インターフェイスの街角 (18)
ブラウザによる個人情報管理

増井俊之

Web による共同作業支援

インターネットの普及とともに高機能なブラウザが登場し、まったく〝World Wide〟ではない Web 上でも、ブラウザを使っていろいろな仕事が簡単にできるようになってきました。

計算機を用いた共同作業支援 (CSCW: Computer-Supported Cooperative Work) に関する研究は、10 年ほど前から活発におこなわれています[1]。ところが、データを共有したり、もともとシングルユーザー向けに開発されたアプリケーションを複数のユーザーで使うための手法に関するコンセンサスが得られなかったため、普及したツールはあまりありませんでした[2]。しかし、この状況も、Web がひろく使われるようになって変わりつつあります。Web 上では、各ユーザーが汎用の高機能なブラウザを用いてサーバー内のデータを共有するだけで協調作業を手軽に実現できます。そこで、このような機能を利用した各種の Web ベースの CSCW ツール (いわゆる〝イントラネット構築ツール〟) が開発・販売されています。特定のアーキテクチャに依存しない Web ベースの CSCW システムは、ブラウザさえあればどこでも誰でも使えるので、幅広い人間がかかわる共同作業の支援にうってつけです。

共同作業では、共有資源の予約やスケジュールの調整が重要なので、イントラネット構築システムでも、会議室などの使用予約やグループのメンバーのスケジュール管理のためのツールを基本にしていることが多いようです。各ユーザーの要求を反映してスケジュールを自動的に調整するような高度なシステムについては、従来からさまざまな研究がおこなわれています。しかし、単純に他人のスケジュールや共有資源の使用状況を Web ブラウザで閲覧できるようにするだけでも、CSCW ツールとしては十分な価値があるでしょう。

一方、現在の Web ベースのインターフェイスの大きな問題は、〝データを眺めるのは楽でも作成するのは大変〟という点でしょう。情報を見聞きするより作るほうがはるかに大変なのは、音楽でも小説でも雑誌の記事でも (?) 同じです。とはいえ、情報の発信者より受信者のほうが圧倒的に多いときは発信者が頑張ればいいわけですから、情報発信にかかる手間はそれほど問題にはなりません。ところが、発信者と受信者との区別がない共同作業では、発信の手間が大きいと作業自体が停滞してしまいます。

Web ベースの情報共有においては、このほかにも解決すべき課題があります。たとえば、データの内容だけでなく見え方についても、発信者がある程度は面倒をみなければなりません。UNIX 上の共有ファイルシステムなどを用いて情報を共有する場合などは、ファイル名と内容だけに注意すればいいでしょう。しかし、Web を介した共有ではページの見え方や操作性などにも配慮が必要なので、過不足なくデータを共有できるようにするにはかなりの手間と時間がかかります。たんにファイルにリンクを張ったりするだけでは、手抜きをしたようにみえはしないかと心配になります。

Web ブラウザでは、情報の閲覧は簡単でも、編集は容易ではないという問題もあります。共有ファイルを編集できるようにするだけでも、エディタとして必要な機能の

1 計算機科学に関する世界最大の学会 ACM (Association for Computing Machinery) では、CSCW Conference を隔年で開催しています。日本でも、情報処理学会グループウェア研究会などで CSCW に関する研究発表がさかんにおこなわれています。

2 Lotus Notes のように一部で普及したツールもありますが、本当に普及した CSCW ツールは NetNews と電子メールくらいでしょう。

図 1　ファイル 105 の内容

```
1999/3/25 11:00-14:00 <a href="http://www.csl.sony.co.jp/">CSL</a>増井氏来社
1999/4/1 14:00-16:00 <a href="smart.html">SMARTプロジェクト</a>キックオフ（小沢）
```

