創刊 3 周年記念号

TSNET スクリプト通信

TSC 編集委員会

ISSN 1884-2798

目次

巻頭言	jscripter	...	3
たったひとつの冴えたやりかた	Y さ	...	4
Python の文法 第 5 回	機械伯爵	...	20
よしおさんとロボ太　最終回	海鳥	...	29
エアチェック・ポッドキャスティング	jscripter	...	31
編集後記	jscripter	...	37

表紙写真: あじさいと水滴
撮影: jscripter
日時: 2011 年 5 月 29 日
場所: ベランダにて
メモ: 1cm マクロ撮影の最初の成果。

巻頭言

jscripter

暑い夏がやってきたが、雨が多い。梅雨明けは早かったというのに。新潟・福島は水害に見舞われている。被災された方には心よりお見舞い申し上げる。東日本大震災に続いて、記録的な豪雨による水害が襲う。地震・津波に誘発された原発事故という人災も解決にはほど遠い。しかしながら、暗くなっていても仕方がないので、我らも前に向いて進むしかない。

TSNET スクリプト通信も 3 周年を迎えることができた。執筆者や周囲で応援いただく方々のお蔭である。

世界は大きな変動の時期を迎え、先進国の経済は変調を来たし、新興国は急激な発展に揺らいでいる。日本国内は、自然の脅威に足元も覚束ない。我等は今何をなすべきかを考える必要がありそうであるが、具体的に何をすべきなのか、先が見えにくい。こういう時代には腰を据えて、じっくりと自分の技を磨き、世の中に貢献することを考えるしかない。どのように役立つことができるのかだが、問題意識は持っていきたい。

3 周年記念号に最初に投稿されたのは、Ｙささんの「たったひとつの冴えたやりかた」という洒落たタイトルの AWK ゲームプログラミングの記事である。前回の「プラス的・エフェクト」の続編なのだが、単なるゲームの作成というよりは次第にゲーム理論について掘り下げて書いておられるので、興味深く読むことができる。是非ご一読いただきたい。

機械伯爵さんの「Python の文法」は第 5 回。最近、Peter Seibel 著、青木靖訳「Coders at Work プログラミングの技をめぐる探求」(Apress Ohmsha、2011 年)を読んでいると、「Smalltalk よりも Python のほうがずーっとよい」とＬ・ピーター・ドイチェという人が言っている。なるほどそういうものなのかと感心している。「Python ではプログラムが何であり、プログラムを実行するとか、プログラムの一部であるというのがどういうことなのか、明確なストーリーがあります。」(428 ページ)。Python 再入門というのもターゲットにしてもよいかも。

海鳥さんの「よしおさんとロボ太」の転載は今回が最終回である。転載は、私のせいで、制作された順序どおりには行わなかったので、かなり前後しているものがある。ご了承いただきたい。いずれ何か整理してみたいと思っている。何はともあれ、転載を快諾いただいた海鳥さんに感謝している。新作をいただいたのも懐かしい思い出になる。また、思いがけず、Postscript や Ruby のプログラミング記事をいただいたのはうれしい出来事であった。ありがとうございました。

私の記事は、以前書いたラジオエアチェックサーバーのスクリプトの出力を iTunes の RSS 出力とすることによって、iTunes の Podcast としてエアチェックを自動的に読み込ませるような仕組みを作るものである。自分で言うのもおかしいが、なかなか便利に使っている。iPod touch へ音声などのメディア・ファイルを読み込ませるのは iTunes の同期機能を使うのが転送が速く、効率がよいのである。

(投稿: 2011 年 7 月 30 日)

たったひとつの冴えたやりかた

written by Yさ

1. この記事について

前回記事の指立て"チックル効果"ゲームスクリプトの続編というか、コンピュータ思考ルーチンの改訂版を作成しました。
いわゆる<二人・ゼロ和・有限・確定・完全情報>ゲーム(※)における、「先読み」ってどうやったらできるのか？ といったお悩みをお持ちの方の、ひょっとしたら何かの役に立つかもしれないので、スクリプトを作成するのにあたっていろいろ調べた事について、書いて残しておきたいと思います。

※二人ゼロ和有限確定完全情報ゲームとは？
詳しくはweb等を検索してくだd(x_x)＼ばき
...非常にざっくりと簡単に言えば、二人で行うゲームで一方が勝者となるなら
他方が敗者となるもの、の事です。

2. 動作環境について

執筆時の動作確認は、
GNU Awk 3.1.8, mawk 1.3.3 MBCS R27
で(WinXPのDOS窓で)行なってます。

ちなみにマシンは
・CPU : Core Duo T2600 / 2.16GHz
・メモリ : 1GB (PC2-5300 DDR2 SD-RAM)
...といった今となってはすっかりショボくなってしまった5年前のノートPCです
(;^^ゞ

3. コンピュータ思考ルーチン≒「ゲームの木」の生成と「先読み」

二人が交互に"手"を打っていくことで新しい局面が作られ進行していくゲームで、ある局面で打つことができるいくつかの"手"があるとします。
コンピュータに、その中からたったひとつの冴えた手を選ばせるにはどうすれば良いのでしょうか？
やや乱暴で一番原始的(?)な方法としては、その局面で可能な手のいずれかを乱数を使って選択するというものがあります...
※effect.awk に使われてない function rnd() が定義されて残っているのは、
そういった理由だったりします d(^^;

世の中一般的な手順としておおまかには、
1)コンピュータの手番の局面で、
1.1)"手"を一つ試し、新しい局面を作る。
その局面を何らかの方法で評価=評価点を算出する。
1.2)試した"手"を打つ前の元の局面に戻す。
1.3)可能な"手"全てについて、上記1.1)～1.2)を繰り返す。

2)全ての可能な"手"を試し、一番高い評価点の局面に進む"手"を選択する。

となります。

基本的には、全ての可能性を系統的に調べるバックトラッキングという手法を用います。

ここで 1.1) において「局面を評価」するにはどうするか？ ですが、

ある局面で"手"を試して新しい局面を作り、

→その新しい局面の手番のプレイヤー(相手)にとって一番高い評価となる"手"
を打ち、さらに新しい局面を生成したとして、

→その新しい局面の手番のプレイヤー(自分)にとって一番高い評価となる"手"
を打ち、さらに新しい局面を...

と繰り返して次の局面を作りながら、(双方打つ手が無くなった)最終局面でのプレイヤーの評価点を算出します。

それを元の局面まで遡って、局面の評価点とします。

なお、一番高い評価点の算出は、次の 2 ステップを行うことで実現できます。

1.1.1) これ以上新しい局面が作成できない局面(最終局面)に達した場合は、
評価点を算出し、呼び元に値を渡す。

1.1.2) そうでなければ、上記 1)～2)と同様に、この局面の手番のプレーヤーが
可能な"手"について全て調べ、後続局面を生成し、その中で一番高い評価点を
算出する。なお「後続局面の生成と評価」は、1.1.1), 1.1.2)を再帰的に行う
ことで実現する。

この局面を展開した結果がまるで枝葉が分かれた木のように見える事から「ゲームの木」と呼びます。また、「ゲームの木」を生成して自分の打つ手を決定することを「先読み」と呼びます。(人間でもエキスパートなプレイヤーは同じことをしていますね。)

ちなみに"チェス"や"将棋","オセロ"といった局面数が膨大となるゲームでは、終盤を除いて、ある程度の時間内で終了するように手数を限定して先読みするように工夫してプログラミングしている模様です。

...なのでどちらが優勢なのか不利なのかといった評価点をゲームの木の『途中』で算出する必要があり、それが難しいといわれています。

余談ですが、着手を評価しながらしらみつぶしに調べるのが一番困難とされる"囲碁"においては、近年では途中評価を諦めて(^^; あり余る CPU パワーを駆使し、ランダムに候補手を打って"終局"まで対局をシミュレーションして、結果その中で最も勝率が高かった着手を選ぶ、といういわゆるモンテカルロ法を応用したアルゴリズムを持つプログラムが登場しているとの事です。

まぁアメリカではクイズ番組で人間のチャンピオンと争って勝利しちゃうコンピュータ・システムが登場する時代ですからねぇ。

4. 局面の評価とミニマックス手続き

双方打つ手が無くなった最終局面でのプレイヤーの評価点ですが、

自分の勝ち=+1, 引分=0, 自分の負け=-1

とするのがもっともシンプルです。

自分が勝ちか少なくとも引分となるような手＝最大値となるものを選ぶ、

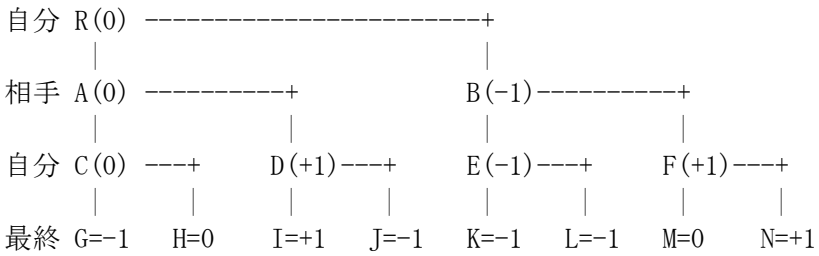
逆に相手も勝ちか少なくとも引分となるような手＝最小値となるものを選ぶ、

という想定で双方が打つ手を探索し決定するアルゴリズムを「ミニマックス手続き」と呼びます。

自分の局面の最大評価点は、相手が選択した最小評価点によって決定します。

今、自分の局面 R から A, B の局面が作れるとします。

相手の局面Aからは C, Dが、Bからは E, Fが作れます。
自分の局面C～Fからは、G～Nが作れるとして、そこが最終局面だったとします。



例えば、Cから作れる局面G=相手の勝ち=-1, H=引分=0だった場合、自分は最大値を選択するので、Cの評価点は0となります。
同様にAの評価点は、C, Dで最小の0となります。
こうして局面Rの評価点は0であることが分かり、RからAを作る手を選べば良い、
となります。

