1セ###ォ########□##B#N#N#N#N#O##P#################################APL・Jシンポジウム資料 2001/12/8

JによるDEBUGコマンド・プログラムと BMPファイルの解析への適用

西川 利男 (Toshio Nishikawa.@kiu.ne.jp)

Jでは10進数の扱いはもちろんのこと，いろいろなシステム処理のプログラミングが可能である．DOS時代にはシステム解析の基本のツールとしてDEBUGコマンドが愛用されたが，Jのシステム上で同じ機能をシミュレートするプログラムを作った．まずこれを紹介する．

つぎにこれを利用してJによりBMPファイルの構造の解析を行う．先月の例会で慶応大の塩谷昌典氏により24ビットBMPファイルの構造についての発表があった．実は，わたくし自身も必要にせまられ，BMPファイルの解析をおこなっている．先の発表と重複する部分もあるが，ここではJがシステム処理にいかに強力であるかを示す例としてとりあげた．

1．JによるDEBUGシミュレーションプログラム

Jのdebug定義のくわしいコーディングは最後の付録にあげた．

例えば，つぎのような簡単なテキストファイル sample.txt を作って，プログラム debug を適用してみよう．

```
------------------------------------------------------
(123)
This is a short sentence.
次は，漢字，ひらがな，カタカナ交じりの日本語の文章です．

ーJのコーディング例ー
  x =. i. 3 4
  x
0 1  2  3
4 5  6  7
8 9 10 11
  +/x
12 15 18 21
  +/"1 x
6 22 38

*** xyz ***
------------------------------------------------------
```

実行方法は，以下のようにファイル名を指定して打ち込めば，DOSのDEBUGコマンドの場合と同じスタイルで，16進表示の表形式でただちに出力される．

```
  debug 'sample.txt'
   -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F
000: 28 31 32 33 29 0D 0A 54 68 69 73 20 69 73 20 61
001: 20 73 68 6F 72 74 20 73 65 6E 74 65 6E 63 65 2E
002: 0D 0A 8E 9F 82 CD 81 43 8A BF 8E 9A 81 43 82 D0
003: 82 E7 82 AA 82 C8 81 43 83 4A 83 5E 83 4A 83 69
004: 8C F0 82 B6 82 E8 82 CC 93 FA 96 7B 8C EA 82 CC
005: 95 B6 8F CD 82 C5 82 B7 81 44 0D 0A 0D 0A 81 7C
006: 82 69 82 CC 83 52 81 5B 83 66 83 42 83 93 83 4F
007: 97 E1 81 7C 0D 0A 20 20 20 78 20 3D 2E 20 69 2E
go_on ? (y or q)
y
   -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F
008: 20 33 20 34 0D 0A 20 20 20 78 0D 0A 30 20 31 20
009: 20 32 20 20 33 0D 0A 34 20 35 20 20 36 20 20 37
00A: 0D 0A 38 20 39 20 31 30 20 31 31 0D 0A 20 20 20
00B: 2B 2F 78 0D 0A 31 32 20 31 35 20 31 38 20 32 31
00C: 0D 0A 20 20 20 2B 2F 22 31 20 78 0D 0A 36 20 32
00D: 32 20 33 38 0D 0A 20 20 0D 0A 2A 2A 2A 20 78 79
00E: 7A 20 2A 2A 2A 0D 0A
*** end ***
```

2. BMPファイルの構造

BMPファイルの基本の考え方は絵画データのヨコ（＝幅）×タテ（＝高さ）のピクセル（画素）に対して数値の配列を対応させる，ことである．ピクセル値の与え方によって2つの方式があり，これは1ピクセルを何ビットで表すかにも関係してくる．

・直接方式　24ビットBMP
　1ピクセルを3バイト＝24ビットにより、そのピクセルの色に対して，直接RGB値で表す
・参照方式　8ビットBMP
　1ピクセルを1バイト＝8ビットの値で表し，その値がどういう色を表すかは別に設けたパレットと呼ぶ参照表を参照して絵画データとする

色の数が少ない初期の頃には，メモリ容量も少なくてすむことから8ビットBMPが用いられたが，現在では色の数も多く24ビットBMPが標準的に用いられる．以下24ビットBMPのみについて述べる．