画面 1　会議室予約状況の確認

多くを CGI などで提供しなければなりません。ふだんは Emacs などの UNIX のエディタを使っているユーザーは、情報共有のためだけにブラウザ付属のエディタを使うのは面倒に感じるのではないでしょうか[3]。

たいして手間をかけずに情報を発信し、Web ブラウザで手軽に情報を共有できるシステムを実現するためには、すくなくとも現時点では、情報の作成や編集には使い慣れたエディタなどの（CSCW 用として開発されたものではない）ツールを使い、これに手を加えて共有する Web コンテンツを半自動生成するのが妥当なところでしょう。

たとえば、UNIX システムでは printcap ファイルを用いてプリンタの設定を定義し、lpq コマンドでプリンタの状況を確認します。しかし、その記述言語やインターフェイスはあまり分かりやすいとはいえません。利用可能なプリンタの一覧を表示し、そのなかから状況を調べたいプリンタを選んで確認できるようなシステムがあると便利でしょう。printcap ファイルや lpq コマンドを用いて適当な HTML ファイルを生成し、ブラウザで見やすく表示してプリンタの状況をチェックできるようにすれば、多くのユーザーにとってはるかに使いやすくなるのではないでしょうか。

現時点では、UNIX ベースで共同作業を支援する場合には何もかもブラウザで操作するのではなく、次のような方針で対処するほうが有効だと思います。

- 共有資源データ（printcap ファイルなど）は、エディタなどの既存のツールで作成・編集する。
- 共有資源の状況（lpq の出力や printcap ファイル）を見やすい HTML ファイルに変換するスクリプトを用意し、ユーザーが簡単に眺められるようにする。

会議室予約システム

今回はまず、ごく簡単な CSCW システムの例として〝会議室予約システム〟を作ってみることにします。上記の方針にもとづいて考えると、次のようにすればいいでしょう。

- 予約の追加・修正は、予約状況を示すテキストファイルの編集によっておこなう。
- 予約状況を Web ブラウザで閲覧できるようにする。

このシステムで生成された HTML ファイルは、ブラウザ上でカレンダーふうに表示されます（画面 1）。

運用に必要なのは、予約状況ファイルと HTML への変換コマンドだけです。しかし、予約状況ファイル名や HTML ファイル名をいちいち指定するのは面倒なので、1 つのコマンドで指定できるようにします。

meetingroom コマンド（末尾のリスト 1）を実行すると、自分がふだん使用しているテキストエディタが起動され、引数で指定された会議室予約状況ファイルが編集できるようになります。図 1 のようなファイル〝105〟を作成して編集作業を終えると自動的に HTML ファイルが作られ、画面 1 のようなカレンダーが生成されます。

予約状況ファイルでは、⟨a⟩ タグを使って別の資料へのリンクを記述しておくと便利です。図 1 のファイルでは、ソニー CSL のホームページへのリンクを指定しているので、画面 1 の 3/25 のエントリにある〝CSL〟の部分をクリックすると、そのページに移動します。

3　こう感じるユーザーは私だけではないでしょう（私は、ブラウザに文章を入力するときはいつも Emacs から切り貼りしています）。文章の作成も編集もつねに Web ブラウザ上でおこなうようになれば、状況は変わるかもしれませんが …… 。

多くの Web ベースのイントラネット構築ツールでは、このような予約システムは CGI で実装されています。しかし、使いやすいものを作るには大がかりなプログラムが必要ですし、新しい予定を追加したり、それぞれの予定に対して編集や削除などの操作ができるようにするには、画面上に多くのボタンやメニューを配置しなければなりません。これに対して、今回の例のような共有テキストの編集による単純な手法は十分に実用的であるばかりでなく、はるかに簡単なプログラムで実現できます。さらに、使い慣れたテキストエディタを編集のために利用できるという利点もあります。