ミニマックス手続きの擬似的なロジックは次となります。

```
-----
  v=-search(自分の手番); # 現在の局面を評価
   :

function search(who,  eval,v)
{
  if(who 手番の最終局面である) return who 手番の評価点を算出して返却;
  現在の局面の状況を退避
  eval=-999;  # 仮に-∞としておく
  for(who 手番の全ての打てる手について){
    手を打ち、次の局面を生成
    v=-search(who 手番の相手);  # v に評価点をセット
    局面を退避していた状況に戻す
    if(v>eval) eval=v;  # 最大の評価点を更新
  }
  return eval;  # 最大の評価点を返却
}
-----
```

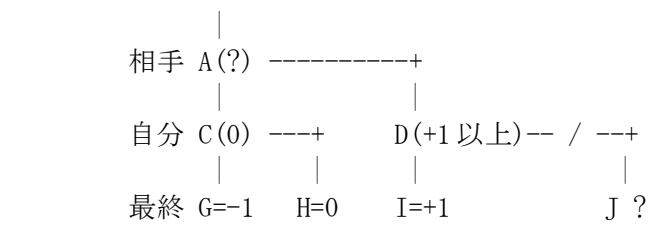
なお、このロジックは相手の評価点も自分の評価点と同じ、
勝ち=+1, 引分=0, 負け=-1
として算出し、符号を反転させて、
v=-search(who 手番の相手, lev-1)
とすることで相手の手番の最小の評価点を算出しています。
こうするとロジックがシンプルになります。
※ちなみにネガマックス(Nega-Max)と呼ぶらしいです。

5. αβ 枝刈り

ミニマックス手続きでは、全ての最終局面について評価点を算出しています。

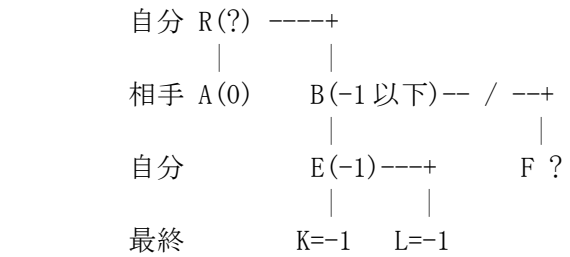
ゲームでは『自分が明らかに勝てる見込みが無い』と分かった局面について、そこから作られる局面を調べても意味がありません。
※ただし、"オセロ"のように石の数の差で最終的な勝負が決まる場合は別です。
負けるとしても、なるべく自分の石が多く残るようにする必要があります。

新たな局面を調べない＝ゲームの木の枝を狩る、という意味で枝刈りと呼びます。
先程の例で、Cの評価点0が分かったとします。



次にDの評価点を算出するために、まずIの評価点を調べたら +1 でした。
自分の手番では最大値を選ぶので、Jを調べなくても、Dの評価点は +1以上 であることが決定します。
この時点で、Cの評価点(0) < Dの評価点(+1以上) が確定するので、Aの評価点は 0 で確定します。
この、Jの評価点を調べない枝刈りを β 枝刈り(β cut)と呼びます。また、今の場合だとCの評価点 0 を、"β 値"と呼びます。

さて、これでAの評価点0が分かりました。続けてBの評価点を算出するとします。



Bの評価点を算出するために、まずEの評価点を調べたら -1 でした。
相手の手番では最小値を選ぶので、Fを調べなくても、Bの評価点は -1以下 であることが決定します。
この時点で、Aの評価点(0) > Bの評価点(-1以下) が確定するので、Rの評価点は 0 で確定します。
この、Fの評価点を調べない枝刈りを α 枝刈り(α cut)と呼びます。また、今の場合だとAの評価点 0 を、"α 値"と呼びます。

α β 枝刈りの擬似的なロジックは次となります。

```
-----
  v=-search(自分の手番, -999, 999);  # 現在の局面を評価
                                     # α 値=仮に-∞, β 値=仮に+∞ としておく
   :

function search(who,alpha,beta,  v)
{
  if(who手番の最終局面である) return who手番の評価点を算出して返却;
  現在の局面の状況を退避
```

```
  for(who 手番の全ての打てる手について){
    手を打ち、次の局面を生成
    v=-search(who 手番の相手, -beta, -alpha);  # v に評価点をセット
                              # α 値=-beta, β 値=-alpha とする
    局面を退避していた状況に戻す
    if(v>alpha) alpha=v;  # 最大の評価点を更新
    if(alpha>=beta) return alpha;  # 枝刈り = 最大の評価点を返却
  }
  return alpha;  # 最大の評価点を返却
}
```

このロジックも相手の手番の最小の評価点を算出する際に、符号を反転させて、

```
  v=-search(who 手番の相手, -beta, -alpha);
```

としてロジックをシンプルにしています。

※ちなみにネガアルファ(Nega-α)と呼ぶらしいです。

実際にミニマックス手続き版と α β 枝刈り版の動作を比較すると、

[#05]

```
 ( 3 )   ( 1 ) : COM
  III     I    .
                .
                .
    I     I    .
 ( 1 )   ( 1 ) : YOU
  [L]     [R]
```

この状態から、COM の次の手を決めるのに、

・ミニマックス手続き版: 評価点算出 4383902 回, 約 120 秒

・α β 枝刈り版: 評価点算出 132319 回, 約 5 秒

といった結果となりました。

やはり枝刈りの効果は大きいですね。

参考までに α β 枝刈り版で、

[#03]

```
 ( 1 )   ( 1 ) : COM
    I     I    .
                .
                .
    I     I    .
 ( 1 )   ( 1 ) : YOU
  [L]     [R]
```

この状態から、COM の次の手を決めさせると、

・評価点算出 1673274 回, 約 60 秒

といった結果となりました。

5. コンピュータ思考ルーチンの改訂版 effect2.awk について

遊び方は、前回記事を参考にしてください。

前回版からの変更点は、CPU の思考ルーチンを選択できるようにした所です。
beginner レベルが前回版の思考ルーチンです。
expert レベルとして「α β 枝刈り」ロジックを採用しています。
なお思考時間の問題で、BEGIN ブロックの SEARCHSTART=4; によって 5 手目から探索を行うように調整しています。

ちなみに自分が先手でも、expert レベルの後手の CPU に勝てなくなりました...
(;^^ゞ

6. その他

本記事のプログラム effect2.awk はフリーソフトです。
著作権は作者であるＹさにあります。
ただし転載、再配布、改造、消去は自由です。
(できましたら素晴らしい改造を加えた後に TSC 編集委員会宛に投稿してくださいませ)
また、このソフトを使用した事による損害が発生したとしても、損害に対しては一切の責任を負いかねます。