BMPファイルは大きく2つの部分，すなわちいろいろな仕様を示すヘッダー部とピクセルデータ部とから構成されている．

２.１ ヘッダー部（ヘッダーデータ（＝狭義のヘッダー）も含む）

なお，アドレスなど4バイトの16進数はパソコンの通例としてバイト逆順で表されている．
例 12 AB EF 00 の場合，実際の値は 00 EF AB 12(=15706898 10進)である

ヘッダー部のメモリ構成はつぎのようになっている．

アドレス(hex)	値と説明
00, 01	'BM' (=42 4D)
02 - 05	ファイルの大きさ(全バイト数)
06 - 09	(予備)
0A - 0D	ピクセルデータの開始アドレス
0E - 11	ヘッダーデータの開始アドレス
12 - 15	幅(ヨコ)の桁数(column)
16 - 19	高さ(タテ)の行数(row)
1C - 1D	BMP 方式の種別 18 hex = 24-BMP, 8 hex = 8-BMP

２.２ ピクセルデータ部

アドレス(0A - 0D)の間接指定で示されるアドレス(24ビットBMPでは 36番地)から最後までがピクセルデータの本体であり，24ビットBMPでは1ピクセル当たり3バイトの連続値である．しかし，一見，奇妙にも思われる以下の仕様には特に注意を要する．

・1ピクセルを3バイト(24ビット)のRGB値であらわすものの，実際のバイト値の順は B(Blue), G(Green), R(Red)と逆順になっている．
例 赤であれば 00 00 FF となる．

・それぞれの幅は1行で表される．必要なバイト数は，(桁数)×3バイトであるが，それに加えて最後に適当なバイト数の 00 を補充して，1行全体が4の倍数になるよう調整する(Zero Padding)．
つまり，1行が
1ピクセルなら 1バイトの 00 を補充して4(=1*3+1)バイトにし，
2ピクセルなら 2バイトの 00 を補充して8(=2*3+2)バイトにし，
.. ..
4ピクセルなら 00 の補充は不要で，12(=4*3)バイトでよい．

・ピクセルデータ全体は行ごとに逆順になっている．つまり，絵画データの一番下の部分からメモリに並べられる．

3．24ビットBMPファイルの解析の実際

小さなBMPファイルをペイントにより作成し，先に示したJのプログラムdebugにより，実際に値を比較して検討してみよう．

```
  sc.bmp
+--------------------+
|青緑赤黒白青緑赤黒白|
+--------------------+
|     青     |
+--------------------+
|     緑     |
+--------------------+
|     赤     |
+--------------------+
|     黒     |
+--------------------+
|     白     |
+--------------------+
```

```
  debug 'sc.bmp'
   -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F
000: 42 4D|F6 00 00 00|00 00 00 00|36 00 00 00|28 00   全バイト数 F6
001: 00 00|0A 00 00 00|06 00 00 00|01 00|18 00|00 00   幅 10， 高さ 6
002: 00 00 C0 00 00 00 CE 0E 00 00 C4 0E 00 00 00 00
003: 00 00 00 00 00 00|FF FF FF FF FF FF FF FF FF FF
004: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF   白(10*3+2)
005: FF FF FF FF 00 00|00 00 00 00 00 00 00 00 00 00
006: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   黒(10*3+2)
007: 00 00 00 00 00 00|00 00 FF 00 00 FF 00 00 FF 00
go_on ? (y or q)
y
   -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F
008: 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00   赤(10*3+2)
009: FF 00 00 FF 00 00|00 FF 00 00 FF 00 00 FF 00 00
00A: FF 00 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 FF   緑(10*3+2)
00B: 00 00 FF 00 00 00|FF 00 00 FF 00 00 FF 00 00 FF
00C: 00 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 FF 00   青(10*3+2)
00D: 00 FF 00 00 00 00|FF 00 00 00 FF 00 00 00 FF 00
00E: 00 00 FF FF FF FF 00 00 00 80 00 00 00 FF 00 00   各種の色
00F: 00 FF FF FF 00 00|
*** end ***
```