個人情報管理システム

次に、上記と同様の手法にもとづき、Web を個人情報管理ツールとして活用する方法を紹介します。

今日では、個人スケジュール管理ツールは、パーソナル・コンピュータや携帯端末上のもっとも重要なツールの 1 つといっていいでしょう。しかし、いろいろな場面で異なる機械に入力したデータを統一的に扱うための手法が悩みの種です。最近、デスクトップ・コンピュータと携帯端末とのあいだでデータを同期させるようなシステムが流行していますが、N種類の計算機間でデータを同期させるには N^2のオーダーの同期ソフトウェアが必要になってしまいます。

このような問題は、個人情報をすべて Web 上で管理すれば解決するでしょう。現在のほとんどの計算機は Web データを扱えるので、スケジュールなどが Web 上にあれば、どの計算機からでも同じデータを参照することができます。最近は、My Yahoo (http://my.yahoo.com/) や Netcenter (htp://www.netscape.com/) などのいわゆるポータルサイトが無料のカレンダー・サービスを提供するようになり、スケジュール管理を完全に Web 上だけでおこなえるようになりました。これらのサービスでは、カレンダーなどが HTML で表示されるため一覧性に優れていますし、どのブラウザでも情報の入力や編集ができるのでたいへん便利です。

このようなポータルサービスは今後もさらに広まっていくと思いますが、問題がないわけではありません。たとえば、社内会議のスケジュールなどをポータルサービスまかせにするわけにはいきませんし、サービス提供者の都合や混雑により使えなくなるのでは困ります。ブラウザ経由で情報を入力したり編集したりする作業は、テキストエディタを用いた編集にくらべるとどうしても隔靴掻痒の感がぬぐえません。Web 上のスケジュール管理システムの便利なところを残したまま、個人で使える情報管理システムが望まれます。

個人情報管理システムの必要条件

個人情報についても、CSCW の場合と同様に UNIX 上のテキストベースのファイルやデータベースを基本データとして利用し、適宜 HTML ファイルに変換して容易に参照できるようにすれば、UNIX 上で Web ブラウザを活用した効率的な管理が可能になります。前述の会議室予約システムと同様な方針にもとづいて、次のようなことが実現できるでしょう。

- テキストエディタで個人情報の入力・編集をおこなう。
- 見やすい形式で全体を簡単に閲覧できるようにする。
- 関連情報へのリンクを張る。

さらにブラウザの機能を活用して、

- 絵や写真を併用する

ように工夫します。日付に関連したデジタルカメラの画像や CrossPad から取り込んだメモなどをカレンダーに貼り付けておけば、その日に何をしたかを簡単に思い出せますし、今後の予定にもアクセスしやすくなります。また、

- 日記、メモ、ファイル、家計簿などとの統合

なども有用だと思います。メモ帳と日記は別々に管理するのが一般的であり、メモ帳は内容別に、日記は日付順に並べるのが普通です。更新したメモやファイルとカレンダーの日付とのあいだにリンクを張っておけば、その時期におこなった仕事の内容などを簡単に参照できます。

一般にファイルは内容別に分類されますが、〝超整理法〟的にカレンダーを使い、時間順にファイルをアクセスできれば、〝2 カ月ほど前に書いていた報告書〟なども簡単にみつけられるようになります。超整理法と同様な考え方にもとづき、計算機内のすべてのデータを時間順に並べて管理しようという Lifestreams システム [1] と同様の利用法が実現できそうです。

図 2　/user/masui/1999/3/23/の中身

```
% ls /user/masui/1999/3/23
19990323101926@  19990323160036@  19990323203526@  diary
19990323112424@  19990323173635@  19990323210654@  event
19990323120521@  19990323175616@  19990323212802@  expense
19990323130211@  19990323184828@  19990323214021@
19990323142522@  19990323193650@  19990323222356@
19990323145825@  19990323203202@  19990323224418@
19990323155032@  19990323203333@  19990323231324@
% cat /user/masui/1999/3/23/diary
風邪がひどくて鼻水がとまらない！
%
```