7. effect2.awk

本記事のプログラム effect2.awk を以下に示します。

[effect2.awk]

```
##  Even-Odd Effect rev.2  written by Ｙさ

function rnd(N){ return int(N * rand()); }  ##  乱数

BEGIN{
  srand();      ##  乱数の初期化

  # ゲームモード決定
  gm=GAMEMODE;
  if(gm!="0" && gm!="1" && gm!="2"){
    print " <Solitaire>   - 0"
    print " <Match-up>"
    print "  first move   - 1"
    print "  2'nd         - 2"
    printf("¥nWhich?[0-2]> ");
    do{
```

```
    gm=""; getline gm; gm=tolower(gm);
  }while(gm!="0" && gm!="1" && gm!="2");
}
GAMEMODE=gm+0;
print "¥n";

# CPU レベル決定
SEARCHSTART=4;
gm=CPULEVEL;
if(GAMEMODE!=0 && gm!="0" && gm!="1"){
  print " <CPU LEVEL>"
  print "  beginner - 0"
  print "  expert   - 1"
  printf("¥nWhich?[0-1]> ");
  do{
    gm=""; getline gm; gm=tolower(gm);
  }while(gm!="0" && gm!="1");
  CPULEVEL=gm+0;
  print "¥n";
}

## Turn;      ##  順番 0:人,1:COM
Turn=(GAMEMODE!=1)?(0):(-1);  #ゲーム開始時に+1
Times=0; #回数
## Sc[2];     ##  得点 0=人,1=COM
Sc[0]=Sc[1]=0;
## Bit[9];    ##  指 [人<1:左,3:右>, COM<9:左,7:右>]
Bit[1]=Bit[9]=0;
Bit[3]=Bit[7]=1;
## 場所判別 他
pNo[1,1]=7;  pNo[1,0]=9;
pNo[0,0]=1;  pNo[0,1]=3;
cnvPos[10]=pNo[0,1];  cnvPos[11]=pNo[1,1];  cnvPos[12]=pNo[1,0];
cnvPos[30]=pNo[0,0];  cnvPos[31]=pNo[1,0];  cnvPos[32]=pNo[1,1];
cnvPos[70]=pNo[1,0];  cnvPos[71]=pNo[0,0];  cnvPos[72]=pNo[0,1];
cnvPos[90]=pNo[1,1];  cnvPos[91]=pNo[0,1];  cnvPos[92]=pNo[0,0];
mv["LR"]=mv[12]=10;  mv["LU"]=mv[14]=11;  mv["LX"]=mv[15]=12;
mv["RL"]=mv[32]=30;  mv["RU"]=mv[36]=31;  mv["RX"]=mv[35]=32;
mv[78]=70;  mv[74]=71;  mv[75]=72;
mv[98]=90;  mv[96]=91;  mv[95]=92;

##  ゲームメイン
for(;;){
  #状況表示
  ++Times;
  dispHands(0);
  #GAME OVER / 1回休み判定
  if(gameOver(Turn)>0) exit;
  if(GAMEMODE!=0) if(++Turn>1) Turn=0;
```

```
    setSkip();

    do{
      #移動指定 -> 影響判定
      sw=(Turn==0)?(man()):(com());
      ev=canMove(sw);
      if(Turn==0 && ev<0){
        printf(" No effect, retry.");  pushEnter();
        dispHands(0);
      }
    }while(ev<0);
    #変化
    change(sw,ev);
  }
  exit;
}
END{
  print "¥n";
  if(finish(0)) printf("     Congratulations !!  (SCORE:%05d0)¥n", Sc[0]);
  else{         printf("     ****** GAME OVER ******¥n");
                printf("     ...YOU LOSE (SCORE:%05d0)¥n", Sc[0]);
  }
  print "";
}
function pushEnter(  tmp){ printf(" <PUSH ENTER>");  getline tmp; }
function yorn(  yn)
{
  do{ yn=""; getline yn; yn=toupper(yn); }while(yn!="Y" && yn!="N");
  return yn;
}


##
## 状況表示
##
function dispHands(sw,  d,ev,i)
{
  delete d;
  delete ev;
  makeDispTbl(sw,d,ev);

  print "";
  for(i=((GAMEMODE==0)?(6):(0)); i<=8; ++i) print d[i];
  print "";
}
function makeDispTbl(sw,d,ev,  s0,s1)
{      #0....5....0....5....0....5....0...
  d[0]="  <s1> .<s2>   <s3>. <s4>";
  d[1]="      :             : ";
```

```
  d[2]="<s5> : zzzzz   xxxxx : COM [cccc0]";
  d[3]="     . zzzzz   xxxxx .";
  d[4]="     .               .";
  d[5]=d[4];
  d[6]="     . @@@@@   &&&&& .";
  d[7]="<s0> : @@@@@   &&&&& : YOU [yyyy0]";
  d[8]="        [L]     [R]   ";

  if(sw>=90){       s0=    mkBitStr(Bit[9],1);
                    s1=    mkBitStr(Bit[9],0);
  }else if(sw>=70){ s0=rev(mkBitStr(Bit[7],1));
                    s1=rev(mkBitStr(Bit[7],0));
  }else if(sw>=30){ s0=    mkBitStr(Bit[3],0);
                    s1=    mkBitStr(Bit[3],1);
  }else if(sw>=10){ s0=rev(mkBitStr(Bit[1],0));
                    s1=rev(mkBitStr(Bit[1],1));
  }
                         #0....5....0....5....0....5....0...
  if(sw==90){           d[2]="<s5> : zzzzz" s0 "    : COM [cccc0]";
                        d[3]="     . zzzzz" s1 "    .";             }
  if(sw==70){           d[2]="<s5> :    " s0 "xxxxx : COM [cccc0]";
                        d[3]="     .    " s1 "xxxxx .";             }
  if(sw==30){           d[6]="     . @@@@@" s0 "    .";
                        d[7]="<s0> : @@@@@" s1 "    : YOU [yyyy0]"; }
  if(sw==10){           d[6]="     .    " s0 "&&&&& .";
                        d[7]="<s0> :    " s1 "&&&&& : YOU [yyyy0]"; }
                         #0....5....0....5....0....
  if(sw==31 || sw==91){ d[4]="     .         " s0 " .";
                        d[5]="     .         " s1 " .";  }
  if(sw==32 || sw==92){ d[4]="     .    " s0 "      .";
                        d[5]="     .    " s1 "      .";  }
  if(sw==12 || sw==72){ d[4]="     .      " s0 "    .";
                        d[5]="     .      " s1 "    .";  }
  if(sw==11 || sw==71){ d[4]="     . " s0 "         .";
                        d[5]="     . " s1 "         .";  }
             #0....                                #0....
  cnvHandStr(7,"zzzzz",sw,1,d,3,2);   cnvHandStr(9,"xxxxx",sw,0,d,3,2);
  cnvHandStr(1,"@@@@@",sw,1,d,6,7);   cnvHandStr(3,"&&&&&",sw,0,d,6,7);

  setAllEval(ev);
  # YOU
  sub(/s0/, ev[0], d[7]);
  sub(/yyyy/, sprintf("%04d",Sc[0]), d[7]);
  # COM
  sub(/s1/, ev[1], d[0]);
  sub(/s2/, ev[2], d[0]);
  sub(/s3/, ev[3], d[0]);
  sub(/s4/, ev[4], d[0]);
  sub(/s5/, ev[5], d[2]);
```

```
  sub(/cccc/, sprintf("%04d",Sc[1]), d[2]);
}
function cnvHandStr(hand,regexp, dispSw,revSw,dTbl,ln1,ln2,  s0,s1)
{                                                    #0....
  if(unpackUpper(dispSw)==hand && unpackLower(dispSw)!=0) s0=s1="     ";
  else{
    s0=mkBitStr(Bit[hand],0);
    s1=mkBitStr(Bit[hand],1);
    if(revSw){ s0=rev(s0);
               s1=rev(s1); }
  }
  sub(regexp, s0, dTbl[ln1]);
  sub(regexp, s1, dTbl[ln2]);
}
function mkBitStr(n,pos,  s,i)
{
  if(pos==0){
    for(i=1; i<=n; ++i) s=s "|";
    return substr(" " s "    ", 1, 5);
  }
  if(pos==1)
    return (n==5)?("- 5 "):(sprintf("( %d )", n));
}
function rev(src,  dst,len)
{
  for(len=length(src); len>0; --len) dst=dst substr(src, len, 1);
  if(substr(dst, 1, 1)==")")  dst="(" substr(dst, 2);
  if(substr(dst, 5, 1)=="(")  dst=substr(dst, 1, 4) ")";
  return dst;
}
function setAllEval(ev  )
{
  delete ev;
  ev[0]=evenOddStr(Bit[1], Bit[3]);
  ev[1]=evenOddStr(Bit[3], Bit[7]);
  ev[2]=evenOddStr(Bit[1], Bit[7]);
  ev[3]=evenOddStr(Bit[3], Bit[9]);
  ev[4]=evenOddStr(Bit[1], Bit[9]);
  ev[5]=evenOddStr(Bit[7], Bit[9]);
}
function evenOdd(b1,b2){ return ((b1+b2)%2); }
function evenOddStr(b1,b2){ return ((evenOdd(b1,b2))?("od"):("ev")); }


##
## 影響判定
##
function canMove(sw){ return canMoveFromTo(unpackUpper(sw),cnvPos[sw],Turn); }
function canMoveFromTo(from,to,who)
```

```
{
  # 初手
  if(Times<=2)
    if(other(who,to)) return -1;
  # 奇数
  if(evenOdd(Bit[from],Bit[to])) return ((Bit[from]<5)?(1):(-1));
  # 偶数
  if((Bit[to]<1) || (Bit[from]>=4)) return -2;
  return 2;
}
function other(who,to,  oth){
  oth=another(who);
  if(to==pNo[oth,0] || to==pNo[oth,1]) return 1;
  return 0;
}
function another(who){ return ((who==0)?(1):(0)); }
function total(who){ return (Bit[pNo[who,0]] + Bit[pNo[who,1]]); }
function finish(who){ return (total(who)==10); }
function noEffectLR(l,r){ return (Bit[l]==4 && Bit[r]==4); }
function noEffect(who){ return noEffectLR(pNo[who,0],pNo[who,1]); }
function gameOver(who)
{
  if(finish(who)) return 3;
  if(finish(another(who))) return 2;
  if(noEffect(0) && (GAMEMODE==0 || noEffect(1))) return 1;
  return 0;
}
function setSkip()
{
  if(isSkip(Turn)){ Turn=another(Turn);  printf(" Can't effect.");  pushEnter(); }
}
function isSkip(who)
{
  return ((noEffect(who) && evalOddEven(another(who))==4)?(1):(0))
}
function evalOddEven(who){ return evalOddEvenLR(pNo[who,0],pNo[who,1]); }
function evalOddEvenLR(l,r,  minOE, maxOE)
{
  minOE=Bit[getMinPos(l,r)]%2;
  maxOE=Bit[getMaxPos(l,r)]%2;
  if(minOE==1 && maxOE==0) return 1; # 奇 < 偶
  if(minOE==0 && maxOE==1) return 2; # 偶 < 奇
  if(minOE==1 && maxOE==1) return 3; # 奇 <= 奇
  if(minOE==0 && maxOE==0) return 4; # 偶 <= 偶
  return 0;  # err
}
function getMinPos(l,r){ return ((Bit[l]<Bit[r])?(l):(r)); }
function getMaxPos(l,r){ return ((getMinPos(l,r)==l)?(r):(l)); }
function change(sw,ev,  from,to)
```

```
{
  from=unpackUpper(sw);
  to=cnvPos[sw];
  countUp(ev,from,to);

  #得点加算
  ++Sc[Turn];
  if(ev==2 && other(Turn,to)) Sc[Turn]+=2;
  if(finish(Turn))
    Sc[Turn]+=50;
  else if(Bit[pNo[Turn,0]]==5 || Bit[pNo[Turn,1]]==5)
    Sc[Turn]+=5;
}
function countUp(ev,from,to)
{
  # 奇数
  if(ev==1){
    if(Bit[from]<5) ++Bit[from];
  }
  # 偶数
  if(ev==2){
    if(Bit[to]>=1 && Bit[from]<4){
      --Bit[to];
      Bit[from]+=2;
    }
  }
}
function cnvMoveNo(from,to,  dir,n)
{
  n=from*10;
  for(dir=0; dir<=2; ++dir)
    if(cnvPos[n+dir]==to) return n+dir;
  return -1;  # err
}
function find(bit,who,  dir)
{
  for(dir=0; dir<=1; ++dir)
    if(Bit[pNo[who,dir]]==bit) return pNo[who,dir];
  return -1;  # not found
}
function findOEPos(dir,tgtOE,who,  list,p,st,iim)
{
  delete list;
  list[0]=getMinPos(pNo[who,0],pNo[who,1]);
  list[1]=findOtherPos(list[0],pNo[who,0],pNo[who,1]);
  st=(dir==1)?(0):(1);
  lim=(dir==1)?(2):(-1);
  for(p=st; p!=lim; p+=dir)
    if(Bit[list[p]]%2==tgtOE) return list[p];
```

```
  return -1;  # not found
}
function findOtherPos(tgt,l,r)
{
  if(tgt==l) return r;
  if(tgt==r) return l;
  return -1;  # not found
}
function move(from,to){ return moveMain(cnvMoveNo(from,to),from,to); }
function moveBySW(sw){ return moveMain(sw,unpackUpper(sw),cnvPos[sw]); }
function moveMain(sw,from,to)
{
  dispHands(sw);
  printf("[#%02d] ... %s to %s.", Times, dirName(from), dirName(to));
  pushEnter();
  return sw;
}
function dirName(pos)
{
  return sprintf("%s %s",
    ((pos>=7)?("My"):("Your")), ((pos==1 || pos==9)?("Left"):("Right")));
}


##
##  人間main
##
function man(  sw)
{                                          #0...
  do{ sw=which(); dispHands(sw); printf("    OK?[Y/N]> "); }while(yorn()!="Y");
  return sw;
}
function which(  sw,tmp)
{
  printf("[#%02d] Which", Times);
  for(;;){
    printf("?> ");  tmp="";  getline tmp;
    sw=mv[toupper(tmp)];  if(sw+0==0) sw=mv[tmp+0];
    if(GAMEMODE==0){
      if(sw==10 || sw==30) return sw;
    }else{
      if((10<=sw && sw<=12) || (30<=sw && sw<=32)) return sw;
    }
  }
  return 0;  # err
}


##
```

```
##  COM main
##
function com()
{
  if(CPULEVEL==0) return beginner();
  return expert();
}
function beginner(  ev,minP,maxP,yEv,yMin,yMax,me,you)
{
  me=1; you=0;
  ev=evalOddEven(me);
  minP=getMinPos(pNo[me,0],pNo[me,1]);
  maxP=findOtherPos(minP,pNo[me,0],pNo[me,1]);
  yEv=evalOddEven(you);
  yMin=getMinPos(pNo[you,0],pNo[you,1]);
  yMax=findOtherPos(yMin,pNo[you,0],pNo[you,1]);

  if(yEv!=4)
    if((Bit[minP]==3 && Bit[maxP]==5) || (Bit[minP]==4 && Bit[maxP]==4))
      return move(minP, findOEPos(1,1,you));
  if((Bit[minP]==3 && (Bit[maxP]==4 || Bit[maxP]==3)) && (Bit[yMin]==3 && Bit[yMax]==5))
    return move(minP,yMin);
  if(Bit[minP]==2 && Bit[maxP]==5){
    if(yEv!=3 && Bit[findOEPos(1,0,you)]>0)
      return move(minP, findOEPos(1,0,you));
    return move(minP,maxP);
  }
  if(Bit[minP]==2 && Bit[maxP]==4 && Bit[yMin]==2 && Bit[yMax]==5)
    return move(minP,maxP);
  if(Bit[minP]==3 && Bit[maxP]==3 && Bit[yMin]==3 && Bit[yMax]==3)
    return move(minP,yMin)
  if(Bit[minP]==1 && Bit[maxP]==1 && Bit[yMin]==1 && Bit[yMax]==1)
    return move(minP,yMin);

  if(Times>2 && ev!=4 && yEv!=4){
    if(Bit[minP]%2)
      return move(minP,findOEPos(1,1,you));
    if(Bit[maxP]<5 && Bit[maxP]%2)
      return move(maxP,findOEPos(1,1,you));
  }

  if(Bit[yMin]==4 && Bit[yMax]==5)
    if((Bit[minP]==0 || Bit[minP]==2) && (Bit[maxP]==0 || Bit[maxP]==2 || Bit[maxP]==4))
      return move(minP,yMin);

  if(Bit[yMin]==4 && Bit[yMax]==5){
    from=find(3,1);  if(from>0) return move(from,yMax);
    from=find(1,1);  if(from>0) return move(from,yMax);
  }
```

```
  if(ev==1 || ev==2){
    from=findOEPos(1,0,me);
    to=findOtherPos(from,pNo[me,0],pNo[me,1]);
  }else{
    from=minP;
    to=maxP;
  }
  return move(from,to);
}
function expert(  mv,alpha,beta,cnt,v,p,lev)
{           #0....5
  printf("¥n      ... thinking");  srand(); st=srand();
  mv=0; alpha=-999; beta=999;
  cnt=((Times>SEARCHSTART)?(setApproach(1, 0, (lev=99))):(0));
  if(cnt>0) backUp(lev);
  for(p=0; p<cnt; ++p){
    tryMove(approach[lev, p]);
    v=-search(0, lev-1, -beta, -alpha);
    restore(lev);
    if(v>alpha){ alpha=v; mv=unpackUpper(approach[lev, p]); }
    if(alpha>=beta) break;
  }
  srand(); ed=srand(); printf(" (%dsec)¥n", ed-st);
  if(mv) return moveBySW(mv);
  return beginner();
}
function search(who,lev,alpha,beta,  cnt,v,p,oth)
{
  if((v=gameOver(who))>0) return phaseEvaluate(v,lev);
  oth=another(who);
  cnt=setApproach(who, oth, lev);
  if(cnt==0) return search(oth, lev, alpha, beta);
  backUp(lev);
  for(p=0; p<cnt; ++p){
    tryMove(approach[lev, p]);
    v=-search(oth, lev-1, -beta, -alpha);
    restore(lev);
    if(v>alpha) alpha=v;
    if(alpha>=beta) return alpha;
  }
  return alpha;
}
function phaseEvaluate(v,lev)
{
  if(v==3) return 500+(99-lev); # 手番 who の勝ち
  if(v==2) return -500-lev;     # 手番 another(who)の勝ち = 手番 who の負け
  return 0;  # 双方手詰まり = 引分
}
```

```
function tryMove(tgt,  sw)
{
  sw=unpackUpper(tgt);
  countUp(unpackLower(tgt), unpackUpper(sw), cnvPos[sw]);
}

# bkBit[レベル, 1～9]    (レベルはスタック代り)
function backUp(lev)
{
  bkBit[lev,7]=Bit[7];   bkBit[lev,9]=Bit[9];
  bkBit[lev,1]=Bit[1];   bkBit[lev,3]=Bit[3];
}
function restore(lev)
{
  Bit[7]=bkBit[lev,7];   Bit[9]=bkBit[lev,9];
  Bit[1]=bkBit[lev,1];   Bit[3]=bkBit[lev,3];
}

# approach[レベル, 添字] (レベルはスタック代り)
function setApproach(who,oth,lev,  cnt,sw,p,d,lr,dLast,hLast,ev)
{
  cnt=0;
  dLast=((Bit[pNo[oth,0]]==Bit[pNo[oth,1]])?(1):(2));
  hLast=((Bit[pNo[who,0]]==Bit[pNo[who,1]])?(0):(1));
  for(d=0; d<=dLast; ++d)
    for(lr=0; lr<=hLast; ++lr){
      sw=pack(pNo[who,lr], d);
      ev=canMoveFromTo(pNo[who,lr],cnvPos[sw]);
      if(ev>0) approach[lev, cnt++]=pack(sw,ev);
    }
  return cnt;
}
function pack(upper, lower){ return (upper*10+lower); }
function unpackUpper(src){ return int(src/10); }
function unpackLower(src){ return (src%10); }
```

