MSP/Fortran 使用法

講習会資料

1996 年 5 月 17 日

九州大学大型計算機センター

MSP/Fortran 利用法 ’94 *

肥田木 直子 † 渡部 善隆 ‡

本稿は九州大学大型計算機センターで Fortran プログラムを MSP で実行させるための基本的なことがらを述べています．表題にある“MSP/Fortran”とは，富士通株式会社提供の OS IV/MSP Fortran77 EX システムと OS IV/MSP Fortran77 EX/VP システムの総称を意味します．

本文は次の条件が満たされていることを暗に利用者に要求して書かれています．

- Fortran の文法を知っていること．
- PFD が利用できる端末環境であること．
- MSP のエディタを使えること．

最初の条件はとても大事です．MSP 上で動作する Fortran コンパイラの Fortran77 EX と Fortran77 EX/VP は，双方一般の Fortran77 の拡張であり，正しい文法で記述されている限り，他の Fortran 言語と互換性が保証されています．ただし，あくまでも「正しい文法」で記述されている場合に限ります．Fortran77 EX は Fortran77 に比べて文法チェックが厳しいので，別の Fortran コンパイラではエラーなしして通ったプログラムが Fortran77 EX ではエラーを出して引っかかる場合がありますので御注意下さい．Fortran77 EX に関する詳細は [5], [6], [7] を参照下さい．

二番目の条件は，本文の説明で PFD を用いて説明を行なう箇所がたくさんあるためです．ただし，フルスクリーン環境がサポートされていない場合でも Fortran の実行は可能です．

三番目の条件は，本文をお読みの方にデータセットを編集する MSP 上のエディタの知識があることです．MSP のエディタは `EDIT` コマンドで起動されるエディタと，`ED` コマンドで起動される PFD のフルスクリーンエディタとがあります．フルスクリーン環境がある場合は，PFD/EDIT を利用するのが断然便利です．PFD/EDIT は mifes, red や vi, emacs などのパソコン，ワークステーションのエディタと比べれば少し遜色するかもしれませんが，大変に優れたエディタです．文献 [11], [12], [13] を片手に，一日パチパチ端末を叩いていれば，最低限必要な機能はすぐに覚えられるはずです．

また，MSP の TSS コマンドのリファレンスマニュアルとして [14] が，本文では説明を省略した JOB 文，DD 文等を解説したバッチ処理のマニュアルとして [10] がそれぞれセンターより発行されています．各利用の手引は，各地の連絡所を通じて九州大学大型計算機センターの共同利用掛まで申し込むことで入手可能です．

なお，本稿は広報 Vol.26, No.3 の解説記事に 1993 年 4 月に開催された Fortran 使用法講習会の資料を追加し，更に全面的な改訂を加えたものです．

---

*1994 年 4 月 15 日受理

†九州大学大型計算機センター・システム管理掛
‡九州大学大型計算機センター・研究開発部

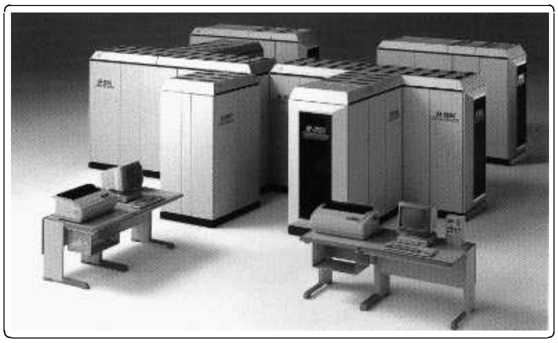
目次

1 MSP/Fortranの使用可能な計算機環境

九州大学大型計算機センターには，主計算機として汎用計算機 Fujitsu M1800/20U と，ベクトル計算機 Fujitsu VP2600/10 が導入されている．M1800，VP2600 には各々 OS として MSP(Multidimensional System Products), UXP(UNIX Product) がサポートされている．Unix OS である UXP での Fortran プログラムの実行方法は [1], [2], [3] を参照されたい．MSP での Fortran プログラムの実行は，M1800 はバッチ及び TSS 処理で，VP2600 はバッチ処理のみで行なわれる．

1.1 汎用機とベクトル計算機の違い

汎用計算機 Fujitsu M1800/20U は，多目的用に設計された一般的な計算機である．英語で書けば"general purpose computer"．M1800 上では，言語処理プログラム Fortran, C, Pascal, Cobol などの他にも，様々なアプリケーション・ソフトウェア（SAS, SPSSX, TEX, xv, etc.）が動作する．



汎用計算機 Fujitsu M1800/20U

ベクトル計算機 Fujitsu VP2600/10 は，科学技術計算の分野で比重の大きい Fortran 配列に関する演算の高速処理に適した計算機．VP2600 上で動作する言語処理プログラムは，VP 用の Fortran および UXP では C も利用可能（[3]）．VP 上でのアプリケーション・プログラムとしては構造解析ソフトウェア MARC が代表的である．

ベクトル計算機 Fujitsu VP2600/10

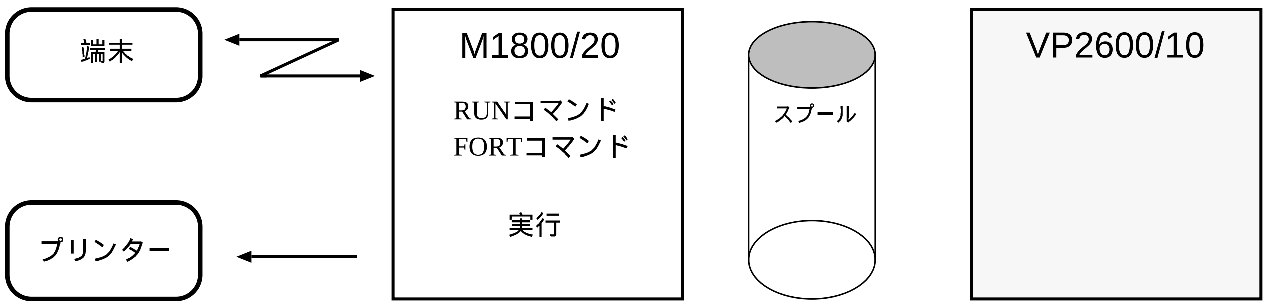
VP2600 は，ベクトルプロセッサ (Vector Processor)，通称"VP"と呼ばれ，文字通り DO ループを並列にベクトル処理することで演算の高速化を追求する計算機．そのため，利用者がベクトル化率（VU 率）の高い Fortran プログラムを書けば，そのまま処理が高速に行なわれ，課金が安く済む．逆に言うと，お金と時間を節約するためには高速にベクトル処理されうるプログラミング技術が必要ということである．計算機科学の世界は「このようにプログラムをチューニングすればベクトル化が可能だよ」といった種類の論文で賑わっている．一方で，「論理的に同値なのだからそんなことは計算機の方で勝手にやってくれ」との意見もあり，これももっともな意見である．更に，世の中の大型計算機の流れは"並列化"に向かっている様で，processor を横に 128 個も 256 個も並べた"並列計算機"が既に登場している．しかしながら，横にずらっと並んだ processor に如何にして資源を割り振り，高速化するかは，なかなか難しい問題であり"自動並列化コンパイラ"の開発に各研究機関が現在躍起になっている（そうである）．つまり，読者のあなたが，もし black box 的な高速並列化ソフトを開発すれば，大いに功成り名を遂げる可能性が高い．論文のネタを捜している方は是非"並列化プログラミング"をやってみられることをお勧めする．

1.2 TSS & バッチ

プログラムを計算機で実行させる処理形態には TSS(Time Sharing System) とバッチ (batch) の 2 つの方法がある．

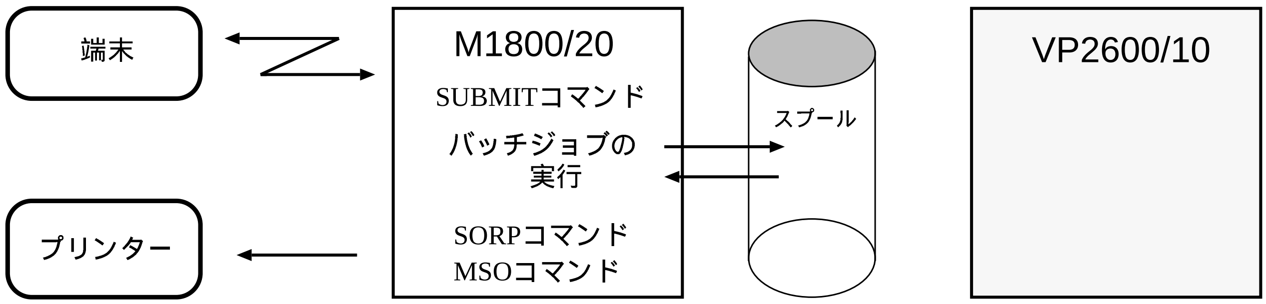
TSS はコマンドを用いて計算機と会話的にやりとりをする方法で，小規模なプログラムの実行や，編集中のプログラムの動作確認に適している．また，PFD(Programing Facility for Display user) の editor によって作成中のプログラムを，editor 中から直接実行することもできる．ただし，確保できる region size は最大 50MB までであり，それ以上の region size が必要な場合は，バッチ処理になる．

TSSの実行 (M1800/20)



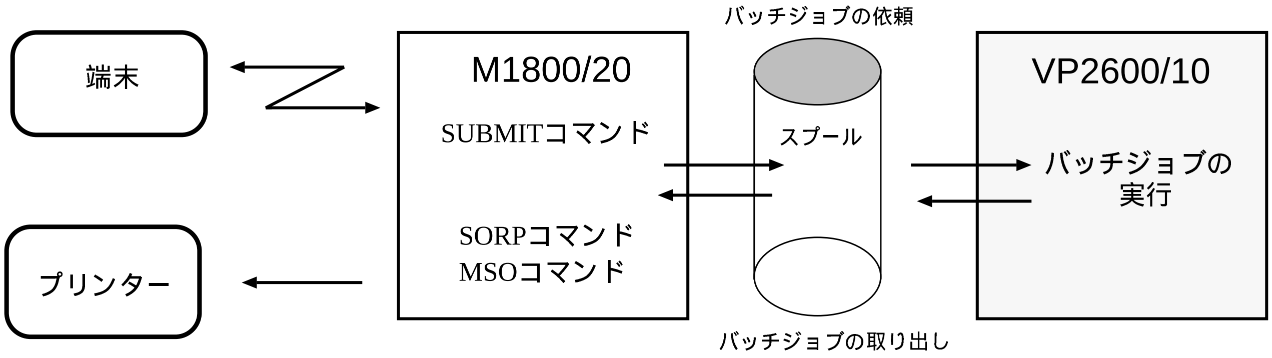
バッチは，実行に必要な情報を一括して計算を依頼する処理方法．TSS セッションから切り離してバックグラウンドで処理するので，実行中でもセッション内で他の作業が可能であり，バッチを依頼したままでセッションを終了することもできる．そのため，実行時間の長い大容量処理などに適している．

バッチジョブの実行 (M1800/20)



ベクトル計算機 VP2600 での Fortran プログラムの実行はバッチ処理のみであり，TSS 処理による Fortran プログラムの実行は出来ない．また，利用者による VP2600 へのジョブの依頼は，全て M1800 を通じて行なわれ，利用者は VP2600 と直接会話することは出来ない．

バッチジョブの実行 (VP2600/10)



1.3 確保出来る region size

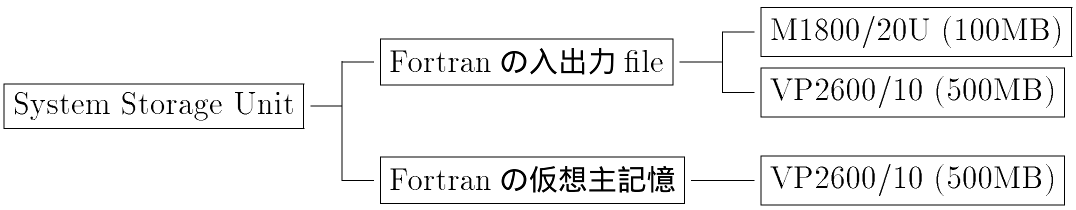
利用者は，拡張アドレスモード (AE モード) の指定を行なうことで，一般のパーソナルコンピュータ，ワークステーションでは通常確保出来ない大容量の領域を確保することが可能である．記憶域の最大値は次の通り．

処理形態	計算機	最大記憶域
TSS	M1800/20U	50MB
バッチ	M1800/20U	200MB
	VP2600/10	300MB

SSU の利用

上の region size は，主記憶 (memory) に確保できる大きさである．この主記憶とは別に，SSU と呼ばれるシステム記憶装置を Fortran の配列として割り付けることも可能．

SSU(System Storage Unit) は，主記憶と disk との中間に位置する記憶装置．配列データとして SSU に割り当てることで，仮想主記憶として SSU を最大 500MB 使用できる．また，入出力ファイル (VIO/F) として SSU を割り当てると，ディスクに割り当てる場合よりも高速の処理が行なわれる．また，ファイルアクセス回数 (EXCP) が軽減されることで，課金も安く済む． SSU の Fortran 配列としての利用は， VP2600 のみである． SSU の一時データセットへの割り付けは M1800， VP2600 双方使用可能．なお SSU の使用方法は，この記事の守備範囲を逸脱しているので，ここでは紹介のみにとどめる．参考文献 [4] を参照されたい．確保できる SSU の最大値は， M1800 で 100MB, VP2600 で 500MB．



SSU の計算機別用途

ただし， VP2600 での SSU を入出力ファイルと仮想主記憶に同時に使用する場合は，合計が 500MB 以下になるように設定すること．

TSS での AE モードの指定

TSS の標準 region size は 10MB である．TSS 処理で拡張アドレスモードを使用するためには，セッションを開く LOGON 時に SIZE パラメータの指定が必要．

標準モード	$\Longrightarrow$	拡張モード
10MB	size parameter	50MB

以下は，利用者 A79999A が LOGON 時に拡張領域を 50MB 確保する例．
LOGON TSS A79999A に続けて S(50) を追加する．
なお，以下本稿では ↵ 記号を Return キーまたは Enter キーの入力とする．

```
JCET010 SYSTEM READY
LOGON TSS A79999A S(50)  ↵                              <---- MSP に LOGON
+ PASSWORD ?
       ↵                                                <---- password の入力
 KDS40613I USER(A79999A) LAST ACCESS DATE(1994.01.17),TIME(20:45:43)
 KEQ56455I A79999A LOGON IN PROGRESS AT 20:46:14 ON JANUARY 17, 1994
 JOB NO = TSU8632 CN(01)
 KEQ56951I NO BROADCAST MESSAGES
 THIS SESSION IS AE MODE. REGION SIZE IS  50MB.  <---- AE mode での LOGON
```

S(50) は最大値 50MB を示す．この値は必要に応じて 7 ~ 50MB を指定する．SIZE パラメータを指定しないで LOGON する場合は標準値 6MB となる．

また，既に標準モードで LOGON している状態ならば，READY モードから次のように再 LOGON 可能．

```
 READY
LOGON A79999A S(50)  ↵
  RETURN CODE : 0000
 CPU TIME(   0.21SEC.) USE TIME(   1MIN.) REGION SIZE(6144KB)
 INPUT(     1LINES) OUTPUT(      25LINES) EXCP(      22TIMES)
 SESSION CHARGE(TSU7034,11:19:55)                             7YEN
 TOTAL CHARGE SINCE 04/01/93 (EXCEPT THIS SESSION'S)       6,858YEN
 KEQ56470I A79999A LOGGED OFF AT 11:20:06 ON MARCH 3, 1994+
 + PASSWORD ?
```

この後 Fortran 実行時に AE オペランドを指定することで拡張モードでの実行が行なわれる．バッチ処理では，ジョブ制御文の CLASS パラメータに拡張クラスを指定することで AE モードが使用できる．

バッチ処理の依頼は TSS セッションの中から SUBMIT コマンドで行なう．これにより処理されるバッチジョブは FIB ジョブ (Foreground Initiated Background) と呼ばれ，TSS のセッション中の他のコマンドとは無関係に処理される．セッション中にバッチ処理が終了すれば，MSO コマンド，SORP コマンド，あるいは PFD のオプション 3.8 でその結果を見ることができる．バッチ処理中にセッションを終了させても，バッチ処理は続行される．

1.4 region size の算出方法

実行したい Fortran プログラムが，どの位の region size を必要とするかは，定義した配列の大きさから電卓を用いて簡単に概算できる．

以下のことを知っていればよい．

1 変数の必要とする byte 数は

変数		byte 数
単精度実数	real	4
倍精度実数	double precision	8
4 倍精度実数	quadruple precision	16
単精度複素数	complex	8
倍精度複素数	double precision complex	16
4 倍精度複素数	quadruple precision complex	32

あとは

$$
\begin{aligned}
1\mathrm{MB} &= 2^{10}\mathrm{KB} \\
&= 2^{20}\mathrm{byte} \\
&= 1,048,576\mathrm{byte}
\end{aligned}
$$

を使って計算する．

例として次のプログラムをあげる．

```
      PROGRAM TEST1
       PARAMETER(NP=4096)
       REAL*8 MX(NP,NP),X(3),Y(2)
       INTEGER IN,N
             :
```

プログラムの中で最も大きい配列 MX は 4096×4096 の倍精度 2 次元配列である．倍精度変数 1 つは 8 バイトである．従って，次の計算によって配列 MX のとる領域の大きさが分かる．単精度変数の場合は 1 変数 4 バイトで計算すればよい．

$$
\begin{aligned}
\text{配列 MX の大きさ} &= \text{変数の合計} \times 8\ \mathrm{byte} \\
&= 4,096 \times 4,096 \times 8\ \mathrm{byte} \\
&= 134,217,728\ \mathrm{byte} \\
&= 134,217,728/1,024/1,024\ \mathrm{MB} \\
&= 128\ \mathrm{MB}
\end{aligned}
$$

ここで，$1\mathrm{MB}=1024\mathrm{KB}=1024^2\mathrm{byte}$ を用いている．

逆に配列 MX に 200MB の領域を確保する場合の NP の値も，$\mathtt{NP}^2 \times 8 = 200 \times 1024^2$ の解として $\mathtt{NP} = 5120$ がおおよその目安となる．

1.5 ジョブクラスと制限値

九州大学大型計算機センターでは，利用者の処理するプログラムの規模に応じてジョブクラスを設定している．ジョブクラスは M1800，VP2600 で設定が異なる．利用者は自分の処理するプログラムの規模に応じ自身でジョブクラスを陽に指定する必要がある．以下，1994 年 4 月現在のジョブクラスと制限値の一覧をあげる．表中，下線部はパラメータを指定しない場合の省略値を表す．

	バッチジョブ				TSS	
	標準		MT	AE ジョブ		
ジョブクラス	A	B	N	F	標準	AE
CPU 時間 (分)	10	180	10	180	60	
ファイルアクセス (万回)	20	50	20	50	無制限	
リージョンサイズ (MB)	10			200	10	50
端末接続時間 (分)	—				1435	

M1800/20U

	ＡＥジョブ			
ジョブクラス	A	B	V	
CPU 時間 (分)	10	180	10	180
ファイルアクセス (万回)	20	50		
リージョンサイズ (MB)	50		300 (SSU 500)	

VP2600/10

上の表が何を意味しているのかさっぱり分からない人のための解説

ジョブクラスと制限値が切実に効いてくるのは，バッチ処理の世界である．バッチ処理の説明が出てくるのはまだ先なので，以下の説明を見てもさらに分からなくなる恐れがあるが，“なんとなく理解した”気になって，構わず先に進まれたい．バッチ処理を少し試みられたあとで，この部分を改めて読めば，他愛のない内容だったことに気づくはずである．

“MT”とは何か？

Magnetic Tape の略．磁気テープの処理に使用するジョブクラス．利用者が磁気テープを使用する場合，ジョブクラス N を指定する． MSP の場合，磁気テープの使用はバッチで行なう．センター 2 階の MT 処理装置を利用して，センターの課題を持つ人は誰でも MT を扱える．もちろん，磁気テープは各自で用意すること．詳細は [8] を参照．

制限値を越えたジョブはどうなるのか？

CPU 時間を越えたジョブは，越えた時点で強制的にキャンセルされる．例えば，TSS で Fortran プログラムを実行する場合，セッション開設以来の CPU 時間の合計が 3600 秒に達すると，システムから強制的に LOGOFF される．バッチ処理も同じく，実行中に CPU 時間またはファイルアクセス回数が制限値を越えた場合，それより先の処理を打ち切り，それまでの処理結果を利用者に通知する．

何故ジョブクラスの中に CPU 時間の分類があるのか？

CPU 時間は，ジョブ制御文 (Job Control Language) と呼ばれるバッチ処理を記述した手続きの TIME パラメータに，例えば `TIME=10` などとして利用者が指定する．特に TIME パラメータに何も指定しなければ，下線の省略値が設定される．ジョブクラスの中に CPU 時間の分類がある理由は，TIME パラメータとして CPU 時間をもう一方の値 以下 に設定すると，ジョブが優先的に処理されるためである．VP2600 の V ジョブでは，`TIME=10` と指定することで，指定なしの他の V ジョブより，処理が優先される．その代わり，CPU 時間が 10 分を越えてしまった場合，実行はキャンセルされるので，設定には十分注意されたい．また，プログラムの暴走や無限ループによる CPU 時間の浪費を防ぐため，陽に TIME パラメータを指定することも効果的であるが，この場合 TIME パラメータ以上の CPU 時間は使用できないことにくれぐれも注意願いたい．

“ファイルアクセス回数”とは何か？

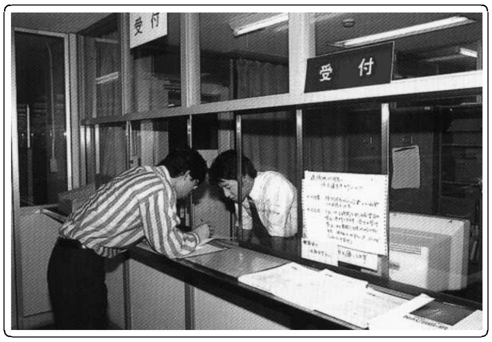
MSP のデータセットでは，一度に読み書きできる最小単位をレコードと呼ぶ．Fortran では，ひとつの“行”のことだと思ってよい．レコードが幾つか集まったものをブロックと呼ぶ．ファイルアクセス回数とは，ディスクに記録されている 1 ブロックを buffer に何回読みこんだかを指す．従って，外部データセットと read, write でデータのやりとりを頻繁に行なう Fortran プログラムはファイルアクセス回数が多くなる．

“端末接続時間”とは何か？

TSS 特有の制限値．要するに，セッション開始から 1435 分経過すると強制的に LOGOFF されることを意味する．強制 LOGOFF の前にはその旨のメッセージが表示されるので，その場合，直ちに処理中の仕事を終らせて，一度 LOGOFF する必要がある．