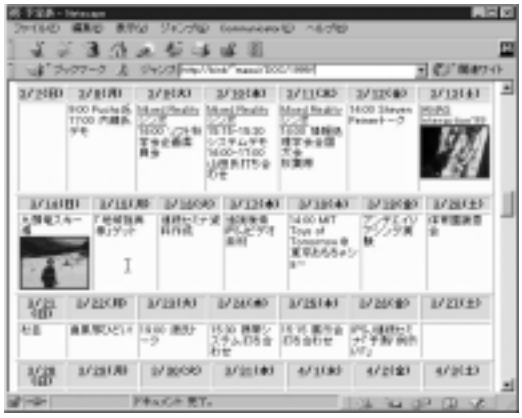
文章を書くときには、カテゴリーではなく日付にもとづいたファイル名のほうが都合がよいこともあります。たとえば、雑文やアイデアノートなどのように、作成時にはカテゴリーが不明な文章を書く場合には、事前にディレクトリやファイル名を決めるのは難しいでしょう。これを、たとえば〝アイデア〟などという名前のファイルと日付とを関連させて保存するようにすれば、文章を書く前にファイル名について悩まずにすみます。

実装の方針

以上の点を考慮し、下記の方針にもとづいてファイルの作成や Web ブラウザでの閲覧ができるようにします。

- 予定表や日記、雑文などは、日付に対応したディレクトリに格納する。
  例：/user/masui/1999/3/23
- event、diary、essay など種類別のコマンドを起動し、日付に対応したディレクトリの下にコマンドと同名のファイルを作成する。
  例：1999/3/23/essay、1999/3/23/diary など
- 普通のファイルの編集時に RCS の ci コマンドでチェックインをおこなったとき、日付に対応したディレクトリからそのファイルへのリンクを作成する。
  例：1993/3/23/19930323112345 → DOC/meibo/Masui.Toshiyuki のように、時刻を表現するファイルからシンボリック・リンクを張る。
- デジタルカメラの画像や CrossPad によるメモなども日付ディレクトリに格納する。
- 日付ディレクトリ内のデータをまとめて HTML ファイルを生成し、予定や日記、写真、ファイルなどを年間/月間カレンダーから簡単に参照できるようにする。

画面 2　予定表



たとえばルート・ディレクトリとして/user/masui を指定した場合、/user/masui/1999/3/23/の中身は図 2 のようになります。

編集コマンドの実装

event、diary などのコマンドはリスト 2 のような簡単なもので、現在時刻からディレクトリ/ファイル名を決定してテキストエディタを起動するだけです。同じプログラムに event や diary など異なる名前を付け、diary コマンドとして起動すると日付ディレクトリ中の diary ファイルが、event コマンドとして起動すると event ファイルが編集できるようになっています。

こうしておけば、編集対象となるファイルの名前やディレクトリを気にする必要がなく、気軽に起動して編集をおこなうことができます。時刻などについても、各コマンドに対応したテンプレートを用いて自動的に置換するようになっています。

画面 3　日記

HTML の生成

リスト 3 のプログラム **makecalendar** を使って、各日付に対応した event、diary などのファイルをカレンダー形式の HTML に変換すると、Web 上の予定表ができあがります（画面 2）。

同様の方法で月間カレンダーや各日付に対応するページを index.html として作成し、それぞれをリンクしておくとさらに便利です。画面 3 の日記では、編集したファイルへのリンクや写真が並んでいるため、それぞれの日にどのような仕事をしていたかなどが一目瞭然です。同様にして各日付に対してもページを作成して、さらに詳しい情報を見られるようにします。

PDA との連携

このように、個人情報を Web データに変換しておけば、Web ブラウザが利用できる計算機上で個人情報を簡単に参照できるようになります。ただし、PalmPilot などの小型携帯端末を使いたいときには、データ同期ツールを利用する必要があるかもしれません。このような場合でも、個人情報データをおもに UNIX 上のテキストファイルにしておけば、テキスト形式で日付ごとに格納されている予定表などのデータと PDA のデータ形式とを相互変換するプログラムを用いて、比較的容易に両者を同期させることができます。

