TSSでFORTRANを使ってみて

荻野綱男（東京大学文学部）

1. はじめに

筆者はいくつかのコンピュータのTSSを利用しているが，TSSの利用に完全に満足しているわけではない。それぞれのTSSにどうも使いにくいところがあるのである。TSSの使い勝手は，いろいろな点から考えることができる。特に使用頻度の高い「エディタ」とその周辺については，すでに筆者なりの考えを述べたことがあるので，ここではそれを省略し，TSSの中のFORTRAN処理系について，いくつかの処理系を比較しつつ，その使い勝手を考えていきたい。

2. とりあげる処理系

①TOSBAC 5600/160 (GCOS) <電子技術総合研究所EPICS主計算機>
FORTRAN処理系は一つしかない。以下これをTと略称する。実質的にACOSシリーズと同じである。

②MELCOM COSMO-900 (UTS/VS) <東京大学教育用計算機センター>
FORTRAN処理系は三つある。それらを総称してMと呼ぶことがある。
1) FLAG (Fortran Load And Go) 処理系が主記憶に常駐。以下MFと略称。
2) FORTRAN IV 文法が大幅に拡張されている。以下M4と略称。
3) 拡張FORTRAN FORTRAN77に準拠。以下MXと略称。

③HITAC 8800/8700 (OS7) <東京大学大型計算機センター '80年8月まで>
処理系は一つしかないが，その利用法が二つに分けられる。
1) メーカ提供のコマンドによる。以下HSと略称。
2) 東大センタで用意したマクロコマンドTOOLによる。以下HTと略称。

④HITAC M-200H (VOS3) <東京大学大型計算機センター '80年7月から>
東大センターで標準的に使うのは最適化FORTRAN 77である。以下HVと略称。

3. TSSでは自由書式入出力が絶対必要である。

自由書式入出力というのは，FORMAT文を指定せずに'自由な'形式で端末とプログラムの間でデータの受け渡しをすることである。たとえば入力のときはデータを
1, 123, -23, 0 のように入れ，出力のときは出力並びを書くだけでそれぞれの型にしたがって適当に変換されて端末に印字される。

TSSにおいては，特に端末からの少量データの入力（'会話'のため），プログラムのデバッグ用の出力のために，この機能が不可欠である。今後はFORTRAN77の処理系が普及するだろうから，問題はなくなるだろう。

表1 各処理系の自由書式入出力文

処理系	T	MF・M4	MX	HS・HT	HV	(JIS)
入力	READ	INPUT	{READ*, INPUT}	なし	READ*	なし
出力	PRINT	OUTPUT	{PRINT*, OUTPUT}	なし	PRINT*	なし

4. 同じ処理をするのに要する入力コマンドの量は，少ないほうがいい。

ログオンしてから，簡単なプログラムを作成し，実行し，一部修正し，再実行する操作を考えてみよう。プログラム本体とデータの入力を含めず，ファイル名はなるべく短くし，<CR>だけの応答は1行0文字として，それらの一連の操作に必要な入力量を算出すると表2のようになる。HSの使いにくさは一目瞭然である。MXではファイルの保存ができないので，他の処理系でもファイルの保存をしないようにすれば，Tが23文字で最小文字数になる。

表2 同一処理に要する入力

処理系	MX	T	MF	HV	M4	HT	HS
最小行数	9	6	9	6	11	9	16
最小文字数	27	29	30	38	40	49	133

5. 処理系は不必要な情報を出さないほうがよい。

何が不必要かは人によって異論もあろうが，筆者は「正常に処理が進むなら他に何もメッセージを印字することはない」と考える。過多の改行もめいわくである。

4. で使ったプログラムの場合，改行を1行0文字としてかぞえると，表3に示したような不必要なメッセージが出てくる。印象的にいうと，Hは何のシステムでもやたらとメッセージを出したがるようだ。電話端末の利用者のことなど全然考えていない。

表3 不必要なメッセージ

処理系	T	MF	HV	MX	HS	HT	M4
行数	7	13	6	16	21	24	49
文字数	12	35	168	201	615	667	934

M4：コンパイル時にかなりのメッセージが出る。オプションNSの指定で大幅に改善される。これをデフォールトにするべきだ。なお，ローダから長々としたメッセージが出るが，こちらは消しようがない。まったくのムダであろう。