1.6 制限値を越えるジョブの処理はどうするか？

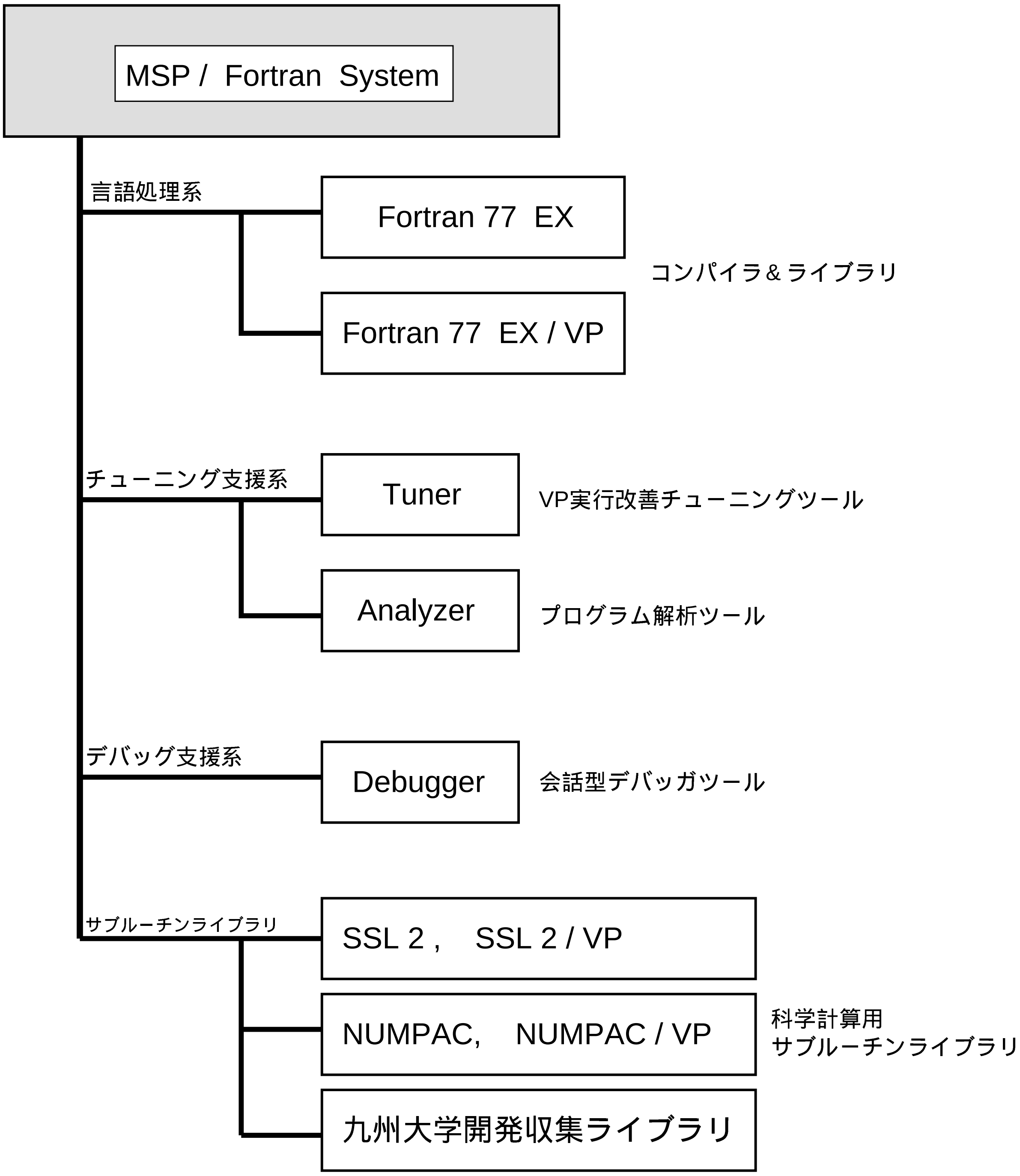
ジョブグラスと制限値で定められた制限値を越えるジョブは「要審査ジョブ」としてセンターに処理を依頼すると，審査を経て処理がされる (D ジョブ)．提出方法は，センター二階受付横の所定の用紙に必要事項を記入し，受付に提出する．また遠隔地の利用者は，共同利用掛を通して依頼する．最大リージョンサイズは VP で 400MB．



【九州大学大型計算機センター 2 階受付】
機器のトラブル時の相談や，予約端末，要審査ジョブの受付を行なう．センターでは，新入でも容赦なく受付業務をこなさせられるので，時には対応の具合が少し鈍い場合もありうるが，大きな心で見守って下さい．逆に受付に来る利用者も“品定め”の対象になっていることをお忘れなく．

2　MSP/Fortran の概要

MSP/Fortran システムは言語処理系，チューニング支援系，デバッグ支援系，及びサブルーチンパッケージで構成されている．



以下，各項目について概要を述べる．

2.1 Fortran77 EX, Fortran77 EX/VP

Fortran77 EX は M1800 用のオブジェクトモジュールを生成し，処理を行なうコンパイラ／ライブラリ．Fortran77 EX/VP は VP2600 用のオブジェクトモジュールを生成し，処理を行なうコンパイラ／ライブラリである．

Fortran77 EX, Fortran77 EX/VP の主な特徴をあげる．

- "!"によって行の途中から注釈が書ける
- 暗黙の型宣言を抑止できる
- 英小文字，アンダースコアー (_) が使用できる
- 変数，定数名の最大長は 31 文字 (外部名は先頭 7 文字だけ有効)

以上は新しい言語規格 Fortran90 への先行採用である．

プログラム例をあげる．

```
      program main
       implicit none
       real*8 a,x(10),b/0.0/         ! This is comment
       real*8 interval_vector(10,10) ! interval vector
       write(6,*) b
      end
```

また，一文の継続行数が 99 行まで可能であり．その他にも最新の最適化技術，ベクトル化技術の導入，システム記憶 (SSU) への対応，などをおこなっている．詳細は [7] を参照．

2.2 チューニング支援系

ベクトル計算機 (VP) での Fortran プログラムの実行では，VP 向けのプログラムを作成することで，処理速度の高速化が可能となる．そのためには，CPU 時間が集中的に消費されるプログラムの箇所を特定し，ベクトル化のための改良 (チューニング) を行なうことが必要である．

プログラムの動的な振舞いを解析して，ベクトル化のための情報を提供するツールとして Tuner および Analyzer が提供されている．Analyzer によって解析された情報をもとに Tuner でベクトル化のガイダンス，プログラムの解析を行なう．Tuner から Analyzer を起動することも可能であるが，実行時間が大幅にかかる場合や，プログラムの規模が大き過ぎて実行出来ない場合が多いため，あらかじめバッチ処理で Analyzer を実行させ，解析情報を収集するチューニングが望ましい．機能の解説は [9], [10], [14] を参照．

2.3 デバッグ支援系

Fortran プログラムのバグを取り除く作業の支援として，TSS で会話的にデバッグを行なうツール Debugger が提供されている．Debugger はあらかじめデバッグ用の情報を付加したロードモジュールを作成する必要がある．また `FORT` コマンドに `GODBG` オプションを指定することで editor 内から起動することも出来る．機能の解説は [15] を参照．

2.4 サブルーチンライブラリ

科学計算用のサブルーチンライブラリとして，富士通提供の SSL II, SSL II/VP，名古屋大学大型計算機センター提供の NUMPAC, NUMPAC/VP がサポートされている．また，九州大学で開発収集したライブラリも利用できる．

ざっと次のことができる．

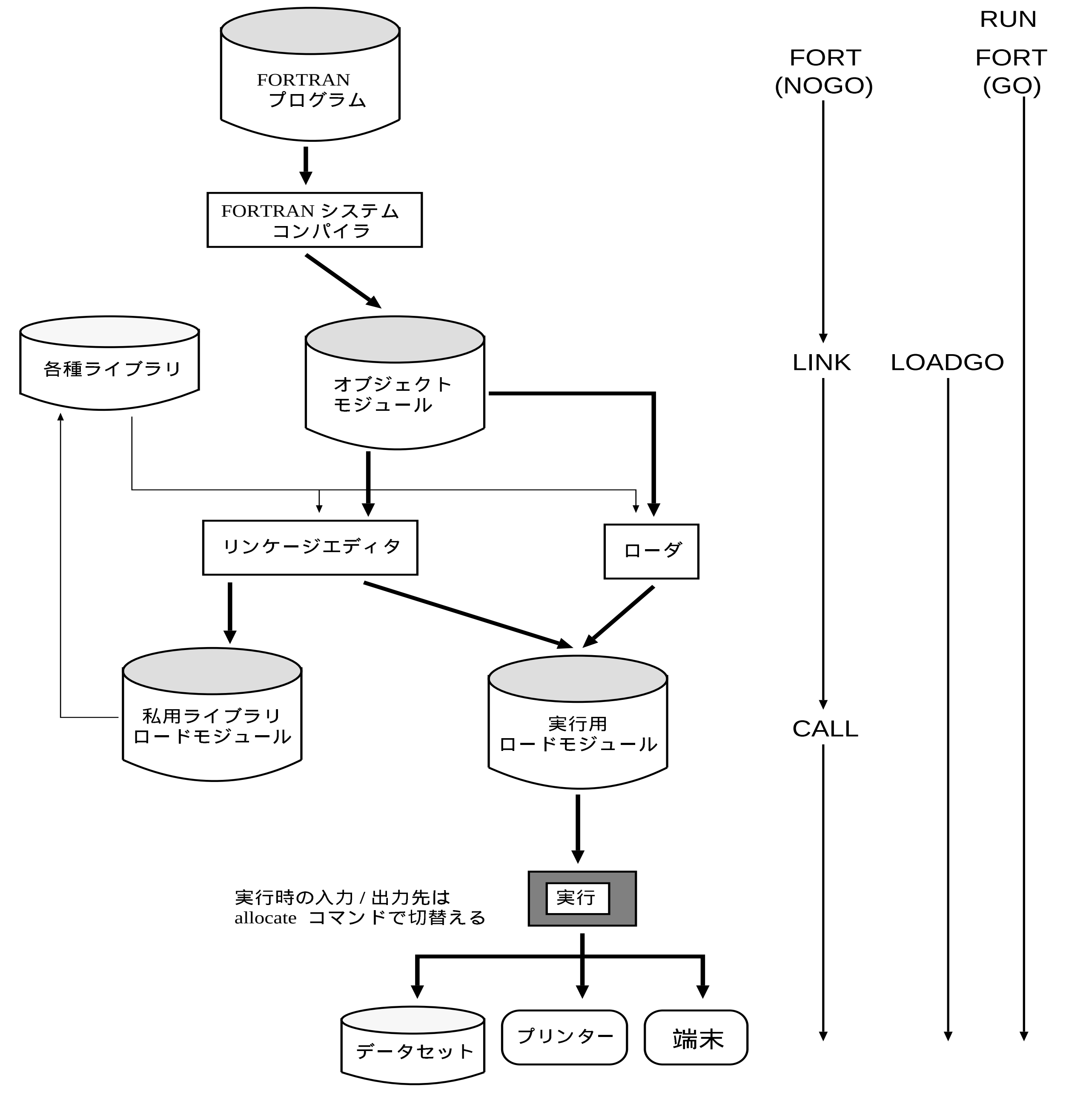
- 行列格納モードの変換
  — 一般行列と対称，対称バンド行列との変換
- 行列操作
  — 行列の和・積・差．行列とベクトルの積
- 連立一次方程式の解法
  — Cholesky 法， Bunch 法，共役勾配法， SOR 法， Gauss 消去法，解の反復改良, etc.
- 行列の三角分解，逆行列
  — LU 分解， LDL^T 分解， MDM^T 分解，および逆行列の解法．
- 最小二乗解
  — 特異値分解法， Householder 法
- 固有値，固有ベクトル
  — Householder-Bisection 法， QR 法， Jennings 法，準直接法, etc.
- 非線形計算
  — 高次代数方程式，実超越方程式，複素超越方程式，連立非線形方程式
- 極値問題
  — 多変数関数の極小化，関数自乗和の極小化，線形計画問題，非線形計画問題
- 補間，平滑化
  — B-spline 補間， Hermite 補間， C^k 級補間
- 級数，変換
  — cosine 級数展開， sine 級数展開， Chebyshev 級数, Laplace 変換, Fourier 変換
- 数値微積分
  — B-spline 補間による数値微分，二重指数関数型積分公式，多次元積分
- 微分方程式
  — 連立常微分方程式， Runge-Kutta 法, etc.
- 特殊関数
  — Fresnel 積分， Gamma 関数， β 関数， Jacobian 楕円関数, etc.
- 疑似乱数

サブルーチンの内容，機能は [16], [17], [18] を参照．

3　Fortran プログラムの TSS 処理の流れ

3.1　流れを図でみてみよう

Fortran 言語で書かれたプログラムは，TSS 処理で下図のように翻訳・結合編集・実行という三つの処理過程を通り，それぞれの手順に従った結果が得られる．



Fortran 実行流れ図

3.2 処理過程の組合せと方法

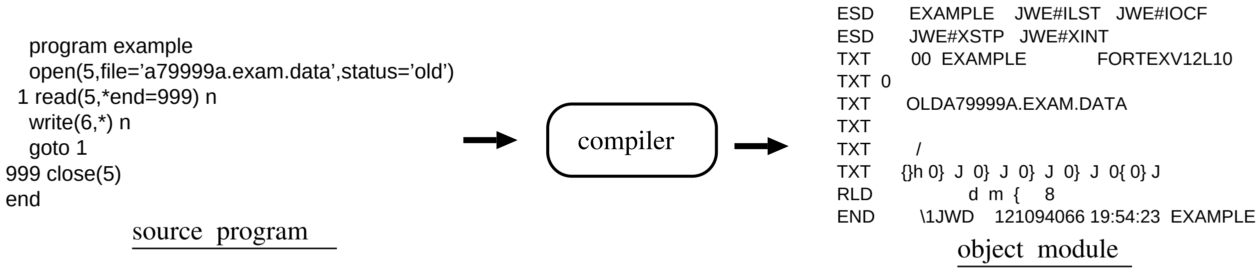
流れ図で示した各処理過程の組み合わせと，MSP コマンドの一覧を目的に応じて述べる．

処理過程の組合せ	方法	コマンド
翻訳 (compilation)	Fortran コンパイラをオプション NOGO を指定して起動	FORT
結合編集 (link/edit)	リンケージエディタを起動	LINK
実行 (execution)	ロードモジュールを実行	CALL
翻訳–結合編集	step1：Fortran コンパイラをオプション NOGO を指定して起動	FORT
	step2：リンケージエディタを起動	LINK
翻訳–結合編集–実行		
方法 a	step1：Fortran コンパイラをオプション NOGO を指定して起動	FORT
	step2：リンケージエディタを起動	LINK
	step3：ロードモジュールを実行	CALL
方法 b	翻訳の後ローダを起動し結合編集と実行を同時に行う	FORT
方法 c	RUN コマンドを用いる	RUN
結合編集–実行		
方法 a	step1：リンケージエディタを起動	LINK
	step2：ロードモジュールを実行	CALL
方法 b	ローダを起動	LOADGO
デバッグ	会話型デバッガによる実行	DEB
プログラム解析	チューナによる実行	TUNER
私用ライブラリ作成	step1：Fortran コンパイラを OBJ, NAME を指定して起動	FORT
	step2：リンケージエディタをオプション NCAL を指定して起動	LINK

3.3 用語集

コンパイラ (compiler)

Fortran で書かれたプログラムを機械語の形に翻訳するプログラム．



オブジェクトモジュール (object module)

コンパイラで機械語に翻訳された結果のプログラム．

リンケージエディタ (linkage editor)

オブジェクトモジュールの段階はまだ実行できる形にはなっていない．リンケージエディタは，実行に必要なライブラリを呼び出し，組み込んで実行可能なロードモジュールにするプログラム．

ローダ (**loader**)

オブジェクトプログラムを読み込み，編集を行い，さらにそのプログラムを実行させるプログラム．リンケージエディタと違いロードモジュールをライブラリとして出力したり，オーバーレイ・プログラムの作成，INCLUDE 文による追加入力などの機能がない．

実行可能なロードモジュール (**load module**)

リンケージエディタで処理した結果得られるもので，すぐにでも実行できる形式．

私用ライブラリとしてのロードモジュール

システムライブラリとして FORTLIB[1], SSL II[2], 九大作成ライブラリ等が用意されている．それ以外に個人として必要なサブルーチンや関数を，ライブラリとして作成・利用することができる．

サブルーチン (**subroutine**)

プログラムの中で何度も同じことを行ったり，特定の機能を持つ部分を集約して，いつでも参照実行できる形にしたプログラム．利用者が必要に応じて自由に作成するが，メーカー提供サブルーチン，センター開発収集サブルーチン等もある．

スプール (**spool**)

データ処理システムの入出力処理とジョブ実行を並行処理させ，効率的な運用を行なう中間装置．

順データセット (**sequential data set**)

データセット内のレコード (行) が順番に並んでいるデータセット．

区分データセット (**partitioned data set**)

データセット内をいくつかの区域に分割して，各区域が順データセットとして利用できる形式のデータセット．ディレクトリのようなもの．

オペランド (**operand**)

コマンドが必要とする情報のこと．

DD 名 (**data definition name**)

ジョブで使用するデータセットに対して与えられる名前．一般にシステムや処理プログラム固有の名前が決められている．システムはこの DD 名を見ることでデータセットをアクセスするための情報を得て処理プログラムとデータセットとの結合を行なう．

---

[1] sin, log などの基本的関数の入ったライブラリ

[2] 科学技術計算用サブルーチンライブラリ群

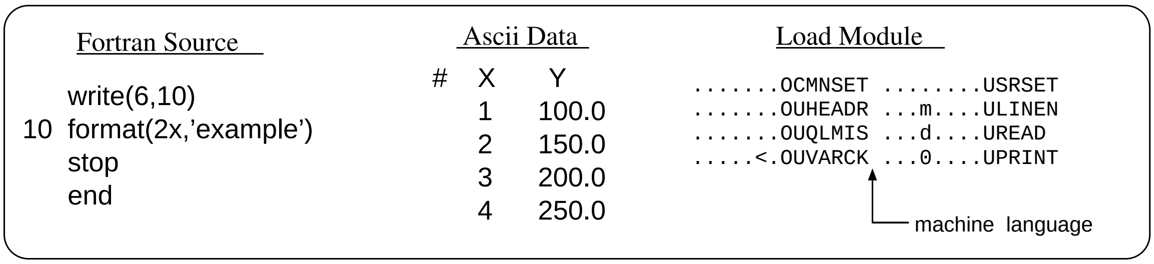
4　データセット概説

OS としての MSP の大きな特徴の一つは“データの管理が行き届いてる”ことである．しかし，裏を返せば“データの管理方法がややこしい”わけで，“固定長ブロック化レコード形式”とか，“順編成データセット”とかの MSP 特有の方言が乱れ飛ぶデータ管理は，リダイレクション (<, >) 機能に慣れた Unix 愛好者にとっては頭を抱える状況である．

そこで，本節では MSP のデータセットについて，実際使いこなす観点から，簡単な説明を行なう．

4.1　データセットとは

データセット (data set) とは，ソースプログラムや実行形式のプログラム，データや文書などの情報を集合として格納する入れもののこと．Unix や MS-DOS の“ファイル”に対応する．利用者の Fortran のソースプログラムや SSL II のサブルーチンライブラリも立派なデータセットである．



データセットのいろいろ

4.2　データセット名

各データセットには，当然固有の名前が付与され，個別に管理保存される．

4.2.1　完全データセット名

データセット名は次で構成される．

データセット名 = プロジェクト名 + ライブラリ名 + タイプ名

具体的には，次のようになる．

$$\text{'}\underbrace{\texttt{A79999A}}_{\text{project name}}.\underbrace{\texttt{EXAMPLE}}_{\text{library name}}.\underbrace{\texttt{FORT}}_{\text{type name}}\text{'}$$

プロジェクト名は利用者 ID であり，タイプ名はデータセットの内容 (Fortran プログラムであるか，テキストデータであるかなど) を示す．利用者 ID が `A79999A` の人のプロジェクト名は無条件で `A79999A` となる．利用者が自分の所有するデータセットを取り扱う場合，プロジェクト名を意識する必要はないが，共同研究などで他の利用者とデータセットを共有する場合には意識する必要がある．

たとえば，`A79999A` さんの立場から見れば `TEST.FORT` という名前のデータセットを取り扱っているつもりでも，正式には`'A79999A.TEST.FORT'` というデータセットを取り扱っていることになる．利用者 ID を含めて，さらに引用符 (') で囲った`'A79999A.TEST.FORT'` を，完全データセット名 という．

4.2.2 利用者がつけることのできるデータセット名

利用者のデータセット名は，強制的に頭にプロジェクト名がつく．また，末尾には原則タイプ名をつける．実際はタイプ名は指定しなくてもいいが，後々不便になる．プロジェクト名とタイプ名に挟まれたライブラリ名は利用者が自由に指定できる．データセットを名付ける場合の規則は以下の通り．

- データセットの文字列の合計は 44 文字以内
- 英字で始まる 8 文字以内の文字列をピリオド (.) でつなぐ．

合計文字数はプロジェクト名を含める．A79999A さんがデータセットを作る場合，次の様になる．

○	TEST.FORT	
○	TEST.EX2.FORT	
○	SAS.GRAPH.SAMPLE.DATA	
×	EXAMPLE100.FORT	文字列が 8 文字を越えている
×	TEST.1X.TEXT	数字で始まる文字列がある

もちろん，上の○の部分を完全データセットで表せば，次の様になる．

```
'A79999A.TEST.FORT'
'A79999A.TEST.EX2.FORT'
'A79999A.SAS.GRAPH.SAMPLE.DATA'
```

4.3 レコード形式

次に MSP のデータセットの保存形態について概説する．

4.3.1 トラック & シリンダ

データセットを構成する基本単位をトラック (track) と呼ぶ．トラックの大きさは，

$$1 \text{ トラック} = 47KB.$$

この等式は，データセットの大きさを計算するのに役立つ．

トラックの集合体をシリンダ (cylinder) と呼ぶ．

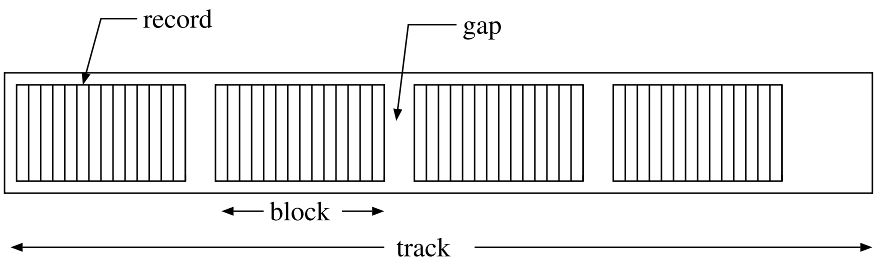
$$1 \text{ シリンダ} = 15 \text{ トラック} = 705KB$$

4.3.2 ブロック & レコード

データセットは，トラックの中に，ブロック (block) という単位に分割されて保存される．Fortran プログラムで read, write を行なう場合，このブロック毎にバッファに読み込んで処理を行なう．

さらに，ブロックは，レコード (record) と呼ばれる単位に分割できる．レコードは，“行” のことである．

図にすると次の様なイメージとなる．



ここまでの包含関係を書くと

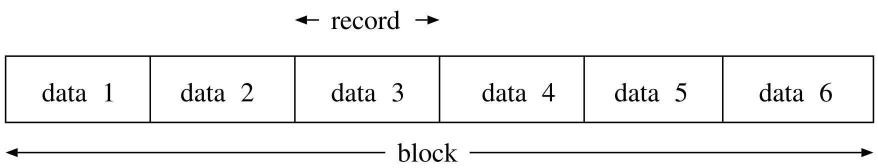
$$\boxed{\text{レコード}} \subset \boxed{\text{ブロック}} \subset \boxed{\text{トラック}} \subset \boxed{\text{シリンダ}}$$

