スーパーコンピュータ SX-1 の
タイム・シェアリング・システム ATSS-AF の使い方（その1）

大阪大学 大型計算機センター 研究開発部
馬野 元秀

1．はじめに

本センターでは、1986 年 5 月より NEC のスーパーコンピュータ SX-1 の運用を始める。スーパーコンピュータ SX-1 は、ほぼ通常の汎用機の機能を持つ CP（control processor）と高速にベクトル計算を行なう AP（arithmetic processor）の2つからできている。スーパーコンピュータを使用するのは AP の高速ベクトル計算の機能を利用するためであるが、プログラムの開発・保守などを行なうために CP を利用する必要がある。SX-1 も HFP と同様に、ACOS システム 1000（以下、ACOS-1000 と略す）のバックエンド・プロセッサとして使用するが、HFP の場合とは異なり、SX-1 の側にファイルを持つことができ、SX-1 の CP の タイム・シェアリング・システム(TSS)も利用できる。

スーパーコンピュータ SX-1 の概要については文献 [1]、[2] ですでに解説されているし、SX-1 の Fortran77 の文法とその使用法については、それぞれ文献 [3]、[4] で説明されている。また、簡易形の TSS と SX-1 へのファイルの転送については、文献 [5] で述べられている。

本稿では、SX-1 の CP 上の TSS である ATSS-AF（advanced time sharing system - advanced function）を ACOS-1000 の TSS-AF から使用する場合の基本的な使い方について説明する。これは SX-1 上にファイルを持つ使い方なので基本形と呼ばれる。なお、このように2つのオペレーティング・システム(OS)の仲介をするシステムを MSF（multi system facility）と呼び、本システムの場合は ACOS-6 上に置かれるので MSF-6 と呼ばれる [6]。

2．ATSS-AFの概要

SX-1 のオペレーティング・システムは SXOS と呼ばれるオペレーティング・システムで、その上の ATSS-AF は ACOS-1000 の ACOS-6/MVX 上の TSS-AF とはかなり異なっている。したがって、SX-1 上にファイルを持つ必要のないユーザは、簡易形によって SX-1 を使用すること勧める [4]、[5]。簡易形では、ATSS-AF に入ることなく SX-1 を利用することができる。SX-1 上にファイルを持つ必要のあるのは、(i) Fortran77 において open 文により動的にファイルを開く必要があるとき、(ii) コンパイル・ユニット(ACOS-1000 のオブジェクト形式に当たる)やロード・モジュール(ACOS-1000 の実行形式に当たる)やデータのファイルを保存したいとき、(iii) アナライザやベクトライザ [7] の機能をフルに使いたいとき、などである。

2.1. 使用開始

SX-1 の ATSS-AF は ACOS-1000 を経由して使用するので、まず、 ACOS-1000 の TSS-AF にログオンし、

```
SYSTEM ?SXTSS↵
```

と打ち込む(下線はユーザの入力とする。以下も同じ。また、↵ は「return」、「送信」、「書き込み」等のキィーを表わす。以下、特に強調する場合以外は ↵ を省略する)と、

```
SX MODE        RON = X00016638
*
```

となり、これ以降は ATSS-AF に入ることになる。ここで、RON は run occurrence number の略で、SXOS が管理に使う番号であり、使用するたびごとに異なる番号となる。

以下、ATSS-AF でプログラムを作成、修正、翻訳、結合、実行する方法について、例を中心に述べよう。 以下の例は、一応、SX-1 で動作を確認してあるが、今後、既定値が変更されたり、機能の追加が行なわれたりすると、以下と同じように入力しても出力が少し異なってしまうことがある。変更については「速報」や「センター ニュース」などに掲載されるので、注意しておいていただきたい。

2.2. ソース・プログラム・ファイル

SX-1 上にソース・プログラムのファイル (ソース・ユニットとも言う)を作成するには、ACOS-1000 上のファイルを転送する方法と SX-1 上に新たにファイルを作成する方法とがある。