PalmPilot を例にとると、**calendar2pdb** と **pdb2calendar** という 2 つの変換ツール[4]を用いてデータをそれぞれに変換し、UNIX と PalmPilot 間の通信ソフト pilot-xfer[5]を利用すれば、以下のようにデータの相互変換が可能になります。

- **UNIX → PalmPilot**

```
% calendar2pdb > DatebookDB.pdb
% pilot-xfer -i DatebookDB.pdb
```

- **PalmPilot → UNIX**

```
% pilot-xfer -f DatebookDB
% pdb2calendar DatebookDB.pdb
```

おわりに

Web の登場によって、CSCW や個人情報の管理方法まで変化しつつあるのは興味深い現象です。今後のインターフェイスは Web を抜きには考えられませんし、ほとんどの仕事は Web ブラウザだけですむようになる可能性があります。文字入力や文書の編集さえも CGI とブラウザだけで実行するのが当然という時代が到来するのかもしれません。

たいへん便利な世の中になったものだと感慨深く思う一方で、誰もが現状の Web ブラウザの機能に満足してしまうと、新たなインターフェイスの提案がかえって困難になるのではないかと心配でもあります。Web ブラウザをさらに画期的に使いやすくするようなインターフェイスが開発され、標準的に利用されるようになってほしいと願っています。

（ますい・としゆき　ソニー CSL）

[参考文献]

[1] Eric Freeman and Scott Fertig, Lifestreams: Organizing your electronic life, In *AAAI Fall Symposium: AI Applications in Knowledge Navigation and Retrieval*, November 1995 (http://www.cs.yale.edu/HTML/YALE/CS/HyPlans/freeman/papers/AAAI/draft.ps)

4 私の Web ページ http://www.csl.sony.co.jp/person/masui/UnixMagazine/で公開しています。

5 ftp://ryeham.ee.ryerson.ca/pub/PalmOS/pilot-link.0.9.1.tar.gz に含まれています。

リスト 1　meetingroom コマンド

```
#!/usr/local/bin/perl
require 'pim.pl';
require 'getopts.pl';
require 'timelocal.pl';

# Webブラウザから閲覧可能な共有ディレクトリ
$dir = '/www/local/meetingroom';
chdir($dir);

$EDITOR = "mule -nw";
$EDITOR = $ENV{'EDITOR'} if $ENV{'EDITOR'};

sub addroom {
  local($room,$name,$attr) = @_;
  push(@rooms,$room);
  $name{$room} = $name;
  $attr{$room} = $attr;
}

&addroom('104', '104会議室', 'color=orange');
&addroom('105', '105会議室', 'color=red');
&addroom('201', '201会議室', 'color=blue');
&addroom('305', '305会議室', 'color=green');
&addroom('big', '大会議室',  'color=black');

$room = shift;
unless($name{$room}){
  print STDERR "% meetingroom [",join(',',@rooms),"]\n";
  exit;
}

($year,$mon,$mday) = &getdate(time); # きょうの日付を取得

link($room,"$room.bak");

system "$EDITOR $room";
chmod 0666,$room;

foreach $room (@rooms){
  open(in,$room);
  while(<in>){
    if(/^(\d+)(\s+|\/)(\d+)\/(\d+)\s+(.*)$/){
      $event{$room,$1,$3,$4} .= "$5\n";
    }
  }
  close(in);
}

open(out,"> index.html");
select(out);

$t = &timelocal(0,0,0,$mday,$mon-1,
  $year<2000 ? $year-1900 : $year-2000);
$t -= &dayofweek($year,$mon,$mday) * $daysec;

print <<EOF;
<html>
<head>
<title>会議室予約状況</title>
</head>
```

```
<body bgcolor=#FFFFDD>

<h1>会議室予約状況</h1>
<p>
EOF

foreach $room (@rooms){
  print "<font $attr{$room}>$name{$room}</font> -\n";
  print "<code>% meetingroom $room</code><br>\n";
}

for $week (0..20){
  print "<table border unit=relative width=100%>\n";
  print "<tr valign=top>\n";
  for $day (0..6){
    ($year,$mon,$mday) = &getdate($t+($week*7+$day)*$daysec);
        printf("<th bgcolor=#ffcccc width=%d%%>%d/%d (%s)</th>\n",
      ($day==0 ? 10 : 15),$mon,$mday,$dayname[$day]);
  }
  print "</tr>\n";
  print "<tr valign=top>\n";
  for $day (0..6){
    ($year,$mon,$mday) = &getdate($t+($week*7+$day)*$daysec);
    printf("<td width=%d%%>\n",
      ($day==0 ? 10 : 15),$mon,$mday,$dayname[$day]);
    foreach $room (@rooms){
      if($_=$event{$room,$year,$mon,$mday}){
        @a = split(/\n/,$_);
        for $a (@a){
          print "<font $attr{$room}>$a</font><br>\n";
        }
      }
    }
    print "<br></td>\n";
  }
  print "</tr>\n";
  print "</table>\n";
}

print <<EOF;
</body>
</html>
EOF

close(out);
```

