閲覧、検索は比較的簡単です。しかし、小型化がかなり進んだとはいえ、現在の PC は〝いつでもどこでも〟手軽に使えるとはいえないように思います。ノート PC はつねに持ち歩けるほど軽くはありませんし、衝撃などに対して十分な強度があるともいえません。PC だけの話ではありませんが、持ち歩けるということは紛失の危険と隣り合わせですから、かならずデータのバックアップをとっておくといった注意も必要です。

また、複数の PC を利用している場合は、それらのあいだでデータを同期させなければならないのも面倒です。

携帯情報端末を使う

Palm や Pocket PC などの PDA はノート PC にくらべてはるかに小型で、しかも頑丈に作られています。したがって、いつでもどこでも持ち歩くのに適しているといってよいでしょう。

つねに同じ PDA を持ち歩いて情報管理に利用するのであれば、〝いつでもどこでも使いたい〟という要望は満たせます。しかし、PDA は PC にくらべて小さいぶん、文字入力や閲覧などの面では見劣りします。また、インターネットや PC 経由で情報をやりとりすることも多く、さらに紛失する危険性も高いので、PC などとのあいだでデータを同期させる必要があります。こういった点を考えると、PDA だけで生活するのはやはり難しそうです。

Wiki を使う

最近はインターネットのネットワーク基盤が整備されたため、ノート PC や PDA などに PIM 情報を入れて持ち歩くよりも、インターネット上のサーバーに情報を置いておき、Web ブラウザなどを使って読み書きする方式のほうが有利になってきました。インターネット上のサーバーにデータを格納しておけば、外出先でデータを紛失してしまうような事故は減るでしょうし、バックアップも比較的簡単にとれます。

Wiki Wiki Web などのシステムで PIM 情報を管理すれば、インターネットに接続された各種のシステムで Web

著者校正 [2004 年 2 月 1 日]

ブラウザを使い、情報を閲覧したり編集したりすることができます。最近はインターネットへのアクセス手段が多様化し、いろいろな場所で Web ページにアクセスできるようになりました。これらの点を考えると、さきほどの要望のうちのいくつかは満たせそうです。

とはいっても、現状ではいつでもどこでも手軽にインターネットにアクセスできるとはかぎらないので、やはり、サーバー上のデータを PC や PDA とやりとりする仕組みが必要になってしまいます。

携帯電話を使う

最近の多くの携帯電話は、PDA 代わりに使えるような、ちょっとした PIM 管理機能を備えています。携帯電話で受け取ったメールに書かれているメールアドレスや電話番号などは、簡単にアドレス帳に登録できます。すくなくとも、この点については普通の PDA よりもはるかに高機能といえるでしょう。しかも携帯電話は PDA よりも小型ですから、まさしく〝いつでもどこでも使える〟といってよいと思います。

とはいえ、PIM に関連するすべての機能を携帯電話で実行できるかというと、やはり無理があるでしょう。入力はけして楽とはいえませんし、画面が PDA よりさらに小さいので情報の閲覧性という点でも劣ります。さらに、ノート PC や PDA よりも紛失する危険性が高く、情報のバックアップも簡単ではありません。

複数の手段の組合せ

こうしてみてくると、現状では最初に述べた PIM への要望を完全に満たすシステムはなさそうです。そもそも、閲覧しやすい大きさの画面を備えていることと携帯性は矛盾するので、単体ですべてを満たすのは不可能ではないかという気もします。

1 つのシステムで要望を満たせないのなら、その時々の状況に応じて異なるシステムを使う手法はどうでしょうか。たとえば、次のような方法が考えられます。

- PC とインターネットが使えるときは Wiki でデータを読み書きする。
- インターネットが使えないときは PC 上の（同期された）データを使う。
- PDA ではデータのサブセットを持ち歩く。
- 携帯電話しか持っていない場合でもアクセス可能な手段を用意する。

私はあらゆる PIM データを PC と PDA で共有しているので、すくなくとも PDA を持ち歩いていれば、ほとんどすべての PIM データを読み書きすることができます。ところが、PIM にレストラン情報を書き込んでおいたのに、肝腎なときに PDA や PC を持っていなかったため情報が参照できず、ひどく残念な思いをしたことがあります。また、その場ですぐに PIM データとしてメモを残しておきたいのに、携帯電話しか手許になかったため、とりあえず自分にメールを送って間に合わせてしまうこともしばしばです。

