東大TSS（HITAC M200H／VOS3）におけるコマンド体系

（東京大学大型計算機センター）
石田晴久，長谷部紀元，鷹野澄，小沢宏，木村友則

1. はじめに

東大センターでは、昭和55年度予算で、計算機借料の予算が、従来の年額約7億円から、来年1月からは年額約12億円余に増額されることになったのに伴い、去る5月24日から待望の新鋭超大型コンピュータの導入を開始した。

この新システムは、汎用機としては世界最高速機である（日立製作所の）HITAC M－200Hプロセッサを中心とするもので、次のような構成をもち、単一システムとしては世界最大級の規模のものとなる。全システムがフル稼動に入るのは来年1月であるが、現在すでに全体の約半分が稼動している。

プロセッサ　M－200H　計8台（2台＊4系）
内蔵型アレイプロセッサ　6台
主記憶　計64メガバイト（16メガバイト＊4系）
ディスク　317.5メガバイト＊96台／200メガバイト＊32台　計 36.8 ギガバイト
マス・ストレージ系（MSS）50メガバイト＊706本＝ 35 ギガバイト
ドラム（TSS系のみ）15メガバイト＊ 8台＝ 120 メガバイト
磁気テープ装置　12台
通信制御プロセッサ　8台（うち3台はネットワーク用）
周辺機器　カードリーダ（7台），ラインプリンタ（13台），OCR（1台）
図形ディスプレイ（7台），XYプロッタ（6台），漢字端末（5台）
TSS端末（館内用50台），その他多数
OS（オペレーティング・システム）----VOS3

2. 東大HITAC M－200Hシステムの特徴

このシステムの特徴は次の通りである。

(1) （各2台のプロセッサからなる）密結合多重プロセッサ4系からなる疎結合プロセッサ・システムになっているため、一元的に運用できて、ユーザ・ファイルは全系で共有でき、システム全体の信頼性や拡張性が高い。当センターでは旧システム（HITAC 8800／8700）で4台の大型汎用機からなる密結合多重プロセッサの技術を初めて実用化したが、今回の疎結合系も先進的な試みといえる。なお、4つの系には、当初次のような役割を割り当てる予定である。ただし、この割当は固定的なものではない。

TSS…CPU2台，メモリ16MBで端末250台の同時使用が目標、
通常バッチ処理
大型バッチ処理
グローバル制御---全体管理，入出力，バッチ

(2) プロセッサが世界最高速である。　M－200Hは現用の8800プロセッサの約3倍の速度をもつ。　ある種のベクトル演算や行列演算は内蔵型アレイプロセッサを用いれば、さらに高速に処理できる。（後者には8800の15倍のスピードで処理されるFORTRANプログラムの例がある。）
M－200Hのスピードは約10MIPSないし24MIPSである。
M－200Hの性能を示す数値を少しあげると次のようになる。
マシン・サイクル（最短命令実行時間）　42 nsec
バッファ・メモリ　アクセス＝　7 nsec，サイクル＝　21 nsec
主メモリ（NMOS）アクセス＝100 nsec，サイクル＝210 nsec

(3) TSSでは同時に250台ないし500台の端末が動かせる。TSS端末は、センターのユーザ向に現在約600台あるが、この数は近いうちに1,000台からさらに2,000台に増えるものと想定している。このうちマイコン（含インテリジェント端末）も多数接続されるものと予想されるため、回線としては、300 bps（ビット／秒）と1,200 bpsの電話線（300回線）のほか、9,600 bpsまでの専用線によるTSS端末接続も可能とする。
当センターのTSSでは、従来から、学生の実習ではなく、研究者ユーザによる利用であるところから、TSSジョブもかなり大きい。　疎結合であるため、各TSSジョブの開始時には、当のTSSプロセッサがグローバル制御プロセッサにジョブ起動の承認を求めるといったオーバヘッドがある。またTSSジョブでは、編集のみならず、完成したプログラムの本番ランもかなりあり、TSS負担はかなり高いのである。　このことを考慮し、TSS性能目標は、M-200H（2台）とメモリ16メガバイトのシステムで250台の同時アクティブ端末をサポートすることに置いている。これが可能であれば、そうしたシステム2系で500端末の同時サポートが可能となる。