リスト 2　event コマンド

```
#!/usr/local/bin/perl
require 'pim.pl';

($progname) = ($0 =~ m#([^/]+)$#);

$EDITOR = $ENV{'EDITOR'};
$EDITOR = "mule -nw" unless $EDITOR;

# 現在の日付/時刻または引数による指定を取得
($year,$mon,$mday,$hour,$min,$sec) = &getdate(time);

$date = shift;
```

```
$time = shift;
if($date =~ /^((\d\d\d\d)\/)?(\d+)\/(\d+)$/){
  $year = $2 if $1;
  $mon = $3; $mday = $4;
  $hour = $min = $sec = 0;
  if($time =~ /^(\d+):(\d+):(\d+)$/){
    $hour = $1; $min = $2; $sec = $3;
  }
}

&mkdir("$rootdir/$year/$mon/$mday");
$f = "$rootdir/$year/$mon/$mday/$progname";

# テンプレートがある場合は日付と時刻を置換して利用
if(open(in,"$templatedir/$progname")){
  $time = sprintf("%02d:%02d:%02d",$hour,$min,$sec);
  $date = sprintf("%4d/%02d/%02d",$year,$mon,$mday);
  open(out,">> $f");
  while(<in>){
    s/%Date%/$date/g;
    s/%Time%/$time/g;
    print out $_;
  }
  close(in);
  close(out);
}

# 編集コマンドを起動
system "$EDITOR $f";
```