携帯電話しか持っていなくても、すべての PIM 情報にアクセスできるのであれば、このような問題は起きなかったはずです。たとえ最低限の PIM 情報であっても、携帯電話からのアクセスを可能にしておけば、〝いつでもどこでも使える PIM システム〟として要件はかなり満たされるでしょう。

コマンドメール

携帯電話から特殊な形式のメールを送ることによって PIM 情報にアクセスする仕組みがあれば、携帯電話で PIM 情報を参照したり、新たな情報を書き込んだりすることができます。携帯電話にはさきほど述べたような欠点はありますが、それでもレストラン情報を参照したり、メモを追加する程度であれば操作にまごつくこともなさそうです。

この場合、まず考えつくのは、pim@example.com や diary@example.com など、用途に応じた特別なメールアドレスを用意し、それぞれにコマンドを送る方法です。しかし、これを実現するにはサーバーの管理者権限が必要ですから、敷居が高くなってしまいます。それよりも、特殊な形式のメール（コマンドメール）を自分宛に送り、それぞれに応じた操作をおこなうようにしたほうが手軽に使えるでしょう。

2003 年 5 月号で、メールの振分け処理をおこなうための〝lens〟というシステムを紹介しました。そのときと同じく、~/.forward ファイルにコマンドメールを処理する

著者校正 [2004 年 2 月 1 日]

プログラムを指定し、そこから PIM を操作する方式にすれば、いろいろなコマンドメールを定義して活用できるようになります。

.forward ファイルは、その名が示すとおり、もともとはおもにメールの転送先を簡便に指定するために使われていました。しかし、最近はメールの分類や SPAM のフィルタリングなど、受け取ったメールに対する各種の自動処理を記述しておくことが多いようです。今回のシステムのなかでも、受け取ったメールを内容に応じて適切に処理する〝ディスパッチャ〟として利用しています。X サーバーに送られたさまざまなイベントが、ディスパッチャによって適切なアプリケーションに割り振られるのと同じように、自分宛に送られてきたメールがディスパッチャによってうまく分類されると考えればよいわけです。

私がふだん利用している.forward ファイルは、以下のようにごく単純なものです。

```
"| /usr/local/bin/lens"
```

ここで指定している lens は、以前に紹介したプログラムを拡張し、以下の処理をおこなうようにしたものです。

- SPAM メールや不要なメールは、削除したり特別な場所に移動したりする。
- メーリングリストから受け取ったメッセージは、それぞれのフォルダに分類する。
- 知人からのメッセージも、それぞれのフォルダに分類する。
- 重要なメールは携帯電話に転送する。

コマンドメールを扱う場合は、これらに加えて、

- 特殊な形式のコマンドメールを受け取ったら、それに応じた適切な処理をおこなう

という機能を追加すればよいことになります。

メールの自動分類や転送には procmail がひろく使われているようです。しかし、procmail は用途が限られているにもかかわらず、特殊な記号や言語を使って振分け規則を記述しなければなりません。したがって、上記のような分類やコマンドメールの処理などに本格的に利用する場合には、メールを自在に扱える柔軟なプログラムを.forward に指定するほうが賢明です。

コマンドメールの形式

メールのどこかにコマンドを指定する必要がありますが、 Subject:行にコマンドを指定し、メール本文に引数を記述することにしました。普通のメールが誤ってコマンドメールとして解釈されないように、文字列の最後に以下のような記号を付け、コマンドメールであることを明示します。

! ：書換え/起動など
? ：検索
＋：追加など

たとえば、

```
Subject: スケジュール?
```

というメールを自分宛に送ると予定表を取り寄せることができ、

```
Subject: スケジュール!
```

```
2004/3/1 13:00
山田氏打合せ
```

というメールを送れば新たな予定を追加できるようにします。

最近は高い山の上でも携帯電話が使えることが多い[1]ので、

```
Subject: 日記!
```

```
立山山頂についた! 富士山まで見える!
```

のような形式で日記を送れるようにしておけば、どこでもリアルタイムに日記が書けるようになります。

時刻指定メール

このコマンドメール・システムでは、スケジュールやメモなど、PIM としての基本的な機能だけでなく、時刻を指定してメールを送る機能も実装してみました。

最近のメールは、送信したらすぐに相手に届くのが普通ですが、到着日時を指定できると便利な場合もあるでしょう。たとえば、 n 日後にメールが自分宛に届くようにすれば、リマインダーとして使えます。あるいは、相手の都合がよいときを見計らってメールが届くように工夫をすることも可能です。