(投稿: 2011年7月5日)

Python の文法

TSNET スクリプト通信版　草稿　第５回

6-2-2-2-3-2 専用の順序列型

収容内容が決まっている順序列型のコレクションオブジェクトには、文字列を扱う'str'クラスと、バイト列を扱う'bytes'と'bytearray'があります。
歴史的に見れば、以前はPython ではバイト列は文字列で扱っていました。
しかしPython2 でUnicode 文字を扱うUnicode 文字列型が登場し、一時期は「ASCII やバイト列は旧文字列型、Unicode はUnicode 文字列」という扱いで区分していました。
しかし、データ構造を基本とした分類は不自然だということで、現在は「文字列」を扱うクラスと「バイト列」を扱うクラスという形に分類されたというわけです。
Python3 からPython を始めた初心者の方は、この分類で覚えて頂けば良いと思いますが、旧Python を愛用されていた方は特にこの部分については慎重にご確認ください。

6-2-2-2-3-2-1 文字列(str)

■クラス名：'str'

Python では、文字列を'str'クラスのインスタンスとして扱います。
'str'型オブジェクト（以後「文字列」）は、「文字」を専用に扱う「不変型」順列型コレクションクラスです。
コレクションとしての「文字列」の特徴は、不変であることとと「１文字でも文字列」であることです。
文字を１文字取り出したりすることや、１文字文字列だけを扱う関数('ord'関数等)もありますが、「文字」は必ず「文字列」になります。

```
>>> T
(10, 20, 30) #【例えばタプルで……】
>>> T[1]
20           #【要素は整数】
>>> S
'あいう'      #【しかし文字列は……】
>>> S[1]
'い'          #【要素も文字列】
```