HS,HT：ラインプリンタのイメージでメッセージを送ってくる。くだらない。

T：プログラムがすぐ実行される感じ。

6. データをファイルに入れてプログラムで読むことが簡単にできなければならない。

T,Mは問題がないが，H（HS,HT,HV）は次の点でまったくひどい。

①端末から入力したデータ（当然可変長レコード）を可変長レコード形式のファイルに入れると，実質的に読めなくなる。入力した各行の文字数Nを完全に知っていて，　READ(1,1)(A(I),I=1,N)　としなければならない。N文字以上読もうとするとエラーになる。

②固定長ファイルに対して，レコード長よりも長く読もうとするとエラーになる。たとえば，80バイト固定長のファイルに対して次のプログラムでは読めない。

```
READ(1,1) (A(I),I=1,132)
1 FORMAT(132A1)
```

③ファイル形式を一つ一つのファイルに対してユーザが指定しなければならない。コマンドが長くなり，ミスタイプが増える。（HS,HT://DTFコマンド，HV:>>FILEコマンド）ファイル形式を指定しないと'未定義形式'にな

るが，これは一層使いにくい形式である。
例 //FT01F001:DTF FN=OUTDATA,FCB=(RECFM=FB,RECL=80,
BLKSIZE=800),FORG=S,SPACE=(TRK,,(3,2),RLSE)
参考：Mでは !SET F:1/OUTDATA;OUT;SAVE
Tではプログラムの実行と同時に *RUN *#OUTDATA"01"
④東大センターの用意したHT は80バイト固定長の世界であって，81バイト以上のレコードを含むデータファイルは作れない。(//DATAコマンド)
HVでは//DATAに相当する>>ASSIGNコマンドがあるが，やはり80バイトの世界である。
⑤HSの//FCREATコマンドで固定長のデータファイルを作ろうとする場合，そのままでは<CR>で区切って1行ずつデータを入れることができない。わざわざ"INPFORM=LINE"という指定をしなければならない。デフォールトがバッチ向け(INPFORM=CARD)にできているのだ。
⑥ファイルの最小サイズ，および増分サイズが大きすぎる。HVでは最小・増分とも19KB単位である。M(1KB)，T(320ワード)ぐらいがよい。なにしろ，ファイルの多くは"小さい"のがあたりまえなのである。MやTの利用者は各自のもっているファイルのサイズの統計をとってみよ。
⑦一度大きくなったファイルは，たとえ小さいものをそこへ入れても自動的に小さくはならない。(これはTも同じ。Mは小さくなる。) HVの区分データセットがもっとひどいことは周知の事実であろう。ファイルのムダづかいをするように設計されているのである。
⑧ファイルは，増分の指定によって15回まで大きくなることができる。しかし，なぜ無限回ではいけないのだろう。無限回できるなら増分サイズは当然最小値をデフォールトにしておける。初期サイズも最小にすればよい。すなわち，ユーザは必要最小限なファイルサイズの計算から解放される。
結論：Mの方式を最善と考える。テキストファイルはすべて可変長レコードであり，それをプログラムで読むとき右端に無限個の空白を補なって解釈する。ファイルのサイズはいつも必要最小限になっている。Tはファイルサイズ以外はMと同じであり，よい。
FORTRANのユーザーにとって，ファイル形式が固定長だとか可変長だとかレコード長がどうだということはまったく不要なことなのであって，書式つき入出力文を使うのか，書式なし入出力文を使うのかが必要にして充分な情報なのである。効率をよくするためのブロッキングなどはOS側で適当にやってくれればそれでよい。
東大大型センターのOS7時代のプログラム相談員のひとりはこう言った。
「ここではファイルが使えるようになったらりっぱに一人前ですよ。とにかく質問の中で最も多いのがファイル関係ですから。」
ファイルはTSSの必需品である。こんな発言が出てくるようでは困る。これではTSSが充分使いこなせるはずがない。VOS3時代になった今，この種の発言はなくなったのだろうか？

