

## Jによるシステム処理－3 APL2セッション・ログの解析と表示・印刷－続き

西川 利男

先月の例会でJによるシステム処理の一つとして、APL文字を表示・印刷することを報告したが、このときは1つの例についてとりあえず試みただけであった。今回、APL2セッション・ログについて、より一般的に行えるようプログラムの改善、修正をおこなった。

プログラムの流れは次のようになる。

① APL2のセッション・ログのファイルは、これまでのセッション履歴のベタ打ちされた文字のコード値の集合であるので、これを改行コードなどを手掛りに、各セッションごとに見やすく整形する。この過程は実用のプログラムとしては、必ずしも必要ではないが、今の場合のように、処理を解析してプログラムを開発するために、見やすく表示することは重要である。

② APL2で決められた文字フォントを参照するコード値を、新たに作製した外字フォントを参照するようにコード値の変換を行う。このとき単なる1バイトごとではなくら1バイトから2バイトへの値の変換が必要になるのが、少々やっかいである。

両方の過程において、文字列の解析などには前回紹介したJのデバッグプログラム debug と 10進ダンプ dec\_dump を用いた。

初めに、APL2のセッション・ログをそのままメモ帳で見たものを表示してみる。

```

IBM MAP2 - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
2      ◀@ PRs+/[1] 0 = (.) . | .      ↑@      PR      !!@1 2 2 3
2      ↑@*      N      ↑@15      N      ↑@      PR      ↑@1 2 2 3
PQ      →@-2 3 5PQ      ←@*      N+100      ↑@105      N+100      @*      N*
100     @      PR      @1 2 2 3 2      @      PR      !@1 2 2 3
2      @      PQ      #@-2 3 5PQ      $@*      *PRIME N      %@* [1]
[0.]    &@*[0] Zs*PRIME N      ↑@ [1]      (@([1] 0 Zs(2 = +/[1] 0 = (
.) . | .) / .)      )@*[2] *      *@*      *PRIME[疹* +@*
[0] Zs*PRIME N* ,@ [1] Zs(2 = +/[1] 0 = (.) . | .) / .      -@+      * 2011-01-15 13
.40.03N) / .      .@*      PRIME 10      /@ 2 3 5 7IME 10      @@+      PRIME 100      1@G2
3 5 7 11 13 17 19      23 29 31 37 41 43 4 7 53 59 61 67 71 73      79 83 89 97
2@r      3@+      A*2 2 . 3 4 . 2      4@      B*3 14
5@r      X*B . A      6@*      A      7@12 3 A      8@14 . 2 A
9@*      B      :@13 14 B      :@*      X      <@13 . 1 X
=@*      )OFF      .,NUMTHEO APL      14 088 1990-01-23 22.24 .56
. ,STATS APL      22 248 1990-02-24 17.42 .58      .,DIFFER APL
8 408 1990-02-25 17.12 .54      .,NSTAT1 APL      7 064 1995-07-07
07.38 .38      .,TRYUTIL APL      7 288 1990-02-24 17.48 .22
. ,MUSIC APL      13 000 1990-03-04 17.46 .40      .,SPIRAL
APL      5 544 1996-08-13 16.19 .46      .,NKEYS APL      3 976
1990-08-03 10.07 .48      .,ISHIMU APL      9 192 1995-07-14 06.39
.42      .,PICALC APL      21 032 1990-01-15 18.13 .42
. ,ESTRAN APL      72 472 1990-01-24 08.44 .48      .,UTILITY APL
48 968 1990-01-31 04.55 .46      .,CPLA APL      51 480 1994-01-25
10.55
    
```

このように、そのままでは到底分からない。

これを、Jの画面の上で、APL文字を用いたセッションとして表示するプログラム aplses の詳細は最後のリスティングにあげてある。

プログラムの概要は次のようになる。

まず、fread のコマンドにより、APL2のセッション・ログのファイルを読み込む。

つぎに、先に述べたプログラム処理の①の部分をおこなう。

APL2のセッションの開始と終了を示す )CLEAR と )OFF の文字列で囲まれる部分で各セッションごとに分割する。

セッション・ログは画面表示のすべてを文字コードで表示してある。このため、空白部分はすべてスペースコード、つまり16進20, 10進32で埋められている。したがって、APL2のコードで不要な空白は取り除く。del\_leadSPとdel\_trailCRはそのためのものである。その他必要な整形処理を行った後、セッションごとに表示する。