(4) バッチ処理では、各ラインプリンタ毎に1台などディスプレイ装置をふんだんに使った高度なセルフサービス方式が使える。　オープン入出力方式（セルフサービス入力、デマンド出力）は当センターが8800用に初めて開発したものであるが、出力リクエストがためられたり、マグ・サイン表示盤が使えるなど、今度のはその拡張強化版である。

(5) TSSとバッチで同じコマンド、同じファイルが使える。現在の商用大型機ではどれもこうなってはいないが、東大センターではTOOLコマンド系でTSSとバッチを統合し、使い勝手のよいシステム作りを目指している。

3. TSSとバッチを統合するTOOLコマンド体系

本システムの導入にあたって、われわれの最も力を入れたのが、TSSとバッチの統一的な使い方、すなわちTSSコマンドとJCL（ジョブ制御言語）の統一である。これはわれわれの旧システム（HITAC 8800／OS7）では実現されていたが、IBM機および日立・富士通のMシリーズ機のバッチでは1964年に

発表されたIBM360伝来の悪評高いJCLがそのまま使われていて、TSS用のTSOコマンドとは全く別の世界になっている。そこで、前にOS7で使っていたTOOL（TOkyo OnLine）コマンド体系にならって、VOS3用にもTOOLコマンド体系を作ったが、その基本的な考え方は次の通りである。

(1) バッチでも（JCLの代りに）原則としてすべてのTSSコマンドが使えるようにし、TSSとバッチでのシステムの使用法の統合をはかる。

(2) ユーザ・ファイル（データセット）は名前さえ指定すれば、中がメンバーに分れる区分ファイルとして自動的に作成されるようにする。

(3) ユーザ・プロファイル機能を活用し、コマンドのパラメータは一度指定すれば、プロファイルに記憶されるようにする。

(4) 各言語毎にコンパイラは1種のみを標準ときめ、Fortran, Pascal, Cobol PL/I, Basic, Algol, アセンブラのソース・プログラムは、同形のコマンドで統一的にコンパイル・リンク・ゴーができるようにする。またコンパイル時にオブジェクト・ファイルは作らず、コンパイル後のオブジェクトは必らずロード・モジュール形式にしてファイルに保存するようにする。

(5) コンパイルなどの過程で一時ファイルおよび一時仮想ファイル（仮想記憶中にとられるVIOファイル）が使えるようにする。

(6) VOS3のもつSAFE機能を使い、他人のファイルの共用と保護が容易にできるようにする。

(7) 自家用コマンドとしてユーザがコマンド・ファイルを作るのを容易にするため、コマンド・プロシージャの機能を強化する。

4. TOOLコマンド体系の実現

TOOLの実現にあたって、最も苦慮したのは、VOS3のTSSコマンドおよび（EXEC文とDD文よりなる）JCLとの共存性であった。1センターだけで独自に全てのTSSコマンドを開発するのは、明らかに手に余るから、共存性は重要である。そこで結局、妥協案として、

(A) あるコマンド（useコマンド）を発行するとTOOLの世界に入る。〔TOOLからぬけ出すのはunuseコマンドによる。〕

(B) TOOL系のコマンドと他の一般TSSコマンドはまぜて使ってよいが、後者にはパラメータの記憶はない。〔このため記憶の有無について、ユーザに多少の混乱が起こりうる。〕

(C) 一般コマンドのうち、editとlinkは、名前はそのまま変えずに、パラメータの一部を変更してTOOLに組入れる。

(D) バッチ・ジョブの開始時にはTSSコマンド・インタプリタを起動させる。

以上のような方針のもとに、TOOLコマンド体系を次のようにして実現した。

(1) TSSとバッチの統合

まず問題だったのは、TSSにはないカード・デックの扱いであるが、これはTSSのプロンプティング記号を次のように変更し、

READY（CR-LF） ⟶ >>
EDIT （CR-LF） ⟶ E>

>>をコマンド接頭辞として使うこととし、次の二つのコマンドを用意した。

TSSでもこの方が見やすいし、改行しない分だけスペースの節約になるからである。

```
>> use sourcefile
>> source member
```