(1) ソース・プログラムの転送

SX-1 ではプログラムは統合ライブラリ(integrated library)に入れる。 統合ライブラリは、ソース・ユニット、コンパイル・ユニット、ロード・モジュールなどを同時にいくつか納めることのできるファイルのファイルと考えればよい。そして、統合ライブラリ中のファイルはサブファイルと呼ばれる。

ACOS-1000 上にあるソース・プログラム・ファイルを SX-1 上の統合ライブラリに入れるには、まず、LIBALLOC コマンドで統合ライブラリを作成し、 .SSTAGE コマンドでファイルの転送を行なう。 いま、ILIB1 という統合ライブラリを作成し、そのサブファイル F1 に ACOS-1000 上のファイル SFILE1 を転送するには、次のようにすればよい。

```
*LIBALLOC ILIB1
ILIB1 ALLOCATED
*.SSTAGE SFILE1:ILIB1(F1):SOURCE FREE2
*
```

LIBALLOC コマンドでは、統合ライブラリ名(ILIB1)に対する指定以外はすべて既定値を使っている。.SSTAGE コマンドのパラメータ FREE2 は、ACOS-1000 側のファイルが NFORM 形式であることを表わしている(しかし、本稿の例では、プログラムの見易さを考えて、ACOS-1000 の $FORM を適用した普通の Fortran の形式を用いる)。 また、 .SSTAGE の先頭の . は ACOS-1000 のコマンドを意味する。したがって、ACOS-1000 側のファイルの情報を見たければ、

```
*.CATALOG
```

とすればよいし、SFILE1 の内容を見たければ、

```
*.LIST SFILE1
```

とすればよい。 なお、ファイル転送については、文献 [5] で説明されているので、詳しくはそちらを参照されたい。

さて、確認のために SX-1 の側のファイルの情報を見てみよう。

```
*FILLIST
                          /***  NAME  LIST  ***/            FILLIST REV.070
  NO. FILE / SUBFILE NAME              FILEORG  TYPE      PWORD  RUAF STATUS
 0001 A60000.ILIB1                     LINKQD   FILE      NO     YES
*
```

これを見ると、 自動的に利用者番号(A60000)と . が統合ライブラリ名の前に付けられているのがわかる。また、統合ライブラリは OS から見ると待機結合編成ファイル(linked queued file)と呼ばれるファイルであるので、FILEORG 欄では LINKQD と表示されている。さらに、ファイルの名前欄の幅がずいぶん広いのに気が付くだろう。SX-1上のファイル名は英字で始まる長さ 16 文字以内の英数字とハイフン(-)と下線(_)からなる文字列である。なお、ファイル名についての詳細は 3.1 節(次号掲載予定)を参照されたい。

サブファイルの確認は

```
*FILLIST ILIB1 SUBFILE
                          /***  NAME  LIST  ***/            FILLIST REV.070
  NO. FILE / SUBFILE NAME              FILEORG  TYPE      PWORD  RUAF STATUS
 0001 A60000.ILIB1                     LINKQD   FILE      NO     YES
    1 F1                                        SL
*
```

で可能である。SL はソース・ユニットであることを表わす型である。 サブファイル名は、英字で始まる長さ 28 文字以内の英数字と下線(_)からなる文字列であればよい。

また、サブファイル F1 の内容は次のようにして調べることができる（以下、コマンドの後の ; 以降はコメントを表わすとする)。

```
*LIST ILIB1(F1) SL                                  ; SL はソース・ユニットを表わす
FILE:ILIB1(F1)                                                    LIST REV.060
     10       PROGRAM FACT
     20       INTEGER N,F
     30     1 READ(5,*) N
     40       IF (N.LT.0) STOP
     50       WRITE(6,*) F(N)
     60       GOTO 1
     70       END
     80 C
     90       INTEGER FUNCTION F(N)
    100       INTEGER N,J
    110       F=1
    120       DO 10 J=1,N
    130         F=J*F
    140    10 CONTINUE
    150       END
TOTAL INPUT RECORDS : 15
TOTAL OUTPUT RECORDS : 15
*
```