文字列オブジェクトでは、不変の汎用順序列オブジェクトである'tuple'クラスのメソッドや演算子やスライス等は漏れ無く使用できます。

```
>>> S = "あいうえお abcde"
>>> S + "xxx"
'あいうえお abcdexxx'
>>> S * 2
'あいうえお abcde あいうえお abcde'
>>> S[2:8]
'うえお abc'
```

「文字列」は不変型のコレクションなので、「文字列を直接操作」して別の文字列に変えることはできません。

同じような効果を得るためには、以下のどちらかの方法を取ります。
１．文字列をリストに変換してそのリストを操作し、'join'メソッドでつなぎ直し、その文字列を再代入する
２．文字列の変換メソッドで得られた文字列を再代入する
たとえば変数Ｓ内の"abcde"を"abXde"に変換するなら、こんな感じです。

```
>>> S = "abcde"
>>> SL = list(S)           #【リストに変換】
>>> SL
['a', 'b', 'c', 'd', 'e']
>>> SL[2] = 'X'
>>> SL
['a', 'b', 'X', 'd', 'e']
>>> S = ''.join(SL)        #【文字列に連結】
>>> S
'abXde'
>>> S = "abcde"
>>> S = S.replace('c','X')       #【入れ替えメソッド】
>>> S
'abXde'
```

文字列オブジェクトでは、文字を加工して新しい文字列を返すメソッドがいくつかあります。

※ただし、Pythonは「文法として」正規表現(regular expression)をサポートしていません。正規表現をPythonで使用する場合は、付属の're'モジュールをインポートして使用してください

【補足】
Python2以前の「文字列」は「１文字」を「１バイト」という記憶容量単位で区切っていましたが、Python3の「文字列」は、文字通り「１文字が単位」です。Python2以前で文字列型をバイトコード操作に使用していた方は、Python3では後述の'bytes'型を使用してください
また、フォーマット演算子である'%'も、執筆時点の最新である'Python 3.2.1'ではまだ使用できるようですが、将来的に廃止されることが予定されているので、より汎用性に富んだ'format'メソッドを使用するように勧告が出されています。従ってここでは、フォーマット演算子についての説明は割愛し、新しい形式である「メソッドを用いたフォーマット」のみを説明することにします。

6-2-2-2-3-2-1-1 フォーマット用メソッド

ここでは、フォーマットに関するメソッドを扱います。
フォーマットとは「書式、及び書式に従って文章を組み立てること」を指します。
以前、'%'演算子で行っていた操作をメソッドにしたものですが、その表現力はより強力に、また明確になっています。
ただし強力すぎて、全部を一度に覚えようとすると大変なので、ここではメソッドの紹介と、「そもそもフォーマットって何するの？」ということを中心に説明したいと思います。
実際の書式の書き方については、次に続く「基礎編」と「詳細編」にわけて説明したいと思います。

■フォーマット用メソッド

書式：S0.format(arg1[, arg2, ..]) ⇒ S1
動作："S0"に書かれた書式に従って、"arg1"などの引数を処理した文字列"S1"を返す。キーワード引数も使用可能

書式：S0.format_map(mapping) ⇒ S1
動作：'format'の引数を辞書オブジェクトでまとめたものを受け取るバージョン(ver.3.2より)

●そもそもフォーマットって何するの？

※この項目は、文字列のフォーマットを行ったことのない初心者の方向けの説明ですので、不要な方は次の「文字列フォーマット構文　基礎編」にお進み下さい。

さて、文字列のフォーマットですが、こんな例を考えてください。

☆プロフィール
氏名　（　　）
年齢　（　　）
住所　（　　）
電話　（　　）

上の例で、実際に（ ）の中は様々でしょうが、「氏名」だの「年齢」だのといった項目の文字は変化しません。

多分、印刷して各人に配り、空所を埋めてもらって使うのでしょう。

この「変化しない部分」と「空所に何を書くのかの指定」のことを「書式」と言います。

'format'メソッドは、「書式」に当たる文字列に「内容」を与えて文章を完成させるメソッドです。

このとき、「内容」が直接指定される場合と、与えられた「データ」を加工して「内容」を作り出す場合とがあります。

これに続く「基礎編」では、直接指定する方法を説明します。

これだけでも十分便利ですので、まずはこちらを使ってみて下さい。

続く「詳細編」では加工の方法……桁数を揃えて整形したり、データを整数化したりする方法などを説明します。

さて、これだけではおもしろくないので、一つ上の例を試してみましょう。

```
>>> S="""☆プロフィール
... 氏名　({0})
... 年齢　({1})
... 住所　({2})
... 電話　({3}) """
>>> sk = S.format('機械伯爵','忘れた','時間城','無いよ')
>>> print(sk)
☆プロフィール
氏名　(機械伯爵)
年齢　(忘れた)
住所　(時間城)
電話　(無いよ)
```

6-2-2-2-3-2-1-1-1 文字列フォーマット構文　基礎編

'format'メソッドは「書式」である文字列の'{ }'(ブレース)で囲まれた部分を引数に指定された部分と入れ替えます。

'{ }'の中には「数字」か「ラベル」が入ります。

「数字」の場合は、最初の引数が'{0}'になり、順に'{1}','{2}'..と続いていきます。

「ラベル」の場合は、キーワード引数で指定します。

「ラベル」となる文字の文字列の条件は、変数のラベルと同じです。

まずは、例を見てみましょう。

```
>>> "{0}-{1}-{2}".format('a','b','c')
'a-b-c'
>>> "{0}-{1}-{0}".format('a','b','c')
```

```
'a-b-a'
>>> "{one}-{two}-{three}".format(three='a',two='b',one='c')
'c-b-a'
>>> "{one}-{one}-{two}".format(three='a',two='b',one='c')
'c-c-b'
```

引数やキーワード引数の順番を変えてもかまいませんし、２回以上引用する引数があっても使用しない引数があっても一向に構いません。

```
>>> "{key}-{0}-{1}".format('a','b', key='c')
'c-a-b'
```

位置引数とキーワード引数を混合する場合、関数の引数に於いてキーワード引数は通常の引数の後、というルールは有効ですが、書式に適用する順序は当然関係ありません。
'format_map'メソッドの使い方も、多分ご想像の通りだと思います。

```
>>> d = {'a':'AAA', 'b':'BBB', 'c':'CCC'}
>>> "{c}-{b}-{a}".format(**d)
'CCC-BBB-AAA'
>>> "{c}-{b}-{a}".format_map(d)
'CCC-BBB-AAA'
```

※辞書オブジェクトについては後述の「写像型オブジェクト」の項で詳しく述べます。

さて、引数は文字列表現を持つ（持たない方が珍しい）全てのオブジェクトを取ることができます。
数字などはそのままですが、たとえば'str'のようなクラスオブジェクトであってもへっちゃらです。

```
>>> '{0}-{1}-{2}'.format(3.14,(lambda x:x), str)
"3.14-<function <lambda> at 0x00C71390>-<class 'str'>"
```

カスタムオブジェクトで'__str___'メソッドか'__repr__'メソッドが実装されている場合は、指定された文字表現で文字列の中に現れます。

```
>>> class X:
...   def __str__(self):return 'xxx'
...
>>> x = X()
>>> x
<__main__.X object at 0x00C72050>
>>> print(x)
xxx
>>> "{0}-{1}-{2}".format(x,x,x)
'xxx-xxx-xxx'
```

「書式」の書き換えはかなり柔軟であり、例えばコレクション型の引数の一つをターゲットにすることも可能です。

```
>>> "{0}".format("abc")
'abc'
>>> "{0[1]}".format("abc")       #【アイテム参照】
'b'
>>> "{0[a]}".format({'a':100,'b':200})
'100'
>>> class C:pass
...
```

```
>>> o = C()
>>> o.x = 'xxx'
>>> "{0.x}".format(o)              #【属性参照も…】
'xxx'
```

なおブレースを２連続で書くと、エスケープ（フォーマットのルールから除外）されます。

```
>>> "{{,{0},}}".format(100)
'{,100,}'
>>> "{{{0}}}".format(100)
'{100}'
```

これを利用すると、書式を作る書式（メタフォーマット）が作れたりします。

```
>>> "{{0[{0}]}}".format(1).format('abcde')
'b'
```

さて、'{}'のように、中を指定しない場合は、次のようになります。
・'{}'が一つならば、第一引数の値が入ります
・'{}'が引数と同じ数あった場合は、引数の順番に入ります。

```
>>> "{}".format(100,200,300)
'100'
>>> "{}-{}-{}".format(100,200,300)
'100-200-300'
```

6-2-2-2-3-2-1-1-2 文字列フォーマット構文　詳細編

「基礎編」では、引数をそのまま適用しました。
「詳細編」では、引数を加工して適用する方法を説明します。
まず、置き換えられる箇所（置換フィールド）の書式を以下に書きます。

１．{フィールド名}
２．{フィールド名!変換}
３．{フィールド名:書式設定}
４．{フィールド名!変換:書式設定}

「フィールド名」は、位置引数の位置（整数）か、キーワード引数のキーワードのどちらかです。
「変換」は'r'と's'と'a'があり、それぞれ'repr'関数（リダイレクトの表現文字列を返す関数）によって返される文字列、'str'関数（print 関数などによって出力される文字表現を返す関数）によって返される文字列、そして'ascii'関数（全てを ASCII 文字表示に変換する関数）によって返される文字列ですが、これが問題になる場合はほとんどありません。
基本範囲を若干逸脱しますが、分数を扱う'fractions'モジュールの'Fraction'型のオブジェクトの例を見てみましょう。

```
>>> import fractions
>>> f = fractions.Fraction(7,2)
>>> f
Fraction(7, 2)
>>> print(f)
7/2
>>> "{0!r}".format(f)
'Fraction(7, 2)'
>>> "{0!s}".format(f)
```

```
'7/2'
```

