Jによるシステム処理－１
debug と dec_dump などシステム操作のツール

西川　利男

Jの有用性は数値計算だけに留まらず、システム処理でも十分発揮される。
いろいろな場合にファイルの構造、中身を見たいことがある。いわゆるシステム処理であり、一般にはC言語などを用いて、普通のユーザにはとうてい無理と思われているが、そんなことはない。Jで十分、かえって手軽に行うことが出来る。

かつて、DOSの時代にはdebugというコマンドがあり、ファイルの中身を見て、構造を解析するのに使った。これと同じ機能のツールdebugをJで作った。さらに16進数でのdebugだけでなく、10進数で行えるdec_dumpを作った。

ここでは、簡単なファイルの処理で使用法を示し、別稿としてAPL文字の表示、印刷に利用した。

１．基本のユーティリティ

・ディレクトリ、ファイルの読み込み、書き込み

```
require 'files'
dir =: fdir
fread（ファイル名）
（データ）fwrite（ファイル名）
```

・スクリーンへの表示、キーボードからの入力

```
wr=: 1!:2&2
rd=: 1!:1
```

・10進数と16進数との相互変換

```
dfh=: 16&#. @ ('0123456789ABCDEF'&i.) f.
hfd=: dfh ^:_1 f.
NB. hex  (adverb)
NB. e.g. 'FF' + hex '8'
hex=: &. dfh

val=: a. & i.
chr=: val ^:_1
hdump=: }:@,@(,"1' '"_)@hfd@val f.
```

２．debug と dec_dump

・ファイルの16進数ダンプ

debug（ファイル名）‥‥すべてダンプ
(開始アドレス)(ダンプのバイト長) debug（ファイル名）‥‥一部ダンプ

・ファイルの10進数ダンプ

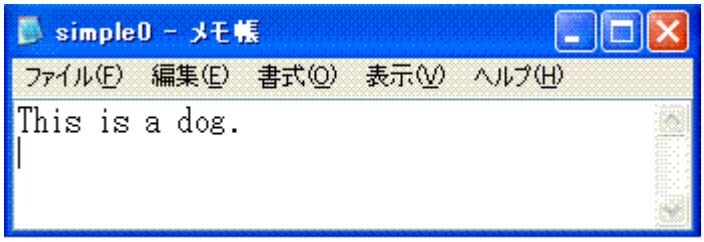
dec_dump（ファイル名）‥‥すべてダンプ
(開始アドレス)(ダンプのバイト長) dec_dump（ファイル名）‥‥一部ダンプ

プログラム定義は最後に挙げてある。

３．debug と dec_dump を使った簡単な例

小さなファイルを作り、ツール debug と dec_dump とを実際に使ってみよう。
まず、16 進数ダンプ debug によりファイル¥APL_test¥simple0.txt を見てみる。つぎのようになる。

```
   debug '¥APL_test¥simple0.txt'
¥APL_test¥simple0.txt
*** Print on NotePad go_on ? (y or q)
y
```




```
（メモ帳を閉じて、さらに進める）
File Length = 16(=10 hex) bytes
Dump Address (hex): from 0000 to 000F
*** Dump go_on ? (y or q)
y
     -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F
000: 54 68 69 73 20 69 73 20 61 20 64 6F 67 2E 0D 0A This is a dog.__
*** end ***
```

10 進数ダンプ dec_dump ではつぎのようになる。

```
   dec_dump '¥APL_test¥simple0.txt'
 |   0   1   2   3   4   5   6   7   8   9
-|----------------------------------------
0|  84 104 105 115  32 105 115  32  97  32 This is a
1| 100 111 103  46  13  10   _   _   _   _ dog.
```

J のツール debug は DOS のコマンド debug と全く同じである。ここでは示していないが、ファイルデータの一部だけのダンプは実際に当たって非常に有用である。さらにツールの中から、メモ帳を開いて確かめるなど、使い易くした。

dec_dump はこれを 10 進数で行うようにしただけのものであるが、使ってみるとここで新たな発見をした。

これまで、システム処理は 16 進数を用いて行うものと頭から決めていたのが、とんでもない思い込みであることが分かった。コンピュータ内部のバイナリデータでも、これを扱うのはあくまで人間である。それならよく慣れた 10 進数で、アドレスも値も扱えばよい。さらには値を 10 進数で表示するにはそのままで良いが、16 進数、A, B, C, D, E, F の表示には改めて文字に直しているので、そのまま演算はできない。コンピュータ内部のバイナリデータは、10 進数を用いることでもっと気楽に扱ったらよい。