ACOS-1000 では統合ライブラリは使用できなかったが、これを用いると、(i) 種々の操作が簡単になる、(ii) ファイル容量を節約できる、(iii) ソース・ユニットの世代管理ができる、(iv) ソース・ユニットを圧縮して保存できる、などの利点がある。また、アナライザやベクトライザなどの性能向上支援ツール [7] を使用するには、 プログラムが統合ライブラリに入っていなければならない。

先ほど、統合ライブラリを作成する(領域を割り付ける)ときには、 統合ライブラリ名(ILIB1)以外はすべて既定値を使った。もちろん、統合ライブラリには多くの属性がある。統合ライブラリの各種属性の値を見るには次のようにする。

```
*FILLIST ILIB1 DETAIL
                          /***  DETAIL  LIST  ***/          FILLIST REV.070
    1 A60000.ILIB1
        <==  CATALOG DESCRIPTION  ==>
        TYPE                  :  FILE
        GENERATION TYPE       :  NO
        MAX GENERATION        :  0
        CURRENT GENERATION    :  0
        PASSWORD PROTECTION   :  NO
        RUAF PROTECTION       :  YES
        <==  VOLUME LIST  ==>
        VOLUME NUMBER :  1
        DEVCLASS      : MS/M800
        MEDIA         : PUB050
```

```
        <==  FILE DESCRIPTION  ==>
        FILE     : ACOS4      BLOCKSZ  : 2036      CREATE   : 86-04-20
        FILEORG  : LINKQD     RECSIZE  : 2032      EXPIR    : 00-00-00
        INCRSZ   : 0          RECFORM  : VB        NODELR   : YES
        DIRSIZE  : 8          LOGTRKSZ : 1         USEDBLKS : 1 BLK
                              EMPTY    : NO        SIZE     : 450 BLK
*
```

いろいろな情報が表示されているが、 最後の SIZE は統合ライブラリの大きさを表わし、 BLOCSZ の大きさをもとにして領域が確保される。ただし、この大きさは初期値であり、ある程度までは自動的に大きくなる。 そして、既定値では領域が小さすぎるときは、LIBALLOC に SIZE パラメータを付けて、例えば、

```
*LIBALLOC ILIB2 SIZE=2000
ILIB2 ALLOCATED
*
```

とすればよい。

(2) ソース・プログラム・サブファイルの作成

すでに存在している統合ライブラリ中に、SX-1 側で新たにサブファイルを作成したいときには、エディタを使って次のようにすればよい。

```
*EDIT ILIB1(A1) NEW FTFREE2                    ; FTFREE2 - Fortran の FREE2 形式
00010      PROGRAM EXAMPLE
00020C   - TEST PROGRAM -
00030      READ(5,*) A,B
00040      WRITE(6,*) 'A+B = ',A+B,'    A-B = ',A-B
00050      END
00060[F                                        ; 入力の終わりは [F
E#L 10:50                                      ; 行番号 10 から 50 までの行を表示
00010       PROGRAM EXAMPLE
00020 C   - TEST PROGRAM -
00030       READ(5,*) A,B
00040       WRITE(6,*) 'A+B = ',A+B,'    A-B = ',A-B
00050       END
E#W                                            ; 現在の内容をファイルに書き出す
CREATED  UPDATED  REV    LINE NO-OF-BYTE LANTYPE ORIGINAL-ENTRY-NAME
86-04-20 86-04-20 000       6        230 FTFREE2 A1
E#E                                            ; エディタを終わる
*
```