'a'変換の例は、日本語を使用すれば若干解り易いかもしれません。

```
>>> j = "あいうえお"
>>> j
'あいうえお'
>>> print(j)
あいうえお
>>> "{0!r}".format(j)
"'あいうえお'"
>>> "{0!s}".format(j)
'あいうえお'
>>> "{0!a}".format(j)
"'\\u3042\\u3044\\u3046\\u3048\\u304a'"
```

●書式設定

主に数値の表示に関係する「書式設定」は、以下の順番で設定します。

{「フィールド名」:「(詰め物)位置」「符号」「記数法表示」「数字前のゼロ」「表示幅」「3桁区切りコンマ」「精度」「数値型」}

それでは、書式設定の説明をします。
まず「表示幅」ですが、これは0以外の正の整数(自然数)で、「最低の」表示幅を表します。つまり数値が何桁であろうとも、そのスペースを確保する、ということです。
例をあげてみましょう。

```
>>> "|{0}|".format(123.45)
'|123.45|'
>>> "-{0:10}-".format(123.45)
'|    123.45|'
```

このスペースに関わってくるのが、「フィールド名」のすぐ横の「(詰め物)位置」です。
位置は左揃え('<')、右揃え('>')、中央揃え('^')、そして「符号と数字の間を開ける」('=')の4種類が指定できます。

```
>>> "|{0:10}|".format(-123.45)
'|   -123.45|'
>>> "|{0:<10}|".format(-123.45)
'|-123.45   |'
>>> "|{0:>10}|".format(-123.45)
'|   -123.45|'
>>> "|{0:^10}|".format(-123.45)
'| -123.45  |'
>>> "|{0:=10}|".format(-123.45)
'|-   123.45|'
```

この位置揃えを指定したとき、開いたスペースをなにかの文字で置き換えることができます。

```
>>> "|{0:a<10}|".format(-123.45)
'|-123.45aaa|'
>>> "|{0:@<10}|".format(-123.45)
'|-123.45@@@|'
>>> "|{0:<^10}|".format(-123.45)
```

```
'|<-123.45<<|'
```

※最後の例のように、位置指定の記号も使用できますが、紛らわしいのでおすすめできません。
位置のデフォルトは右揃えですが、右揃えでも詰め物を指定する場合は'>'の指定が必要になります。

「符号」では、'+','-',' '(空白)が指定できます。
'+'を指定すると、必ずプラスかマイナスが表示されます。
'-'を指定すると、負の数の時だけマイナス記号がつきます（デフォルト）
' 'を指定すると、正の数の時には記号ではなく「記号のスペース」が表示されます。

```
>>> "|{0}|".format(100)
'|100|'
>>> "|{0:+}|".format(100)
'|+100|'
>>> "|{0:-}|".format(100)
'|100|'
>>> "|{0: }|".format(100)
'| 100|'
```

「数字の前のゼロ」は、文字のかわりに'0'で詰め物をすると考えて下さい。

```
>>> "|{0:10}|".format(100)
'|       100|'
>>> "|{0:010}|".format(100)
'|0000000100|'
>>> "|{0:^010}|".format(100)
'|0001000000|'
>>> "|{0:<010}|".format(100)
'|1000000000|'
```

よって下の２つの例のように、位置と一緒に使うとなんだか妙な感じになるので、注意しましょう（意識的に使うなら別ですが）

「３桁区切りコンマ」は、''文字通り大きな桁数の数字を表示するときの３桁毎のコンマを表示します。

```
>>> "|{0}|".format(1000000)
'|1000000|'
>>> "|{0:,}|".format(1000000)
'|1,000,000|'
```

「精度」は数値の表示桁数の指定を意味します。
「精度」を指定する場合は、「浮動小数点数('float')」を使用して下さい（整数を含めて「精度」を設定したい場合には、「数字型」の'f'及び'F'オプションを使用して下さい）
「精度」は'.'の後に整数で表示します。

```
>>> "{0}".format(123.45)
'123.45'
>>> "{0:.4}".format(123.45)
'123.5'
>>> "{0:.4}".format(123.44)
'123.4'
>>> "{0:.3}".format(123.44)
'1.23e+02'
```

数字は切り捨てではなく四捨五入で表示されます。
最後は、小数点部分が無いため、科学計算の概数表示の書式に変換されています。
なお、数値以外を指定した場合は、「精度」は最大の表示スペースを意味します。

```
>>> "{0:.3}".format("abcde")
'abc'
```

「数値型」は、数値を変換して様々な表示型で表す書式です。
このうち'b','d','o','x','X'は引数が整数の場合のみ有効です。
'd'は普通の十進法表示です（ほとんど使用する必要はありません）
'b'は整数を二進法表記します。
'o'は整数を八進法表記します。
'x'と'X'は、整数を十六進法表記します。
二進法、八進法、十六進法の場合は「記数法表記」の位置に'#'を入れると、'0x'などの表記が付きます（'X'はアルファベット部分が大文字になります）

```
>>> "{0}".format(200)
'200'
>>> "{0:b}".format(200)
'11001000'
>>> "{0:#b}".format(200)
'0b11001000'
>>> "{0:o}".format(200)
'310'
>>> "{0:#o}".format(200)
'0o310'
>>> "{0:x}".format(200)
'c8'
>>> "{0:#x}".format(200)
'0xc8'
>>> "{0:X}".format(200)
'C8'
>>> "{0:#X}".format(200)
'0XC8'
```

'e'と'E'は、強制的に科学計算用概数表示にします（'e'と'E'の違いは位表示の大文字／小文字表示の違い）
'f'と'F'は、固定小数点数表示になります（「精度」で指定するか、マシンのデフォルトに従います）

'e','E','f','F'で「精度」を指定した場合、有効数字の桁数ではなく「小数点以下の桁数の精度」の意味になりますので、十分注意して

'g'と'G'は、科学計算表示と固定小数表示とを桁数で分けます。
つまり、桁数が多いと科学計算表示、少なければ固定小数点表示という普通の表示方法を採用します（'G'は、科学計算表示で'E'を採用）
'g'がデフォルトです。
なお、「精度」は「有効数字の桁数」を指します。

'n'は'g'とほぼ同じです（環境設定を変えてない限りは）

```
>>> "{0:.4}".format(123.56)
'123.6'
>>> "{0:.4f}".format(123.56)
```

```
'123.5600'
>>> "{0:.4F}".format(123.56)
'123.5600'
>>> "{0:.4e}".format(123.56)
'1.2356e+02'
>>> "{0:.4E}".format(123.56)
'1.2356E+02'
>>> "{0:.4g}".format(123.56)
'123.6'
>>> "{0:.4G}".format(123.56)
'123.6'
```

″%″は文字通り百分率で表します（「精度」は小数点以下の桁数を指します）

```
>>> "{0:.2%}".format(0.5)
'50.00%'
```

●さらに発展編

「書式設定」は、引数で指定することも可能です。

```
>>> "{0:{1}}".format(100,"#b")
'0b1100100'
>>> "{0:{1[1]}}".format(100,("#b","#o"))
'0o144'
```

オブジェクトの中には、特別な書式が使用できるものもあります。
下の例は、日付や時刻などを扱うオブジェクトの特別な書式設定の適用例です。

```
>>> import datetime
>>> d = datetime.datetime(2011,7,31)
>>> "{:%Y/%m/%d}".format(d)
```

’2011/07/31’

（投稿：2011 年 7 月 31 日）

よしおさんとロボ太

海鳥

「温・特技」

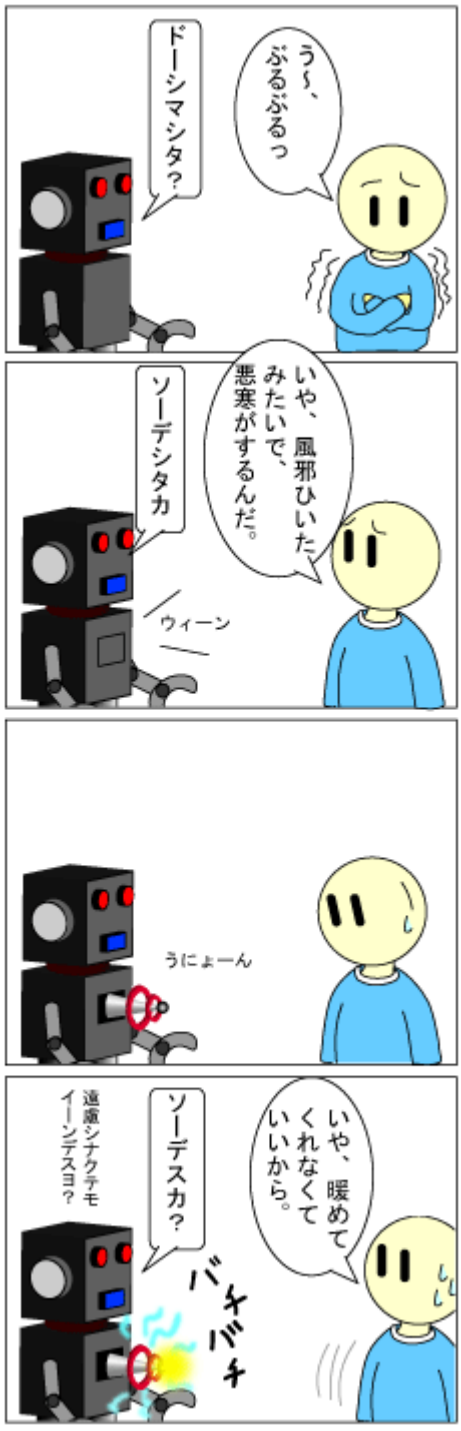
いや、便利だろうけれども。

（転載：2011 年 7 月 30 日）

よしおさんとロボ太

海鳥

「暖・特技」



心も体もすぐポカポカ！（たぶん80度くらいに）

（転載：2011 年 7 月 30 日）

エアチェック・ポッドキャスティング

- エアチェックを iTunes の Podcast に載せる -

jscripter

1. iTunes の登場と Podcasting

Apple I(Apple Computer 創立、1976 年)から数えると「パーソナル・コンピューティングの 35 年」という本が書けるかもしれないが、今や、Post-PC という言葉をスティーブ・ジョブズ自身が唱える時代に突入している。既に Apple の社名から Computer という言葉は 2007 年に消えているのだ。アップルは Post-PC 時代、すなわち PC(Mac)はアップルの一つの要素に過ぎなくなるという既定路線を突っ走っているかに見える。