ファイルから読み込まれたデータはグローバル値SENに入れられている。

```
   SEN
This is a dog.
```

この中から、文字列'is'の最初のアドレスをつぎのようにして求める。

```
   get_adr =: ([ E. ]) # (i.@ (#@]))  NB. get address
   'is' get_adr SEN
2 5
```

つまり、2箇所にある。最初の2は単語'This'の中の文字列のアドレスである。

この文字列'This'を、文字列置き換えのツールamdstr ～ ofにより、つぎのようにして、文字列'That'に置き換える。amdstr ～ ofの定義は最後に挙げてある。

```
   'That' amdstr 'This' of SEN
That is a dog.
```

ちなみに、次のようにすると、両方置換されてしまう。

```
   'at' amdstr 'is' of SEN
That at a dog.
```

```
   SEN1 =: 'That' amdstr 'This' of SEN
```

同様にして、文字列'dog'を'cat'に置き換える。

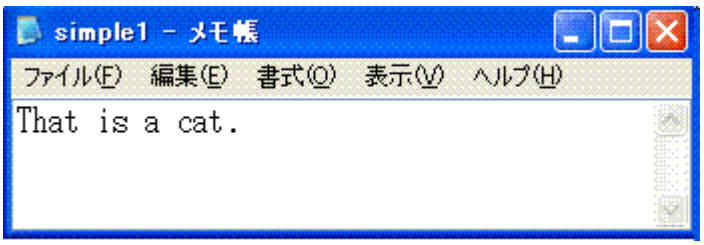
```
   SEN2 =: 'cat' amdstr 'dog' of SEN1
   SEN2
That is a cat.
```

このようにして、修正された文字列SEN2を別の名前でファイルとして、作成する。

```
   SEN2 fwrite '¥APL_test¥simple1.txt'
16
```