このとき、00010 などの後に入力したものと、L サブコマンドで表示したものとでは行番号の後のスペースの数が 1 つ異なっていることに注意しよう。ATSS-AF のエディタには、もちろん画面エディタと行エディタがあるが、本稿では使える端末のことを考えて、以下でも行エディタについてだ

け述べる(画面エディタについては、文献 [8] を参照のこと)。

(3) 統合ライブラリの管理

統合ライブラリのサブファイルどうしのコピーは COPY コマンドで可能である。このとき、サブファイルの型 SL も与えなければならない。

```
*COPY ILIB1(F1) ILIB2(D2) SL                    ; ILIB2 はすでに存在している

 INPUT  NAME : F1
 OUTPUT NAME : D2
*FILLIST ILIB2 SUBFILE
                          /***  NAME  LIST  ***/          FILLIST REV.070
  NO. FILE / SUBFILE NAME          FILEORG  TYPE     PWORD  RUAF STATUS
 0001 A60000.ILIB2                 LINKQD   FILE     NO     YES
    1 D2                                    SL
*COPY ILIB1(F1) (D1) SL     ; 同じ統合ライブラリ中なら、後のライブラリ名は省略可

 INPUT  NAME : F1
 OUTPUT NAME : D1
*FILLIST ILIB1 SUBFILE
                          /***  NAME  LIST  ***/          FILLIST REV.070
  NO. FILE / SUBFILE NAME          FILEORG  TYPE     PWORD  RUAF STATUS
 0001 A60000.ILIB1                 LINKQD   FILE     NO     YES
    1 A1                                    SL
    2 D1                                    SL
    3 F1                                    SL
*
```

また、統合ライブラリ全体のコピーも、COPY コマンドにより可能である。

```
*LIBALLOC ILIB3
ILIB3 ALLOCATED
*COPY ILIB1 ILIB3 ALL

 INPUT  NAME : ILIB1
 OUTPUT NAME : ILIB3
*FILLIST ILIB3 SUBFILE
                          /***  NAME  LIST  ***/          FILLIST REV.070
  NO. FILE / SUBFILE NAME          FILEORG  TYPE     PWORD  RUAF STATUS
 0001 A60000.ILIB3                 LINKQD   FILE     NO     YES
    1 A1                                    SL
    2 D1                                    SL
    3 F1                                    SL
*
```

統合ライブラリ中の不要になったサブファイルは DELETE コマンドで消去できる。このときも、サブファイルの型 SL を指定する必要がある。

```
*DELETE ILIB1(D1) SL
CMND180 DELETION REQUEST SUBFILE: D1 DELETION <Y!N>?
Y↵                                                    ; Y の代わりに、↵だけでもよい
*FILLIST ILIB1 SUBFILE
                          /***  NAME  LIST  ***/            FILLIST REV.070
  NO. FILE / SUBFILE NAME             FILEORG  TYPE      PWORD  RUAF STATUS
 0001 A60000.ILIB1                    LINKQD   FILE      NO     YES
    1 A1                                       SL
    2 F1                                       SL
*
```

また、不要になった統合ライブラリは、DELETE コマンドで消去できる。

```
*DELETE ILIB3
CMND180 DELETION REQUEST FILE: A60000.ILIB3 DELETION <Y!N>?
Y
*FILLIST
                          /***  NAME  LIST  ***/            FILLIST REV.070
  NO. FILE / SUBFILE NAME             FILEORG  TYPE      PWORD  RUAF STATUS
 0001 A60000.ILIB1                    LINKQD   FILE      NO     YES
 0002 A60000.ILIB2                    LINKQD   FILE      NO     YES
*
```