となる．

MSP のデータセットは，レコードの保存方法によって，3 つの形式を持つ．

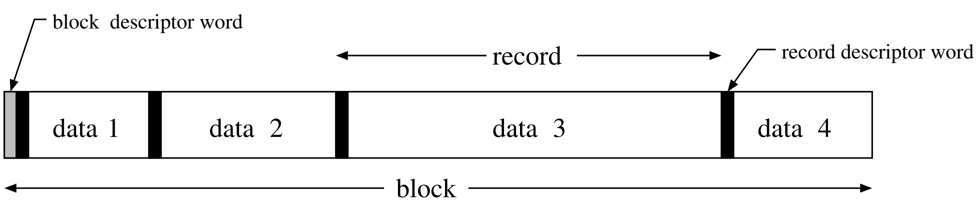
4.3.3 固定長レコード (F) 形式

固定長レコード形式とは，データセットを構成するレコードの全ての長さが一定のものを指す．例えば，固定形式の Fortran プログラムは，一行が 80 バイトとなり，レコード長 80 の固定長データセットである．特に，1 ブロック = 1 レコードを固定長非ブロック化レコード形式 (F 形式)，1 ブロックを幾つかのレコードに分割したものを固定長ブロック化レコード形式 (FB 形式) と呼ぶ．下図は FB 形式のイメージ．



4.3.4 可変長レコード (V) 形式

可変長レコード形式とは，データセットを構成しているレコードの長さがまちまちであるものを指す．例えば，自由形式の Fortran プログラムは可変長データセットである．固定長と同様，1 ブロック = 1 レコードを可変長非ブロック化レコード形式 (V 形式)，1 ブロックを幾つかのレコードに分割したものを可変長ブロック化レコード形式 (VB 形式) と呼ぶ．下図は VB 形式のイメージ．



4.3.5 不定長レコード (U) 形式

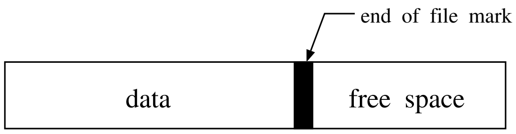
固定長レコード形式，可変長レコード形式のどちらにも属さないデータを取り扱うためのレコード形式．不定長レコード形式は 1 ブロック = 1 レコードで構成される，

4.4 データセットの編成

MSP のデータセットの特徴として，レコード形式とともに，データセット編成がある．ここでは，代表的な順編成と区分編成について述べる．

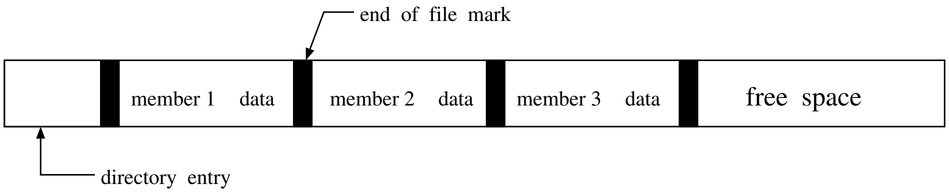
4.4.1 順編成データセット (PS : Physical Sequential)

データがトラックの先頭から順にずらずらと書かれているデータセット．



4.4.2 区分編成データセット (PO : Partitioned Organization)

区分データセットは，データセット内を幾つかの領域に分割して，それぞれがひとつの順編成データセットとして利用できる編成のデータセット．区分データセットの下にぶら下がっているデータを“メンバ (member)”と呼ぶ．



区分データセットは，メンバの追加，更新では常に結果を新規とみなして空領域に追加するため， editor で頻繁に区分データセットのメンバの編集を行なっていると，空スペース不足になりやすい．その場合は `CONDENSE` コマンドで圧縮を行なう ([11])．

4.5 タイプ名

タイプ名は，内容識別子とも呼ばれる．処理目的に応じてレコード形式，レコード長，タイプ名が決められている．利用者はこれに逆らうこともできるが，やってもあまり意味はない． Fortran 処理に関係するタイプ名は次の通り．

処理目的	レコード形式	レコード長	タイプ名
Fortran ソースプログラム (固定形式)	FB	80	FORT
Fortran ソースプログラム (自由形式)	VB	255	FORT
コマンドプロシジャ	VB	255	CLIST
ジョブ制御文	FB	80	CNTL
プログラム入力データ	FB	80	DATA
ジョブの結果出力	FB	80	OUTLIST

4.6 固定形式と自由形式

先ほどから何度か登場しているが， Fortran プログラムの形式は固定形式 (標準形式とよぶ場合もある) と自由形式の 2 種類ある．下に示すような違いがある．

	標準形式	自由形式
レコード形式	F(固定長)	V(可変長)
レコード長	80	255
文字数 / 行	72	最大 255
注釈行	第 1 桁が C，∗	第 1 桁が ”，∗, +
継続行	第 6 桁が数字 0 と空白以外の文字	直前の行の文字の 最後に - (ハイフン)

どちらの形式を選ぶかは利用者の自由だが，現在の状況から次の様にした方がいい．

他の計算機との移植性を考えた場合，固定形式の方が無難

形式を変換したいときには，CONVERT コマンドを用いる．使用方法は [11], [14] 参照．

4.7 データセットの作成方法

TSS でデータセットを新規に作成する主な方法は，次の通り．

1. ED コマンド
2. PFD(Programing Facility for Display user) のデータセット・ユーティリティ
3. PFDE のデータセット作成機能
4. EDIT コマンド
5. ALLOC コマンド
6. COPY コマンドによる既成データセットからのコピー

ED コマンド, EDIT コマンド，PFDE のデータセット作成機能による新規のデータセット作成の場合，システム側で設定する省略値がとられる．利用者側でデータセットの形式，大きさを指定したい場合，ALLOC コマンド，または PFD のデータセットオプションを用いる．ALLOC コマンドの使用方法は，後述する．PFD のデータセットオプションの使用方法は [11] を参照．

4.7.1 ED コマンドによる作成

ED コマンドは，PFD の editor を呼び出すコマンド．順データセット Fortran プログラム TEST.FORT を新規に作成，編集したい場合は次のように指定する．

```
 READY
ED TEST.FORT  ↵
```

すると，PFD のエディット画面に切り替わるので，適当に編集を行なえばよい．ソースプログラムは固定形式で作成すること．

区分編成データセット TEST.EX.FORT を新規に作成して，メンバ EX1 の編集を行ないたい場合は，次のようにする．

```
 READY
ED TEST.EX.FORT(EX1)  ↵
```

4.7.2 PFDE による作成

MSP の full screen が使用できる環境ならば，この方法がデータセット一覧を見ながら編集できるので便利．使用方法の詳細は [13] を参照．先ず，PFDE コマンドでデータセット一覧を表示する．

```
 READY
PFDE  ↵
```

画面が切り替わって，data set list menu が表示される．

```
----------------------------< DATA SET LIST MENU >----------------------------
COMMAND ===>                                                    SCROLL ===> CUR
PREFIX : A79999A
                                                USER-ID - A79999A TIME - 17:36
******  DATA SET NAME ----< COMMAND INPUT FIELD >----------MESSAGE------------
'''''' ABC.TEXT           _
'''''' A79999A.R03047.OBJ _
'''''' CMDPROC            _
'''''' DEMO.TEXT          _
'''''' DEMOCLG.CNTL       _
'''''' DEMOGO.CNTL        _
'''''' EGRETD.CNTL        _
'''''' EGRPRT.DATA        _
'''''' EXCIS.FORT         _
'''''' FUNC1              _
******  END OF LIST **********************************************************
```

カーソルを“''''''”の適当な部分にあわせ，“I” を入力し，↵ とすると，データセット入力モードになる．

```
----------------------------< DATA SET LIST MENU >----------------------------
COMMAND ===>                                                    SCROLL ===> CUR
PREFIX : A79999A
                                                USER-ID - A79999A TIME - 17:36
******  DATA SET NAME ----< COMMAND INPUT FIELD >----------MESSAGE------------
'''''' ABC.TEXT           _
'''''' A79999A.R03047.OBJ _
'''''' CMDPROC            _
==DSN>
'''''' DEMO.TEXT          _
'''''' DEMOCLG.CNTL       _
'''''' DEMOGO.CNTL        _
'''''' EGRETD.CNTL        _
'''''' EGRPRT.DATA        _
'''''' EXCIS.FORT         _
'''''' FUNC1              _
******  END OF LIST **********************************************************
```

Fortran データセット TEST.FORT を新規に作成する場合，==DSN> に続けてデータセット名を入力する．続けて“E”を入力すると，データセット編集画面に切り替わる．

```
----------------------------< DATA SET LIST MENU >----------------------------
COMMAND ===>                                                    SCROLL ===> CUR
PREFIX : A79999A
                                               USER-ID - A79999A TIME - 17:36
******  DATA SET NAME ----< COMMAND INPUT FIELD >----------MESSAGE------------
'''''' ABC.TEXT            _
'''''' A79999A.R03047.OBJ _
'''''' CMDPROC             _
'''''' TEST.FORT           E  ↵
'''''' DEMO.TEXT           _
'''''' DEMOCLG.CNTL        _
'''''' DEMOGO.CNTL         _
'''''' EGRETD.CNTL         _
'''''' EGRPRT.DATA         _
'''''' EXCIS.FORT          _
'''''' FUNC1               _
******  END OF LIST **********************************************************
```

TEST.FORT の属性は，LISTD コマンドで調べることができる．

```
----------------------------< DATA SET LIST MENU >----------------------------
COMMAND ===>                                                    SCROLL ===> CUR
PREFIX : A79999A
                                               USER-ID - A79999A TIME - 17:36
******  DATA SET NAME ----< COMMAND INPUT FIELD >----------MESSAGE------------
'''''' ABC.TEXT            _
'''''' A79999A.R03047.OBJ _
'''''' CMDPROC             _
'''''' TEST.FORT           LISTD  ↵
'''''' DEMO.TEXT           _
'''''' DEMOCLG.CNTL        _
'''''' DEMOGO.CNTL         _
'''''' EGRETD.CNTL         _
'''''' EGRPRT.DATA         _
'''''' EXCIS.FORT          _
'''''' FUNC1               _
******  END OF LIST **********************************************************
```

画面が切り替わって，属性情報が表示される．

```
A79999A.TEST.FORT
--RECFM-LRECL-BLKSIZE-DSORG
  FB    80    23440   PS                <--- 固定長データセット
--VOLUMES--
  PUB154
***
```

4.7.3 EDIT コマンドによるデータセット作成

EDIT コマンドによって起動される editor の使用方法は [12] を参照．

- 固定形式の Fortran プログラム TEST.FORT を新規に作成する．データセットは順編成とする．

```
 READY
E TEST.FORT FORT(FI)
 KEQ52320I DATA SET NOT FOUND, ASSUMED TO BE NEW
 INPUT
 00010        PROGRAM TEST
 00020         PRINT*,'TEST'
 00030        END
 00040
END S
 KEQ52460I SAVED IN DATA SET 'A79999A.TEST.FORT'
 READY
LISTD TEST.FORT                 <--- LISTD コマンドで属性を調べる
 A79999A.TEST.FORT
 --RECFM-LRECL-BLKSIZE-DSORG
   FB    80    3120    PS       <--- 固定長データセット
 --VOLUMES--
   PUB131
 READY
```

- 自由形式の Fortran プログラム EX1.FORT を新規に作成する．データセットは区分編成とし，メンバー名は TMP とする．

```
 READY
E EX1.FORT(TMP)
 KEQ52320I DATA SET NOT FOUND, ASSUMED TO BE NEW
 INPUT
 00010        PROGRAM TEST
 00020         PRINT*,'TEST'
 00030        END
 00040
```

```
END S
 KEQ52460I SAVED IN DATA SET 'A79999A.EX1.FORT(TMP)'
 READY
LISTD EX1.FORT                  <--- LISTD コマンドで属性を調べる
 A79999A.TEST.FORT
 --RECFM-LRECL-BLKSIZE-DSORG
   VB    255    3120    PO    <---- 可変長データセット
 --VOLUMES--
   PUB131
 READY
```

なお，既存のデータセットを EDIT コマンドにより編集する場合，固定長データセット作成時のオペランド“FORT(FI)” は不要．

5　TSS で Fortran を実行するためのコマンド

TSS での具体的な Fortran プログラムの実行方法は次節で紹介する．本節ではコマンドの機能を解説する．

TSS コマンド
RUN
FORT
LINK
CALL
ALLOCATE

その他の関連コマンドとして，`ALLOCATE` コマンドで割り当てるデータセットの属性を登録する `ATTRIB` コマンド，オブジェクトモジュールを実行する `LOADGO` コマンドなどがある．このうち，`ATTRIB` コマンドは，1993 年 4 月からの MSP のレベルアップにより，`ALLOCATE` コマンドで同等の機能が追加されたことから，10 節に後述した．また，`LOADGO` コマンドは用途が特殊なため割愛した．

5.1　記号の説明

- 強調文字の部分は，それ以下が省略可能であることを示す．例えば "**FR**ee" は FREE と入力しなくても，FR で省略できることを意味する．
- "[{ FIXED | FREE }]" とは，オペランドで FIXED または FREE のいずれかを指定するか，何も指定しないことを表す． FIXED と FREE の同時の指定は出来ない．
- " LIB (区分データセット...) " の "..." は，続けて指定可能なことを表す．この場合は区分データセットを複数続けて入力できる．

  【例】
  ```
  LIB('SYS1.TGSLIB' 'LIB.NUMPAC.LOAD' MYLIB.LOAD)
  ```
- アンダーラインは，指定がない場合の省略値を表す．例えば " [AE | <u>NOAE</u>]" は，AE の指定がない場合は省略値として NOAE が採用されることを意味する．

5.2　TSS コマンドの詳細を知るには

MSP の TSS コマンドについて，基本的なコマンドは `HELP` コマンドを用いてオンラインで参照が出来る．

```
 READY
HELP ↵
```

と入力すれば，検索できるコマンドの一覧が表示できる．その他の TSS コマンドについては [11], [12], [14] を参照．

5.3 RUN

RUN コマンドは，モード表示が READY 状態のときに使用可能で，実行までの処理を一括して行う．EDIT 内からも使用できるが，この場合はデータセット名を書く必要がない (RUN サブコマンド)．この RUN コマンドは Fortran だけでなく，それ以外の言語 (Pascal, PL/I, Cobol 等) の実行にも使用できる．

【入力形式】

RUN	入力データセット名
R	[FORT]
	[{ **FI**xed | **FR**ee }]
	[{ **LM**sg | **SM**sg }]
	[LIB (区分データセット...)]
	[**LIBD**d (DD 名)]
	[AE]

【オペランドの説明】

入力データセット名
: Fortran ソースプログラムが格納されているデータセット名を指定する．

FORT
: データセットのタイプ名が “FORT” 以外の場合指定する．

FIXED
: Fortran ソースプログラムの形式が標準形式であることを指定．省略値．

FREE
: Fortran ソースプログラムの形式が自由形式であることを指定．

LMSG
: 詳細な診断メッセージを出力させる．

SMSG
: 簡略化した診断メッセージを出力させる．省略値．

LIB(データセット ...)
: プログラムのリンクの際に使用するライブラリ群を指定．私用ライブラリを使用する場合も指定．データセットの編成は区分編成でなければならない．

LIBDD(DD 名)
: リンクの際，DD 名によるライブラリの検索を行なうことを指定．

AE
: Fortran プログラムを拡張アドレスモードで翻訳，実行するときに指定．

5.4 FORT

Fortran プログラムの翻訳を行い，オブジェクトモジュールを作成する．オプションによってはRUN コマンドと同様にそのまま実行することもできる．

【入力形式】

FORT	入力データセット名
	[' ローダオプション / 実行可能プログラムオプション ...']
	[データセット指定オプション...]
	[コンパイラオプション]
	[AE]
	[VP]

【オペランドの説明】

入力データセット名

コンパイラの入力となる Fortran プログラムを含むデータセットを指定．必須オペランドであり，かつ最初の位置パラメータである．つまり，必ず FORT コマンドの直後にデータセット名を指定する．

' ローダオプション / 実行可能オプション'

コンパイラオプションで GO オプションを指定したとき，ローダおよび実行可能プログラムのオプションを指定する．これは，第 2 番目の位置オペランドとして指定しなければならない．実行可能プログラムオプションを指定する場合は，必ず "/" に続けて指定する．オプションについては [5] 参照．

データセット指定オプション

LIB (データセット名 ...)	:	結合編集時の 1 次ライブラリを指定．私用ライブラリを使用する場合必ず指定．
OBJ (データセット名)	:	オブジェクトモジュールの出力データセットを指定．OBJ オプションを指定した場合，実行処理されない．

主なコンパイラオプション

AE	:	アドレス拡張機能を使用することを指定．
VP	:	VP 用のコンパイラを起動する．但し作成したロードモジュールの TSS での実行は出来ない．VP 用私用ライブラリを作成する場合などに指定．
FREE	:	自由形式の Fortran ソースプログラムを翻訳するとき指定.
NOGO	:	翻訳のみを行ない，実行処理を行なわない．指定しない場合は実行まで行なう．
SMSG	:	詳細な診断メッセージを出力させる．
MAP	:	記憶領域の割付け (マップ) を行なうことを指定．
NAME	:	リンケージエディタの NAME 制御文の出力を行なう．

5.5 LINK

LINK コマンドは，リンケージエディタを呼び出し，オブジェクトモジュールにライブラリの結合編集を行って，実行形式のロードモジュール，または私用ライブラリを作成する．

【入力形式】

LINK	(入力データセット ...)
	[**LO**ad (区分データセット名)]
	[LIB (区分データセット名 ...)]
	[FORTLIB]

【オペランドの説明】

入力データセット

オブジェクトモジュール名を指定．位置オペランドであり，必ず LINK コマンドに続けて指定しなければならない．

LOAD(区分データセット名)

出力となるロードモジュールのデータセット名を指定．メンバ名が指定されない場合，`TEMPNAME` というメンバ名で作成される.

LIB(区分データセット名 ...)

結合に必要となるライブラリのデータセットを指定する. 私用ライブラリを使用する場合や，SSL II/VP，NUMPAC/VP のサブルーチンライブラリを CALL 使用するロードモジュールを作成する場合指定する．ライブラリ名は次の通り

NUMPAC/VP : `'LIB.NUMVP.LOAD'`
SSL II/VP : `'SYS1.SSL2VP'`

FORTLIB

Fortran システムライブラリ `'SYS1.FORTLIB'` を使用する．

NCAL

未解決の外部参照があるとき，LIB, FORTLIB 等で指定したライブラリからの組み込みを抑止する．

5.6　CALL

CALL コマンドは，区分データセットのメンバとして格納してあるロードモジュールの実行を行う．

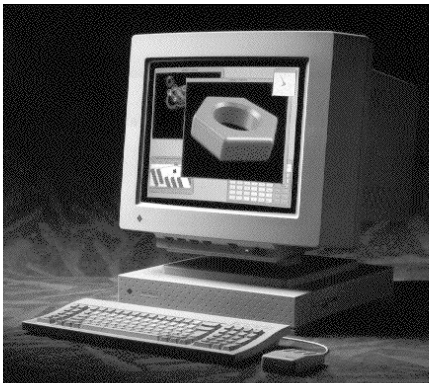
【入力形式】

CALL	区分データセット (メンバ名)

【オペランドの説明】

区分データセット (メンバ名)

　実行させるロードモジュールが格納されている区分データセット名と，そのメンバ名を指定する．



ネットワーク経由でも使えます

5.7 ALLOCATE

ALLOCATE コマンドの機能は，次の 3 つである．

- データセットを論理機番と結び付ける（割り当てる）．
- 入出力にデータセットを利用する場合．
- 種々の属性を持ったデータセットを新しく作る．

なお，1993 年 4 月からのレベルアップより，`ATTRIB` コマンドで行なっていたデータセットの属性指定のオペランドが追加された．

【入力形式】

ALLOCATE	[**FI**le (DD 名)]
ALLOC	[**DA**taset ({ ∗ | データセット名 ... })]
	[**SY**sout (U)]
	[{ **OL**d | **SH**r | **MO**d | <u>**NE**w</u> }]
	[**CA**talog]
	[{ **TRA**cks | **CY**linders }]
	[**SP**ace (初期量 [増分量])]
	[**DIR** (ディレクトリブロック数)]
	[**US**ing (属性リスト名)]
	[**BLK**size (ブッロク長)]
	[**LR**ecl (論理レコード長)]
	[**DSO**rg (PO | PS)]
	[**REC**fm (レコード形式)]
	[**REU**se]

【オペランドの説明】

FILE(DD 名)

割り当てたいデータセットに対する DD 名を指定する．Fortran プログラムの論理機番 `nn` の DD 名は `FTnnF001`．例えば論理機番 1 の DD 名は `FT01F001`．

DATASET

割り当てたいデータセットのデータセット名を指定する．∗ を指定すると，端末が割り当てられる．

SYSOUT(U)

センター内のプリンター装置にデータを出力したいときに指定．

OLD

既にデータセットが存在し，他の作業と並行してこのデータセットを使用しては困るとき指定する．

SHR

既存のデータセットを他の利用者と共有しても使用することを指定．

MOD

データの作成にあたって，すでにあるデータセット内のデータに追加作成するときに指定する．

NEW

新しくデータセットを作成するときに指定する．

CATALOG

新規に作成したデータセットをカタログ簿に登録し，保存することを指定．

TRACKS

SPACE オペランドで指定した初期量，増分量の値がトラック数を表わしていることを指定する．

CYLINDERS

SPACE オペランドで指定した初期量，増分量の値がシリンダ数を表わしていることを指定する．

SPACE(初期量　［増分量］)

データセットの作成に必要な領域の大きさを指定する．

DIR(ブロック数)

区分データセットを作成する場合必須．順データセット作成時には不要．ディレクトリ領域としていくつ必要であるかを指定する．通常は 1 ディレクトリにつき 6 メンバ．

BLKSIZE

データセットを構成するブロック長を指定する．

LRECL

データセット中のブロックを構成する論理レコードの最大長を指定する．

DSORG

データセットの編成を指定する．

PO : 区分データセット
PS : 順データセット

RECFM

データセットのレコード形式を指定する．

USING(属性リスト名)

割り当てるデータセット属性リスト名を指定する．属性リストはあらかじめ `ATTRIB` コマンドで作成しておかなければならない (10 節参照).