歴史的に見れば、マイクロソフトの MS-DOS/Windows と IBM の PC が 1990 年代にコンピューティングの大衆化をもたらしたのは間違いなく、1980 年代の前半に PC に GUI を取り込むという先駆的な役割を果したアップルもスティーブ・ジョブズが離れてからは脇役にすぎなかった。

ジョブズが戻り、21 世紀に入って、アップルは iPod という音楽プレーヤーと iTunes というアプリケーションを生み出した。そして、2003 年に iTunes Music Store で音楽のオンライン販売を始め、iTunes は Windows をサポートした。ここからアップルの快進撃が始まる。2007 年には iPhone、2010 年には iPad を登場させ、驚異の発展を遂げつつある。

アップルは Mac という PC のメーカーであるだけでなく、音楽メディアプレーヤー・メーカーであり、音楽やビデオのオンライン販売業者であり、iPhone というケータイ・メーカーであり、iPhone は単なるケータイではなく、デジカメであり、FaceTime というビデオ通話マシンであり、Web ブラウザでもあり、iPod そのものでもあり、ビデオ再生マシンでもあり、ゲームマシンであり、電子書籍リーダーであり、Apple TV との AirPlay マシンなどでもある。また、iPad という Post-PC メーカーでもある。

iPhone などは、音楽は iTunes Store からダウンロードし、アプリは AppStore から簡単にダウンロードしてインストールできるだけでなく、PC/Mac の iTunes とコンテンツを共有してバックアップなどができる。自在なメディアプレーヤー、コミュニケーション・ツールとして売れないはずはないというものである。その中でも著者が購入している iPod touch(第 3 世代)は、ケータイ機能はないが、32GB メモリをもって、29,800 円という低価格マシンである。

このように、アップルの製品の中核に iTunes があるといっても過言ではない。日本では 2004 年に「iPod & iTunes」(オライリー・ジャパン)が、2005 年には「iPod & iTunes HACKS」(オライリー・ジャパン)が発売される。著者も 2005 年に iPod mini を購入して、晴れてアップルユーザーとなった。

実は、2005 年には「PODCASTING HACKS」(オライリー・ジャパン)という書籍が発売された。この中に、iTunes への Podcast のインポートという項目がある。ジョブズの WWDC 2005 のキーノートの話題の一つが、iTunes の Podcasting 対応であったのである。Podcast は当初は楽曲そのものの配信に使われる可能性もあったと思われるが、著作権の問題のために、音楽サイトのプロモーション、インタビュー、ニュース・トークやラジオ番組の配信などに使われることになった。

2. ラジオエアチェックサーバー

一方、デスクトップには、創刊二周年記念号(2.1 号、May 2010)に書いたように USB 接続の FM/AM ラジオでエアチェックした録音ファイルが存在している。「日曜工作的プログラミング - ラジオサー

バー」の記事には、デスクトップサーバー(ローカルサーバー)上で録音ファイルをリストアップしてそれぞれ再生できるようにする CGI を作成した。

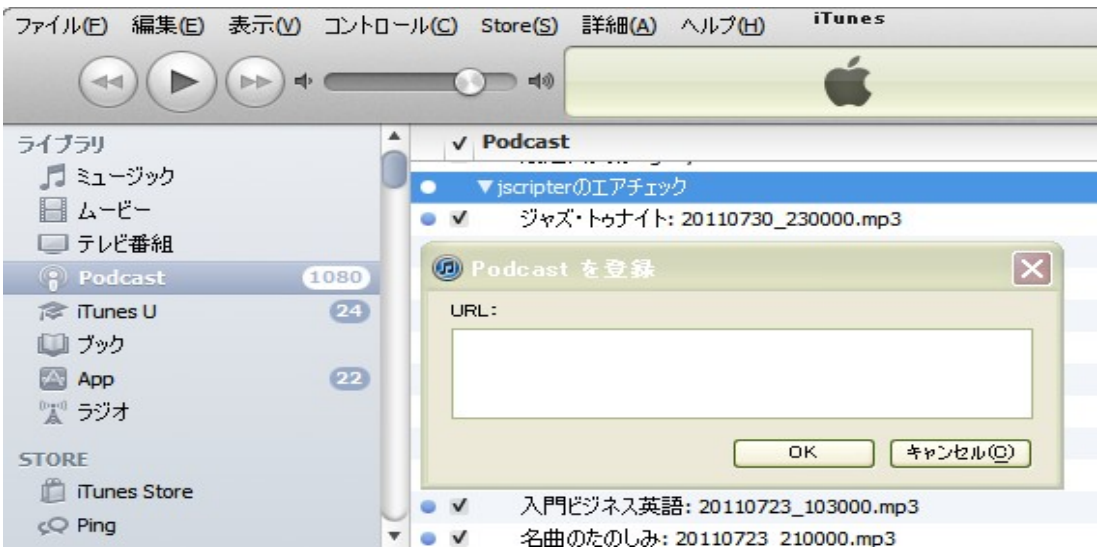
本稿は、前稿の radiorecm.pl の CGI 出力を iTunes の Podcast 用タグを持つ RSS 2.0 の出力に変更して、iTunes に読み込ませてみようという初期的な実験報告である。サーバー設定など実行環境については、「日曜工作的プログラミング - ラジオサーバー」の記事をご参照いただきたい。

3. iTunes の Podcast

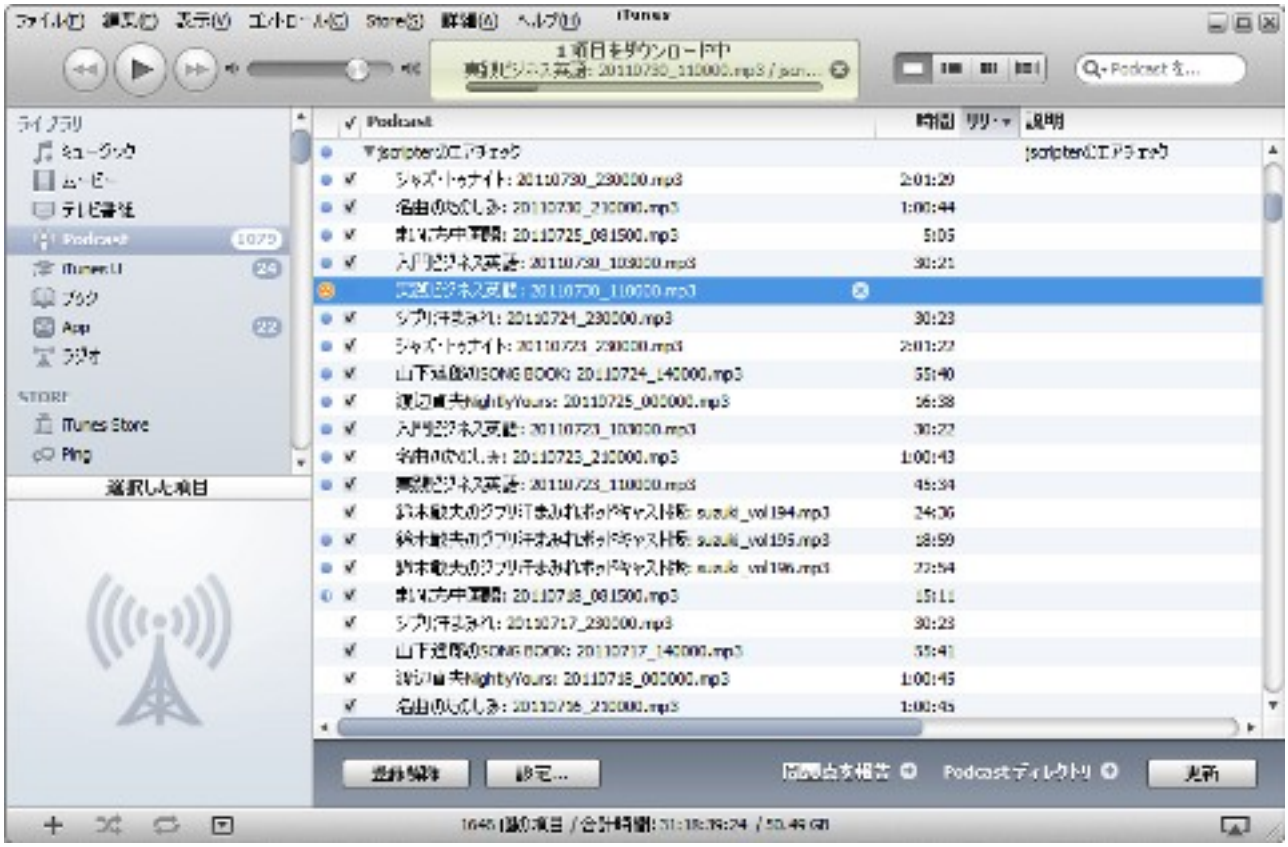
Apple は Podcast に早くから注目し、iTunes 用のタグを策定して、iTunes Store から「Podcast を送信する」ことができるような仕組みを作っている。しかしながら、エアチェックによる録音は私的利用であり、自作のコンテンツを配信するのでない限り、この仕組を利用する必要はない。たとえ利用したとしても Web に公開されていないローカルの URL では動作しないはずだ。

iTunes は Podcast のテスト用に、メニューバーの「詳細」に「Podcast を登録」という項目を持っている。次のような具合である。

この項目をクリックすると URL を登録するポップアップが出る。



ここで、次項に示す iTunes の podcast 用タグを備えた RSS 2.0 を出力する CGI の URL を入力して「OK」ボタンを押す。通常、最初の一件だけがダウンロードされて読み込まれる。他のものは「入手」ボタンを押せばダウンロードされる。ダウンロード画面の一例を次に掲載しておく。