すべてのセッションは相当の量になることもあるので、必要なセッションのみを選択するようにした。

その後、プログラム処理の②の部分のAPL文字の変換を行う。APLフォントのコード変換テーブルAPL\_CFにしたがって、変換プログラムapl\_fconvにより文字列の変換をおこなう。ここで、先に述べた1バイトから2バイトへの値の変換をおこなう。Jのプログラムとしては、ここではループをつかわざるを得なかった。

このようにして、APL文字で表示したAPL2のセッションのようすをJの画面で見ることが出来る。これをメモ帳などに書き込めば、メモ帳で見ることが出来る。

実行のようすは以下のようになる。

```
aplsls '¥APL_test¥ibmap2.txt'
==== Session 0: =====
)CLEAR
    .?      CLEAR WS
EAR
    「?      )LIB .APL
    」?, PROFILE  APL      6728 1988-12-20 12.00.00
        ,?, START   APL      3224 1989-04-24 14.00.50
        ·?, RADIX
==== Session 1: =====
(途中省略)

    ~ ~ ~ ~
    ~ ~ ~ ~
(途中省略)
==== Session 4: =====
)CLEAR
    .      CLEAR WS
EAR
    ▪?      Aス・2
```

```

IBMAP2 - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
2      ◀@ PRs+/[1] 0 = ( . ) . | .      1@      PR      !!@1 2 2 3
2      1@*      N      +@15      N      T@      PR      +@1 2 2 3
2      PQ      →@-2 3 5PQ      ←@*      N+100      @J105      N+100      @*      Ns
100     @      PR      @1 2 2 3 2      @      PR      !@1 2 2 3
2      @      PQ      #@-2 3 5PQ      $@*      *PRIME N      %@[1]
[0 . ]      &@*[0]      ZsPRIME N      @J[1]      (@([1]      Zs(2 = +/[1] 0 = (
. ) . | . ) ) / .      )@.[2]      *      *PRIME[疹*      +@*
[0]      ZsPRIME N*      ,@ [1]      Zs(2=+/[1]0=( . ) . | . ) / .      -@→      * 2011-01-15 13
.40.03N)/ .      .@*      PRIME 10      /@2 3 5 7IME 10      @+      PRIME 100      1@G2
3 5 7 11 13 17 19      23 29 31 37 41 43 4 7 53 59 61 67 71 73      79 83 89 97
2@r      3@+      A*2 2 . 3 4 . 2      4@      B*3 14
5@r      X*B . A      6@      A      7@12 3 A      8@14 . 2 A
9@      B      :@13 14 B      :@      X      <@13 . 1 X
=@      )OFF      . ,NUMTHEO      APL      14 088 1990-01-23 22.24      .56
. ,STATS      APL      22 248 1990-02-24 17.42      .58      . ,DIFFER      APL
8 408 1990-02-25 17.12      .54      . ,NSTAT1      APL      7 064 1995-07-07
07.38      .38      . ,TRYUTIL      APL      7 288 1990-02-24 17.48      .22
. ,MUSIC      APL      13 000 1990-03-04 17.46      .40      . ,SPIRAL
APL      5 544 1996-08-13 16.19      .46      . ,NKEYS      APL      3 976
1990-08-03 10.07      .48      . ,ISHIMU      APL      9 192 1995-07-14 06.39
.42      . ,PICALC      APL      21 032 1990-01-15 18.13      .42
. ,ESTRAN      APL      72 472 1990-01-24 08.44      .48      . ,UTILITY      APL
48 968 1990-01-31 04.55      .46      . ,CPLA      APL      51 480 1994-01-25
10.55