REUSE

一度割当てた DD 名をいったん解除してから再割当てを行う．

ALLOCATE コマンドでのデータセット作成例

- 固定長ブロック化形式の順編成データセット EX.DATA を新規に作成する．1 ブロックの大きさは最大 23440 バイト．

```
 READY
ALLOC DA(EX.DATA) NEW CA SP(10 5) TRA +  ↵              <---   + は継続の印
REC(F B) BLK(23440) LR(80) DSO(PS)  ↵
```

- 可変長ブロック化形式の区分編成データセット EX.TEXT(EX) を新規に作成する．

```
 READY
ALLOC DA(EX.TEXT(EX)) NEW CA SP(10 10) TRA DIR(5) +  ↵ <--- + は継続の印
REC(V B) BLK(23440) LR(255) DSO(PO)  ↵
```

データセットを磁気ディスクに確保するためには，ALLOCATE コマンドの他に，PFD コマンドによって起動される PFD のデータセット作成機能も使用できる．MSP におけるデータセットの編成，レコード形式，データセットの管理等は [11] に詳述してあるので参照されたい．

ALLOCATE コマンドの SPACE オペランドで指定されるデータセットの容量は次のように計算する．上記例のようにトラック単位の領域を確保した場合を考える．最初のデータセット作成では，容量は "SP(10 5)" で決定される．まず，初期容量として 10 トラックが確保される．1 トラックは 47 キロバイトなので，初期量は $47 \times 10 = 470$ キロバイトである．次に，データセットが最初確保した容量を越えた場合，SPACE オペランドの増分量で宣言した容量だけ，必要に応じて <u>最大 15 回</u> の容量の拡張を許す．

SP(10 5) の指定では，増分量が 5 トラックであり，最大増分回数は 15 回であるので，最大の増分値は $5 \times 47 \times 15 = 3,525$ キロバイトである．よって，SPACE(10 5) TRACKS によって確保されるデータセットの最大領域は $470 + 3,525 = 3,995$ キロバイトとなり，ざっと 4MB 弱である．

MSP では，UNIX と違い一度宣言したデータセットに対してはこれ以上の書き込みは出来ない．SPACE パラメータを大きくとれば，更に容量の大きいデータセットが確保できるが，普通はこの程度で十分であろう．

6 TSS での Fortran 実行例

6.1 一気に Fortran プログラムを実行する – RUN,FORT –

MSP のデータセットに作成した Fortran プログラムに対し，翻訳 (コンパイル), 結合編集 (リンク), 実行をまとめて行う例を以下幾つか示す．コマンドは RUN，または FORT で行なう．利用者は A79999A さんとする．

- プログラム TEST.FORT を RUN コマンドで実行する．入力データをキーボードより入力する時，または入力データが必要ない時．出力は端末画面に出す．

```
 READY
RUN TEST.FORT  ↵
```

- プログラム TEST.FORT を FORT コマンドで実行する．入力データをキーボードより入力する時，または入力データが必要ない時．出力は端末画面に出す．

```
 READY
FORT TEST.FORT  ↵
```

- プログラム TEST.FORT を RUN コマンドで実行する．入力データをキーボードより入力する時，または入力データが必要ない時．出力は端末画面に出す．自分用に作成した関数やサブルーチンを呼ぶため，それらが格納されたデータセット MYLIB.LOAD を指定する．

```
 READY
RUN TEST.FORT LIB(MYLIB.LOAD)  ↵
```

- 'SYS1.TGSLIB' というデータセット名のライブラリ群を使用して GRAPH.FORT を実行させる.

```
 READY
FORT GRAPH.FORT LIB('SYS1.TGSLIB')  ↵
```

- PFD/EDIT で編集中の Fortran プログラムを実行する．実行は RUN サブコマンドで行なう．R で省略可能．

```
EDIT --- A79999A.TEST.FORT ------------------------- 表示欄  001 072
コマンド ===> R  ↵                                   移動量 ===> CUR
****** ********************* データの先頭 ************ V10L30 ******
000010      1 READ(5,*,END=999) N
000020     10 FORMAT(I4)
000030        IF(N-N/4*4.NE.0) THEN
000040         WRITE(6,20)N
000050        ELSEIF(N-N/100*100.NE.0) THEN
000060         WRITE(6,30)N
000070         END
****** ********************* データの末尾 **************************
```

- TEST.FORT を RUN コマンドにより実行し，計算結果をオープン室の NLP，CLP に出力する．ALLOCATE コマンドによって，WRITE(6,*) への出力（FT06F001）を端末から NLP に切替え，出力後にもとに戻す．

```
 READY
ALLOC FI(FT06F001) SY(U) REU ↵           ← プリンターへの切替え
 READY
RUN TEST.FORT ↵
     :
 (翻訳メッセージなど)
     :
 READY
ALLOC FI(FT06F001) DA(*) REU ↵            ← 出力先を画面に戻す
```

上記例題で，端末によっては画面への割り当ての時，レコード長が 80 バイトになり，プログラムで出力レコードが 80 バイト以上のものを端末に出力するとエラーとなる場合がある．その場合は，出力先を端末画面に戻す際に次のように属性を修正する．

```
 READY
ALLOC FI(FT06F001) DA(*) REU LR(137) BLK(3120) REC(V B A) ↵
```

- TEST.FORT を RUN コマンドにより実行する．READ(5,*) に対応する入力データはデータセット TEST.DATA から入力．出力は端末画面 ．

```
 READY
ALLOC FI(FT05F001) DA(TEST.DATA) SHR REU ↵
 READY
RUN TEST.FORT ↵
     :
  (実行結果)
     :
ALLOC FI(FT05F001) DA(*) REU ↵              ← 入力元を端末に戻す
```

- TEST.FORT を RUN コマンドにより実行する．READ(5,*) に対応する入力データはデータセット TEST.DATA から読取る．WRITE(6,*) に対応する出力は新規に RESULT.OUTLIST を作成して出力する．

```
 READY
ALLOC FI(FT05F001) DA(TEST.DATA) REU ↵
 READY
ALLOC FI(FT06F001) DA(RESULT.OUTLIST) NEW CA TRA SP(10 10) REU ↵
 READY
RUN TEST.FORT
```

```
    :
（翻訳メッセージなど）
    :
 READY
ALLOC FI(FT05F001) DA(*) REU ↵            ← 入力元を端末に戻す
 READY
ALLOC FI(FT06F001) DA(*) REU LR(137) BLK(3120) REC(V B A) ↵
                                                  ← 出力先を画面に戻す
```

- プログラムの実行におけるデータセットの入出力は，RUN または FORT コマンドの直前に ALLOC コマンドを用いて割り付けるかわりに，Fortran のソースプログラムの中に直接 OPEN 文を入れて動的に割り付ける方法もある．OPEN 文の記述は一般の Fortran 文法書に従う．注意点は，FILE の指定で完全データセットを指定すること．従って，「'」で両方をくくり，ID もあわせて記述すること．

  ○ OPEN(5,FILE='A79999A.TEST.DATA',STATUS='OLD')
  ○ OPEN(6,FILE='A79999A.RESULT.OUTLIST',STATUS='OLD')
  × OPEN(5,FILE=TEST.DATA,STATUS='OLD')
  × OPEN(6,FILE=RESULT.DATA,STATUS='OLD')

  もちろん最後は CLOSE 文を忘れないこと．なお，STATUS=OLD は既にデータセットが存在している場合に指定．

- 入力，出力には，その他の機番も指定できる．
  入力の標準機番は 5（READ(5,*)），出力の標準機番は 6（WRITE(6,*)）であるが，それ以外の機番も指定可能である．

  【例】
  ```
  READ(11,*)    →   ALLOC FI(FT11F001) DA(TEST.DATA) REU
  WRITE(22,*)   →   ALLOC FI(FT22F001) DA(RESULT.OUTLIST) REU
  ```

  機番は 1 から 99 まで指定できる．

6.2 オブジェクトモジュール作成 – FORT –

オブジェクトモジュールとは Fortran コンパイラで機械語に翻訳された結果のプログラムのこと．普通は実行可能なロードモジュールの作成段階に作るデータセットであり，ロードモジュールを作成した後は削除しても構わない．

オブジェクトモジュールの作成は FORT コマンドに OBJ オプションを付加して行なう．オブジェクトモジュールの名前は，利用者が適当につける．

- ソースプログラム TEST.FORT(EX1) に対しコンパイルを行い，オブジェクトモジュール TEST.OBJ(EX1) を作成する．

```
 READY
FORT TEST.FORT(EX1) OBJ(TEST.OBJ(EX1)) ↵
```

- 上記例は次のように省略可能．

```
 READY
FORT TEST(EX1) OBJ(TEST(EX1)) ↵
```

- ソースプログラム TEST.FORT(EX1) に対し翻訳を VP で行ない，オブジェクトモジュール TEST.OBJ(EX1) を作成する．Fortran コンパイラは，Fortran77 EX に代わって，Fortran77 EX/VP が行なう．アドレス拡張機能を使用する．ただし VP 上では翻訳のみ可能であり，TSS での実行はできない．実行はバッチ処理で行なう．

```
 READY
FORT TEST.FORT(EX1) OBJ(TEST.OBJ(EX1)) AE VP ↵
```

FORT コマンドに VP オプションを付加して VP で翻訳を行なうと，ベクトル処理に適したオブジェクトモジュールが作成され，実行が高速化される．ただし，TSS 処理で対話的な実行が出来ない．実行はバッチ処理のみ．

従って VP オプションを付加する場合は，デバックが完全に終了した自分用のサブルーチンや関数を VP 用ライブラリとして登録する時などに利用する．

6.3 実行用ロードモジュールの作成 – LINK –

LINK コマンドによって，実行可能なロードモジュールを作成する．FORT コマンドによって作成されたオブジェクトモジュールからロードモジュールを作成する．ソースプログラムからロードモジュールを作成するには，常に FORT コマンドと LINK コマンドが対になっている．

- Fortran ソースプログラム TEST.FORT からオブジェクトモジュール TEST.OBJ(EX1) を作成し，LINK コマンドによりロードモジュール TEST.LOAD(EX1) を作成する．オブジェクトモジュールは，Fortran プログラムとロードモジュールとの仲介の役割を担うものであり，ロードモジュールを作成すると無用の長物であるので，消去して問題ない．

```
 READY
FORT TEST.FORT(EX1) OBJ(TEST.OBJ(EX1)) ↵
 READY
LINK TEST.OBJ(EX1) LO(TEST.LOAD(EX1)) FORTLIB ↵
 REDAY
DEL TEST.OBJ ↵
```

- 上のコマンドは，タイプ名を省略して次のように不精できる．

```
 READY
FORT TEST(EX1) OBJ(TEST(EX1)) ↵
 READY
LINK TEST(EX1) LO(TEST(EX1)) FORTLIB ↵
```

- NUMPAC/VP を使用したプログラムを Fortran77 EX/VP でコンパイルし，ロードモジュールを作成する．翻訳時の最適化パラメータ（cf.9.4 節）は OPT(E) を指定．

```
 READY
FORT TEST.FORT(EX1) OBJ(TEST.OBJ(EX1)) VP AE OPT(E)  ↵
 READY
LINK TEST.OBJ(EX1) LO(TEST.LOAD(EX1)) LIB('LIB.NUMVP.LOAD') FORTLIB  ↵
```

SSL II/VP を使用する場合は `LIB('SYS1.SSL2VP')` と指定．ただし，作成した `TEST.LOAD(EX1)` を `CALL` コマンドを用いて TSS で実行することは出来ないので注意．

6.4　ロードモジュールの実行 – CALL –

作成したロードモジュールは `CALL` コマンドで実行される．入力データを変えて何度も繰り返しプログラムを実行したい場合などは，あらかじめロードモジュールを作成しておいた方が便利．

- ロードモジュール `TEST.LOAD(EX1)` を実行する．入力データはなし．

```
 REDAY
CALL TEST.LOAD(EX1)  ↵
```

- ロードモジュール `TEST.LOAD(EX2)` を実行する． `READ(5,*)` に対応する入力データを最初は `TEST1.DATA`，次に `TEST2.DATA` にして各々実行する．

```
 READY
ALLOC FI(FT05F001) DA(TEST1.DATA) SHR REU  ↵
 READY
CALL TEST.LOAD(EX2)  ↵
     :
（実行結果）
     :
 READY
ALLOC FI(FT05F001) DA(TEST2.DATA) SHR REU  ↵
 READY
CALL TEST.LOAD(EX2)  ↵
    :
（実行結果）
    :
 READY
ALLOC FI(FT05F001) DA(*) REU  ↵             ←  入力元を端末に戻す
```

6.5　私用ライブラリの作成

自分で作成したサブルーチンや関数を，異なる Fortran プログラムから CALL したい場合，いちいち main プログラムに同じサブルーチンや関数を書き込むのは煩わしい．その場合，自分用のサブルーチンや関数が詰まったロードモジュールを作っておき，そこに `FORT` & `LINK`

コマンドで実行形式にしたプログラムを保存しておけばよい．CALL の際は，LIB パラメータでそのデータセットを指定することで，対応するサブルーチンや関数を捜しに行く．その時，ロードモジュールのメンバ名は，サブルーチン名や関数名に一致するので，重複は避けること．

- FORT コマンドを用い副プログラムを記述したデータセット TEST.FORT(SUB) からオブジェクトモジュール TEST.OBJ を作成する．NAME オペランドによって，副プログラム毎にコンパイルすることを指定する．

```
  READY
 FORT TEST.FORT(SUB) OBJ(TEST.OBJ) NAME  ↵
```

- 上記の方法で登録したオブジェクトモジュール TEST.OBJ から LINK コマンドによりロードモジュール MYLIB.LOAD を作成し，私用ライブラリとして登録する．MYLIB.LOAD が既存の場合は，メンバーが追加される．

```
  REDAY
 LINK TEST.OBJ LOAD(MYLIB.LOAD) NCAL ALIAS  ↵
```

  NCAL は外部参照を組み込まないための指定．ALIAS はサブルーチン中に ENTRY 文がある時に必要

- このようにして作成された私用ライブラリ MYLIB.LOAD を利用して，ソースプログラム EX.FORT を RUN コマンドで実行する．

```
  READY
 RUN EX.FORT LIB(MYLIB.LOAD)  ↵
```

6.6 簡単なコマンドの入力方法

コマンド例での一連のコマンド入力は，似たようなコマンドを入力する場合が多い．頻繁に使うコマンドリストをあらかじめデータセットに作成しておくと少しの修正で済む．

タイプ名 .CLIST のデータセットは，コマンドのリストを保存するもので，これを EX コマンドで実行すると，コマンドリストに記述されたコマンド群を上から順番に実行する．

例えば，よく行なうサブルーチンライブラリの作成について，次のコマンド列をデータセット SUB.CLIST に作成しておく．

```
EDIT --- A79999A.SUB.CLIST ----------------------------- 表示欄   001 072
コマンド ===>                                             移動量 ===> CUR
****** ************************ データの先頭 ************ V10L30 ******
000001 /* COMMENT     <--- /* 以下はコメント
000002  FORT TEST.FORT(SUB) OBJ(TEST.OBJ) NAME VP AE NOGO
000003  LINK TEST.OBJ LOAD(MYLIB.LOAD) NCAL
****** ************************ データの末尾 **************************
```

SUB.CLIST の実行は EXEC コマンドで行なう．EXEC は EX と省略可能であり，SUB.CLIST も SUB と省略できる．

```
 READY
EX SUB ↵        <--- コマンドリストの実行
```

これにより，SUB.CLIST に記述したコマンドが実行される．エディタ内からは，EXEC サブコマンドで起動される．

私用ライブラリの登録では，以下作成した SUB.CLIST でソースプログラム，ロードモジュール名などを修正して EXEC コマンドで実行させれば良い．

同じく，誰もが嫌になる ALLOCATE コマンドも，典型的な作成パターンをデータセットに保存しておくと，いちいちマニュアルを引っ張り出さなくてもよくなる．

```
EDIT --- A79999A.AL.CLIST ----------------------------- 表示欄  001 072
コマンド ===> EX ↵                                   移動量 ===> CUR
****** ************************ データの先頭 ************ V10L30 ******
000001 /* ALLOCATE COMMAND LIST
000002  ALLOCATE DA(PS.DATA) NEW CA SPACE(10 10) TRA +
000003  REC(F B) BLK(23440) LR(80) REU
****** ************************ データの末尾 *************************
```

よく用いるコマンド群はこのようにまとめておくと便利である．また，自分で作成したコマンドを EXEC をつけずに昔からあったコマンドの様に実行することも可能である．詳しくは [12], [14] を参照．

7　バッチ処理 — カタログドプロシジャの利用 —

バッチジョブを実行し，バックグラウンドでの Fortran プログラムの処理をさせるためには，ソースプログラムや入力データセットの他に，計算機に仕事を任せるためのジョブ制御文 (JCL) が必要である．ジョブ制御文の中心は，カタログドプロシジャ (Cataloged Procedure) と呼ばれる一文であり，これが TSS コマンドの役割を果たす．以下， Fortran 処理に関係するジョブ制御文の記述方法を，カタログドプロシジャの紹介を中心に述べる．バッチ処理に関する詳細は [10] を参照．

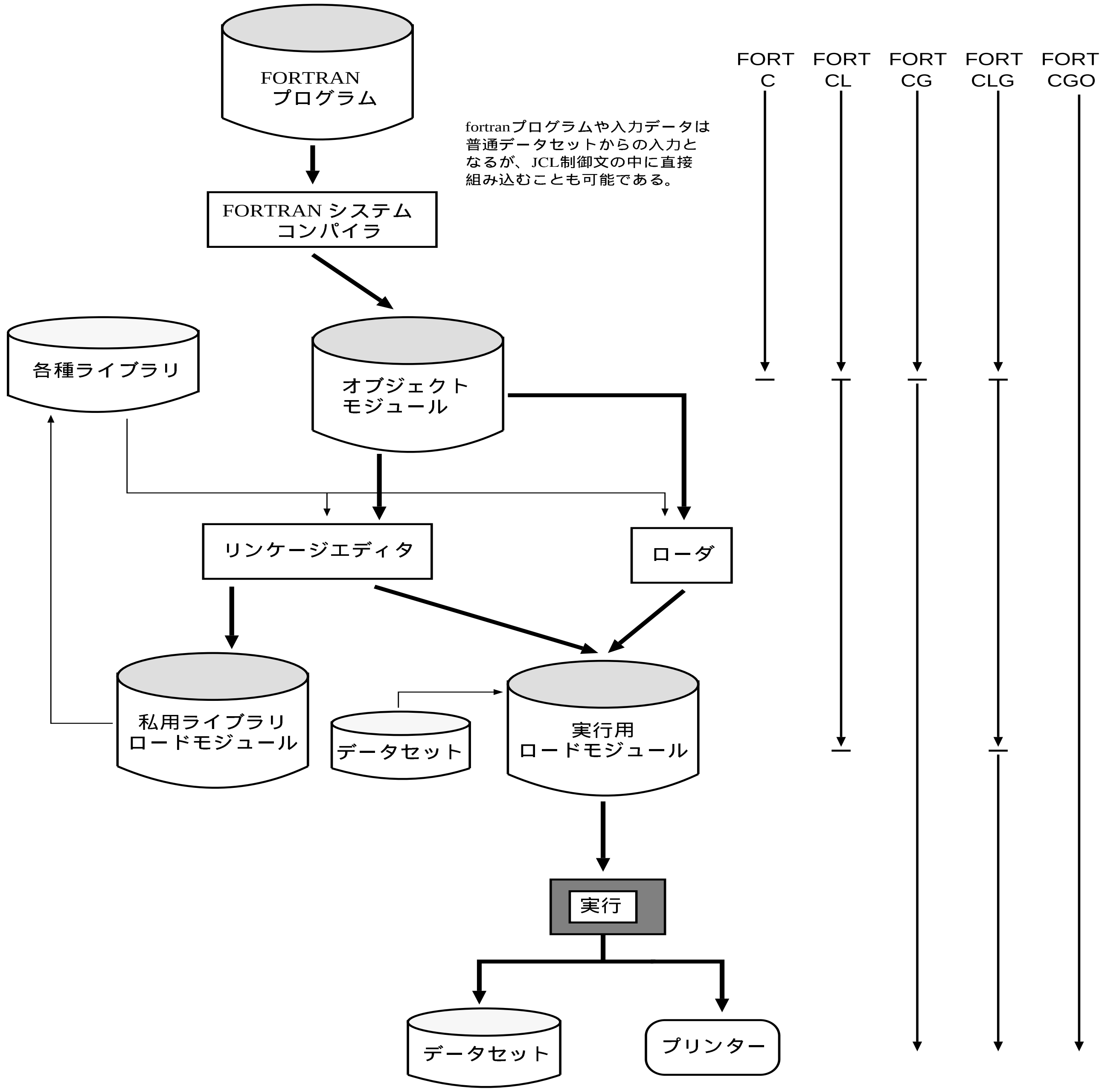


図 1: Fortran 実行流れ図

7.1 FORT

Fortran コンパイラによる翻訳，結合編集，実行を行う．

	主なパラメータ
FORT	[, SYSOUT = { <u>A</u> ❘ K ❘ S ❘ O ❘ U ❘ H }] [, STEP = { C ❘ CGO ❘ CG ❘ CL ❘ <u>CLG</u> }] [, OPT = { <u>B</u> ❘ E ❘ F ❘ NO }] [, PRVLIB = ' データセット名'] [, OPTION = ' コンパイラオプション'] [, VP = YES]

記号パラメータの説明

SYSOUT : 出力クラスを指定する．省略値は A．
STEP : 処理過程を選択する．省略値は CLG．
C ... 翻訳のみ行う．
CGO ... 翻訳，結合編集，実行を 1 ステップで行う．
CG ... 翻訳，ローダによる結合編集，実行をする．
CL ... 翻訳，リンケージエディタによる結合編集まで行い，
私用ライブラリあるいは実行用のロードモジュールを作成．
CLG ... 翻訳，リンケージエディタによる結合編集，実行をする．
OPT : 最適化のレベルを指定する．省略値は B．
PRVLIB : 組み込みたい私用ライブラリのデータセット名を指定する．
OPTION : コンパイラオプションのリストを記述する．
VP=YES : VP で実行する時指定する．省略すると M1800 で実行される．

オプション	ジョブステップ	関連する DD 名	データセット
STEP=C	FORT	FORT.SYSIN	ソースプログラム
STEP=CL	FORT LKED	FORT.SYSIN LKED.SYSIN	ソースプログラム リンケージエディタ制御文
STEP=CLG	FORT LKED GO	FORT.SYSIN LKED.SYSIN LKED.SYSLMOD GO.SYSIN	ソースプログラム リンケージエディタ制御文 私用ライブラリあるいは ロードモジュール保存 実行時の入力データ
STEP=CG	FORT LOADGO	FORT.SYSIN LOADGO.SYSIN	ソースプログラム 実行時の入力データ
STEP=CGO	FORTCGO	FORTCGO.SYSIN FORTCGO.SYSGO	ソースプログラム 実行時の入力データ

7.2 LKED

リンケージエディタによってオブジェクトモジュールから私用ライブラリを作成したり，結合編集してロードモジュールを作成する．

	主なパラメータ
LKED	[, SYSOUT = { A ∣ K ∣ S ∣ O ∣ U ∣ H }] [, PRVLIB = ’データセット名’]

記号パラメータの説明

SYSOUT ：出力クラスを指定する．省略値は A．
PRVLIB ：組み込みたい私用ライブラリのデータセット名を指定する．

7.3 GO

ロードモジュールを実行する．

	主なパラメータ
GO	, PROG = プログラム名 , LOADDS = ’区分データセット名’ [, SYSOUT = { A ∣ K ∣ S ∣ O ∣ U ∣ H }]

記号パラメータの説明

PROG ：実行するプログラム名を指定する．
LOADDS ：ロードモジュールが格納されているデータセット名を指定する．
SYSOUT ：出力クラスを指定する．省略値は A．

Fullcolor Printer CP3000

7.4 カタログドプロシジャの使用例

- Fortran ソースプログラム TEST.FORT を実行する．FT05F001(READ(5,*)) からの入力は TEST.DATA から行なう．STEP=CLG として，翻訳，結合編集，実行のそれぞれのジョブステップで行う．

```
//A79999A1 JOB CLASS=A
//   EXEC FORT,STEP=CLG                            ← EXEC の前には空白を
//FORT.SYSIN DD DSN=A79999A.TEST.FORT,DISP=SHR
//GO.SYSIN   DD DSN=A79999A.TEST.DATA,DISP=SHR
//                                                 ← 制御文の終わり
```

- Fortran ソースプログラム TEST.FORT を STEP=CLG で実行する．TEST.FORT が READ(5,*) 等の入力が不必要な自身で完結したプログラムの場合はこれだけで OK．

```
//A79999A2 JOB CLASS=A
//   EXEC FORT,STEP=CLG
//FORT.SYSIN DD DSN=A79999A.TEST.FORT,DISP=SHR
//
```

- Fortran ソースプログラム TEST.FORT を VP2600 上 STEP=CLG で実行する．最適化レベルとして OPT(E) を指定する．

  また最適化のメッセージを示すコンパイラオプション OPTMSG を指定する．

```
//A79999A3 JOB CLASS=B
//   EXEC FORT,VP=YES,STEP=CLG,OPT=E,             ← 継続は , で
//        OPTION='OPTMSG'
//FORT.SYSIN DD DSN=A79999A.TEST.FORT,DISP=SHR
//
```