7. プログラムの形式は自由でなければならない。
自由な形式とは，1～5カラムの文番号，6カラムの継続行のマーク，7～72

表4　各処理系の許すプログラムの自由度

	T	MF	M4	MX	HS	HT	HV
文番号の位置は1～5カラム以外も可か	○	×	×	×	×	×	○
文が1～6カラムから始まってもいいか	○	×	×	×	×	×	○
文の右端は何カラムか	595?	72	72	72	72	72	72

の本文という制限がない形式のことである。表4に示したように，Tが最も自由である。自由な形式はドキュメンテーションの観点からはよろしくないが，手軽にプログラミングできる効果は大きい。ちょっとテストプログラムを作って走らせるときなど，重宝する。文の右端が72カラムに制限されている（JIS準拠）のはTSSではよろしくない。長い文字列欄記述子（nH や '……'）をもつようなFORMAT文を端末から正しく入れられようか。たとえ入れられたとしても，あとでコンマの抜けに気づいたとき，エディタでうまく直せるだろうか。

なお，HVは自由形式でも解釈でき標準形式でも解釈できるようなプログラムが書けない（注釈行・継続行の指定のちがい）。また，オプションを指定しないと自由形式がコンパイルできない。この2点でTよりも劣っている。

自由形式のプログラムを標準形式に直すソフトウェアはぜひ必要だが，HVにあってTにはない。東北大の大型センター（ACOS）のFORMコマンドがほぼそれに近い。

8．コンパイルと（リンクと）実行が一つのコマンドでできるほうがよい。

FORTRANのユーザから見れば，処理系がコンパイラだろうとインタプリタだろうと，あるいはリンクエディットしようとしまいと，そんなことはまったく関係ない。とにかくプログラムが走ればいいのである。

この条件を満たすのは，T，MF，MX，HS，HVであり，満たさないのはM4とHTである。ただしHTは複合コマンド //TFO;TRU が使える。またHSではコンパイルと実行のコマンド//FORTCGあるいは//FORTCLGが一つのジョブステップ内で一度しか使えないという大きな欠点がある。

9．プログラムの作成・修正・実行は有機的に結ばれていなければならない。

この観点から見て，よくないのはMF，M4，HS，HTである。いずれもエディタとFORTRAN処理系が独立してTSSのTにぶら下がっている。最もすぐれているのはTで，FORTRAN処理系からエディタが簡単に呼び出せること，FORTRAN処理系内で一行だけのエディタコマンドを入れることによって一時的にエディタが呼び出せること，エディタの中からプログラムの実行ができること，などの特徴がある。次によいのがMXで，FORTRAN処理系の中にエディット機能が組み込まれている。しかし，この修正コマンドが普通のエディタのそれとやや異なっているのがまずい。似ているだけに誤解のもとである。次によいのがHVで，エディタの中からプログラムの実行ができる。しかし，エディタの中で使えるコマンドに制限があって，ファイルの割り当てなどで不都合があること，コンパイラオプションの指定ができないこと（少なくともマニュアルには書いてない）などの欠点がある。8．と9．で述べたことによって4．のような差が生じるのだろう。

10. 実行時のブレイク処理については不満が多い。

FORTRANプログラムをTSSで走らせるメリットの一つは，途中経過などを実際に目で追いながらプログラムの動きを監視することができる点である。実行中のプログラムはいつでも<BREAK>キーを押すことによって実行を中断できる。

ブレイクのあと，そのプログラムの実行が再開できるほうが便利である。TやHVは再開できない。さて，ブレイクから再開までのあいだにどういうコマンドが入れられるべきなのだろうか。MやHS,HTでは，CPU消費時間の表示など，かなり制限されたことしかできない。これでは不充分であり，ブレイクのメリットが生かせない。筆者がほしいと思うブレイク直後のコマンドには，次のようなものがある。

①プログラムで出力中のデータファイルの現在の大きさを表示する。
②プログラムで出力中のデータファイルの内容を端末上に表示する。
③プログラムで入力中のデータファイルの内容とそのうちどこまで入力済みかを端末上に表示する。
④ソースプログラムファイル自身（あるいは他のファイル一般）を表示する。
⑤ソースプログラムのどのルーチンの何行目を実行中かを表示する。
⑥各ユーザに許可されている最大ファイル使用量に対して，現在どれくらい余裕があるかを表示する。（①と関連）