```

A  
 1 2 3 4 5 6 7 8 9 10 11 12  
@ P s 3 4 檻

```

IBMAP2 - メモ帳
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)
2      ◀@ PRs+/[1] 0 = ( . ) . | .      1@      PR      !!@1 2 2 3
2      1@*      N      +@15      N      T@      PR      +@1 2 2 3
2      PQ      →@-2 3 5PQ      ←@*      N+100      @J105      N+100      @*      Ns
100     @      PR      @1 2 2 3 2      @      PR      !@1 2 2 3
2      @      PQ      #@-2 3 5PQ      $@*      *PRIME N      %@[1]
[0 . ]      &@*[0]      ZsPRIME N      @J[1]      (@([1]      Zs(2 = +/[1] 0 = (
. ) . | . ) ) / .      )@.[2]      *      *PRIME[疹*      +@*
[0]      ZsPRIME N*      ,@ [1]      Zs(2=+/[1]0=( . ) . | . ) / .      -@→      * 2011-01-15 13
.40.03N)/ .      .@*      PRIME 10      /@2 3 5 7IME 10      @+      PRIME 100      1@G2
3 5 7 11 13 17 19      23 29 31 37 41 43 4 7 53 59 61 67 71 73      79 83 89 97
2@r      3@+      A*2 2 . 3 4 . 2      4@      B*3 14
5@r      X*B . A      6@      A      7@12 3 A      8@14 . 2 A
9@      B      :@13 14 B      :@      X      <@13 . 1 X
=@      )OFF      . ,NUMTHEO      APL      14 088 1990-01-23 22.24      .56
. ,STATS      APL      22 248 1990-02-24 17.42      .58      . ,DIFFER      APL
8 408 1990-02-25 17.12      .54      . ,NSTAT1      APL      7 064 1995-07-07
07.38      .38      . ,TRYUTIL      APL      7 288 1990-02-24 17.48      .22
. ,MUSIC      APL      13 000 1990-03-04 17.46      .40      . ,SPIRAL
APL      5 544 1996-08-13 16.19      .46      . ,NKEYS      APL      3 976
1990-08-03 10.07      .48      . ,ISHIMU      APL      9 192 1995-07-14 06.39
.42      . ,PICALC      APL      21 032 1990-01-15 18.13      .42
. ,ESTRAN      APL      72 472 1990-01-24 08.44      .48      . ,UTILITY      APL
48 968 1990-01-31 04.55      .46      . ,CPLA      APL      51 480 1994-01-25
10.55

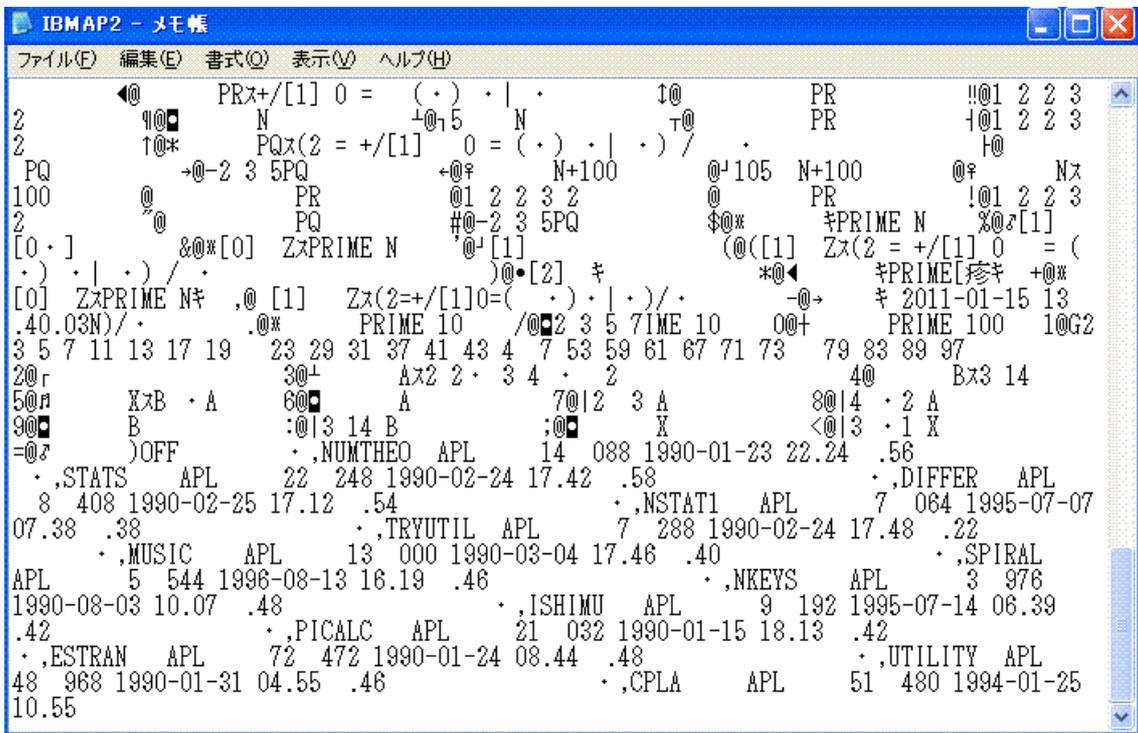
```