- Fortran ソースプログラム TEST.FORT を VP2600 上 STEP=CLG で実行する．入力データセットとして TEST.DATA を，出力データセット (WRITE(6,*)) として RESULT.DATA(既存) を各々指定する．また TIME パラメータを指定して，ジョブの優先度を上げる．

```
//A79999A4 JOB CLASS=V,TIME=10
//   EXEC FORT,VP=YES,STEP=CLG,OPT=E,             ← 継続は , で
//        OPTION='OPTMSG'
//FORT.SYSIN  DD DSN=A79999A.TEST.FORT,DISP=SHR
//GO.FT06F001 DD DSN=A79999A.RESULT.DATA,DISP=SHR
//
```

- Fortran ソースプログラム TEST.FORT(EX) を STEP=CLG で実行する．出力メッセージを日本語で出力させる．またジョブ処理の詳細な情報を出力させる．

```
//A79999A5 JOB CLASS=A,MSGLEVEL=(1,1),SPARM='LANG=J'
//   EXEC FORT,STEP=CLG,OPT=B
//FORT.SYSIN DD DSN=A79999A.TEST.FORT(EX),DISP=SHR
//
```

- Fortran ソースプログラム TEST.FORT を翻訳し，実行可能なロードモジュール SAMPLE.LOAD(TIME) を作成する． STEP=CL であり，ロードモジュールは新規に作成する．

```
//A79999A6 JOB CLASS=A
//   EXEC FORT,STEP=CL
//FORT.SYSIN   DD DSN=A79999A.TEST.FORT,DISP=SHR
//LKED.SYSLMOD DD DSN=A79999A.SAMPLE.LOAD(TIME),
//     UNIT=PUB,DISP=(NEW,CATLG),SPACE=(TRK,(10,10,1),RLSE)
//
```

- Fortran ソースプログラム TEST.FORT を実行する．その際，私用ライブラリ MYLIB.LOAD を引用して実行する．

```
//A79999A6 JOB CLASS=B
//   EXEC FORT,STEP=CLG,PRVLIB='A79999A.MYLIB.LOAD'
//FORT.SYSIN DD DSN=A79999A.TEST.FORT,DISP=SHR
//
```

- Fortran ソースプログラム TEST.FORT を実行する． STEP=CLG で実行．ソースプログラムを直接ジョブ制御文に書き込む．

```
//A79999A7 JOB CLASS=A
//   EXEC FORT,STEP=CLG
//FORT.SYSIN DD *        ← * は以下ソースプログラムであることを示す
      INTEGER ADD,SUB,MUL
       REAL  DIV
       WRITE(6,'(4X,''I  J  ADD  SUB  MUL  DIV''/)')
       DO 10 I=1,10
       DO 10 J=1,10
       WRITE(6,'(5I6,F12.6)') I,J,ADD(I,J),SUB(I,J),MUL(I,J),DIV(I,J)
    10 CONTINUE
      END
/*                       ← ソースプログラムの入力終わり
//
```

- Fortran ソースプログラム EX.FORT を STEP=CGO により，翻訳，結合編集，実行を 1 ジョブステップで行う．

```
//A79999A8 JOB CLASS=A
//   EXEC FORT,STEP=CGO
//FORTCGO.SYSIN DD DSN=A79999A.EX.FORT,DISP=SHR
//
```

- STEP=CGO でソースプログラムや入力データをそのまま書き込む場合．ただし，ジョブ制御文に書き込むのは昔のカード時代の風習の名残であり，デバッグ等で後々苦労する．データセットは独立に作成しておいた方が良い．

```
//A79999A9 JOB CLASS=A
//  EXEC FORT,STEP=CGO,OPT=E
//FORTCGO.SYSIN DD *
      INTEGER X,H,M,S           ←  ここからソースプログラム
   10 READ (5,*,END=99) X
      M=X/60
      S=M-X*60
      H=M/60
      M=H-M*60
      WRITE (6,20) X,H,M
   20 FORMAT(1H,4X,I5,'秒=',I5,'時間',I5,'分')
      GO TO 10
   99 STOP
      END
/*
//FORTCGO.SYSGO DD *
  3567                          ←  ここからデータ
  2467
 13499
   245
/*
//
```

- Fortran ソースプログラム TEST.FORT を STEP=CGO で実行する．入力データ TEST.DATA．

```
//A79999AA JOB CLASS=A
//   EXEC FORT,STEP=CGO,OPT=B
//FORTCGO.SYSIN DD DSN=A79999A.TEST.FORT,DISP=SHR
//FORTCGO.SYSGO DD DSN=A79999A.TEST.DATA,DISP=SHR
//
```

- Fortran ソースプログラム TEST.FORT を STEP=CLG で実行する．データを機番 1 で入力する場合． TEST.FORT の内容は次の通り．

```
****** ******************************************************
          INTEGER X,H,A,M,S
       10 READ (1,*,END=99) X         ← 入力元機番を 1 にする
           H=X/3600
           A=X-3600*H
           M=A/60
           S=A-60*M
          WRITE (6,20) H,M,S
       20 FORMAT(1H ,3X,I2,'時間',I2,'分',I2,'秒')
          GO TO 10
       99 STOP
          END
****** ******************************************************
```

ジョブ制御文を次のように作成．

```
//A79999AB JOB CLASS=A
//   EXEC FORT,STEP=CLG,OPT=B
//FORT.SYSIN DD DSN=A79999A.TEST.FORT
//GO.FT01F001 DD *
3567                                 ← ここからデータ
2467
13499
245
/*                                   ← データの終り
//
```

- オブジェクトモジュール EXAM.OBJ(MEM) から，リンケージエディタによってロードモジュール EXAM.LOAD(MEM1) を作成する． EXAM.LOAD は新規に作成．

```
//A79999AC JOB CLASS=A
//   EXEC LKED
//LKED.SYSLIN   DD DSN=A79999A.EXAM.OBJ(MEM),DISP=SHR
//LKED.SYSLMOD  DD DSN=A79999A.EXAM.LOAD(MEM1),
//        UNIT=PUB,DISP=(NEW,CATLG),SPACE=(TRK,(10,10,1),RLSE)
//
```

- 作成したロードモジュール `SAMPLE.LOAD(TIME)` を実行する．入力データは `TEST.DATA`.

```
//A79999AD JOB CLASS=A
//   EXEC GO,PROG=TIME,LOADDS='A79999A.SAMPLE.LOAD'
//GO.SYSGO DD DSN=A79999A.TEST.DATA,DISP=SHR
//
```

- 私用ライブラリとしてのロードモジュールを作成する．新規に作成される `MYLIB.LOAD` に登録する．

```
//A79999AE JOB CLASS=A
//   EXEC FORT,STEP=CL,OPTION=NAME,
//   PARM.LKED='NCAL'               ← このソースプログラムの場合は
//FORT.SYSIN   DD *                   ALIAS をかかなくてもよい
      FUNCTION ADD(X1,X2)
      INTEGER ADD,X1,X2
      ADD=X1+X2
      END

      FUNCTION SUB(X1,X2)
      INTEGER SUB,X1,X2
      SUB=X1-X2
      END

      FUNCTION MUL(X1,X2)
      INTEGER MUL,X1,X2
      MUL=X1*X2
      END

      FUNCTION DIV(X1,X2)
      INTEGER X1,X2
      REAL DIV
      DIV=X1*1.0/X2
      END
/*
//LKED.SYSLMOD  DD DSN=A79999A.MYLIB.LOAD,DISP=(NEW,CATLG),UNIT=PUB,
//  SPACE=(TRK,(10,10,5),RLSE)
//
```

- `SSU`(cf. 1.3) を `VIO/F` ファイルとして使用する． `Fortran` ソースプログラムは `TEST.FORT`，入力データセットは `TEST.DATA`，作業用ファイルとして `FT01F001 (READ(1,*),WRITE(1,*))`

を利用．

```
//A79999AF JOB CLASS=F
//  EXEC  FORT,STEP=CLG
//FORT.SYSIN  DD DSN=A79999A.TEST.FORT,DISP=SHR
//GO.SYSIN    DD DSN=A79999A.TEST.DATA,DISP=SHR
//GO.FT01F001 DD UNIT=SSU,SUBSYS=(VPCS,'SPACE=100M'),DISP=(NEW,DEL)
//
```

- SSU 配列機能を VP2600 で使用する．Fortran ソースプログラムは TEST.FORT，入力データセットは TEST.DATA，なお TEST.FORT は SSU 使用を COMMON 文で宣言する必要がある．

```
//A79999AG  JOB  CLASS=V
//  EXEC  FORT,STEP=CLG,OPTION='SSU(共通ブロック名)',VP=YES
//FORT.SYSIN  DD  DSN=A79999A.TEST.FORT,DISP=SHR
//GO.SYSIN    DD  DSN=A79999A.TEST.DATA,DISP=SHR
//GO.SSARRAY  DD  UNIT=SSU,SUBSYS=(VPCS,'SPACE=400M'),DISP=(NEW,DEL)
//
```

- SSU を SSU 配列と VIO/F ファイルで使用する．

```
//A79999AH JOB CLASS=V
//  EXEC  FORT,STEP=CLG,OPTION='SSU(共通ブロック名)',VP=YES
//FORT.SYSIN  DD DSN=A79999A.TEST.FORT,DISP=SHR
//GO.SYSIN    DD DSN=A79999A.TEST.DATA,DISP=SHR
//GO.FT01F001 DD UNIT=SSU,SUBSYS=(VPCS,'SPACE=100M'),DISP=(NEW,DEL)
//GO.SSARRAY  DD UNIT=SSU,SUBSYS=(VPCS,'SPACE=300M'),DISP=(NEW,DEL)
//
```

7.5 メッセージを日本語で出力したい時は

Fortran プログラム翻訳時のメッセージを日本語で出力させたい場合，次のように制御文に追加をする．

`//A79999A1 JOB CLASS=A,`<u>`SPARM='LANG=J'`</u>

ただし，これを指定したジョブ内だけ有効である．SPARM='LANG=J' の指定を省略すると，英語のメッセージが出力される．

7.6 その他バッチジョブについては

ここでは，Fortran プログラムをバッチで動作させる環境について述べたが，用語の説明等は省略気味に解説している．バッチジョブに関しての詳細は [10] を，Fortran77 EX, Fortran77 EX/VP のオプションその他については [5] を参照．[5] については PLUM コマンドで

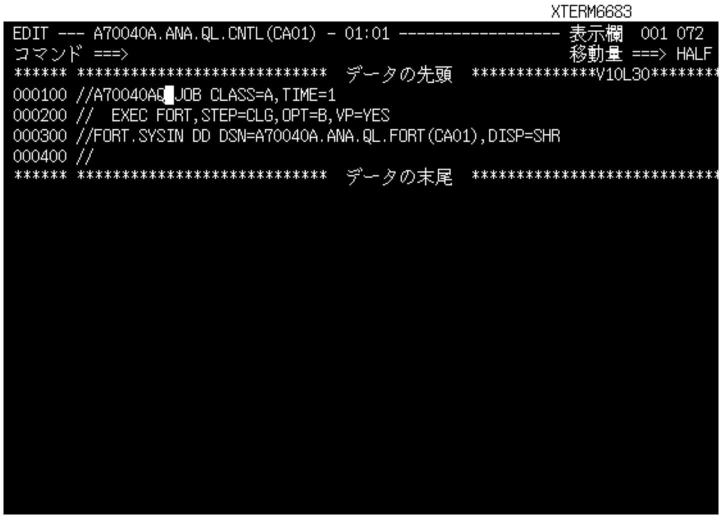
オンラインマニュアルとして検索できる．ただし，端末形式やエミュレータによっては起動できない場合があるので注意されたい．

7.7 ジョブ制御文はデータセットに書きます

ここまで示した“//”で始まるジョブ制御文は，タイプ名 CNTL のついたデータセットに記述すること．TSS のコマンドとして，例えば

```
 READY
//A79999AH JOB CLASS=V  ↵
```

などと打ちこむのではない．editor を起動し，データセットとして書き込むこと．以下は，PFD/EDIT での編集画面．



タイプ名 CNTL は必ず指定すること．編成は区分編成でも順編成でも構わない．

8　バッチジョブ関連で用いる TSS コマンド

8.1　SUBMIT

SUBMIT コマンドはバッチジョブを依頼するコマンドであり，バッチ処理では一番重要なコマンド．バッチジョブを依頼するためには，まずジョブ制御文をタイプ名 CNTL のデータセットに作成する．そして SUBMIT コマンドを用いてジョブを依頼する． SUBMIT は EDIT 内からの入力も可能で，その場合データセット名は不要である．

【入力形式】

SUBMIT **SUB**	データセット名

【使用例】

- ジョブ制御文を記述したデータセット EXAMPLE.CNTL について処理を依頼する．

```
 READY
SUB EXAMPLE ↵                                <--- .CNTL は省略可
 KEQ5602I ***        A79999AZ        :  (RECEIVED)   ***
 KEQ5602I JOB A79999AZ(JOB09999) SUBMITTED
 READY
```

- PFD/EDIT 内からジョブを依頼する． SUB と打つだけで，データセット名は省略出来る．

```
EDIT --- A79999A.EXAMPLE.CNTL ---------------------- 表示欄   001 072
コマンド ===> SUB  ↵    … バッチ処理の依頼．            移動量 ===> CUR
****** *********************** データの先頭 ************ V10L30 ******
000001 //A79999AZ JOB CLASS=V,TIME=10
000002 //  EXEC FORT,STEP=CLG,VP=YES,OPT=E
000003 //FORT.SYSIN  DD  DSN=A79999A.EXAM1.FORT,DISP=SHR
000004 //GO.SYSIN    DD  DSN=A79999A.EXAM1.DATA,DISP=SHR
000005 //
****************************** データの末尾 *************************
```

【注意】

バッチジョブを行なう場合， SUBMIT コマンドでジョブを依頼するデータセット名は，タイプ名 “CNTL” が必要である．順編成，区分編成は問わない．

[例]
EXAMPLE.CNTL
FEM.CNTL(EX1)

8.2 STATUS

STATUS コマンドは，依頼したバッチジョブの処理状況（実行待ち，実行中，出力待ち）を表示する．

【入力形式】

STATUS
ST

【使用例】

```
 READY
ST ↵
 KEQ56192I JOB A79999A2(JOB06492) IS WAITING FOR EXECUITION    ← 実行待ち
 KEQ56192I JOB A79999A2(JOB06495) IS EXECUTING                 ← 実行中
 KEQ56221I JOB A79999A#(TSU07684) IS EXECUTING ON THIS SESSION ← 現 session
 KEQ56192I JOB A79999A1(JOB06490) IS WATING FOR OUTPUT         ← 出力待ち
```

8.3 STATE

STATE コマンドは現在の日時，現在の TSS 利用者数，バッチジョブの実行状況および実行待ち件数を表示する．

【入力形式】

STATE

【使用例】

```
 READY
STATE ⏎

TIME=10.46.16  DATE=94.02.04
TSS USER 0029
JOBNAME  STEP     SNO CLS REGION E-REGION CPU TIME  START    ACCEPT         SYS
 A79999A2 GO         3  F  7232KB    193MB 00:07:11  10:37:32 10:37:20 02/04  M
 A79999AC GO         3  B  8192KB     41MB 00:57:33  06:21:58 17:37:43 02/03  V

  ***  YOUR JOB WAITING FOR EXECUTION *** AT 02/04/94 10:46:17
  JOBNAME(JOBNO)   CLASS   SYSTEM   ORDER   CLASSTOTAL
 A79999AA(J4961)    *F      M         0          1

 ***** INFORMATION OF WAITING JOB FOR EXECUTION 10:46 ON 02/04/94 *****
 * : JOBCLASS :     A     :  B  :     F     :     V     :  L  :  N  : *
 * :   TYPE   :  S  :  L  :     :  S  :  L  :  S  :  L  :     :     : *
 * +----------+-----------+-----+-----------+-----------+-----+-----+ *
 * :  M1800   :   0 :   0 :   0 :   0 :   2 :    ---    :   0 :   0 : *
 * :  VP2600  :   0 :   0 :   3 :    ---    :   0 :   9 : --- : --- : *
 **********************************************************************
 READY
```

【注意】

- 上記表の “TYPE” は TIME パラメータの指定によって振り分けられる．具体的な TIME パラメータの設定方法および注意については 9.6 を参照．
- また，表は「実行待ち」の人数の合計である．実行中のジョブは含まれていない．このため，例えば数字が 0 のジョブクラスに，この画面の直後に投入しても，実行中のジョブが存在していたときは，直ちに実行処理されない場合がある．

8.4 MSO

MSO は，バッチジョブをメニュー形式で出力検索する．フルスクリーン端末でバッチジョブの操作（状態表示・検索・消去・出力）がメニュー形式で行える．動作は TTY 端末，FNVT 上でも可能．

【入力形式】

MSO

【使用例】

MSO コマンドを実行すると，次に示すような画面が表示される．バッチジョブがない時は，何も表示されない．

```
-----------------------<  M S O  V02/L02  >--------------- BY KYUSHU UNIV.
                                                                 94.06.05
                     YOUR JOBS ARE AS FOLLOWS.

(SEQ-NO.) (JOBNAME)                     (SEQ-NO.) (JOBNAME)
    1     A79999A1 (J3362)       (1)
  * 2     A79999AF (J3881) EXEC  (2)
  * 3     A79999AH (J3883) WAIT  (3)
  * 4     A79999AH (J3882) WAIT

 SEQ-NO.   ==>  (4)                         SORP-DSPRINT(PF5) DATASET
 OPTION    ==>  (5)   < B,NB,E,NE,C >                ==>   (7)
 NEW CLASS ==>  (6)   < H,O,K,S,E,U OPR-NO >
 TSS       ==>
---------------------------------------------------------------------------
         PF1 ==> HELP  PF2 ==> RE-FRESH  PF3 ==> END  ← (8)
```

表示後の画面の説明

(1) 出力待ちのジョブであることを示す．

(2) 実行中のジョブであることを示す．

(3) 実行待ちのジョブであることを示す．

(4) 画面に表示されているジョブ一覧の番号を選択する．特に 99 を指定すると，次項 (5) の C オプション，あるいは，(6) の項を指定した場合に限り，すべてのジョブが対象となる．ただし 1 度選択したジョブは対象とならない．

(5) 出力検索時の環境を指定する．

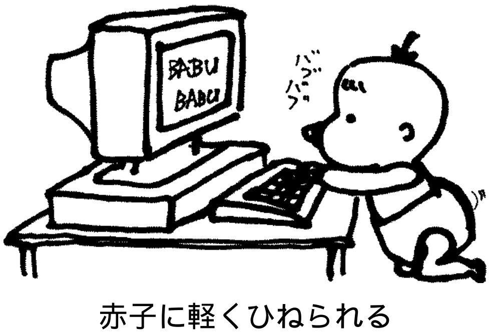
空白	..... SORP コマンドによる出力検索を行う．
B	..... PFD の BROWSE オプションの出力検索を行う．
NB	..... PFD の日本語 BROWSE オプションの出力検索を行う.
E	..... PFD の EDIT オプションの出力検索を行う．

NE　　..... PFD の日本語 EDIT オプション風の出力検索を行う．
C　　..... (4) で選択したジョブをキャンセルする．

(6) ジョブを出力する場合の出力クラス，あるいは，OPR のプリンタ名を選定する．
(7) SORP コマンド実行時にページ抽出するデータセットを指定する．
(8) PF1 を押すと HELP 画面が表示される．PF2 を押すとジョブの状態の再検索および画面の再出力を行う．PF3 を押すと MSO コマンドは終了する．

【留意事項】

1. TTY 型端末で TSSPFD エミュレータを使用される方は最新のエミュレータを使用すること. 最新のエミュレータはセンター 2 階オープン室でコピー可能．
2. MSO コマンドについては，広報 No.23, No.5 を，また SORP コマンドの使用法については，TSS で MANUAL コマンド（MANUAL SORP）を入力し「SORP コマンド使用の手引」を出力して参照されたい．
3. バッチの出力結果は 2 週間を過ぎると強制的に消去されるので，必要なものはその期限内に出力すること．
4. PFD3.8(cf. 8.5) でも検索が可能である．
5. PFD の日本語 EDIT オプション風検索オプションで検索内容を編集しても，実行結果が書き変わることはない．



赤子に軽くひねられる

8.5 PFD の OUTLIST を利用したジョブの出力

PFD を利用し，フルスクリーン端末からメニュー形式でバッチジョブの出力検索を行う．

【入力形式】

PFD 3.8

【使用例】

```
 REDAY
PFD 3.8  ↵
```

```
-----------------------<  OUTLIST ユティリティ  >-------------------------
オプション ===>

   D    - ジョブの出力結果をスプールから削除する。
   L    - ジョブの状態を通知する。
   P    - ジョブの出力結果をプリントしてスプールから削除する。
   R    - ジョブの出力クラスを変更する。
   E    - 日本語データを含まないジョブの出力結果を編集する。
   J    - 日本語データを含むジョブの出力結果を編集する。
   N    - 日本語データを含むジョブの出力結果を表示する。
   空白 - 日本語データを含まないジョブの出力結果を表示する。

以下のパラメタを指定して下さい。
  SYSOUT クラス     ===>
  ジョブ識n別番号 ===> J6492
  新 SYSOUT クラス ===>                 (オプション'R'を選択した場合)
  印刷制御文字      ===>                (A-ANSI,M-MACHINE,BLANK-NONE)
--------------------------------------------------------------------------
```

使用方法は次の通り．

- オプション欄に ST コマンドを入力してジョブの実行状態を知り，ジョブ名およびジョブ識別番号を確認する．同一ジョブ名が 1 つしかない時はジョブ名だけでよい.
- ジョブ名 (およびジョブ識別番号) を入力．
- オプションとして N または 空白を入れると結果の表示．
- オプションとして E または J を選択すると表示結果の編集が可能．出力として不要な部分を削除，種々のコメントの追加等が可能．
- MSO と比較して，表示部分の移動がかなり簡単．

8.6 CANCEL

起動したバッチジョブ (FIB ジョブ[3]) を取り消す．また，マルチセッション下での TSS セッションのキャンセルも行なえる．

【入力形式】

CANCEL	[ジョブ名] [**P**urge]

【オペランドの説明】

ジョブ名 : 取り消すジョブ名を指定する．
PURGE : ジョブの実行を取り消し，さらに実行結果リストも消去することを指定する．

【使用例】

- ジョブ A79999AB を取り消す．

```
 READY
CANCEL A79999AB  ↵
```

- ジョブ ID が同じ場合は (JOB*****) まで指定する必要がある．
  ジョブ A79999AK(JOB00002)，A79999AK(JOB00010) のうち，A79999AK(JOB00002) をキャンセルする．出力 HOLD キュー，出力キューもその対象とする．

```
 READY
CANCEL A79999AK(JOB00002) P  ↵
```

- 3 個のジョブを A79999AB，A79999AX，A79999A9 を取り消す．

```
 READY
CANCEL (A79999AB A79999AX A79999A9)  ↵
```

【注意事項】

- 起動したジョブの状態を知るにためには，STATUS コマンドを用いる．

[3] TSS のもとでバッチジョブを端末から入力する処理形態

8.7 もっと簡単にバッチで実行したいが

バッチで Fortran プログラムを実行するためには，ジョブ制御文なるものが必要なのは既に説明した．その補助として，簡単な制御文ならば必要なオペランドを指定するだけで，制御文を自動作成し実行までしてくれるありがたい TSS コマンド AF77 がある．AF77 は Fortran プログラムを M1800，または VP2600 上でバッチ処理するためのコマンドである．

【入力形式】

```
AF77    データセット名
        [ JOBID ( { ジョブ ID | A } ) ]
        [ CLASS ( { ジョブクラス | A } ) ]
        [ MSGCLASS ( { A | H | K | S | O | U } ) ]
        [ MSGLEVEL ( '{ 0 | 1 | 2 }, { 0 | 1 } ' ) ]
        [ TIME ( ' 分 , 秒 ' ) ]
        [ VPREGION ( ' リージョンサイズ ' ) ]
        [ COPT ( 翻訳時オプション ) ]
        [ LOPT ( 結合・編集時オプション ) ]
        [ GOPT ( 実行時オプション ) ]
        [ OPT ( { B | E | F } ) ]
        [ LIB ( 私用ライブラリ名 [, ... ] ) ]
        [ UNIT(装置番号 : { データセット名 | 出力クラス | * } ]
        [ LOAD ( ロードモジュール名 ) ]
        [ C ]
        [ VP ]
        [ JAPANESE ]
        [ MAIL ]
        [ N ]
```

オペランドの説明

データセット名

Fortran ソースプログラムのデータセット名を指定．

JOBID

ジョブ ID（0 ~ 9，A ~ Z）を指定する．省略すると A となる.