これらのほかに，もっと一般的な処理を行ないたいこともあるので，それを考慮すれば，Mのようにコマンドでブレイク直後の状態を保存しておけることが望ましい。

11. TSSはデバッグ向きである。

TSSが最もその威力を発揮するのは，プログラムのデバッグに関してである。TSSによるデバッグでは二つの点が重要である。①ブレイクによってプログラムの実行が中断できること。②プログラムが簡単に変更できること。この二つを生かすと，ソースプログラムのレベルでのダンプとトレースが手軽に実現できる。プログラムの中に自由書式出力文をいくつかはさんでおいて途中経過をどんどん出すようにして，適当なところでブレイクを押せばよい。

プログラムが簡単に変更できるためには，FORTRAN処理系がエディタと密接に関連していなければならない（9. 参照）。それに，できればBASIC風の修正機能がほしい。行番号を用いて手軽に削除・挿入・置換をする機能である。これが備わっているのはT，MX，HVである。

12. ていねいな処理系ほど使いやすい。

デバッグには処理系のていねいさがものをいう。コンパイル時に検出されるエラーに関しては，それほど処理系による差が大きくないので，ここではふれない。むしろ実行時のエラーのほうが重大であり，かつわかりにくい。デバッグというのは実行時のエラーをなくすことなのである。以下では経験上おかしやすい誤りをいくつかあげ，それぞれを各FORTRAN処理系がうまく見つけて正しくユーザに知らせてくるかどうかを比較する。実行時エラーとして次の五つをとりあげる。

①添字の値が宣言された大きさよりも大きくなる。

②実引数と仮引数の個数が異なる。
③実引数と仮引数の型が異なる。
④サブルーチン内で仮引数の配列の宣言を忘れ，実引数の配列を関数として実行する。
⑤関数の型が，呼ぶ側と呼ばれる側とで異なる。

表5　各処理系のていねい士

処理系 (モード)	T	MF		M4	MX		HS, HT		HV	
		ノーデバッグ	標準(デバッグ)		標準	ミニデバッグ	標準	デバッグ	標準	SUBCHK
①添字オーバー	X	X	○	X	X	不充分	X	X	X	○
②引数の個数	X	X	○	X	X	○	不充分	不充分	X	X
③引数の型	X	X	○	X	X	○	X	X	X	X
④配列と関数	X	X	○	X	X	○	X	不充分	X	X
⑤関数の型	X	X	X	X	X	X	X	X	X	X
(コンパイル)	X	X	○	X	X	X	○	○	○	○

↳ 変数に値が代入されているが，どこにも参照されていない。

○はエラーの種類とそれがソースプログラムの何行目で起きたかを知らせることを表わす。この表でわかるようにMFが最もていねいであり，次がMXのミニデバッグモードである。MFはデバッグモードが標準であることに注意してほしい。

プログラムを実行させるときは，“速く能率よく”よりも“正しく”が必要なことであり，デバッグ向き処理系を使うのがよい。それによるコスト増は保険のかけ金みたいなもので，たかが知れている。カーニハン[2]もいっている。

「虫取り向きコンパイラを使おう」

HS, HTではFORTRANの文の一つにDEBUG文というのがあり，これで添字の範囲のオーバーなどは見つかる。しかし，DEBUG文はそれぞれのプログラム単位ごとに入れなければならない。これでは，サブルーチンのたくさんあるプログラムの場合，一体どうしたらいいのか！HVではコンパイラオプションSUBCHKがあり，この問題は一応解決されている。

13. オンライン・デバッガを使いにくい。

MX, M4, HVでオンラインデバッガが使えるが，覚えるコマンドが多くて使いにくい。使いやすいエディタがあれば，そのほうがデバッグが楽である。(詳細略)

14. シンタクス・チェッカは不要の長物である。

OS7の//FORTSYN, VOS3のエディタのSCANオプション，それにMXでシンタクス・チェッカが使えるが，TSSプログラミングには不向きである。(詳細略)

参考文献

1) 荻野綱男，TSSエディタの使い勝手，bit, Vol. 11, No. 1～8, 1979
2) カーニハン，プローガー著(木村泉訳)プログラミング書法，共立出版，1976