@ P

```

@
1 2 3 4
@
5 6 7
==== Session 5: =====
)CLEAR
  @ CLEAR WS
EAR
  @
  Nス5
  @ PRス+/[1] 0 = (.) . | .
  @ PR
  @
1 2 2 3 2

```



```

μ@ N
$ @5
N
@ PR
==== Select Sesion #: ?
5
StartADR = 5899 EndADR = 1276

===== APL2 Session No.5=====
)CLEAR
CLEAR WS

```

N←5  
 PR←+/[1] 0 = (ι N) →. | ι N  
 PR  
 1 2 2 3 2  
 N  
 5

PR  
 1 2 2 3 2  
 PQ←(2 = +/[1] 0 = (ι N) →. | ι N) / ι N  
 PQ  
 2 3 5

N+100  
 105  
 00

N←100  
 PR  
 1 2 2 3 2  
 PR  
 1 2 2 3 2  
 PQ  
 2 3 5

○PRIME N  
 [1] [0•0]  
 [0] Z←PRIME N  
 [1]  
 [1] Z←(2 = +/[1] 0 = (ι N) →. | ι N) / ι N  
 [2] ○  
 ○PRIME[•]○  
 [0] Z←PRIME N

[1] Z←(2=+/[1]0=(ι N)→. | ι N)/ι N  
 ○ 2011-01-15 13.40.03  
 ι N  
 PRIME 10  
 2 3 5 7  
 10  
 PRIME 100  
 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

A←2 2 ρ 2 3 4 ♦2

X←B ▪ A  
A  
2 3  
4 ♦2  
B  
3 14  
X  
3 ♦1  
)OFF

```

NB. APL Character Code Conversion =====
NB. Left Arrow '←', Right Arrow
APL_CF =: ((189);(129 169)),: (184);(129 168)
NB. Iota 'ι', Rho 'ρ'
APL_CF =: APL_CF, ((236);(131 199)),: ((230);(131 207))
NB. Domino 'ι', Quad 'ρ'
APL_CF =: APL_CF, ((146);(240 64)),: ((144);(240 65))
NB. Overbar '◊', Del '⊖'
APL_CF =: APL_CF, ((253);(240 68)),: ((183);(240 69))
NB. Jot '→'
APL_CF =: APL_CF, ((248);(240 70))

apl_fconv =: 3 : 0
:
CF =: x.
APLSes =. , y.
i =. 0
while. i < #CF
do.
'Code Font' =. i { CF
Acode =. chr Code
Afont =. chr Font
Adr =. (Acode = APLSes) # (i. # APLSes)
APLSes =. ; (<Afont) Adr } <"(0) APLSes
NB. wr > <;. 1 , APLSes
NB. if. (1 = go_on_quit '') do. '*** quit ***' return. end.
i =. i + 1
end.
APLSes
NB. > <;. 1 , APLSes
)

del_leadSP =: 3 : 0
(0 i.~'' = y.) }. y.
)

del_trailCR =: 3 : 0
(1 i.~ CR = y.) {. y.
)

NB. Usage:
NB. aplses '¥APL_test¥ibmap1.txt' => display all raw session
aplses =: 3 : 0
ASENO =: fread < y.
CLEADR =: (1 = ')CLEAR' E. ASENO) # (i. #ASENO)