2.3. テスト実行

最も簡単な実行のしかたは、エディタの中から行なうものである。

```
*EDIT ILIB1(F1)                                                    ; OLD が既定値
CREATED  UPDATED  REV    LINE NO-OF-BYTE LANTYPE ORIGINAL-ENTRY-NAME
86-04-20 86-04-20     0000000 0000000000 FTFREE2 F1
E#L ^:¥                                            ; ^ - 最初の行、¥ - 最後の行
00010       PROGRAM FACT
00020       INTEGER N,F
00030     1 READ(5,*) N
00040       IF (N.LT.0) STOP
00050       WRITE(6,*) F(N)
00060       GOTO 1
00070       END
00080 C
00090       INTEGER FUNCTION F(N)
00100       INTEGER N,J
00110       F=1
00120       DO 10 J=1,N
00130         F=J*F
00140    10 CONTINUE
00150       END
E#RUN
FORT77 COMPILER STARTED
EXECUTION STARTED
```

```
?5
120
?10
3628800
?-1
E#
```

この方法は ACOS-1000 での使い方に似ており、処理が終ってもエディタの中にいる状態に戻るので、プログラムにエラーがある場合でも、すぐに修正することができる。ただし、これは CP で実行されるので、デバッグ中のテスト実行にしか使えない。

RUN のようなエディタのサブコマンドは、現在編集中のプログラムを処理対象とするので指定が簡単で使い易いが、そのようなコマンドはあまり多くない。しかし、コマンドの前に * を付けると、ほとんどのコマンドをエディタの中から実行できる（新たに、エディタを起動することも可能である）。

```
E#*FILLIST
                          /***  NAME  LIST  ***/            FILLIST REV.070
  NO. FILE / SUBFILE NAME              FILEORG  TYPE      PWORD  RUAF STATUS
 0001 A60000.ILIB1                     LINKOD   FILE      NO     YES
 0002 A60000.ILIB2                     LINKOD   FILE      NO     YES
E#
```

このときは単にエディタの中からコマンドを起動しているだけなので、現在編集中のプログラムを処理対象とすることはできない（W サブコマンドで編集中のものをサブファイルに書き出して、そのサブファイルを指定することはできる）。コマンドやパラメータの書き方などは、* のプロンプトで入力するのとまったく同じである。デバッグ中は、普通、同じプログラムを何度も編集するので、エディタから出ずに一般のコマンドを起動できる機能はなかなか便利である。

さらに、文法のチェックのために翻訳だけを行ないたいときには、次のようにすればよい。

```
E#FORT77                                          ; FORT77 サブコマンド
FORTRAN 77 ENTERED
PROGRAM : FACT
NO ERROR
NO OBJECT PROGRAM PRODUCED
PROGRAM : F
NO ERROR
NO OBJECT PROGRAM PRODUCED
E#
```

このときはコンパイル・ユニットもロード・モジュールも作成されない。作成するには、パラメータを与える必要がある。コンパイル・ユニットやロード・モジュールの作成については、5 節(次号掲載予定)を参照されたい。

また、プログラムやコマンドの実行を中断するには「break」キィーを押せばよいが、プロンプトが ** に変わってしまうことがある。このときは、

```
《「break」キィーを押す》
**CANCEL                                    ; ** が出ずに E# になることも多い
E#
```

とすればよい。また、誤って「break キィー」を押したので、もとの続きをしたければ

```
《「break」キィーを押す》
**↵
E#
```

とすればよい。ここで、ILIB1(F1) の編集を終えよう。

```
E#E                                         ; ILIB1(F1) の編集を終わる
*
```

2.4. プログラムの編集

プログラムの修正は、SX-1 の ATSS-AF のエディタを使う方法と ファイルを ACOS-1000 へ逆転送 [5] して TSS-AF のエディタで修正し、もう一度 SX-1 に転送し直す方法とがある(もちろん、ACOS-1000 側に同じファイルがあるときは、そのファイルを修正し SX-1 に転送するだけでよい)。ここでは、ATSS-AF のエディタを使用する場合について述べよう。

最も単純な修正のしかたは、行全体を書き直すことである。これは、

```
*EDIT ILIB1(A1)
CREATED  UPDATED  REV    LINE NO-OF-BYTE LANTYPE ORIGINAL-ENTRY-NAME
86-04-20 86-04-20 000       6       230 FTFREE2 A1
E#30    10 READ(5,*,END=99) A,B
E#50    99 END
E#
```