ソース・カード・デック

```
>*
>>data
```

データ・カード・デック

```
>*
```

この場合、ソース・ファイル名は一時ファイルでも、保存ファイルでもいずれでもよい。この擬似JCLでは、>>印はOSによる認識が不十分で、>* を抜かすと、>> がデータとみなされることがありうる。

次に問題なのは、TSSとバッチでのファイル書込みのぶつかり合いである。TSSでは、同一ユーザのバッチ・ジョブでその時丁度書き込みを行っているファイルにアクセスしようとすると、アクセス不可能のエラーとなって、ぶつかったことが分るからよいが、バッチ・ジョブでアクセスしようとしたファイルに、TSSで書込みをしようとしていた場合は、本方式ではバッチ・ジョブが異常終了してしまう。本来のバッチならば当のファイルがあくまで待ち合せが行われるが、TSSコマンドをバッチで使うときは、そうはいかないのである。　このぶつかり合いを避ける方法としては、ダミーのファイルをTSSでold（共用不可）に指定し、バッチでは、「そのファイルのoldが解除されるまで待つ」コマンドを頭につけるとか、さらには、TSSでバッチ・ジョブの実行を一時保留（hold）させるのもひとつの手であるが、これについては使い勝手やデットロックなどの問題があり、現在まだ検討中である。このほか、本方式ではジョブ・ステップの概念がなくなって困ることがあるので、free all（すべてのファイルを解除）コマンドなどの導入を予定している。

(2)　ユーザ・ファイルの自動作成

VOS3の主要ファイル編成は、順と区分であるが、

(A) ファイルの最小サイズは19キロバイト（1トラック）とかなり大きいが、順編成では、小さなファイルが1つしかそこに入らず、スペース効率が悪い。

(B) ファイル・ディレクトリ（カタログ）がツリー状（ハイアラーキー）になっていないので、順編成ファイルの一括扱いができない。

などの理由で、TOOLの標準ファイルは、ソースおよびロード・モジュールの両方とも区分ファイルにした。これなら、小さいメンバーが多数ひとつのファイルに入れられるし、ワイルド・カード文字などの使用で、メンバーの一括扱いが可能となる。TOOLでは、原則として次の形にソース・ファイルとロードモジュール・ファイルをジョブの頭で宣言するのを標準にすることにした。これらのファイル名は記憶される。

```
>>use sourcefile,loadmodulefile
```

ただし、VOS3で使われる区分ファイルにはメンバー内容を変更するとゴミ（無効）領域ができるため、ときどきパックする必要がある。

(3)　コマンドのパラメータの記憶

VOS3のコマンドでは、パラメータは指定しないと、一般にその値を問合せて

くるようになっているが、TOOLでは無指定時には記憶されている値を使うことにした。TOOLで記憶されるパラメータの例は次の通りで、>>state コマンドで表示できるようになっている。

```
>>state
  CODE      VAL(FORT)・・・・・プログラム言語の種別
  SDSN      VAL(F.FORT)・・・ソース・ファイル名
  SMEM      VAL(PAI)・・・・・・同上内のメンバー名
  MDSN      VAL(F.LOAD)・・・・ロード・モジュール・ファイル名
  MMEM      VAL(PAI)・・・・・・同上内のメンバー名
  FOPAR     VAL(NOSOURCE)・・FORTRANコンパイラ用パラメータ
```

各プログラム言語については、コンパイル時オプションとライブラリ名が記憶の対象となる。

ところで、この記憶方式で問題なのは、上述のパラメータが指定する毎に常に記憶され、従って記憶内容がその都度変ってしまうことである。これは困ることも多いので、パラメータを記憶しないモードの設定やmemorize コマンドの付加を現在検討している。パラメータを指定しなかった時に、どんな仮定値をとるのがよいか、問合せをする方がよいのかは、なかなか難しい問題である。