```

```

i =. 0
while. i < #CLEADR do.
  wr '==== Session ', ("i), ': =====', CR , 200 {. (i{CLEADR} ). ASEN0
  i =. i + 1
end.
label_skip.

wr '==== Select Sesion #: ?'
ii =. ". rd 1

ASEN1 =: (ii{CLEADR} ). ASEN0
OFFADR =: (1 = ')OFF' E. ASEN1) # (i. #ASEN1)
wr 'StartADR = ', ("ii{CLEADR} , ' EndADR = ', ("{. OFFADR)
wr ' '
wr '===== APL2 Session No.', ("ii), '=====
ASEN2 =: ( 4 + {. OFFADR) {. ASEN1
ASEN3 =: <|. 2 (ASEN2, CR)
ASEN4 =: del_leadSP L:0 ASEN3
ASEN5 =: del_trailCR L:0 ASEN4
ASEN6 =: ((i.#>ASEN5) -. 0, 2 ) { > ASEN5
ASEN7 =: ' )CLEAR', 3 }. "(1) ASEN6
ASEN8 =: ASEN7 , "(1) CR
ASEN9 =: APL_CF apl_fconv ASEN8
)

```



以下、順に1ステップずつ追っていきます。また  $N = 5$  の場合の途中経過もいっしょに示します。

(1) 整数  $M$  を  $N$  で割った余りは関数  $|$  で求められます。

$$N | M$$

(2) 整数  $M$  を  $1 \sim N$  までの整数で割った余りは、次のようになります。

$$(1N) | M$$

(3) さらに  $1 \sim M$  までの整数を、 $1 \sim N$  までの整数でそれぞれ割った余りを出すには、どうしたらよいでしょう。これは外積演算で行うことができます。また問題とするものは、 $1 \sim N$  までの整数に対して行うことから

$$(1N) \circ \cdot | 1N$$

結果は  $N \times N$  の二次元配列になります。

なぜ  $(1N) | 1N$  ではいけないかは考えてみて下さい。

$$N \leftarrow 5$$

としたときの結果は次のようになります。

$\circ \cdot  $	1	2	3	4	5
1	0	0	0	0	0
2	1	0	1	0	1
3	1	2	0	1	2
4	1	2	3	0	1
5	1	2	3	4	0

(4)  $0 = (1N) \circ \cdot | 1N$

これは、配列のそれぞれの要素が0に等しいかどうかテストします。

0に等しければ フラグ1 (真)

0に等しくなければ フラグ0 (偽)

になります。

$0 =$	1	2	3	4	5
1	1	1	1	1	1
2	0	1	0	1	0
3	0	0	1	0	0
4	0	0	0	1	0
5	0	0	0	0	1

(5) いま求めたフラグ1 (真) が1とNの2回だけえられるNが素数です.

これは, 先の配列をまずタテ方向に, すなわち[1] 軸で低減します. 結果は一次元のベクトルになります.

$$+ / [1] 0 = (1 N) \circ . | 1 N$$

1 2 2 3 2

(6) 上のベクトルの要素が2に等しいかどうかのテストをして, フラグを得ます.

$$2 = + / [1] 0 = (1 N) \circ . | 1 N$$

0 1 1 0 1

(7) このフラグベクトルにより, もとの1~Nの整数から, フラグ1 (真) のものを取り出します. つまり, 素数の条件にあったものが取り出されます.

$$(2 = + / [1] 0 = (1 N) \circ . | 1 N) / 1 N$$

2 3 5

以上の操作で1~Nまでの間の素数がとり出されます.

このようにたった1行のAPLプログラムにより, 素数が得られます. これを, 他のプログラム言語で書いた場合, 何行ぐらいになるでしょうか.

しかし, ここで注意したいことがあります. APLプログラマは, えてしてコンパクトなプログラムを作りたがるのですが, あまり短くして, わかりにくくすることはさけなくてはなりません. あくまでも, これはここまで使えるというようにとどめるべきと思います.