CLASS

ジョブクラスを指定する．省略すると A となる．

MSGCLASS

システムメッセージを出力する装置を出力クラスにより指定．

MSGLEVEL

システムメッセージの出力レベルを指定する．

TIME

ジョブクラスによる制限の範囲内で，ジョブ実行の打ち切り CPU 時間を指定する．省略すると各ジョブクラスの制限値となる．

VPREGION

VP オペランドを指定した場合に有効で，実行時のリージョンサイズを指定する．省略すると各ジョブクラスの標準値となる．

COPT

翻訳時オプションを指定する．

LOPT

結合編集時オプションを指定する．

GOPT

実行時オプションを指定する．

OPT

翻訳時の最適化レベルを指定する．

LIB

組み込みたい私用ライブラリのデータセット名を指定する．

UNIT

入出力に使用する装置番号と，それに対応させるデータセット名または出力クラスを指定する．

LOAD

翻訳および結合編集されたロードモジュールを格納するデータセット名を指定する．本オペランドを指定すると，プログラムの実行は行われない．

C

翻訳だけ行う．本オペランドを指定すると， LOAD オペランドは無視される．

VP

VP で処理を行う．本オペランドを指定しないと， M1800 で処理される．

JAPANESE

Fortran メッセージを日本語で出力する．

MAIL

実行結果を自動的に出力し，返却を連絡所送りとする．

N

ジョブ制御文の作成のみを行う．制御文は `@JOB.SUBMIT.CNTL` に作成される．指定しなければ，このデータセットは自動的に消去される．

【注意】

1. AF77 コマンドで指定するファイルは，すべて既存のものでなければならない．
2. オペランドに複数の値を指定する場合は，その値をコンマ（,）で区切り，全体を引用符（'）で囲んで指定する．引用符の中に引用符を指定する場合は，引用符を 2 つ続けて指定する．

   ［例］
   ```
   LIB('MYLIB.LOAD,''APP1.TACLIB.LOAD''')
   UNIT('6:PROG.LIST(DATA1),5:PROG.DATA(DATA1)')
   ```

3. COPT，LOPT，および，GOPT オペランドに括弧 '( )' を含むオプションを指定する場合は，その値は引用符で囲んで指定する．

   ［例］
   ```
   COPT(NOSOURCE)
   COPT('JCONST(N)')
   COPT('FREE,NOSOURCE,''JCONST(N)''')
   ```

4. 機番 6 番からデータを出力する場合（WRITE(6,*)）は，UNIT オペランドで指定する必要はない．

【AF77 の使用例】

- TEST.FORT を M1800 でバッチ処理する．なお，機番 5 番から PROG.DATA をデータとして読み込む．

```
 READY
AF77 TEST.FORT UNIT(5:PROG.DATA)  ↵
 ****** A79999AA (J1234) A79999A : (JOB ACCEPTED) *** FIB   CN(01)

 READY
STATUS  ↵                                   <---- 処理中かどうかの確認
 KEQ56221I JOB A79999A#(TSU0123) IS EXECUTING
 KEQ56192I JOB A79999AA(JOB1234) IS WAITING FOR OUTPUT
 READY
```

- TEST.FORT を VP 側で実行する．なおジョブ ID を 1，ジョブクラスを V とする．

```
 READY
AF77 TEST.FORT JOBID(1) CLASS(V) VP   ↵
```

- TEST.FORT を VP 用に翻訳，結合編集したロードモジュールを PROG.LOAD のメンバ TEST に格納する．

```
 READY
AF77 TEST.FORT LOAD(PROG.LOAD(TEST)) VP   ↵
```

- PROG.FORT のメンバ KEISAN を実行する．機番 1 番からデータ INPUT.DATA を読み込み，出力されるデータを機番 2 番に割り当てた OUTPUT.DATA に格納する．

```
 READY
AF77 PROG.FORT(KEISAN) UNIT('1:INPUT.DATA,2:OUTPUT.DATA')  ↵
```

- TEST.FORT の翻訳のみを行う．翻訳時オプションに NOSOURCE(ソースプログラムはつけない意味) と FREE(ソースプログラムの形式は自由形式) を指定する．また，Fortran メッセージを日本語で出力する．

```
 READY
AF77 TEST.FORT C COPT('NOSOURCE,FREE') JAPANESE   ↵
```

- PROG.FORT のメンバ COMPUT を VP で実行するためのジョブ制御文を作成する．機番 5 番からデータ INPUT.DATA を入力し，6 番から出力されるデータはリストとして得る．

```
 READY
AF77 PROG.FORT(COMPUT) UNIT(5:INPUT.DATA) N VP  ↵
```

9　VP のカタログドプロシジャ紹介

ここでは VP2600 上で Fortran プログラムを実行させるためのカタログドプロシジャについて説明する．カタログドプロシジャは FORT であり，Fortran77 EX/VP コンパイラによる翻訳，結合編集，実行を行う．

FORT の形式は，VP を用いない通常の FORT と VP=YES の有無 で区別される
他の大部分は共通である．詳細は [9] を参照．

	主なパラメータ
FORT	[, SYSOUT = { A ❙ K ❙ S ❙ O ❙ U ❙ H }] [, STEP = { C ❙ CGO ❙ CG ❙ CL ❙ CLG }] [, PRVLIB = ' データセット名'] [, OPTION = ' コンパイラオプション'] [, VREGION = 'm'] [, OPT = { B ❙ E ❙ F }] ,VP=YES

【記号パラメータの説明】

SYSOUT　：出力クラスを指定する．省略値は A．
STEP　：処理過程を選択する．省略値は CLG．
　C　.... 翻訳のみ行う．
　CGO .... 翻訳から実行までを 1 ステップで行う．
　CG　.... 翻訳，ローダによる結合編集，実行を行う．
　CL　.... 翻訳，リンケージエディタによる結合編集まで行い，
　　私用ライブラリあるいはロードモジュールを作成する．
　CLG .... 翻訳，リンケージエディタによる結合編集，実行を行う．
PRVLIB　：組み込みたい私用ライブラリのデータセット名を指定する．
　私用ライブラリも VP 上で翻訳されていたほうがよい．
OPTION　：コンパイラオプションを指定する．
VREGION　：VP での実行時のリージョンサイズを指定する．
　単位 MB である．省略すると各ジョブクラスの制限値となる．
OPT　：最適化レベルを指定する．省略値は B．
VP　：VP2600 で実行する時必ず YES を指定する．

オプション	プロシジャステップ	関連する DD 名	データセット
STEP=C	FORT	FORT.SYSIN	ソースプログラム
STEP=CL	FORT	FORT.SYSIN	ソースプログラム
	LKED	LKED.SYSIN	リンケージエディタ制御文
STEP=CLG	FORT	FORT.SYSIN	ソースプログラム
	LKED	LKED.SYSIN	リンケージエディタ制御文
		LKED.SYSLMOD	ロードモジュール保存
	GO	GO.SYSIN	実行時の入力データ

（注意） VP の場合なるべく STEP=CLG を使用すること．

9.1 Fortran77/VP オプションパラメータの指定例

- FORT で DEBUG オプションを指定する時は，同時に NOADV を指定する．

```
//  EXEC FORT,STEP=C,OPTION='NOADV,DEBUG(ARGCHK)',VP=YES
```

- 副プログラム（プログラム名 TENKAI）の内部展開による最適化の指定．

```
//  EXEC FORT,STEP=CLG,OPTION='PI(TENKAI)',VP=YES
```

- リスト情報を出力しない．

```
//  EXEC FORT,STEP=CLG,OPTION='NOPRINT',VP=YES
```

- ベクトル化表示プログラムリストを出力しない．

```
//  EXEC FORT,STEP=CLG,OPTION='NOSOURCE',VP=YES
```

9.2 FORT の使用例 (VP)

ジョブの投入，検索のコマンドはすべて M1800 のバッチ処理と共通．ジョブの投入は SUBMIT コマンドで行なう．

```
EDIT --- A79999A.VP.CNTL -------------------------- 表示欄   001 072
コマンド ===> SUB  ⏎   ... バッチ処理の依頼．          移動量 ===> CUR
****** *********************** データの先頭 ************ V10L30 ******
000001 //A79999AX JOB CLASS=A
000002 //  EXEC FORT,STEP=CLG,VP=YES,OPT=E
000003 //FORT.SYSIN  DD DSN=A79999A.EXAM1.FORT,DISP=SHR
000004 //GO.FT06F001 DD DSN=A79999A.EXAM1.LIST,UNIT=PUB,
000005 //          DISP=(NEW,CATLG),SPACE=(TRK,(10,10),RLSE),
000006 //          DCB=(LRECL=137,BLKSIZE=23427,RECFM=FB)
000007 //GO.FT05F001 DD  DSN=A79999A.EXAM1.DATA,DISP=SHR
000008 //
****** *********************** データの末尾 *************************
```

```
KEQ56208I ***         A79999AX         : (RECEIVED) ***
        *** A79999AX(J1234)A79999A : (JOB ACCEPTED) *** FIB CN(01)
KEQ56250I JOB A79999AX(JOB01234) SUBMITED  ***
```

9.3 長時間のジョブはどの STEP で実行すべきか

九州大学大型計算機センターでは，通常，サービス終了時間になるとジョブ処理を中断し，計算機の電源を切断している．このため，これまでは実行中断されたジョブは翌日にジョブステップの最初から実行を再開し，前日のジョブステップの CPU 時間が無駄になっていた．

1992 年 7 月 13 日より，Savehalt 機能を新規に導入したことによって，ジョブのターンアラウンド時間が短くなり，数日にわたるジョブの連続実行が可能となった．

ただし，STEP=CGO, STEP=CG 等の様に 1 つのステップで複数の処理をおこなうマルチタスクは Savehalt の対象とならない．

従って，Fortran ジョブステップで STEP=CGO, STEP=CG が指定された場合，従来通りのジョブステップリスタートとなり，実行を中断されたジョブは最初から再実行となる． CPU 時間の長いジョブは省略値である STEP=CLG を指定されたい．

9.4 最適化オプションについて

特に VP 上で Fortran プログラムを実行させる場合，プログラムによっては最適化レベルの設定によって，実行時間が大幅に異なる可能性がある． DO ループを複雑に絡み合わせた大規模計算を企む利用者は，以下の文章を必ず読むこと．

- 最適化の目的
  最適化の目的は，プログラムを可能な限り高速に実行できる命令列およびデータ域を生成することにある． “強め” の最適化オプションを指定することで，プログラムの実行時間が短縮できる．

- 最適化のレベル
  OPT=NO，OPT=B，OPT=E，OPT=F の 4 つの段階がある．ただし，VP 上では OPT=NO は指定できない．さらに，コンパイルオプション XOPT，INLINE を指定することで機能の一部の調節が可能である．

- 各最適化レベルの特徴

  - OPT=NO
    限定された最適化が行われる．
    デバッグ（プログラムの誤りを見つけて修正および確認する作業）やデバッグが済んだ小・中規模プログラムの翻訳と実行を何度も繰り返して行う場合に適している．

  - OPT=B
    局所的な最適化に加えて，広域的な最適化も行われる．
    正しいプログラムであれば OPT=NO と同じ結果が得られる．デバッグが済んだ中・大規模なプログラムを高速に実行するためのオブジェクトモジュール（計算機が直接実行可能な機械語の集合）を作成する場合に適している．

  - OPT=E
    OPT=B での機能に加えて実行結果に副作用の生じる可能性がある最適化も実施される．正しいプログラムであり，不変式の先行評価と演算評価方法の変更による影響がなければ，OPT=NO と同じ実行結果が得られる．適するプログラムは OPT=B と同じ．

– <u>OPT=F</u>

OPT=E の機能に加えて，プログラム単位間の最適化も実施される．正しいプログラムであり，不変式の先行評価と演算評価方法の変更による影響がなければ，OPT=NO と同じ実行結果が得られる．適するプログラムは OPT=B と同じ．

- OPT=E, OPT=F の設定に伴う副作用について

最適化オプション OPT=E, OPT=F は，以下に示す副作用の伴う可能性のある最適化を行っている．もちろんできるだけ副作用が生じない範囲に限定された最適化であり，実際に副作用が発生する確率は極めて少ないと考えられる．またコンパイラオプション OPTMSG を指定することで，副作用の有無を示すメッセージを出力させることができる．

【 副作用 1 】

OPT=E, OPT=F を指定した場合，不変式の先行評価を行う．この場合，プログラムの論理上は実行されないはずの命令が実行され，エラーになる場合がある．ただし，計算結果およびその精度には影響ない．

【例】

```
    DO 10 I=1,100
     IF(X.GE.0.0) THEN
      A(I)=SQRT(X)
     END IF
10  CONTINUE
```

IF 文で SQRT の引数が 0.0 以上のときだけ関数を呼び出すので OPT=B, OPT=NO では副作用は生じない

最適化後のソースイメージ

```
    T=SQRT(X)
    DO 10 I=1,100
     IF(X.GE.0.0) THEN
      A(I)=T
     END IF
10  CONTINUE
```

不変式 SQRT(X) の最適評価によって，SQRT(X) がループ外に移動される．その結果 X の値が負のとき SQRT で引数が 0.0 以下であるメッセージが出力されてエラーとなる

対策として OPT=B または OPT=NO を設定して最適化レベルを下げるか，OPT=E のままコンパイルオプションで XOPT(NOPREEX) と指定することでエラーを回避できる．

【副作用 2 】

同じく OPT=E, OPT=F を指定した場合，演算評価方法の変更によって計算誤差に影響が生じることがある．また，非常にまれに指数桁あふれなどの演算例外が発生することがある．

【例】

```
    DO 10 I=1,100
     A(I)=B(I)/C
10  CONTINUE
```

変数 C はループ内で一定．演算評価の変更により C の逆数をループの外側で計算する．

最適化後のソースイメージ

```
    T=1.0/C              T=1.0/C の計算結果が C の型が持つ精度内に納まらない
    DO 10 I=1,100        場合は OPT=B や OPT=NO に対して精度差が生じる．
     A(I)=B(I)*T
10  CONTINUE
```

【最適化オプションの指定例】

– Fortran プログラム 'A79999A.TEST.FORT' を VP2600 上で実行させる．

```
//A79999A1 JOB CLASS=V
//  EXEC FORT,STEP=CLG,VP=YES
//FORT.SYSIN DD DSN=A79999A.TEST.FORT,DISP=SHR
//
```

VP2600 上で最適化オプションを陽に指定しないジョブを翻訳する場合，最適化レベルは OPT=B で実行される．

– 最適化オプションを利用者自身で指定する例

```
//A79999A2 JOB CLASS=V
//  EXEC FORT,VP=YES,STEP=CLG,OPT=E
//FORT.SYSIN DD DSN=A79999A.TEST.FORT,DISP=SHR
//
```

– 最適化オプションを指定して，実行結果に影響がある可能性のある最適化を行う場合，その旨のメッセージを出力させる例

```
//A79999A3 JOB CLASS=V
//  EXEC FORT,VP=YES,STEP=CLG,OPT=F,OPTION='OPTMSG'
//FORT.SYSIN DD DSN=A79999A.TEST.FORT,DISP=SHR
//
```

VP2600 上，または M1800 上で Fortran プログラムを OPT=E, OPT=F で実行する場合は，OPTION='OPTMSG' を指定することでメッセージが確認できる．大規模なプログラムを実行させる場合，最適化のレベルやプログラムによっては翻訳・実行時間にかなりの差が出る．センターの最適化レベルの省略値は M1800, VP2600 ともに OPT(B) である．特に VP 上での Fortran プログラムの実行では，上記副作用の発生する可能性を考慮した上で，OPT(E), OPT(F) を指定することで，実行の高速化が可能である．センターの省略値を信用することなく，利用者自身で最適化レベルを積極的に設定するようお勧めする．

9.5 コンパイラオプションについて

Fortran77 EX, Fortran77 EX/VP の翻訳時のオプションは，前節の最適化オプションの他にも他数サポートされている．詳細は [5], [6] を参照．

9.6 TIME パラメータについて

Fortran プログラムを `CLASS=V` で実行させる場合， CPU 時間が 10 分以下であれば，`TIME` パラメータを陽に 10 以下に指定することで，ジョブの優先度を高めることができる．ただし，`TIME` で定めた値を越えた場合，処理は打ち切られるので注意．

また，ベクトル化率が低いジョブは M1800 の `CLASS=F` にジョブを依頼した方が結果が早く得られる場合もある．

使用リージョンと CPU はジョブの処理結果，およびプリントアウトされるセパレータに印刷されている．これを参考に最適のジョブクラスを選択されたい．

【TIME パラメータ指定例】

```
//A79999AK JOB CLASS=V,TIME=10
//  EXEC FORT,VP=YES,STEP=CLG,OPT=E,OPTION='OPTMSG'
//FORT.SYSIN DD DSN=A79999A.TEST.FORT,DISP=SHR
//
```

10 知っていると便利なコマンド

10.1 ATTRIB

ATTRIB コマンドは ALLOCATE コマンドで割り当てるデータセットの属性リストを，前もって作成し，その名前をシステムに登録しておく．

なお，1993 年 4 月からのレベルアップより，ATTRIB コマンドで行なっていたデータセットの属性指定のオペランドが ALLOCATE コマンドに追加された．従って，ATTRIB コマンドの機能は全て ALLOCATE コマンドに包含されている．

【入力形式】

ATTRIB	属性リスト名
ATTR	[**BL**ksize (ブッロク長)]
	[**LR**ecl (論理レコード長)]
	[**DS**org (PO | PS)]
	[**REC**fm (レコード形式)]
	[**I**nput]
	[**REU**se]

【オペランドの説明】

属性リスト

属性リストに対してシステムに登録する名前を指定する．

BLKSIZE

データセットを構成するブロック長を指定する．

LRECL

データセット中のブロックを構成する論理レコードの最大長を指定する．

DSORG

データセットの編成を指定する．

PO : 区分データセット
PS : 順データセット

RECFM

データセットのレコード形式を指定する．

INPUT

データセットを入力に用いることを指定する．

REUSE

一度割当てた DD 名をいったん解除してから再割当てを行う．

【使用例】

- 固定長ブロック化形式の順編成データセットの属性リスト ABC を新規に作成する．レコード長 80 バイト，1 ブロックの大きさ 23440 バイト．更に ALLOCATE コマンドで，属性 ABC を用いてデータセット PS.DATA を作成する．

```
  READY
 ATTR ABC REC(F B) BL(23440) LR(80) REU   ↵
  READY
 ALLOC DA(PS.DATA) NE CA SP(10 10) T US(ABC)  ↵
```

- 可変長ブロック化形式の区分編成データセットの属性リスト XYZ を新規に作成する．レコード長 255 バイト，1 ブロックの大きさ 23440 バイト．更に ALLOCATE コマンドで，属性 XYZ を用いてデータセット PO.TEXT(EX1) を作成する．

```
  READY
 ATTR XYZ REC(V B) BL(23440) LR(255) REU  ↵
  READY
 ALLOC DA(PO.TEXT(EX1)) NE CA SP(10 10) T DIR(5) US(XYZ)  ↵
```

ATTRIB コマンドは ALLOCATE コマンドの中で引用して初めて効果がある．

10.2 FREE

FREE コマンドは，セッション中不要になったデータセットの割り当てを解放するのに用いる．また ATTRIB コマンドで作成，登録した属性リストを削除するのにも使う．割り当てを解放した後，再度 ALLOCATE コマンドで割り当てを行ないたい時は ALLOCATE コマンドの REUSE オペランドを用いた方が便利である．

【入力形式】

FREE	[**F**ile (DD 名 ...) | **DD**name (DD 名 ...)] [**DA**taset (データセット ...)] [**ATTR**list (属性リスト名 ...)]