のように入力すればよい。NEW を指定したときの行番号のプロンプトに対する入力よりスペースの数を1つ多くする必要がある。つまり、L サブコマンドで表示されるのと同じ形で入力すればよい。これは、行番号と行の内容との区切り記号として、スペースを使っているからである。したがって、手順付きの画面端末や無手順の画面端末で画面の情報を再利用できるもの(例えば、Basic の term 文で動作中のパソコンなど)では、L サブコマンドで表示したリストを直接修正できる。

もちろん、プログラムにない行番号の行を入力すると行の追加になるし、行番号のみを入力するとその行は削除される。

```
E#45        GO TO 10
```

```
E#20
E#L ^:¥
00010       PROGRAM EXAMPLE
00030    10 READ(5,*,END=99) A,B
00040       WRITE(6,*) 'A+B = ',A+B, '     A-B = ',A-B
00045       GO TO 10
00050    99 END
E#RUN
FORT77 COMPILER STARTED
EXECUTION STARTED
?1 2
A+B = 3.000000     A-B = -1.000000
?3 4
A+B = 7.000000     A-B = -1.000000
?//EOD                                          ; データの終わり(end of data)
E#
```

このエディタは、バッファに統合ライブラリのサブファイルからプログラムを読み込み、バッファ上で編集を行ない、それを統合ライブラリのサブファイルに書き出すという形で使用する。したがって、適当な時機に W サブコマンドを実行しておくか、編集の終わりに

```
E#E W                              ; サブファイルに書き出して、エディタを終わる
CREATED  UPDATED  REV    LINE NO-OF-BYTE LANTYPE ORIGINAL-ENTRY-NAME
86-04-20 86-04-20 001       6        231 FORTRAN A1
*
```

として、編集している統合ライブラリのサブファイルにバッファの内容を書き出しておく必要がある(変更をして書き出さずに、E サブコマンドを実行すると注意はしてくれる)。

2.5. データ・ファイル

プログラムの実行に必要なデータ・ファイルも、ACOS-1000 から転送する方法とエディタで作成する方法とがある。SX-1 では、データはファイルの形で保存する。

(1) データ・ファイルの転送

データ・ファイルを転送するには、まず、PREALLOC コマンドによりファイルの領域を確保しておかなければならない。

```
*PREALLOC DF1 RECFORM=VB BLOCKSZ=1000 SIZE=300 FIXTRK NODELR

A60000.DF1                                  ===  ALLOCATED  ===

*.SSTAGE DFILE1:DF1:DATA
*
```

ここで、RECFORM を VB (variable blocked record : 可変長ブロック化レコード) にしておくと、BLOCKSZ の大きさは適当な値でよい。多くのデータを入れるファイルが必要な場合には、SIZEパラメータの値を大きくする。この大きさは初期値であり、ある程度までは自動的に大きくなることは、統合ライブラリの場合と同じである。Fortran で使用するデータ・ファイルには、FIXTRK とNODELR の２つのパラメータを指定する必要がある。ファイルの中身は、list コマンドで表示できる。

```
*LIST DF1                                                    ; 型は DATA が既定値
FILE:DF1                                                          LIST REV.060
      1 ABCDEFGHIJKLMNOPQRSTUVWXYZ01234567890
      2 +-*/()!"#%&':;?,.
TOTAL INPUT RECORDS : 2
TOTAL OUTPUT RECORDS : 2
*
```

(2) データ・ファイルの作成

エディタでデータ・ファイルを作成するには、次のようにすればよい。

```
*PREALLOC DF2 RECFORM=VB BLOCKSZ=1000 SIZE=300 FIXTRK NODELR

A60000.DF2                                    ===  ALLOCATED  ===

*EDIT DF2 NEW DATA                                    ; 言語タイプの指定は DATA
000100 10
000200 5
000300 8
000400[F
E#E W
CREATED  UPDATED  REV    LINE NO-OF-BYTE LANTYPE ORIGINAL-ENTRY-NAME
                            3          7 DATA    DF2
*
```

