

Jによるシステム処理－2 APLのセッション表示の解析と APL 文字の表示と印刷

西川 利男

0. はじめに－APL文字を表示、印刷したい

筑波の研究所にいた頃 FORTRAN でいろいろプログラムを組んでいたが、ふとしたきっかけで APL を知りそのとりこになってしまった。最初はメインフレーム上の VSAPL について APL2 と使い、やがてパソコン上の APL2/PC をはじめとして、日本語 APL、IAPL などいろいろな APL に接した。

1990 年の前後、10 年ほどは毎年のように、世界各地で行われた APL Conference に出席していながら、Iverson, Hui らの「APL\?」の論文発表は全く知らなかった。その後、研究所を退職する頃、コンピュータ環境がパソコンだけになってしまったのに合わせるかのように、普通のキーボードで出来る J を愛用するようになった。気がついたらそれからもう 10 年以上経っている。

最近、いろいろなところで J と比較するため、APL の実際を見せる必要が起きている。せめて、APL 文字がどういうものか見せたいと思うが、現在は筆者のパソコン環境では文字バケして、印刷もできなくなってしまった。幸い DOS レベルの APL2/PC は動いて APL 文字も表示できる。そこでテキスト文書を介して、外字作製により APL 文字を表示、印刷することを試みた。

1. APL 文字の外字作製、登録

印刷された APL 文字をスキャナーにより読み取り、画像データとしてこれを元にアクセサリの外字作製ツールを使って、以下のように登録した。

F040: ◻ F041: ◈ F042: ◆ F043: ◆ F044: ◆ ……

2. DOS-APL2 のセッション・ログの解析

最初は***.APL として保存される APL のプログラム・ファイルを対象とすればよいと思ったが、これはプログラム・コードだけであり、APL の実際を反映していないのでとりやめた。代わりに、APL のセッションの記録であるセッション・ログのファイルに対して行うことにした。

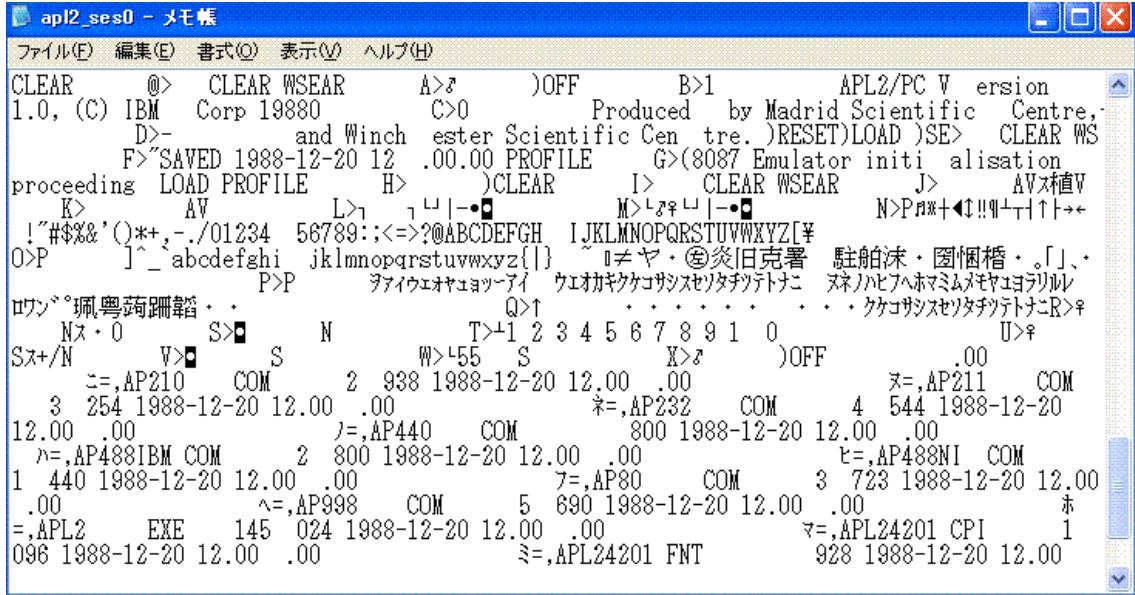
パソコン IBM ThinkPad 701C の DOS 上で APL2/PC を起動し、実行する。通常のとおり終了した後、ファイル APL2\$SES.LOG にはこれまでのすべてのセッション画面のログが保存されている。これを利用することにした。

