新しいオペレーティング・システムの仮想記憶機能

大阪大学大型計算機センター
研究開発部　藤井　護

（本稿は、速報（No.７５　増強システム特集）に掲載した「新しいオペレーティング・システムの概要」に、２・３節と３章の全部を追加したほか、適宜加筆修正したものです。とくに３章だけでも目を通して頂きたいと思います。）

０．はじめに

速報（No.７５）でお知らせしたように、追加導入したシステム（ＡＣＯＳシステム９００モデル２，８ＭＢ）によるサービスを９月１日から開始しています。このシステム（システムⅡと呼びます）では、仮想記憶（Ｖｉｒｔｕａｌ　Ｓｔｏｒａｇｅ，ＶＳと略す。）の機能を持ったオペレーティング・システム（ＯＳと略す）であるＡＣＯＳ－６のＲ７．１版を採用しています。これまではＣジョブでも、プログラムの大きさが１８０ＫＷまでという制限がありましたが、Ｒ７．１版のＯＳでは、ＶＳ機能によりこの制限が大幅に緩和され、16,000ＫＷまでのプログラムがかけられるようになりました。ただし、この機能は当面は、バッチ（会話型リモート・バッチ，リモート・バッチを含む）ジョブでしか使えません。*

なお、このＲ７．１版には、デマンド・プリント機能など当センターの運用上なくては困る機能が備わっていないため、主力機である在来のシステム（システムⅠと呼びます）では、これらの機能が備わり、システムⅡとのディスクの共有が実現する１１月末までは、従来のＲ５．２版をそのまま用います。１２月からは両システムともに、Ｒ７．１版を使う予定です。

以下にＲ７．１版のＯＳのＶＳ機能を簡単に紹介します。使い方については、センター・ニュース本号の「速報」及び「お知らせ」の集録の**３．システムⅡのジョブ制御言語**を御覧下さい。

１．ＡＣＯＳのＯＳの名称

ＡＣＯＳ９００のＯＳは、ＡＣＯＳ－６という名前ですが、継続的に機能の拡張・整備が行われており、ほぼ年に１回（原則的には各年度末）、新版のＯＳがメーカーから提供されます。**
版名はその時に１つ増えます（Ｒ４．１からＲ５．１になるように）。このほか、ＯＳの機能拡張

---

*　ＴＳＳジョブでも使えるようになるのは、２年後のＲ９．１版からの予定です。

**　この提供後、センターの運用に合わせたソフトウエアの開発やテストを行なうため、使えるようになるのに何か月かかゝります。

の本流とは別系統に機能の拡張整備が行われる場合がありますが、この場合は版名は0.1増えます（Ｒ４．１からＲ４．２になるように）。各版が利用可能になった年度を表１に示します。ちなみに、当センターで初めてＡＣＯＳ－６を使ったのは、ＡＣＯＳシステム７００を導入したとき（昭和５１年１０月）で、Ｒ３．１版だったと思います。

表１　　Ｒシステムと Ｖシステム

年度 / システム	51	52	53	54	55	(56)
Ｒシステム	Ｒ３.１	Ｒ4.１	Ｒ4.２ Ｒ5.１	Ｒ5.２	Ｒ５.３	
Ｖシステム				Ｒ6.１	Ｒ７.１	(Ｒ８.１)

ＡＣＯＳ－６のどのマニュアルにも，はしがきの最後の備考欄に「本書は，リリース５・３／７・１に対応しています」などと書かれていますが，この「５．３」や「７．１」は，上記のＯＳの版名です。また，ＴＳＳを使う時，最初に「ＡＣＯＳ－６　ＴＳ１（Ｒ５．２）　ＯＮ　１０／１６／８０……」といったメッセージが出力されますが，この「Ｒ５．２」もそうです。*

ＶＳの機能は，Ｒ６．１版でひとまず実現されましたが，** Ｒ６．１版，Ｒ７．１版などＶＳ機能のあるＯＳをＶシステム，そうでないものをＲシステムといいます。また，Ｖシステムには，ＶＳ機能を使ってプログラムを動作させるＶモードと，従来と同様の機能をもつＲモードがあり，ジョブ制御言語（Ｊｏｂ Ｃｏｎｔｒｏｌ Ｌａｎｇｕａｇｅ，ＪＣＬと略す）で指定するようになっています（後述）。