PREALLOC をせずに直接エディタで作成すると、FIXTRK と NODELR の指定がされていないので、Fortran で読み込んだり、書き出したりできない。

(3) データ・ファイルへの入出力切り換え

さて、標準入力(standard input : SIN)から読み、標準出力(standard print : SPR)に書き出す簡単なプログラムを考えよう。標準入力は特に指定しなければ、端末のキィーボードであり、標準出力は端末のディスプレィ(またはプリンタ)である。

```
*.SSTAGE SFILE2:ILIB1(S1):SOURCE FREE2
*EDIT ILIB1(S1)
CREATED  UPDATED  REV    LINE NO-OF-BYTE LANTYPE ORIGINAL-ENTRY-NAME
86-04-20 86-04-20     0000000 0000000000 FTFREE2 S1
E#L ^:¥
```

```
00010       PROGRAM STDIO
00020       CHARACTER LINE(80)
00030    10 READ(5,100,END=99) (LINE(I),I=1,80)
00040   100 FORMAT(80A1)
00050       WRITE(6,200) (LINE(I),I=1,80)
00060   200 FORMAT(1H ,80A1)
00070       GOTO 10
00080    99 END
E#RUN
FORT77 COMPILER STARTED
EXECUTION STARTED
?HELLO, FRIENDS !
HELLO, FRIENDS !
?HOW ARE YOU ?
HOW ARE YOU ?
?//EOD                                  ; 入力データの終わり
E#
```

これの標準入力 SIN をデータ・ファイル DF1 に切り換えて (SIN を DF1 に割り当てて)実行するには、

```
E#ASSIGN SIN EFN=DF1                    ; ASSIGN はサブコマンドとして実行可能
E#RUN
FORT77 COMPILER STARTED
EXECUTION STARTED
ABCDEFGHIJKLMNOPQRSTUVWXYZ01234567890
+-*/()!"#%&':;?,.
E#
```

とすればよいし、さらに、標準出力 SPR をデータ・ファイル DF3 に切り換えて実行するには、

```
E#$PREALLOC DF3 RECFORM=VB BLOCKSZ=1000 SIZE=300 FIXTRK NODELR

A60000.DF3                                  ===  ALLOCATED  ===

E#ASSIGN SPR EFN=DF3
E#RUN
FORT77 COMPILER STARTED
EXECUTION STARTED
E#$LIST DF3 PR                              ; 印刷イメージのファイルの表示
ABCDEFGHIJKLMNOPQRSTUVWXYZ01234567890
+-*/()!"#%&':;?,.
TOTAL INPUT RECORDS : 2
TOTAL OUTPUT RECORDS : 2
E#
```

とすればよい。 SIN や SPR などはプログラムから見たときのファイル名に当たるもので、内部ファイル名(internal file name : IFN)と呼ばれる。それに対して、今まで述べてきたファイル名や

ライブラリ名は外部ファイル名(external file name : EFN)と呼ばれる (Fortran における内部ファイル名は、装置番号の 5 が SIN で、6 が SPR である。 また、その他の 1 桁の装置番号 n は FFOn に、2 桁の装置番号 mn は FFmn になっている)。そして、内部ファイル名と外部ファイル名を結び付ける働きをするのが、ASSIGN コマンドである。