まず、たとえばファイル名を apl2_ses0.txt のように rename する。このファイルを Windows のメモ帳で見ると、セッション・ログは 8K バイトもある大量の「ベタ」ファイルであり、この中から APL の興味のある部分を取り出すのは容易なことではない。結局、セッション・ログの表示構造の解析をおこなうはめになってしまった。その後、APL 文字への変換をおこなうことになる。

以下、具体的にそのようすをお目にかけてよう。

3. APLセッション・ログの実際

まず、最初のセッション・ログのファイルapl2_ses0.txtをメモ帳で見るとつぎのようになる。



これで見るように、起動の度ごとに表示される最初のメッセージを含めて、これまでのセッションの履歴がすべて保存されている。この中から、興味のあるAPLのコーディングを取り出すにはどうすればよいか。

最近の記録、つまりファイルの最後から見て

```
)CLEAR
```

と

```
)OFF
```

との間の文字列データをとりだすことにした。

DOS ファイルは、つぎのようにしてJのメモリに取り込まれる。

```
NSENO := fread <'¥APL_test¥apl2_ses0.txt'
```

ファイルの中で、上の文字列 ')CLEAR'は何箇所もあり、その最新の位置は次のようにして、求められる。

```
CLEADR := (1 = ')CLEAR' E. NSENO) # (i. #NSENO)
```

```
StADR := {: CLEADR
```

同様にして、文字列 ')OFF'の位置も求められる。

```
OFFADR := (1 = ')OFF' E. NSENO) # (i. #NSENO)
```

```
EdADR := {: OFFADR
```

```
StADR
```

6251

```
EdADR
```

6911

```
EdADR-StADR
```

660

前報で述べたツール10進ダンプ dec_dump を用いて、バイトごとの中身を検討してみよう。

```
(6251, 670) dec_dump '¥APL_test¥apl2_ses0.txt'
| 1 2 3 4 5 6 7 8 9 10
-----
625| 41 67 76 69 65 82 13 32 32 32 )CLEAR
626| 32 32 32 73 62 9 67 76 69 65 I> CLEA
627| 82 32 87 83 13 69 65 82 13 32 R WS EAR
628| 32 32 32 32 32 74 62 13 32 32 J>
629| 32 32 32 32 65 86 189 144 65 86 AV AV
630| 13 32 32 32 32 32 32 75 62 9 K>
631| 32 32 32 32 32 32 65 86 13 32 AV
632| 32 32 32 32 32 32 32 32 32 76 L
633| 62 2 9 10 2 3 4 5 6 7 >
(途中省略)
674| 192 193 194 195 196 197 198 82 62 12 R>
675| 32 32 32 32 32 32 78 189 236 49 N 1
676| 48 13 32 32 32 32 32 32 32 83 0 S
677| 62 8 32 32 32 32 32 32 78 13 > N
678| 32 32 32 32 32 32 32 32 32 32
679| 32 84 62 21 49 32 50 32 51 32 T> 1 2 3
680| 52 32 53 32 54 32 55 32 56 32 4 5 6 7 8
681| 57 32 49 0 0 48 13 32 32 32 9 1 0
682| 32 32 32 32 32 32 32 32 32 32
683| 32 32 32 32 32 85 62 12 32 32 U>
684| 32 32 32 32 83 189 43 47 78 13 S +/N
685| 32 32 32 32 32 32 32 86 62 8 V>
686| 32 32 32 32 32 32 83 13 32 32 S
687| 32 32 32 32 32 32 32 32 32 87 W
688| 62 3 53 53 13 32 32 32 83 13 > 55 S
689| 32 32 32 32 32 32 32 32 32 32
690| 32 88 62 11 32 32 32 32 32 32 X>
691| 41 79 70 70 13 32 32 32 32 32 )OFF
```