(4) コンパイラの標準化

主要言語については、どの言語でも、次のコマンドでコンパイル・リンク・ゴーができるようにした。 オブジェクト・ファイルは使わない。

```
>>use    sourcefile,loadmodulefile [,languagecode]
>>compile  member [,loadmodulefile]
>>link   module [,loadmodulefile]
>>go     module [,loadmodulefile]
>>cgo    member          (ロード・モジュールは作らない)
```

ここでmoduleというのは、ひとつのメンバーであり、〔〕内は省略可能であることを示す。もちろん記憶値を使うのであれば、module やソース member の指定もいらない。

次にcompile, link, cgo などで、どのようにしてコンパイラが選択されるかというと、それはソース・ファイルにつける(.fortもの ような)データ識別子あるいは、useコマンドで指定する言語コードによるのである。

ソース・ファイル名例	言語コード	レコード長	コンパイラ
my.asm	asm	80バイト固定	拡張アセンブラ
my.fort	fort		FORTRAN77
my.cobol	cobol		拡張COBOL
my.pli	pli		最適化PLI
〃	cpli		チェック型PLI
my.basic	basic	255バイト可変長	BASIC
my.pascal	pascal		PASCAL(日立版)

こうしたファイル名を守ると、ひとつのファイル中に異なる言語のソースをメンバーとして入れることはやりにくくなるが、この名付け方は、どんな言語のソースかが一目でわかって便利であろう。

(5) 一時ファイルの使用

VOS3のTSSコマンド体系では、もともとは、コンパイル時などに一時ファイルは使えないようになっていた。TOOLでは、センター・オウン・コーディング・ルーチンを組むことにより、¥や¥¥の記号をつけたファイルを一時ファイルとして使えるようにした。また3日目に消される短期ファイルも使用可能とした。ファイル名の例は次の通りである。

¥my.fort	一時ファイル
¥¥my.pascal	高速一時ファイル(VIOファイル)
@my.pli	短期ファイル

(6) ファイルの共用

これは、VOS3のSAFE機能とセンター・オウン・コーディングの組合せで、他人のファイルは、共用を許されていれば次の名前で使えるようにしている。

#userid.his.fort

ファイル共用に関したコマンドとしてVOS3には、permit, share, inhibit, dsalterなどがあるが、今後の機能強化予定としては、execute-only(read不可)の許可、ファイル名のワイルド・カード文字の導入によるすべてのファイルを特定のユーザに共用させる機能、ファイルを他人にゆずる機能などを計画している。

(7) 自家用コマンド作成のためのコマンド・プロシージャ作成機能

これは基本的にはTSO流のものであるが、われわれはユーザによる自家用コマンド作成の方法としてこれを重視し、VOS3にいろいろな注文をつけ改良をはかってきた。今後さらに、位置指定パラメータへの省略時仮定値の導入、メンバー名の一括指定機能の導入などを計画している。

5. おわりに

Mシリーズのような IBM汎用機のTSSは、TSO(TSS Option)ということばが示すように、オプションとしてスタートしたもので、Mシリーズは本来はTSSには余り向いていないシステムである。それを無理に巨大TSSとして使おうというのであるから、われわれの試みに困難がつきまとうのは当然である。しかし、VOS3の場合には、関係者の努力により、従来のよりは格段に使いやすくなったQEDエディタや論文清書プログラム(ROFF)やPASCALコンパイラやAPLがメーカ・サポートで提供されることなども含めて、かなり使いやすいTSSになりつつある。VOS3では、端末、特にインテリジェント端末の多様化に対応して、バルク入出力の導入や調歩同期無手順方式の9600 bpsへの高速化や全2重化などのサポートも実現されつつある。回線断時の自動ファイル・バックアップや再接続機能はすでに実現された。われわれとしては、使用経験にもとづいて今後もハードウェアを含めて改良を加え、少くとも今後数年にわたって使うことになるVOS3をコンピュータには素人である研究者ユーザにとって、手軽に高度な機能の使えるTSSに育ててゆきたいと考えている。

最後に本システムの開発にたずさわった日立製作所、ファコム・ハイタック社、東大大型計算機センターの関係者の方々に謝意を表する。