さて、現在のファイルの割り当ての状態は、どうなっているだろうか。これを見るには、

```
E#ASSLIST                                    ; ASSLIST もサブコマンドとして実行可能
IFN=CENTERCD EFN=OUC.SYSTEM.CM
IFN=******** EFN=OUC.SYSTEM.CMD.DEFLIB
IFN=******** EFN=A60000.ILIB1
IFN=******** EFN=X00016638.AWORK01
IFN=******** EFN=X00016638.BWORK01
IFN=******** EFN=X00016638.FWORK01
IFN=******** EFN=A60000.DF2
IFN=SIN EFN=A60000.DF1
IFN=SPR EFN=A60000.DF3
E#
```

とすればよい。このとき、センターで特別に使用したり、コマンドが自分の処理用に割り当てているファイル(IFN=CENTERCD や IFN=********)も表示されている。

ファイルの割り当ては FREE コマンドを用いて解放できる。

```
E#FREE EFN=DF1               ; EFN で解放できる。FREE もサブコマンドとして実行可
E#FREE IFN=SPR               ; IFN でも解放できる
E#ASSLIST
IFN=CENTERCD EFN=OUC.SYSTEM.CM
IFN=******** EFN=OUC.SYSTEM.CMD.DEFLIB
IFN=******** EFN=A60000.ILIB1
IFN=******** EFN=X00016638.AWORK01
IFN=******** EFN=X00016638.BWORK01
IFN=******** EFN=X00016638.FWORK01
IFN=******** EFN=A60000.DF2
E#
```

そして、ファイルの割り当ては、FREE コマンドを用いて解放し、ASSIGN コマンドを用いて新たに割り当てるまで変わらない。

また、内部ファイル FF10 を端末に割り当てるには、

```
E#ASSIGN FF10 FILESTAT=TERMINAL
E#
```

とすればよい。

なお、Fortran におけるファイルの割り当ての詳細については、文献 [3]、[4] を参照されたい。

2.6. 使用終了

SX-1 の ATSS-AF を終了して、ACOS-1000 の TSS-AF に戻るには、プロンプトが * のときに、DONE を入力すればよい。

```
E*E                                                    ; エディタを終わる
*DONE
SYSTEM ?
```

これで、ACOS-1000 の ATSS-AF に戻った。

以上、SX-1 のタイム・シェアリング・システム ATSS-AF の基本的な使い方について述べた。
次号掲載予定の「その２」では、

３．ライブラリとファイルの管理
４．行エディタ
５．コンパイル・ユニットとロード・モジュールの保存
６．その他

について、説明する予定である。

[参考文献]

1. 渡辺、近藤、端山、大中、藤井（1986）：「スーパーコンピュータ SX-1 の概要（1） － アーキテクチャを中心に －」、大阪大学 大型計算機センター ニュース、Vol.15、No.4（1986 年 2 月号、第 60 号）、pp.109-129。
2. 藤井（1986）：「スーパーコンピュータ SX-1 の概要（2） － 運用を中心に －」、大阪大学大型計算機センター ニュース、Vol.15、No.4（1986 年 2 月号、第 60 号）、pp.131-136。
3. 大中、後藤（1986）：「SX Fortran77 概説（1） － 文法を中心として －」、大阪大学 大型計算機センター ニュース、Vol.16、No.1（1986 年 5 月号、第 61 号）。
4. 後藤、大中（1986）：「SX Fortran77 概説（2） － 利用法を中心として －」、大阪大学 大型計算機センター ニュース、Vol.16、No.1（1986 年 5 月号、第 61 号）。
5. 多喜（1986）：「ACOS-1000 と SX-1 とのファイル転送について」、大阪大学 大型計算機センター ニュース、Vol.16、No.1（1986 年 5 月号、第 61 号）。
6. 日本電気(1986)：MSF-6 利用説明書、SX ソフトウェア GJF11-1。
7. 三原、山本、後藤（1986；掲載予定）：「スーパーコンピュータ SX-1 の性能向上支援ツールの概要」、大阪大学 大型計算機センター ニュース、Vol.16、No.2（1986 年 8 月号、第 62号）。
8. 日本電気(1986)：ATSS 画面エディタ説明書、SX ソフトウェア GED12-1。