【オペランドの説明】

FILE(DD 名)

解放するデータセットなどの DD 名を指定する．DD 名を複数個指定すると，一つ一つに対して解放処理を行う．

DDNAME(DD 名)

解放するデータセットなどの DD 名を指定する．ログオンプロシジャや ALLOCATE コマンドなどで連結されたデータセットのグループに与えられた DD 名を指定した場合，連結グループに属する全てのデータセットの解放処理を行う．DD 名を複数個指定すると，一つ一つに対して解放処理を行う．

DATASET(データセット)

解放するデータセットを指定．

ATTRLIST(属性リスト)

解放する属性リストを指定．

10.3 PROFILE

PROFILE コマンドは Fortran プログラム翻訳時のメッセージを日本語で出力させたい場合に使う．

【入力形式】

PROFILE	[TL(J)]
PROF	[TL(E)]

【オペランドの説明】

TL(J)

メッセージを日本語で出力する．

TL(E)

メッセージを英語で出力する．

【使用例】

```
 READY
PROF TL(J)  ↵
```

この指定は，以後メッセージを英語で出力するコマンド PROF TL(E) を入力するまで有効となる．

11　エラーメッセージと対策

この節は，Fortran プログラムの実行の際に起きるエラーと，その対策を具体的な例に沿って解説する．

11.1　LM コマンド — Fortran メッセージの検索 —

MSP の Fortran77 EX，および MSP システムの発行するメッセージは，`LM` コマンドによって端末からメッセージの意味を調べることができる．

【入力形式】

LM	メッセージ ID [LINE]

【オペランドの説明】

メッセージ ID
: Fortran コンパイラ，MSP システムが発行するメッセージを入力する．

LINE
: TTY 手順日本語端末の場合必ず指定．パソコン端末の場合も必ず指定する．指定しない場合，画面が乱れて出力される場合がある．

【使用例】

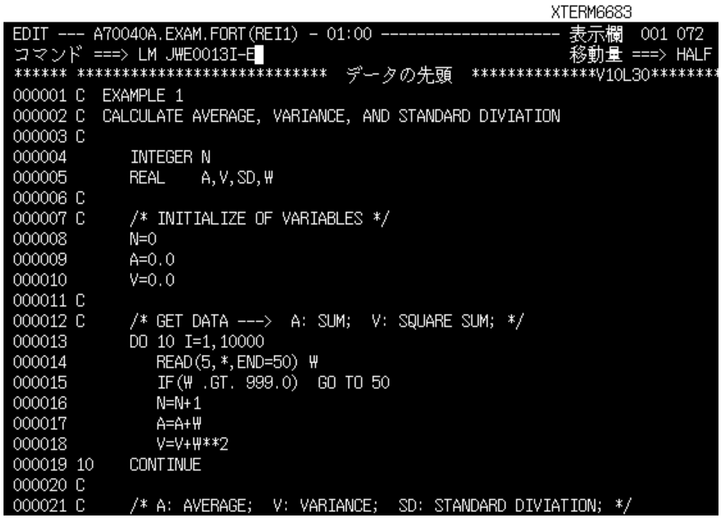
Fortran プログラムを TSS で実行中に，次のエラーメッセージが出力された．

```
                                                     XTERM6683
 【FORTRAN77 EX 翻訳開始】
 【翻訳終了】，完了コード＝00
 00140 ?
/*
 JWE0013I-E 浮動小数点数の除算で除数が0です.PSWは0F8D0000802FB33Cです.
           INSTRUCTION=3D02              LOCATION=802FB33A
           GR0 =00241C70  GR1 =002FB264  GR2 =00241D38  GR3 =002FB070
           GR4 =002FB2B8  GR5 =002FB070  GR6 =003058D8  GR7 =00000000
           GR8 =00000000  GR9 =00000001  GR10=00000000  GR11=00002710
           GR12=002FB2B8  GR13=002FB070  GR14=002FB324  GR15=002FD798
           FR0 =0000000000000000         FR2 =0000000000000000
           FR4 =0000000000000000         FR6 =0000000000000000
  ERROR OCCURS AT MAIN     SSN 00000220     LOC 002FB33A OFFSET 00000336
  MAIN        AT LOC 802FB000 CALLED FROM O.S.
 TAKEN TO (STANDARD) CORRECTIVE ACTION, EXECUTION CONTINUING.
   AVERAGE            = 0.00000000E+00
   VARIANCE           = 0.00000000E+00
   STANDARD DIVIATION = 0.00000000E+00
 ERROR SUMMARY (FORTRAN77 EX)
 ERROR NUMBER  ERROR LEVEL  ERROR COUNT
   JWE0013I        E           1
 TOTAL ERROR COUNT =     1
 【実行終了】，完了コード＝08
 ***
```

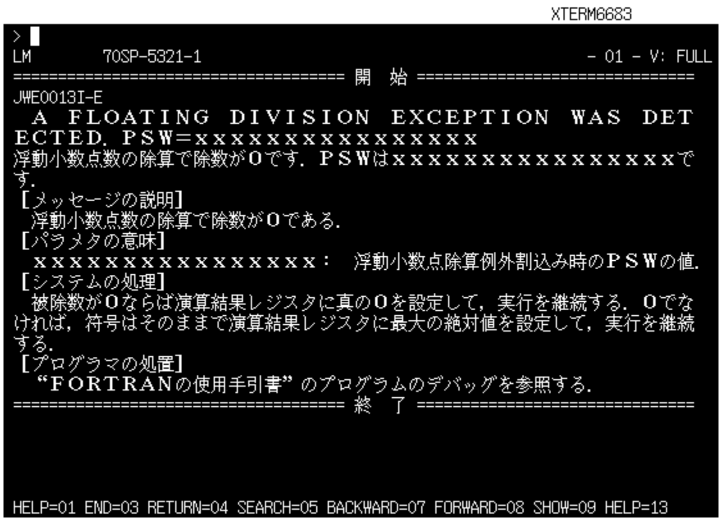
エラーの原因は，思いっきり表示されているが，とりあえず原因不明として先に進む．

11.1.1　センター内端末の場合

フルスクリーン環境が使用できるセンター内専用端末の場合は，`LM` コマンドに続けてエラーコードを入力する．この場合は“`JWE0013I-E`”の意味を調べる．`LM` コマンドは，READY モードだけでなく，次のように，PFD の EDIT 画面からもそのまま入力できるので便利．



`LM` コマンド入力後，画面が切り替わり“`JWE0013I-E`”の意味が表示される．



エラーメッセージの内容を参考にプログラムの修正を行なう．もとの編集画面に復帰するには PF3 キーを入力する．

11.1.2　パソコン端末の場合

パソコンからモデム等を介して MSP に接続している場合は，`LINE` オペランドを付加する．表示されるメッセージは同じ．

XTERM6683

```
 READY
LM JWE0013I-E LINE
 JWE0013I-E
  A FLOATING DIVISION EXCEPTION WAS DET
 ECTED. PSW=xxxxxxxxxxxxxxxx
 浮動小数点数の除算で除数が0です. PSWはxxxxxxxxxxxxxxxxで
 す.
  [メッセージの説明]
   浮動小数点数の除算で除数が0である.
  [パラメタの意味]
   xxxxxxxxxxxxxxxx:  浮動小数点除算例外割込み時のPSWの値.
  [システムの処理]
   被除数が0ならば演算結果レジスタに真の0を設定して，実行を継続する. 0でな
 ければ，符号はそのままで演算結果レジスタに最大の絶対値を設定して，実行を継続
 する.
  [プログラマの処置]
   “FORTRANの使用手引書”のプログラムのデバッグを参照する.
 READY
```

11.1.3　メッセージを英語にする

これまでのメッセージ表示は日本語で行なったが，英語での表示も可能．切替えは，10.3 節の `PROFILE` コマンドで行なう．以下はメッセージを日本語から英語に切替えて表示した例．

XTERM6683

```
 READY
PROF TL(E)
 READY
LM JWE0013I-E LINE
 JWE0013I-E
 A FLOATING DIVISION EXCEPTION WAS DETECTED. PSW=XXXXXXXXXXXXXXXX
 <EXPLANATION>
 IN A FLOATING-POINT DIVISION OPERATION, THE DIVISOR IS ZERO.
 <PARAMETER>
 XXXXXXXXXXXXXXXX:
 SHOWS THE VALUE IN THE PSW AT THE POINT AT WHICH THE FLOATING-POINT DIVISION
 EXCEPTION INTERRUPT OCCURRED.
 <SYSTEM ACTION>
 IF THE DIVIDEND IS ZERO, SETS A TRUE ZERO IN THE OPERATION RESULT REGISTER, AND
 CONTINUES PROCESSING. IF IT IS NOT ZERO, SETS THE MAXIMUM ABSOLUTE VALUE IN THE
 OPERATION RESULT REGISTER, LEAVING THE SIGN UNCHANGED, AND CONTINUES
 PROCESSING.
 <PROGRAMMER RESPONSE>
 REFER TO THE SECTION ON PROGRAM DEBUGGING IN THE FORTRAN USER'S GUIDE.
 READY
```

11.2 open 文

次のプログラムを実行する．内容は a79999a さんが，既存のデータセット EXAM.DATA(EX1) からデータを読み込んで，そのまま書き出すもの．

```
EDIT --- A79999A.EXAM.FORT(EX1) - 01:01 ----------------- COLUMNS 001 072
COMMAND ===> R ↵                                         SCROLL ===> HALF
****** ************************** TOP OF DATA *************V10L30********
000100       PROGRAM EX1
000200       OPEN(5,FILE=EXAM.DATA(EX1),STATUS='OLD')
000300     1 READ(5,*,END=999) N
000400       WRITE(6,*) N
000500       GOTO 1
000600   999 CLOSE(5)
000700       STOP
000800       END
****** ************************* BOTTOM OF DATA *************************
```

次のエラーメッセージが出力される．表示は PROFILE コマンドで英語にしている．

```
FORTRAN77 EX COMPILER ENTERED
FORTRAN77 EX DIAGNOSTIC MESSAGES: PROGRAM NAME(EXAMPLE),FLAG(I),OPTIMIZE(B)
JWD2006I-I                EXAM IS NOT USED.
JWD1035I-S      00020002  INVALID OPERATOR.
END OF COMPILATION,HIGHEST SEVERITY CODE=12
***
```

“JWD1035I-S”の内容を LM コマンドで調べればわかるが，INVALID OPERATOR. と表示された 200 行の OPEN 文の記述に誤りがあるようである．

このエラーの原因は“FILE=EXAM.DATA(EX1)”を完全データセットで指定していないために起きたものである．この部分を，以下のように課題番号込みの完全データセットで指定すれば正常に動作する．

```
EDIT --- A79999A.EXAM.FORT(EX1) - 01:01 ----------------- COLUMNS 001 072
COMMAND ===> R ↵                                         SCROLL ===> HALF
****** ************************** TOP OF DATA *************V10L30********
000100       PROGRAM EX1
000200       OPEN(5,FILE='A79999A.EXAM.DATA(EX1)',STATUS='OLD')
000300     1 READ(5,*,END=999) N
000400       WRITE(6,*) N
000500       GOTO 1
000600   999 CLOSE(5)
000700       STOP
000800       END
****** ************************* BOTTOM OF DATA *************************
```

下線部が修正した部分．このプログラムを実行すると，次のようにデータの読み書きが正

常に行なわれる．

```
FORTRAN77 EX COMPILER ENTERED
END OF COMPILATION,HIGHEST SEVERITY CODE=00
1994
1990
1700
1600
END OF GO,HIGHEST SEVERITY CODE=00
***
```

また，'A79999A.EXAM.DATA(EX1)' は既に存在するデータセットである必要がある．MSP の場合は OPEN 文でデータセットに書き込みを行なう場合にも，一度あらかじめデータセットを作成した上で作業を行うことをお勧めする．

11.3 記憶領域オーバー

1.4 節で region size の概算方法を述べたが，1.5 節の制限値表は，利用者が確保できる"総容量"であり，例えば TSS の標準の region size が 10MB であるから，丸々 10MB の配列がとれる訳ではない．次のプログラムを実行する．

```
EDIT --- A79999A.EXAM.FORT(EX2) - 01:01 ----------------- COLUMNS 001 072
COMMAND ===> R ↵                                          SCROLL ===> HALF
****** ************************** TOP OF DATA *************V10L30********
000100       PROGRAM EX2
000200       PARAMETER(MAXN=105)
000300       REAL*8 MATRIX(MAXN,MAXN,MAXN)
000400       MATRIX(1,1,1)=2.0D0
000500       WRITE(6,*) MATRIX(1,1,1)
000600       STOP
000700       END
****** ************************* BOTTOM OF DATA *************************
```

1.4 節の計算方法では，倍精度 3 次元配列 `MATRIX` の確保できる領域は約 8.8MB であり，制限値の 10MB から見ると実行できそうであるが、

```
FORTRAN77 EX COMPILER ENTERED
JWD8642I-U TOO LARGE PROGRAM SIZE.
END OF COMPILATION,HIGHEST SEVERITY CODE=16
```

となり，領域サイズオーバーで実行できない．

領域が足りない原因の一つは，Fortran プログラムを PFD/EDIT 内から実行していることである．TSS は `MATRIX` の領域の他に，PFD/EDIT の用の領域を確保している．EDITOR を抜けて READY モードから `RUN` または `FORT` コマンドで実行する場合は，EDITOR 用の領域が必要ないので，上のプログラムはかろうじて実行できる．

しかし，`MAXN=128` となると，`MATRIX` の領域だけでも 16MB が必要であり，TSS の標準

モードでは実行不可能である．その場合は，1.3 節の手順に従って，拡張モードで再 LOGON を行なう．さらに，拡張モードでの実行を意味する AE オペランドを付加することで，最大 50MB までの領域を確保できる．

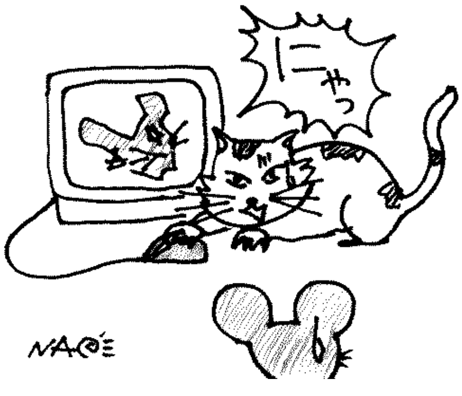
拡張モードで再 LOGON 後，AE オペランド付加してプログラムを実行する．

```
EDIT --- A79999A.EXAM.FORT(EX2) - 01:01 ----------------- COLUMNS 001 072
COMMAND ===> R AE ⏎                                       SCROLL ===> HALF
****** ************************** TOP OF DATA *************V10L30********
000100        PROGRAM EX2
000200        PARAMETER(MAXN=126)
000300        REAL*8 MATRIX(MAXN,MAXN,MAXN)
000400        MATRIX(1,1,1)=2.0D0
000500        WRITE(6,*) MATRIX(1,1,1)
000600        STOP
000700        END
****** ************************* BOTTOM OF DATA *************************
```

拡張モードにより，実行がうまくいく．

```
FORTRAN77 EX COMPILER ENTERED
END OF COMPILATION,HIGHEST SEVERITY CODE=00
2.0000000000000000
END OF GO,HIGHEST SEVERITY CODE=00
READY
***
```

TSS の拡張モードでの実行も出来なくなった場合は，確保する領域が 50MB を越えたことを意味するので，バッチ処理で実行するしかない．



ネコの手を借りて失敗

11.4 external 文

以下の Fortran プログラムを考える．

```
EDIT --- A79999A.EXAM.FORT(EX3) - 01:01 ----------------- COLUMNS 001 072
COMMAND ===> R   ↵                                        SCROLL ===> HALF
****** ************************** TOP OF DATA *************V10L30********
000100        PROGRAM EX3
000200        CALL LMINF(-5.0,5.0,FUN,0.0,400,F,ICON)
000300        WRITE(6,*) F
000400        STOP
000500        END
000600
000700        FUNCTION FUN(X)
000800        FUN=(((X-4.0)*X-6.0)*X-16.0)*X+4.0
000900        RETURN
001000        END
****** ************************* BOTTOM OF DATA *************************
```

“LMINF”は SSL II のサブルーチンで，関数の最小値を求めるもの．700 行以下は関数 FUN の定義．このプログラムを実行すると，エラーとなる．

```
FORTRAN77 EX COMPILER ENTERED
END OF COMPILATION,HIGHEST SEVERITY CODE=00
JWE0019I-U THE PROGRAM WAS TERMINATED ABNORMALLY.
           SYSTEM ABEND CODE=00C1-0000 PSW=0F8D3E0080000004 SDWA=0020D568
           UNPERMITTED INSTRUCTION  INSTRUCTION=0000         LOCATION=80000002
           ENTRY NAME OF THIS PROGRAM IS **GO      AT LOCATION=002B6000
           GR0 =3E107C3B GR1 =002B85E0 GR2 =00000190 GR3 =002B83E0
           GR4 =002B7C10 GR5 =002B83E0 GR6 =002C0E00 GR7 =00000002
           GR8 =00000001 GR9 =00000000 GR10=0022674C GR11=00000002
           GR12=002B7C10 GR13=002B83E0 GR14=802B7D3C GR15=00000000
           FR0 =3E40000000000000       FR2 =3E40038D087FD000
           FR4 =3C10000000000000       FR6 =3C10000000000000
 ERROR OCCURS AT LMINF                     LOC 00000002
 LMINF       AT LOC 002B8380 CALLED FROM LOC 002B6188 IN EX3      SSN 00020000
 EX3         AT LOC 802B6000 CALLED FROM O.S.
ERROR SUMMARY (FORTRAN77 EX)
ERROR NUMBER  ERROR LEVEL  ERROR COUNT
  JWE0019I        U          1
TOTAL ERROR COUNT =     1
JQB199I-U  USER PROGRAM TERMINATED ABNORMALLY.
END OF GO,HIGHEST SEVERITY CODE=240
COMMAND ABORTED
COMMAND ABORTED WITH CODE 0000F0.
```

メッセージの内容から，SSL II のサブルーチン LMINF でエラーが発生している．

これは，サブルーチン LMINF で引数として引用している FUN が外部手続き，この場合関数であることを宣言する external 文を書き忘れたため発生したエラーである．external 文については，Fortran の文法を扱った書物には必ず載っているので，そちらを参照．

プログラムを下線部の通り修正する．

```
EDIT --- A79999A.EXAM.FORT(EX3) - 01:01 ----------------- COLUMNS 001 072
COMMAND ===> R   ↵                                         SCROLL ===> HALF
****** ************************** TOP OF DATA *************V10L30********
000100       PROGRAM EX3
000110       EXTERNAL FUN
000200       CALL LMINF(-5.0,5.0,FUN,0.0,400,F,ICON)
000300       WRITE(6,*) F
000400       STOP
000500       END
000600
000700       FUNCTION FUN(X)
000800       FUN=(((X-4.0)*X-6.0)*X-16.0)*X+4.0
000900       RETURN
001000       END
****** ************************* BOTTOM OF DATA *************************
```

external 文を追加することで，正常に実行される．

```
FORTRAN77 EX COMPILER ENTERED
END OF COMPILATION,HIGHEST SEVERITY CODE=00
-155.99992
END OF GO,HIGHEST SEVERITY CODE=00
***
```

11.5 配列の定義外へのアクセス

DO ループの回し損ねると，このエラーは頻繁に起きる．配列で宣言した範囲を越えてアクセスを行なう場合である．

```
EDIT --- A79999A.EXAM.FORT(EX4) - 01:01 ----------------- COLUMNS 001 072
COMMAND ===> R  ↵                                        SCROLL ===> HALF
****** ************************* TOP OF DATA *************V10L30********
000100       PROGRAM EX4
000200       REAL M(10,10)
000300       DO 10 I=1,5
000400       DO 20 J=I,I*2+I
000500       M(I,J)=(I+J)/10.0
000600 20    CONTINUE
000700 10    CONTINUE
000800       WRITE(6,*) M
000900       STOP
001000       END
****** ************************* BOTTOM OF DATA *************************
```

例では，下線部の J の値が，明らかに配列の大きさ 10 を越えている．エラーメッセージは次の通り．

```
FORTRAN77 EX COMPILER ENTERED
END OF COMPILATION,HIGHEST SEVERITY CODE=00
JWE0019I-U THE PROGRAM WAS TERMINATED ABNORMALLY.
           THE VALUE OF SUBSCRIPT MAY EXCEED THE SIZE OF ARRAY.
           COMPILE WITH DEBUG(SUBCHK,ARGCHK,UNDEF) OPTION AND EXECUTE AGAIN.
           SYSTEM ABEND CODE=00C4-0000 PSW=0F8D2000802B4404 SDWA=002BF988
           INVALID MEMORY ACCESS  INSTRUCTION=402E1479     LOCATION=802B4400
           ENTRY NAME OF THIS PROGRAM IS **GO      AT LOCATION=002B4000
           GR0 =0020BC70 GR1 =00000747 GR2 =0020BD38 GR3 =002B4070
           GR4 =00000028 GR5 =0000000B GR6 =00000108 GR7 =0000000D
           GR8 =00000001 GR9 =00000050 GR10=00000294 GR11=00000000
           GR12=002B4350 GR13=002B4070 GR14=00000005 GR15=00000001
           FR0 =41C0000000000000       FR2 =411E666600000000
           FR4 =4110000000000000       FR6 =4214000000000000
 ERROR OCCURS AT EX4      SSN 00070000     LOC 002B4400 OFFSET 00000400
 EX4         AT LOC 802B4000 CALLED FROM O.S.
ERROR SUMMARY (FORTRAN77 EX)
ERROR NUMBER  ERROR LEVEL  ERROR COUNT
  JWE0019I         U           1
TOTAL ERROR COUNT =     1
JQB199I-U  USER PROGRAM TERMINATED ABNORMALLY.
END OF GO,HIGHEST SEVERITY CODE=240
COMMAND ABORTED
***
```

さて，メッセージを良く見ると，下線部のメッセージが目につく．

意味は「翻訳時に DEBUG(SUBCHK,ARGCHK,UNDEF) オプションを指定しなさい」とのことである．括弧内の検査は次の通り．

名称	機能
SUBCHK	添字式と部分列の値の検査
ARGCHK	引数の対応の検査
UNDEF	未定義データの引用検査

コンパイラのメッセージに従って，再実行する．翻訳オプションを付加する場合はFORTコマンドで行なう．RUNでも可能だが，指定方法が面倒．

```
EDIT --- A79999A.EXAM.FORT(EX4) - 01:01 ----------------- COLUMNS 001 072
COMMAND ===> FORT DEBUG(SUBCHK,ARGCHK,UNDEF) ↵           SCROLL ===> HALF
****** ************************* TOP OF DATA ************V10L30********
000100       PROGRAM EX4
000200       REAL M(10,10)
000300       DO 10 I=1,5
000400       DO 20 J=I,I*2+I
000500       M(I,J)=(I+J)/10.0
000600 20    CONTINUE
000700 10    CONTINUE
000800       WRITE(6,*) M
000900       STOP
001000       END
****** ************************* BOTTOM OF DATA ************************
```

すると，翻訳段階で配列のチェックが行なわれ，下記のように，おかしな場所を教えてくれる．

```
 FORTRAN77 EX COMPILER ENTERED
 END OF COMPILATION,HIGHEST SEVERITY CODE=00
 JWE0320I-W 00050000 THE SUBSCRIPT OR SUBSTRING(M) IS OUT OF THE SPECIFIED RANGE
. REFERENCE VALUE IS 4,11, SPECIFICATION VALUE IS 1:10,1:10.
  ERROR OCCURS AT EX4      SSN 00050000     LOC 002B14B8 OFFSET 000004B4
  EX4         AT LOC 802B1000 CALLED FROM O.S.
 TAKEN TO (STANDARD) CORRECTIVE ACTION, EXECUTION CONTINUING.
 JWE0320I-W 00050000 THE SUBSCRIPT OR SUBSTRING(M) IS OUT OF THE SPECIFIED RANGE
. REFERENCE VALUE IS 4,12, SPECIFICATION VALUE IS 1:10,1:10.
  ERROR OCCURS AT EX4      SSN 00050000     LOC 002B14B8 OFFSET 000004B4
  EX4         AT LOC 802B1000 CALLED FROM O.S.
                           :
```

バッチ処理で同様の処理をする場合は，ジョブ制御文に下線の部分を翻訳オプションとして追加する．ただし，多分プログラムの書換えが生じるので，配列の大きさをTSSでも実行できる程度に宣言した上で，TSSでデバックを行なった方がよろしいかと思う．

```
//A79999AA JOB CLASS=A
//   EXEC FORT,STEP=C,OPT=B,OPTION='DEBUG(SUBCHK,ARGCHK,UNDEF)'
//FORT.SYSIN DD DSN=A79999A.EXAM.FORT(EX4),DISP=SHR
//
```

11.6 恐怖のゼロ割り

ゼロ割りとは

$$\frac{A}{B}$$

の計算で $B = 0$ となること．数学的にはこんな値は定義出来ないので，計算結果としても良くない結果であることは異論のないところであろう．

次のエラーメッセージが出力される．

```
FORTRAN77 EX COMPILER ENTERED
END OF COMPILATION,HIGHEST SEVERITY CODE=00
JWE0013I-E A FLOATING DIVISION EXCEPTION WAS DETECTED. PSW=0F8D0000801D0188
           INSTRUCTION=7D00D098          LOCATION=801D0184
           GR0 =00127C70  GR1 =00127D28  GR2 =00127D38  GR3 =001D0070
           GR4 =001D0180  GR5 =001D0070  GR6 =001DA370  GR7 =00000000
           GR8 =00000000  GR9 =00127C28  GR10=0014074C  GR11=001D1C00
           GR12=001020A0  GR13=001D0070  GR14=801D0052  GR15=001DA370
           FR0 =0000000000000000         FR2 =0000000000000000
           FR4 =0000000000000000         FR6 =0000000000000000
 ERROR OCCURS AT EX5      SSN 00020000     LOC 001D0184 OFFSET 00000184
 EX5         AT LOC 801D0000 CALLED FROM O.S.
TAKEN TO (STANDARD) CORRECTIVE ACTION, EXECUTION CONTINUING.
***
```

ゼロ割りの原因は次が考えられる．

- プログラミングがまずい
- プログラミングはまずくないが，最適化レベルをあげ過ぎた．

最初は利用者レベルの問題なので，エラーメッセージをよく見て，ゼロ割りが発生した行番号を頼りにデバックを行なうこと．

二番目のエラーは，利用者が翻訳オプションとして最適化レベルを E 以上に指定したときに，しかも極まれにしか発生しない（省略値は B）．最適化レベルの指定は，バッチ処理で行なう場合が多い．もし，バッチ処理でゼロ割りのメッセージが発生した場合は，プログラムのチェックと併せて，次の様に最適化レベルを下げて再度実行してみると，正常に処理される場合がある．

```
//A79999AB JOB CLASS=V
//   EXEC FORT,STEP=CLG,VP=YES,OPT=B
//FORT.SYSIN DD DSN=A79999A.EXAM.FORT(EX5),DISP=SHR
//
```