これを使って、手ごろの大きさの文字列を得ることが出来た。

```
NSEN1 =: (500+StADR) }. (4+EdADR) {. NSENO
```

実際の作業には、前報で述べたツール debug、とくに10進ダンプ dec_dump により、文字のコード値を見ながら、ひとつずつ丹念に行うことになる。

トライアンドエラーで見つけた表示の構造はつぎのようになることが判った。

```
表示行番号(1バイト) + '>' + 行文字数(1バイト)
```

```
SP(6バイト) + APLコード + SP(適宜バイト) …… + CR
```

いくつかの不要な文字の取り除き、修正をおこなった(詳細は稿末のリスティングを参照)あと、つぎのように、整形したAPLのコーディングを取り出すことが出来た。

```

      NSEN5
>
>      N←・0
>      N
>1 2 3 4 5 6 7 8 9 10
>      S←+/N
>      S
>55

```

ここで、いよいよ文字バケてしまった APL 文字の変換の処理を行う。そのためのプログラムは以下のようなになる。

```
APL_CF =: ((189);(129 169)),: ((236);(131 199))
```

```

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
    wr > <.; 1 , APLSes
    if. (1 = go_on_quit '') do. '*** quit ***' return. end.
    i =. i + 1
  end.
)

```

ここで APL_CF は、APL コードに対する作製した外字フォントの値への変換テーブルである。つまり例えば、文字「←」の場合

APL コード 189 は外字フォント 129 169

に対応する。なお、これらの値は 16 進ではなく 10 進で行っていることに注意。

関数定義 apl_fconv は変換のプログラムである。工夫を要したところは、1 バイトから 2 バイトへの置換で、ボックスでくくったものに行うことで解決した。そのためループを使って、1 つずつ行わざるを得なかった。

途中経過のチェックも含めて、実行すると次のようになる。

```

      APL_CF apl_fconv NSEN5
>
>      N←・0
>      N
>1 2 3 4 5 6 7 8 9 10
>      S←+/N

```

```

>      S
>55
go_on ? (y or q)
y
>
>      N←ι 10
>      N
>1 2 3 4 5 6 7 8 9 10
>      S←+/N
>      S
>55
go_on ? (y or q)
y
2

```

以上の結果をファイルとするには、各行の行末にCRLFを付加して、fwriteにより書き出せばよい。

```

NSEN7 =: NSEN6 , "(1) CRLF
NSEN7 fwrite '¥APL_test¥apl2_ses7.txt'

```

175

プログラム・リスト

```

NB. APL Session Display in APL characters =====
aplses0 =: 3 : 0
NSEN0 =: fread <'¥APL_test¥apl2_ses0.txt'

```

```

CLEADR =: (1 = ' )CLEAR' E. NSEN0) # (i. #NSEN0)
OFFADR =: (1 = ' )OFF' E. NSEN0) # (i. #NSEN0)
StADR =: {: CLEADR
EdADR =: {: OFFADR
NSEN1 =: (500+StADR) }. (4+EdADR) {. NSEN0 NB. last session
wr 'Original:'

```

```

wr NSEN1
NSEN2 =: > < ;.1 ' >', NSEN1    NB. punctuate at ' >' character
NB. delete spaces between CR and ' >' -character =====
NB. using function 'binf'
FCR =: 13 = val , NSEN2
FAR =: 62 = val , NSEN2
FF =: FCR + FAR
NSEN3 =: (binf FF)#, NSEN2
NB. delete after last CR =====2010/12/21=====
CRADR =. (CR = NSEN3)#(i. #NSEN3)
NSEN3 =: CRADR {. NSEN3
NB. delete less than 31 characters ===2010/12/21=====
FLes31 =: 31 & <"(0) val NSEN3
NSEN4 =: FLes31 # NSEN3
NSEN5 =: > < ;.1 ' >', NSEN4
NB. NSS =. (chr 189);(chr 129 169) apl_conv NSEN5
wr 'Pretty Formated:'
wr NSEN5

wr 'APL_characters converted:'
NB. convert APL_Font =====2011/1/9=====
APL_CF apl_fconv NSEN5
'*** end ***'
)

APL_CF =: ((189);(129 169)), : ((236);(131 199))

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
wr > < ;. 1 , APLSes
if. (1 = go_on_quit '') do. '*** quit ***' return. end.
i =. i + 1
end.
)

```