この結果は、Jの中から以下のコマンドにより、メモ帳を起動して見ることが出来る。

```
wd'winexec "notepad.exe ¥APL_test¥simple1.txt" ;'
```



```
NB. Basic Utilities ================================================
require 'files'
dir =: fdir

wr=: 1!:2&2
rd=: 1!:1
each=: &. >

deb=: #~ (+. 1&|.@(> </¥))@(' '&~:)

h=:  '0123456789ABCDEF'
dfh=: 16&#. @ (h&i.) f.
hfd=: dfh ^:_1 f.

NB. hex  (adverb)
NB. e.g. 'FF' + hex '8'
hex=: &. dfh

val=: a. & i.
chr=: val ^:_1
hdump=: }:@,@(,"1' '"_)@hfd@val f.
NB. non-print characters are converted to Space
chrn =: 3 : 0 "(0)
if. (32 > val y. ) +. (127 < val y.) do. ' '
else. y.
end.
)

NB. string amend program from amendstr.ijs ==============
str2box=: 3 : 0
:
Z1=. x. E. y.
Z2=. (|. x.) E. (|. y.)
Z3=. 1, }: |. Z2
Z4=. Z1 +. Z3                NB. revised to (+.) 2006/6/28
Z5=. Z4 <;.1 y.
AM=: ((<,x.) = Z5) # i. #Z5   NB. AM is global value
Z5
)
of=: str2box                 NB. alias for string amend
NB. Usage:
NB. revised 2006/6/28, 7/3
NB.   'pq' amdstr 'xyz' of 'xyzdefgxyzhi'
NB. pqdefgpqhi
NB.   'pq' amdstr 'x' of 'xyzdefgxyzhi'
```

```
NB. pqyzdefgpqyzhi
amdstr=: 3 : 0
:
; (<,x.) AM } y.                NB. revised 2006/7/3
)


NB. debug ===================================================
NB. e.g. dir '¥APL_test¥*.txt'
NB. e.g. debug '¥APL_test¥simple0.txt'
NB. debug - DOS command simulated
NB.   by T. Nishikawa 2001/11/26
NB.   using hdump, hgd, subs
NB.   revised corresponding characters added 2010/11/15

debug =: 3 : 0
SEN =. fread < y.
(0, #SEN) debug y.
:
NB. Print on NotePad option / 2010/11/28 ==========================
FileName =. y.
wr FileName
if. (1 = go_on_quit '*** Print on NotePad ')
 do. goto_printend.
 else.
  command =. '"notepad.exe ', (": FileName),'"'
  wd 'winexec ', command , ' ;'
end.
label_printend.

NB. ===============================================================
SEN =: fread < y.
if. _1 = SEN do. 'File_Read Error!' return. end.
if. 1 = L. x.
  do.   NB. Addres in hex
    if. 2 = # x.
      do.
        Start =. dfh > {. x.
        Length =. dfh > {: x.
      else.
        Start =. dfh > {. x.
        Length =. (#SEN) - (dfh > {. x.)
      end.
  else. NB. Addres in decimal
    if. 2 = # x.
```

```
      do.
        Start =: {. x.
        Length =. {: x.
      else.
        Start =: {. x.
        Length =. (#SEN) - ({. x.)
      end.
end.

wr 'File Length = ', (": #SEN), '(=', (hfd #SEN), ' hex) bytes'
SENA =: Length {. Start }. SEN  NB. extract by x. / e.g. (70, 100)
SEN =: SENA
SD =: hdump SEN
N =: >. ($SD) % 48
DA =. SD, ((N*48)-$SD)#' '
DN =: (N, 48)$DA
NNA =. i. N
NB. NNA =: (dfh {: hfd Start) |. i. N
NB. NN0 =. ({. x.)
NN0 =. <. Start % 16
NNB =: NN0 +"(0) NNA
NN =.  NNB



wr 'Dump Address (hex): from ', (hgd {. NNB), '0 to ', (hgd {: NNB), 'F'

if. (1 = go_on_quit '*** Dump ') do. '*** quit ***' return. end.

NB. dump ============================================================
Ind_CR =: (13 = val SEN)#i.$SEN
Ind_LF =: (10 = val SEN)#i.$SEN
Ind_CRLF =: Ind_CR, Ind_LF
if. (0<$Ind_CR) *. (0<$Ind_LF)
  do.
    SENX =. '_' Ind_CR } SEN
    SENX =. '_' Ind_LF } SENX
  else.
    SENX =: SEN
end.
NB. SENX =. '_' Ind_CRLF } SEN
SENX =. chrn SENX , 16#' '
NB. SENX =. chrn SENX
NB. wr ((>.(#SENX)%16), 16)$SENX
SN =: (N, 16)$SENX
```

```
NB. HEAD revised 2010/12/2
NB. HEAD =. '      -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F'
NR_HEAD =: (dfh {: hfd Start)
HEAD =. (5#' '), , '-',"(1) (hfd NR_HEAD |. i.16),"(1) ' '

label_loop.
if. ({.$DN) > 8
  do.
     DM =: 8 {. DN
     DN =. 8 }. DN

     SM =: 8 {. SN
     SN =. 8 }. SN

     NM =. 8 {. NN
     NN =. 8 }. NN

     wr HEAD, ((hgd NM) ,"(1) ': ') ,. DM ,. SM

     if. (1 = go_on_quit '') do. '*** quit ***' return. else. goto_next. end.

  else.
     wr HEAD, ((hgd NN) ,"(1) ': ') ,. DN ,. SN
     '*** end ***'
     return.
  end.
label_next.
goto_loop.

'** end **'
)

NB. ================================================================
go_on_quit =: 3 : 0
'q' go_on_quit y.
:
     wr y. ,'go_on ? (y or q)'
     yn =. rd 1
     if. (0 = #yn)
       do. 0
       else.
        if. (x. = yn) do. 1 else. 0 end.
     end.
)
```

```
NB. decimal dump (debug in decimal) ==============================
NB. decimal dump = 2010/11/8 by T. Nishikawa ====================
NB.    left arg: (start address, number of values)
NB.    right arg: file name
dec_dump =: 3 : 0
SEN =: fread < y.
(0, #SEN) dec_dump y.
:
SEN =: fread < y.
if. 1 = L. x.
  do.   NB. Addres in hex
    if. 2 = # x.
      do.
        Start =. dfh > {. x.
        Length =. dfh > {: x.
      else.
        Start =. dfh > {. x.
        Length =. (#SEN) - (dfh > {. x.)
      end.
  else. NB. Addres in decimal
    if. 2 = # x.
      do.
        Start =: {. x.
        Length =: {: x.
      else.
        Start =: {. x.
        Length =: (#SEN) - ({. x.)
      end.
end.

SENA =: Length {. Start }. SEN  NB. extract by x. / e.g. (70, 100)
SENB =. chrn SENA       NB. using chrn function
SDD =: val SENA
Row =. >. ({. $SDD) % 10
DA =. (Row, 10)$SDD,9#_
DAA =. 4 ": DA
DAN =: (10 | Start) + i.10   NB. revised 2010/12/3
DAB =: (' ', }. 4 ": DAN) , (({: $ DAA)#'-') , DAA
DC =: ' ',"(1) ((2, 10)$' '), (Row, 10)$SENB,9#' '
RR =: <. (Start%10) + i. Row
RRA =. ": ,. RR
RRR =. ' ',(({:$RRA)#'-'),RRA
RRR,"(1) '|',"(1) DAB ,"(1) DC
)
```