11.7 負の実数のベキ乗

Fortran77 EX が文法チェックに厳しい例をあげる．次のプログラムは，一般の Fortran コンパイラの場合はエラーなしで実行されるのが普通である．

```
EDIT --- A79999A.EXAM.FORT(EX6) - 01:01 ----------------- COLUMNS 001 072
COMMAND ===> R  ↵                                         SCROLL ===> HALF
****** ************************** TOP OF DATA *************V10L30********
 000100       PROGRAM EX6
 000200       A=-2.0
 000300       B= 3
 000400       WRITE(6,*) A**B
 000500       END
****** ************************* BOTTOM OF DATA ************************
```

上のプログラムを RUN コマンドによって実行する．

```
FORTRAN77 EX COMPILER ENTERED
END OF COMPILATION,HIGHEST SEVERITY CODE=00
JWE0263I-E IN X1**X2(X1,X2:REAL*4),X1.LT.0.0 .AND. X2.NE.0.0
(X1=-0.20000000E+01,X2=-0.30000000E+01).
 ERROR OCCURS AT F#ARXR                     LOC 002B7E68
 F#ARXR  (F) AT LOC 002B807C CALLED FROM LOC 002B66FE IN EX6      SSN 00040000
 EX6         AT LOC 802B6000 CALLED FROM O.S.
TAKEN TO (STANDARD) CORRECTIVE ACTION, EXECUTION CONTINUING.
0.12500000
ERROR SUMMARY (FORTRAN77 EX)
ERROR NUMBER  ERROR LEVEL  ERROR COUNT
  JWE0263I         E           1
TOTAL ERROR COUNT =     1
END OF GO,HIGHEST SEVERITY CODE=08
***
```

Fortran77 EX の場合，このように文句をいわれる．Fortran77 EX は，実数 a, b についての演算 a^b の計算で $a < 0$ としたことを怒っているのである．Fortran の文法書をみると，「このような演算は定義しない」と書いてある．

その数学的根拠は，

> a を底とする指数関数 (Exponential function to the base a) $f(x) = a^x$ の定義では，$a < 0$ となることはありえない

というのが理由である．

これに異議がある場合は，指数関数とその逆関数である対数関数に関する定義を自ら行ない，しかる後に初等関数論を全て再構築されることをお勧めする．間違いなくその業績により博士号が取得できる．

「そんなバカな．だって $(-2)^3$ は -8 じゃない．」などと反論してはいけない．"(-2) ^{3}" は確かにベキ乗であるが，"指数関数から導かれる数"ではない．

"(-2) ^{3}" は "$-2 \times -2 \times -2$" の略記であって，Fortran では変数 B は何も指定がない場合，実数と解釈されるので，Fortran77 EX コンパイラはそれに忠実に従って，A**B を A

を底とする指数関数の代入結果として計算しようとして，文法上の定義に反してしまったのである．

結局，プログラムを組んだ人は指数関数を計算する気などなくて，$(-2.0)^3 = (-2.0) \times (-2.0) \times (-2.0)$ 即ち A の 3 乗を計算したかったのである．b が整数ならば，指数関数の定義を持ち出さなくても計算は単なるかけ算の集まりなので，$a < 0$ でもいっこうに構わない．

計算をエラーなしにするためには，INTEGER B の宣言をして，A**B の計算が“A を B 回かける”意味だと教えてやらねばならない．

```
EDIT --- A79999A.EXAM.FORT(EX6) - 01:01 ----------------- COLUMNS 001 072
COMMAND ===> R  ⏎                                             SCROLL ===> HALF
****** ************************** TOP OF DATA *************V10L30********
000100       PROGRAM EX6
000110       INTEGER B
000200       A=-2.0
000300       B= 3
000400       WRITE(6,*) A**B
000500       END
****** ************************* BOTTOM OF DATA *************************
```

以上の文法チェックは普通の Fortran コンパイラではやってくれない．しかし，プログラミングの意図とは別に Fortran 文法の解釈上，数学的に間違っていることを指摘してくれた Fortran77 EX コンパイラに「厳し過ぎるぞ」と文句を言うのは筋違いである．素直に感謝するのがマナーである．

11.8 まとめ

Fortran プログラムのデバッグは，

- LM コマンドでメッセージを調べる
- DEBUG オプションで文法チェックをする

ことで大部分解決できる．特に，

他の機種の Fortran では動いていたのに，センターでは動かない

場合は，間違いなく利用者のプログラムミスである．これは MSP, UXP 共通の特徴で，センターの Fortran は文法チェックがかなり厳しいことを了解いただきたい．

12　実際に使ってみよう

最後に TSS での Fortran プログラムの実行例を，具体的な例に沿って述べる．

12.1　RUN コマンドによるプログラムの実行

「あなたが生まれた年は閏年か？」を調べるプログラム LEAP.FORT を実行する．
まず，データを端末から入力し，端末に結果を出力する場合．

```
 READY
PROF TL(J) ↵                ......... 日本語でメッセージを出力させる．
 READY
ED LEAP.FORT ↵              ......... データセットの編集
```

画面が切り替わり，編集画面になるので，プログラムを入力する．プログラム作成後 RUN を入力し ENTER キーを押す．

```
EDIT --- A79999A.LEAP.FORT -------------------------- 表示欄   001 072
コマンド ===> RUN ↵                                     移動量 ===> CUR
****** ************************ データの先頭 ************ V10L30 ******
000100     1 READ(5,*,END=999) N
000200    10 FORMAT(I4)
000300       IF(N-N/4*4.NE.0) THEN
000400        WRITE(6,20)N
000500       ELSEIF(N-N/100*100.NE.0) THEN
000600        WRITE(6,30)N
000700       ELSEIF(N-N/400*400.NE.0) THEN
000800        WRITE(6,20)N
000900       ELSE
001000        WRITE(6,30)N
001100       ENDIF
001200       GOTO 1
001300   999 STOP
001400    20 FORMAT(1H ,I4,' IS NOT A LEAP YEAR.')
001500    30 FORMAT(1H ,I4,' IS A LEAP YEAR.')
001600       END
****** ************************ データの末尾 *************************
```

すると Fortran77 EX コンパイラが起動される．

```
【FORTRAN77 EX 翻訳開始】
【翻訳終了】，完了コード =00
 10000 ?                    .........  入力要求．
1994 ↵                      .........  ４桁の数（西暦）を入れる．
 1994 IS NOT A LEAP YEAR.
 10000 ?
1992 ↵
 1992 IS A LEAP YEAR.
```

```
 10000 ?
/*  ↵                        .........    入力終了．
    【実行終了】，完了コード=00
    ***                       ......... ENTERキーを押したら画面が変わる印．
```

実行が終了すると，もとのEDIT画面に戻る．保存，終了する場合はPF3キーを押す．保存しないで終了したい場合はCANCELサブコマンドを入力する.

EDコマンドでデータセットを作成すると，PFDの2の指定時と同じ画面になる．このコマンドで作成するプログラムに日本語を使用する場合は，

```
 READY
ED LEAP.FORT NI   ↵
```

と指定しなければならない．またPFDEコマンドにより，データセット一覧を検索しながら編集を行なうことも可能である．

次に，同じプログラム LEAP.FORT に対して，データを端末から入力し，センター内のプリンターに結果を出力する．PFD/EDIT内からRUNコマンドで実行することは，前と同じ．ただし，プログラムを作成しRUNコマンドを入れる前に，次のコマンドを入力する．

```
EDIT --- A79999A.LEAP.FORT -------------------------- 表示欄   001 072
コマンド ===> ALLOC FI(FT06F001) SY(U) REU  ↵          移動量 ===> CUR
****** ************************ データの先頭 ************ V10L30 ******
000001     1 READ(5,*,END=999) N
000002    10 FORMAT(I4)
000003       IF(N-N/4*4.NE.0) THEN
000004        WRITE(6,20)N
                     :
```

FT06F001は装置機番6のことである．それをオープン室のNLP, CLPに割り当てる．続いてRUNサブコマンドで実行する．

```
EDIT --- A79999A.LEAP.FORT -------------------------- 表示欄   001 072
コマンド ===> R   ↵                                   移動量 ===> CUR
****** ************************ データの先頭 ************ V10L30 ******
000001     1 READ(5,*,END=999) N
000002    10 FORMAT(I4)
000003       IF(N-N/4*4.NE.0) THEN
000004        WRITE(6,20)N
                     :
```

画面が切り替わり，データ入力待ちになる．

```
【FORTRAN77 EX 翻訳開始】
【翻訳終了】，完了コード=00
 10000 ?                   ......... 入力要求．
1994  ↵                    ......... ４桁の数（西暦）を入れる．
 10000 ?                   ......... 結果は出力先をプリンターに割り当てているため，
```

```
1992 ⏎                              端末には出力されない．
 10000 ?
/* ⏎                         ......... 入力終了．
【実行終了】，完了コード=00
 ***
```

実行が終了すると，データセット編集画面になるので，ALLOCATE コマンドで出力先を端末に戻す．

```
EDIT --- A79999A.LEAP.FORT -------------------------- 表示欄   001 072
コマンド ===> ALLOC FI(FT06F001) DA(*) REU ⏎          移動量 ===> CUR
****** ************************ データの先頭 ************ V10L30 ******
000001     1 READ(5,*,END=999) N
000002    10 FORMAT(I4)
000003       IF(N-N/4*4.NE.0) THEN
000004        WRITE(6,20)N
                    :
```

実行後出力結果を受け取るために，プリンター横のコンソールで課題番号とパスワードを入れる．すると，自分のジョブの一覧が表示されるので，現在使用中のセッションからの出力なので，TSS ジョブの印"#"に対応した，例えば

```
A79999A# (TSU1234) EXECUT
```

の行に OUTPUT の "O" を入力することでプリンターに出力される

次に，データセットからデータを入力し，実行する場合をやってみる．まず，入力用のデータ LEAP.DATA を編集する．

```
 READY
ED LEAP.DATA ⏎                             ......... データセットの新規作成
```

```
EDIT --- A79999A.LEAP.DATA   --------------------------- 表示欄   001 072
コマンド ===>                                              移動量 ===> PAGE
****** ********************* データの先頭 ***************** V10L30 ******
000001 1994                              ......... データの作成．
000002 1992
000003 1600
000004 1700
****** ************************ データの末尾 *****************************
```

PF3 キーで LEAP.DATA 保存終了する．保存だけで終了しない場合は SAVE サブコマンドを入れる．

次に ALLOCATE コマンドで READ(5,*) に対応する FT05F001 にデータセット LEAP.DATA を割りあて RUN コマンドで実行する．

```
 READY
ALLOC FI(FT05F001) DA(LEAP.DATA) SHR REU    ↵
 READY
RUN LEAP.FORT   ↵
【FORTRAN77 EX 翻訳開始】
【翻訳終了】, 完了コード =00
 1994 IS NOT A LEAP YEAR.
 1992 IS A LEAP YEAR.                         ......... 実行結果 .
 1600 IS A LEAP YEAR.
 1700 IS NOT A LEAP YEAR.
【実行終了】, 完了コード =00
 READY
ALLOC F(FT05F001) DA(*) REU   ↵               ......... 入力元を端末に戻す .
 READY
```

READY 状態でデータセットからデータを入力し，結果もデータセットへ出力する場合も ALLOCATE コマンドが活躍する．

入力データセットは LEAP.DATA，出力データセットは LEAP.LIST を新規に作成して割りあてる．

```
 READY
ALLOC FI(FT05F001) DA(LEAP.DATA) SHR REU    ↵
 READY
ALLOC FI(FT06F001) DA(LEAP.LIST) NEW CA TRA SP(1 1)+    ↵
LR(80) BLK(23440) REC(F B) REU   ↵
 READY
RUN LEAP.FORT   ↵
 【FORTRAN77 EX 翻訳開始】
 【翻訳終了】, 完了コード =00
 【実行終了】, 完了コード =00
 READY
ALLOC F(FT05F001) DA(*) SHR REU   ↵           ......... 入力元を端末に戻す .
 READY
ALLOC F(FT06F001) DA(*) SHR REU   ↵           ......... 出力先を端末に戻す .
 READY
L LEAP.LIST   ↵                               ......... 出力内容をみる .
```

また，Fortran プログラムに直接 OPEN 文でデータセットを割り当てることも可能．入力データセットを LEAP.DATA，出力データセットを KEKKA.LIST に割り当てる場合は，次の様にプログラムを書けばよい．ともに既存のデータセットとする．

```
      PROGRAM LEAP
      OPEN(5,FILE='A79999A.LEAP.DATA',STATUS='OLD')
      OPEN(6,FILE='A79999A.KEKKA.LIST',STATUS='OLD')
C
    1 READ(5,*,END=999) N
      IF (N - N/4*4 .NE. 0) THEN
        WRITE(6,20) N
      ELSEIF (N - N/100*100 .NE. 0) THEN
        WRITE(6,30) N
      ELSEIF (N - N/400*400 .NE. 0) THEN
        WRITE(6,20) N
      ELSE
        WRITE(6,30) N
      ENDIF
      GO TO 1
  999 CLOSE(5)
      CLOSE(6)
      STOP
   20 FORMAT(1H ,I4,' IS NOT A LEAP YEAR.')
   30 FORMAT(1H ,I4,' IS A LEAP YEAR.')
      END
```

12.2　FORT, LINK, CALL コマンドでのプログラムの実行

先の Fortran プログラム LEAP.FORT を翻訳，編集結合し，ロードモジュールを作成して，それを実行する一連の動作を以下示す．

```
 READY
FORT LEAP.FORT OBJ(LEAP.OBJ) ↵        ...... オブジェクトモジュールの作成．
 【FORTRAN77 EX 翻訳開始】
 【翻訳終了】，完了コード = 00
 READY
LINK LEAP.OBJ LO(PROG.LOAD(LEAP)) FORTLIB  ↵  ......ロードモジュールの作成
 ** MEMBER NAME **  LEAP    NOW ADDED TO LIBRARY.
 READY
CALL PROG.LOAD(LEAP) ↵               .........  ロードモジュールの実行．
 10000 ?                             .........  入力要求．
1994 ↵                               .........  ４桁の数（西暦）を入れる．
 1994 IS NOT A LEAP YEAR.
 10000 ?
```

```
1992 ↵
 1992 IS A LEAP YEAR.
 10000 ?
/* ↵                                         ......... 入力終了．
 READY
```

次に，私用ライブラリの作成と実行例を示す．関数として，四則演算式それぞれに名前をつけて登録し，それを引用する場合のプログラムを SISOKU.FORT に作成する．データセットは次のような中身．

```
EDIT --- A79999A.SISOKU.FORT   ------------------------ 表示欄   001 072
コマンド ===>                                            移動量 ===> CUR
****** ************************ データの先頭 ************ V10L30 ******
000001       FUNCTION ADD(X1,X2)
000002       INTEGER ADD,X1,X2
000003       ADD=X1+X2
000004       END
000005
000006       FUNCTION SUB(X1,X2)
000007       INTEGER SUB,X1,X2
000008       SUB=X1-X2
000009       END
000010
000011       FUNCTION MUL(X1,X2)
000012       INTEGER MUL,X1,X2
000013       MUL=X1*X2
000014       END
000015
000016       FUNCTION DIV(X1,X2)
000017       REAL DIV
000018       INTEGER X1,X2
000019       DIV=X1*1.0/X2
000020       END
****** *********************** データの末尾 **************************
```

プログラムを FORT コマンドにより翻訳し，さらに LINK コマンドにより結合編集し，私用ライブラリ PROG.LOAD に登録する．メンバ名は関数名がそのまま対応する．

```
 READY
FORT SISOKU.FORT OBJ(SISOKU.OBJ) NAME  ↵   ...... オブジェクトモジュールの作成
【FORTRAN77 EX 翻訳開始】
【翻訳終了】
 READY
LINK SISOKU.OBJ LO(PROG.LOAD) NCAL  ↵   ...... ロードモジュールの作成
 ** MEMBER NAME ** ADD    WAS NOT FOUND BUT HAS BEEN ADDED TO LIBRARY.
 ** MEMBER NAME ** SUB    WAS NOT FOUND BUT HAS BEEN ADDED TO LIBRARY.
 ** MEMBER NAME ** MUL    WAS NOT FOUND BUT HAS BEEN ADDED TO LIBRARY.
 ** MEMBER NAME ** DIV    WAS NOT FOUND BUT HAS BEEN ADDED TO LIBRARY.
 READY
```

次に，それらの関数を呼ぶプログラムを作成し，RUN コマンドで実行する．

```
EDIT --- A79999A.KEISAN.FORT   ------------------------ 表示欄   001 072
コマンド ===> RUN LIB(PROG.LOAD)                          移動量 ===> CUR
****** ************************ データの先頭 ************ V10L30 ******
      INTEGER ADD,SUB,MUL
      REAL DIV

      PRINT('(//30X,''TABLE OF ADDITION (I+J)''//)')
      PRINT  600,(I,I=1,10)

      DO 10 J=1,10
        PRINT '(I4,4X,10I6)',J,(ADD(I,J),I=1,10)
   10 CONTINUE

      PRINT'(''1''//30X,''TABLE OF SUBTRACTION (I-J)''//)'
      PRINT  600,(I,I=1,10)

      DO 20 J=1,10
      PRINT '(I4,4X,10I6)',J,(SUB(I,J),I=1,10)
   20 CONTINUE

      PRINT'(''1''//30X,''TABLE OF MULTIPLICATION (I*J)''//)'
      PRINT  600,(I,I=1,10)

      DO 30 J=1,10
      PRINT '(I4,4X,10I6)',J,(MUL(I,J),I=1,10)
   30 CONTINUE

      PRINT'(''1''//30X,''TABLE OF DIVISION (I/J)''//)'
      PRINT  610,(I,I=1,10)

      DO 40 J=1,10
      PRINT '(I4,7X,10F10.6)',J,(DIV(I,J),I=1,10)
   40 CONTINUE

      STOP

  600 FORMAT(5X,' I',10I6/'   J'/)
  610 FORMAT(5X,' I',10I10/'  J'/)
      END
****** ************************ データの末尾 *************************
```

実行結果が画面に表示される．あらかじめ

```
ALLOC F(FT06F001) SY(U) REU
```

と入力しておくとオープン NLP, CLP に結果がページ毎に出力される．上の Fortran プログラムでは，WRITE(6,600) の代わりに PRINT 600 と指定しており．また，WRITE(6,'I4,4X,10I6') の代わりに PRINT'(I4,4X,10I6)' と指定している．

12.3 バッチ処理によるプログラムの実行

バッチ処理によるプログラムの実行で，結果をプリンターに出力するまでの流れは次の通り．

制御文の作成

↓

SUBMIT サブコマンドでのジョブの依頼

```
EDIT --- A79999A.LEAP.CNTL     ------------------------ 表示欄  001 072
コマンド ===> SUBMIT                                     移動量 ===> CUR
****** ************************ データの先頭 ************ V10L30 ******
000001 //A79999AX JOB CLASS=A
000002 //   EXEC FORT,STEP=CLG,OPT=E
000003 //FORT.SYSIN DD DSN=A79999A.LEAP.FORT,DISP=SHR
000004 //GO.SYSIN   DD DSN=A79999A.LEAP.DATA,DISP=SHR
000005 //
****** ************************ データの末尾 *************************
```

↓

```
    KEQ56208I ***           A79999AX        :  (RECEIVED) ***
        *** A79999AX (J1234) A79999A  : (JOB ACCEPTED) *** FIB  CN(01)
    KEQ56250I  JOB A79999AX(JOB01234) SUBMITTED
***
```

↓

実行後出力結果を受け取るためにプリンター横のコンソールに立たずむ

↓

課題番号とパスワードを入れる．自分のジョブの一覧が表示される

↓

OUTPUT 命令を与える

↓

プリンターに出力される

12.4 付録

簡単な Fortran ソースプログラム例です．適当なサンプルがない場合にお使い下さい．

- 出来ないと居残りさせられた算数教育の原点・九九の表

```
      integer table(9,9)
      write(6,10)(n,n=1,9)
   10 format(1h1,9x,'***A Multiplication Table***'/
     +       1h0,5x,9('(',i1,')',2x))
      do 20 i=1,9
      do 20 j=1,9
         table(i,j)=i*j
   20 continue
      write(6,30)(i,(table(i,j),j=1,9),i=1,9)
   30 format(1h0,'(',i1,')',9i5)
      stop
      end
```

- 整数を入力すると秒だと勝手に解釈して時間 / 分 / 秒に換算してくれるもの

```
      integer x,h,a,m,s
 10   read(5,*,end=99) x
      h=x/3600
      a=x-3600*h
      m=a/60
      s=a-60*m
      write(6,20) h,m,s
 20   format(1h ,3x,i2,'時間',i2,'分',i2,'秒')
     go to 10
 99   stop
      end
```

参考文献

[1] 「UXP/Fortran 使用法」九州大学大型計算機センター・ Fortran 講習会資料, 1996.

[2] 「UXP/M for the Working Scientists」 佐藤 周行, 九州大学大型計算機センター・ UXP 講習会資料, 1994. UXP の`/usr/local/doc/uxpguide.dvi` に TEX の dvi file あり

[3] 「Unix 環境でスーパーコンピューティングをする人のためのガイド」佐藤 周行，九州大学大型計算機センター広報, Vol.26, No.4, pp.452–484, 1993.

[4] 「VP2600 のシステム記憶とその使用法」 島崎 眞昭，九州大学大型計算機センター広報, Vol.25, No.3, pp.203–209, 1992.

[5] 「OS IV/MSP FORTRAN77 EX 使用手引書」(79SP-5031)，富士通株式会社．
（PLUM コマンドでオンラインマニュアルとして参照可）

[6] 「OS IV/MSP FORTRAN77 EX/VP 使用手引書 V12 用」(79SP-5041)，富士通株式会社

[7] 「FORTRAN 新コンパイラの公開について」 竹生 政資, 九州大学大型計算機センター広報, Vol.24, No.5, pp.523–540, 1991.

[8] 「MT, CMT の利用法」山嵜 信広，川嵜 正子，九州大学大型計算機センター広報, Vol.27, No.3, pp.170–194, 1994.

[9] 「MSP/FORTRAN を VP で実行するには」肥田木 直子，渡部 善隆，九州大学大型計算機センター広報, Vol.26, No.4, pp.397–451, 1993.

[10] 「利用の手引・バッチジョブ編」 九州大学大型計算機センター・ライブラリ室, 1992.

[11] 「利用の手引・ TSS 編」 九州大学大型計算機センター・ネットワーク室, 1993.

[12] 「OS IV/MSP TSS/E コマンド文法書」(79SP-4091)，富士通株式会社．

[13] 「PFD と PFDE の使用法」 平野 広幸，九州大学大型計算機センター広報, Vol.25, No.2, pp.119–165, 1992.

[14] 「利用の手引・ MSP コマンド編」九州大学大型計算機センター・ライブラリ室, 1993.

[15] 「デバッガ機能の紹介」 渡部 善隆，平尾 耕二, 九州大学大型計算機センター広報, Vol.26, No.1, pp.1–47, 1993.

[16] 「SSL II 使用手引書（科学用サブルーチンライブラリ）」（99SP–0050）, 富士通株式会社．

[17] 「SSL II 拡張機能使用手引書（科学用サブルーチンライブラリ）」（99SP–4070）, 富士通株式会社．

[18] 「ライブラリ・プログラム利用の手引（数値計算編： NUMPAC ）」 名古屋大学大型計算機センター．

【著者略歴】
アポロが月面着陸した頃水俣に生まれる．顔は父親似，性格は母親似．貯金少なし口数多し．特技はイベントの企画，進行．
加齢からくる脳細胞破壊に歯止めをかけるため，只今社会人大学生．毛が生えた蚤(のみ)の心臓を持つ公務員 6 年目． AB 型．

眉目秀麗, 多情仏心な著者近影