4. radiorecpodcast.pl

スクリプトは単なるテスト用のものなので次のように短い。今のところ、RSS には録音ファイルの URL とタイトル名を記載しているだけである。曲の長さは itunes:duration に記載すべきなのだろうが、自動的に iTunes が読み取って表示するようだ。詳細はスクリプトのコメントを参照ください。

Podcast 用の RSS 2.0 は下記のスクリプトから読み取ることも容易だが、アップルが「Podcast を作成する」というページを作成しているので、そちらを参照するのが理解への早道である。URL を紹介しておく。元々 Podcast を自作して公開できるようにするためのもので、詳細に丁寧に書かれている。

アップル - iTunes - Podcast - Podcast を作成する
http://www.apple.com/jp/itunes/podcasts/specs.html

```
#!/Perl5.10/bin/perl.exe
use URI::Escape;
use Encode;
```

```
# デスクトップサーバーの IP アドレス・ポート番号の設定
my $server = "xxx.xxx.xxx.xxx:xxxx";# IP adress and port number of your server
# USB ラジオの録音ファイル格納ディレクトリの設定
my $radiorecdir = "E:/RadioREC";# Radio Recording Directory

# ディレクトリ名(mp3 の拡張子を持たないディレクトリ要素)の取得
opendir(DIR, $radiorecdir);
my @rdirs = grep {!/(^[.][.]?¥z|¥.mp3)/i} readdir DIR;
closedir(DIR);

# CGI 出力: RSS 2.0 の出力の最初の部分
# iTunes 用の RSS タグを追加する設定とチャネル設定部分
# ユーザー毎にチャネルの内容などは変更する必要がある。(今後、改造する予定)
print <<HEADER;
Content-type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>
<rss xmlns:itunes="http://www.itunes.com/dtds/podcast-1.0.dtd" version="2.0">

<channel>
	<title>jscripter のエアチェック</title>
	<link>http://$server/cgi-bin/radiorecpodcast.pl</link>
	<description>jscripter のエアチェック</description>
	<language>ja</language>
	<copyright>Copyright(c) 2011 jscripter</copyright>
	<image>
		<url></url>
		<title>jscripter のエアチェック</title>
		<link></link>
	</image>
	<pubDate></pubDate>
	<itunes:subtitle></itunes:subtitle>
	<itunes:author>jscripter</itunes:author>
	<itunes:summary></itunes:summary>
	<itunes:owner>
		<itunes:name>jscripter</itunes:name>
		<itunes:email>jscripter9@gmail.com</itunes:email>
	</itunes:owner>
	<itunes:image href="" />
	<itunes:category text="Music">
		<itunes:category text="Jazz"/>
		<itunes:category text="Classics"/>
		<itunes:category text="Alternatives"/>
	</itunes:category>
	<itunes:category text="Music Media">
		<itunes:category text="FM"/>
		<itunes:category text="AM"/>
		<itunes:category text="Podcast"/>
	</itunes:category>
```

```
HEADER

# USB ラジオの録音ファイルは予約名で設定されたディレクトリに格納される。
# ディレクトリ名は cp932(SJIS)の文字コードで読み取り、utf8 にエンコードし、
# さらに URI エスケープすることによって、ディレクトリを URL のパスとして
# アクセスできるようになる。
# 詳細は TSNET スクリプト通信 2.1 号(May 2010)、79-80 ページを参照のこと。
foreach my $rdir (@rdirs){
	my $encode_rdir = encode("utf8", decode("cp932", $rdir));
	my $esc_rdir = uri_escape($encode_rdir);
	opendir(DIR, "$radiorecdir/$rdir");
	my @rmp3s = grep(/¥.mp3/i, readdir(DIR));
	closedir(DIR);
	foreach my $rmp3 (@rmp3s){
		my $rmp3_url = "http://$server/radio/$esc_rdir/$rmp3";

# CGI 出力: RSS 2.0 の item 要素出力
# Podcast の番組一つ一つの内容設定
	print <<ITEM;
	<item>
	<title>${encode_rdir}: $rmp3</title>
	<link>$rmp3_url</link>
	<description><![CDATA[]]></description>
	<itunes:author></itunes:author>
	<itunes:subtitle></itunes:subtitle>
	<itunes:summary></itunes:summary>
	<enclosure url="$rmp3_url" length="" type="audio/mpeg" />
	<guid>$rmp3_url</guid>
	<pubDate></pubDate>
	<itunes:duration></itunes:duration>
	<itunes:keywords></itunes:keywords>
	</item>
ITEM
	}
}

# CGI 出力: チャネル要素と RSS 要素の終了タグ
print <<FOOTER;

</channel>
</rss>
FOOTER
```

5. おわりに、さらなる今後の構想

radiorecpodcast.pl + iTunes + iPod touch は大変便利で、音楽生活には手放せなくなった。次の

ステップでは、さらに iTunes の RSS タグを利用して、radiorecm.pl 経由で更新日記(著者ブログ)や Web 情報などを結びつけるデータを生成するようにしてみようかと考えている。そのデータを radiorecpodcast.pl で読み取って、RSS 2.0 の出力に埋め込む。デスクトップサーバーの音楽生活へのさらなる拡張となるだろう。

問題は、どの録音ファイルにどのような内容の曲や話が入っているとか、どのような人が出演しているとか、そのような情報を如何にして取得するかという方法にある。こまめに日記に書いておけば、録音ファイルに対応した記事を radiorecm.pl のリストから表示することも可能だろう。ラジオ番組の情報を持つサイトや特定の番組の感想が書かれているサイトが検索などで見つかっていれば、日記などにリンクを記載しておくと便利だろう。

もちろん、将来はセマンティック・ウェブ化が進んで、特定の情報を容易に取得できるようになるかもしれない。あるいはあなたが情報収集のスクリプトを書いても良いわけだ。

(投稿: 2011 年 7 月 31 日)

編集後記

jscripter

さて、いつものことながら、まだ、自分の記事に問題があるのを抱えたままで、巻頭言に続いて編集後記を書くという荒業を続けている。残っていると安心して自分の記事に取り組めない。

「よしおさんとロボ太」全43編が、1.2号から始まって、4.1号で無事に最終回を迎えることができた。ここまで、刊行を継続できたことをまずは素直に喜びたい。37回の「ロボOS」が、TSNETスクリプト通信への掲載をお願いした後に書かれたものである。最初はたくさん在庫(失礼^^;)があるので、惜しまずに投入したのだが、残り少なくなると数を絞らざるを得なくなった。次号に肩の凝らないビジュアル系で何か掲載希望があればありがたい。誌面にゆとりが出る。海鳥さんは海鳥黙示録のサイトを閉鎖されたこともあり、継続は難しそうだ。第10号の表紙写真に、小麦粘土で製作されたロボ太を取り上げさせていただいたのも思い出深い。

次号は10月末刊行予定で進めたい。7,10,1,4月のサイクルになりそう。日本の年度的数え方では少しサイクルがずれたかもしれない。最初は5,8,11,2月のサイクルぐらいのつもりだったが、年4回のサイクルが維持できればと思う。参加者がもう少し増えてくれればと思う。もっと宣伝が必要かもしれない。

(投稿: 2011年7月31日)

ＴＳＮＥＴスクリプト通信

ISSN 1884-2798　出版地: 広島市

2011 年 7 月 31 日　　4.1.001 版

投稿規程

TSNETWiki ： 「投稿規程」のページを参照のこと

編集委員会(投稿順)

Ｙさ　saw[at]s7[dot]wh[dot]qit[dot]ne[dot]jp
機械伯爵　kikwai[at]livedoor[dot]com
海鳥　kaityo256[at]gmail[dot]com
jscripter　jscripter9[at]gmail[dot]com

著作権

使用許諾・配布条件

免責事項

編集ソフトウェア

OpenOffice.org 3.2.1 Writer

発行所

一次配布所: TSNET スクリプト通信刊行リスト
http://text.world.coocan.jp/TSNET/?TSNET%E3%82%B9%E3%82%AF%E3%83%AA%E3%83%97%E3%83%88%E9%80%9A%E4%BF%A1%E5%88%8A%E8%A1%8C%E3%83%AA%E3%82%B9%E3%83%88

TSNETスクリプト通信 第4巻第1号(通算第13号)
発行: TSC編集委員会 発行日: 2011年7月31日
ISSN: 1884-2798 出版地: 広島市 創刊: 2008年5月7日