リスト 3　makecalendar コマンド

```
#!/usr/local/bin/perl
require 'pim.pl';
require 'getopts.pl';
require 'timelocal.pl';

&Getopts('ay:');

($year,$mon,$mday) = &getdate(time);

@years = ($year);
@years = (1991..2030) if $opt_a;
@years = ($opt_y) if $opt_y;

for $yy (@years){
  next unless "$rootdir/$yy";
  open(out,"> $rootdir/$yy/index.html");
  select(out);

  # 1月1日の週の日曜日の時刻を求める
  $t = &timelocal(0,0,0,1,0,$yy<2000 ? $yy-1900 : $yy-2000);
  $t -= &dayofweek($yy,1,1) * $daysec;

  &prologue($yy);

  for $week (0..53){
    print "<table border unit=relative width=100%>\n";
    for $day (0..6){
      ($year,$mon,$mday) = &getdate($t+($week*7+$day)*$daysec);
```

```
      printf("<a name=%04d%02d%02d>\n",$year,$mon,$mday);
    }
    print "<tr valign=top>\n";
    for $day (0..6){
      ($year,$mon,$mday) = &getdate($t+($week*7+$day)*$daysec);
           printf("<th bgcolor=#ffcccc width=%d%%>" .
              "<a href=\"../$year/$mon/\">%d</a>/" .
        "<a href=\"../$year/$mon/$mday/\">%d</a>(%s)</th>\n",
        ($day==0 ? 10 : 15),$mon,$mday,$dayname[$day]);
    }
    print "</tr>\n";

    print "<tr valign=top>\n";
    for $day (0..6){
      ($year,$mon,$mday) = &getdate($t+($week*7+$day)*$daysec);
      printf("<td width=%d%%>  <!--- %d/%d(%s) --->\n",
        ($day==0 ? 10 : 15),$mon,$mday,$dayname[$day]);
      $dir = "$rootdir/$year/$mon/$mday";
      $event = "$dir/event";
      if(-f $event){
        open(in,"nkf -e < $event |");
        while(<in>){
          chop;
          print "$_<br>\n";
        }
        close(in);
      }
      else {
        print "<br>\n";
      }
      $image = "$dir/index.jpg";
      if(-f $image){
        $ref = "../$year/$mon/index.html#$mday";
        print "<a href=\"$ref\"><img src=\"../$year/$mon/$mday/index.jpg\"></a><br>\n";
      }
      print "</td>\n";
    }
    print "</tr>\n";
    print "</table>\n";
  }

  &epilogue($yy);

  close(out);
}

sub prologue {
  local($year) = @_;
  $lastyear = $year-1;
  $nextyear = $year+1;
  print <<EOF;
<html>
<head>
<title>予定表</title>
</head>
<body bgcolor=#FFFFDD>
<h1>予定表 ($year)</h1>
<a href="../$lastyear/">${lastyear}予定表</a>
<a href="../$nextyear/">${nextyear}予定表</a>
EOF
```

```
}

sub epilogue {
  local($year) = @_;
  $lastyear = $year-1;
  $nextyear = $year+1;
  print <<EOF;
<a href="../$lastyear/">${lastyear}予定表</a>
<a href="../$nextyear/">${nextyear}予定表</a>
</body>
</html>
EOF
}
```

リスト 4　pim.pl

リスト 1〜3 のプログラムを実行するには、下記の pim.pl が必要です。Web ブラウザから参照可能な URL とディレクトリを$rooturl、$rootdir に設定すればよいでしょう。

```
# pim.pl
sub getdate { # 時刻から年月日を得る
  local($t) = @_;
  local($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst);
  ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) =
    localtime($t);
  if($year >= 70 && $year < 100){ $year += 1900; }
  elsif($year < 40){ $year += 2000; }
  $mon++;
  ($year,$mon,$mday,$hour,$min,$sec);
}

sub dayofweek { # 年月日から曜日を得る
  local($year,$mon,$mday) = @_;
  if($mon == 1 || $mon == 2){
    $year--;
    $mon += 12;
  }
  ($year + int($year/4) - int($year/100) + int($year/400)
    + int((13*$mon+8)/5) + $mday) % 7;
}

sub mkdir { # 階層的にディレクトリ作成
  local($dir) = @_;
  local(@d,$i,$h);
  $h = ($dir =~ s/^\/// ? '/' : '');
  @d = split(/\//,$dir);
  for($i=0;$i<=$#d;$i++){
    $dir = $h . join('/',@d[0..$i]);
    mkdir($dir,0755) unless -d $dir;
  }
}

$daysec = 24 * 60 * 60;
@dayname = ('日','月','火','水','木','金','土');

$rooturl = "http://bird.csl.sony.co.jp/~masui/DOC";
$rootdir = "/user/masui/public_html/DOC";
$templatedir = "/user/masui/DOC/template";
}
```