参考文献

R. Dazeley, "J is for JPEG: Windows Bitmap to JPEG Baseline Compression", Vector, Vol.18, No.1 p.103 (2001).

付録 debug simulated routine

```
wr=. 1!:2&2
rd=: 1!: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. conjunction for substitute
subs =: [. & ((((e.&) ((# i.@#)@)) (@])) })
NB. decimal to hex in three digits
NB. dec 10 -> hex 'A' -> hex '0A'
hgd =: 3 : 0"(0)
'0' subs ' ' _3 {. hfd y.
)

NB. read DOS-file into J's data
fread=. 1!:1
NB. write J's data into DOS-file
fwrite=. 1!:2

NB. debug - DOS command simulated using hgd, subs
NB.   by T. Nishikawa 2001/11/26
NB.   NB. e.g. debug 'test.txt'
HEAD =: '     -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -A -B -C -D -E -F'
debug =: 3 : 0
SEN =: fread < y.
SD =: hdump SEN
N =: >. ($SD) % 48
DA =. SD, ((N*48)-$SD)#' '
DN =. (N, 48)$DA
NN =. i. N
label_loop.
if. ({.$DN) > 8
 do.
   DM =. 8 {. DN
   DN =. 8 }. DN
   NM =. 8 {. NN
   NN =. 8 }. NN
   wr HEAD, ((hgd NM) ,"(1) ': ') ,.DM
   wr 'go_on ? (y or q)'
   if. 'q' = rd 1 do. '*** quit ***' return. end.
 else.
   wr HEAD, ((hgd NN) ,"(1) ': ') ,.DN
   '*** end ***'
   return.
 end.
goto_loop.
)
```

BMPでは ▼###ｲ###……□##w#□##……####t#ﾄ###……ﾉ###o#>###……C###j#####……　###e#~###……□##`
#□##……©###[#p###……####################################p###u###v#□##……□##……####t#ﾄ#

##.....ﾉ###o#>###.....C###j#####..... ###e#~###.....□##`#□##.....©###[#p###.....################
################\###ヲ###.....ｲ###.....ﾒ###n#□##n#□##..... ###k#"###....."###^#l###^#n###^#
###^#□##^#H###^#□##.....□##.....####

<#########h##<#

<#########h##□##ｱ###.....□##.....□##.....□##.....####.....####.....####.....'###.....2###.....:###.....G###.....R###.....[###....._###.....l###.....ヲ###.....ｲ###n#>###n#<#

<#########h##>###@###.....A###.....C###.....E###.....\###.....□##.....ハ###.....####.....8###.....o###.....ヲ###.....ン###.....####.....K###.....]###.....`###.....□##.....ホ###.....# ##.....< ##.....#h##< ##s ##.....ɪ ##.....□##.....#
##.....%
##.....'
##.....)
##.....+
##.....−
##...../
##.....I
##.....K
##.....3###n#Q###n#ゥ###n#ナ###n#G

##n#I

##n##

<#########h##I

##

<#########h##ソ###□##.....####.....X###.....Z###.....s###n#g###n#i###n#□##n#####n#####n#ホ###n#□##n#'###n#n###n#□##n#へ###n#マ###n#9###n##

<#########h##9###:###n#<###n#f###n#□##n#□##……□##……####……####……;###……S###……k###……□##……□##……ウ###……ヒ###……□##……□##……####l##<

<#########h######+###y#-###.....@###.....w###.....シ###.....####.....7###.....p###.....ウ###.....□##.....-###.....d###.....v###.....y###.....ー###.....□##.....*###.....m###.....、###.....□##.....##<#□######.....＿###.....□##.....」###.....･###.....ッ###.....ア###.....□##.....-###.....．###.....M###.....O###.....\###.....h###.....j###.....□##.....□##.....ー###.....テ###.....ワ###.....##<#ワ###□##.....□##..... ###.....-###...../###.....P###.....}###.....。###.....ト###.....ヨ###.....□##.....□##.....□##.....####.....%###.....G###.....V###.....X###.....□##.....ュ###.....##<#ュ###ホ###.....####.....####.....1###.....B###.....V###.....r###.....□##.....□##.....□##.....ッ###.....キ###.....ヒ###.....゜###.....□##.....####.....1###.....M###.....□##.....□##.....##<#□##オ###.....ノ###.....ワ###.....□##.....□##.....□##.....□##.....□##.....□##.....□##.....ッ###.....キ###.....ヒ###.....゜###.....□##.....####.....1###.....M###.....□##.....□##.....##<###
#########チ######
007: 00 00 00 00 00 00|00 00 FF 00 00 FF 00 00 FF 00
go_on ? (y or q)
y
  -0#####MS 明朝###6 -7 -8 -9 -A -B -C -D -E -F
008: 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 FF 00 00 赤(10*3+2)
009: FF 00 00