1 立山でも月山でも、山頂で au の携帯電話が使えました。

図 1　時刻指定メール送出

```
def delaymail(message)
  to = message.header['To']
  subject = 'Delayed mail'
  st = Time.now

  while true do
    line = message.body.shift
    break if line.nil?
    line.chomp!
    if line =~ /^\s*$/ then
      break
    elsif line =~ /^\S+@/ then
      to = line
    elsif (ta = ParseDate.parsedate(line)) != [nil] * 8
      ta[0] = Time.now.year if ta[0].nil?
      st = Time.local(*ta[0..5])
    else
      subject = line
    end
  end

  delayfile = "/tmp/delaymail#{$$}"
  File.open(delayfile,'w'){ |f|
    f.write(<<CMDEND)
/usr/local/bin/ruby -x
#!ruby
require 'net/smtp'
Net::SMTP.start('smtp.example.com', 25) { |smtp|
  smtp.send_mail(<<EOF,'masui@example.com','masui@example.com')
Subject: #{subject.tojis}
To: #{to}

#{message.body}
EOF
}
CMDEND
  }
  attime = sprintf("%02d:%02d %d/%d/%d",
    st.hour,st.min,st.mon,st.mday,st.year)
  system "at #{at_time(attime)} -f #{delayfile}"
  File.unlink(delayfile)
end
```

Subject:行の `時刻指定!´という文字列でコマンドを指定し、本文部分に宛先や配信時刻、サブジェクト、遅延配信したいメールの本文を記述しておきます。

たとえば、次のような形式のメールを自分宛に送ったとします。

```
Subject: 時刻指定!

masui@example.com
2005/7/1
免許証更新

免許証更新忘れずに!
```

この場合、2005 年 7 月 1 日に以下のようなメールを受け取ることになります。

```
To: masui@example.com
Subject:免許証更新

免許証更新忘れずに!
```

コマンドメールの本文にコマンドメールを記述することもできます。たとえば、次のようなメールを送っておくと、2004 年 3 月 20 日午前 8 時にスケジュールを問い合わせるメールが送られ、その時点のスケジュール・データが返

著者校正 [2004 年 2 月 1 日]

送されます。

```
Subject: 時刻指定!

masui@example.com
2004/3/20 8:00
スケジュール?
```

このようにいろいろなコマンドを用意し、それらを複合的に利用すると便利だと思います。

コマンドメールの実装

lens プログラムにすこし手を加えれば、Subject:行の記述によってコマンドを判定する処理を簡単に追加できます。

PIM とのデータのやりとりに関する処理は、対象となる PIM のデータ構造によって異なります。私自身は 2001 年 12 月号で紹介した Wiki を利用しているので、``Subject: Wiki?''という形式のメールを受け取ったら、該当する Wiki データを送り返すようにしています。

時刻指定メールの送信には at コマンドを使います。at コマンドでは、

```
% at -f file 8:00 3/27/2004
```

のような形式で、時刻を指定してファイルを実行することができます（at コマンドでは、2004/3/27 のような``年/月/日''形式は認識してくれないようです）。時刻指定メールで指定された時刻をこの形式に変換し、メールを送るプログラムのファイルを作成して at コマンドを起動すれば、時刻指定メールを実現することができます。

使ってみた感想

携帯電話でデータを本格的に編集するのは大変ですが、単純なデータの追加や取得程度であれば、それほど難しくはないでしょう。必要とする PIM データをいつでもどこでも扱えるのはたいへん便利ですし、携帯電話や PDA を紛失してもデータ自体は残るという意味では安心感があります。

私自身、これまでに PIM データを管理するためのいろいろなシステムを試してきましたが、現在のところは``これ''といった万能の解決方法はなさそうです。当面は、Wiki や携帯電話などをうまく組み合わせて使うのがよさそうに思います。

おわりに

メールにコマンドを記述して実行させるという考え自体は、とくに目新しいものではありません。しかし、私の知るかぎりでは、実際に活用されているケースは少ないようです。

メールのメッセージは、現在もっとも汎用的に使えるデータ形式かもしれません。どんなに特殊な計算機であっても、メールの送受信さえできれば、データの交換が可能になります。たとえば、いわゆる``情報家電''を操作するプロトコルとしてメールが使えれば、ハードディスク・レコーダにメールを送って録画を指示したりすることもできるでしょう。

携帯電話やメールの応用範囲は、まだまだ大きくひろがっていく余地があるのではないかと思います。

（ますい・としゆき　産業技術総合研究所）

著者校正 [2004 年 2 月 1 日]