２．ＶＳの方式

ＶＳについては，数多くの教科書や文献(1～4)がありますが，以下にもＶＳの原理や各種の方

＊　バッチ処理の場合のリストには出ていないようです。（ＦＯＲＴＲＡＮのジョブで，ソースリストの各頁の先頭に出る「ＡＣＯＳ－６　Ｒ００７……」のＲ００７はＦＯＲＴＲＡＮコンパイラの版名です。）

＊＊　この版には，マルチＴＳＥ，予算管理，ジョブ処理状況表示ー問合わせ等の諸機能のほか，前述のデマンド・プリント機能がなく，当センターでは運用上採用できませんでした。

式を簡単に紹介します。

2.1 ＶＳの原理

使用する計算機に実装されている，あるいはアーキテクチュア上制限されている主記憶の容量にかかわりなく，十分大きなプログラムでも実行できるようにするためには，プログラマが使える〝番地〟と，実装されている主記憶の番地とを分離して考える必要があります。プログラマが使える番地とその集合を，それぞれ仮想アドレス，仮想空間といいます。また，主記憶の番地とその集合を，それぞれ実アドレス，実空間といいます。ＶＳの機構を一口でいうと，この仮想空間から実空間への変換をシステムが動的に行うことです。ＶＳの機能がないシステムでプログラムの大きさが実空間の大きさを越える場合には，この変換を〝オーバレイ（Overlay）〟技法を用いてプログラマが行っていました。即ち，プログラマがそのプログラムを予めいくつかの部分に（各部分の大きさが実空間のそれよりも小さくなるように）分割してディスクなどの外部記憶に蓄えておき，各部分は必要になった時点で主記憶に転送され，不要になった部分が占めていた領域に〝オーバレイ〟するように制御していました。

ＶＳ方式では，このプログラムの分割（分割された各部分をブロックと呼ぶことにします）や，外部記憶と主記憶間のブロックの転送をシステムが自動的に行います。

このブロックへの分割のし方には大別して二通りの方法があります。メイン・プログラム，サブルーチン，配列など，論理的なまとまりごとにブロック化する方式をセグメンテーション方式と呼び，このブロックをセグメントと呼びます。この場合，ブロック（セグメント）の大きさは，一定ではありません。もう一つは，プログラムを一定の大きさのブロックに機械的に分割する方式で，ページング方式と呼びます。また，この場合のブロックをページと呼びます。いずれの場合でも，仮想空間上のプログラムの番地（仮想アドレス）は，ブロック番号とブロック内アドレスの対で表現されます。例えば図１で，仮想アドレス l は，（i，d）で表現されます。

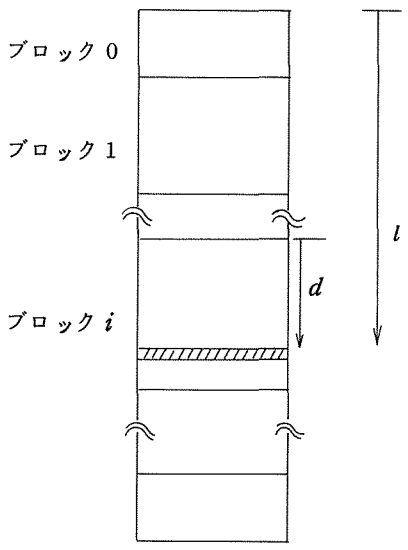


図１　仮想空間上のプログラムと仮想アドレス

仮想アドレスから実アドレスへの変換は，動的アドレス変換機構（Dynamic Address Translation mechanism，DATと略す）というハードウエアによって行われます。DATは，ブロック・テーブル（セグメンテーション方式のときはセグメント・テーブル，

ページング方式のときはページ・テーブルといいます）を内蔵しています。このテーブルは，仮想空間上の各ブロックについて，実空間（主記憶）にあるときはそのブロックの先頭の実アドレスが，ないときは特別な記号ϕが記入されています（図２）。 ＣＰＵから仮想アドレス（i，d）が発せられると，ＤＡＴはブロックテーブル中のブロックに対応する実アドレスを探します。

① これがi'（$\neq\phi$）のとき，ブロックiは主記憶のi'番地から格納されていることを意味しており，ＤＡＴは実アドレス（$i'+d$）を生成して主記憶にアクセスします。

② これがϕのとき，ブロック不在フォルトと呼ぶ割込みが発生し，記憶管理プログラムが作動して，ブロックiを外部記憶から主記憶の空き領域（これがなければ，〃不要不急〃のブロックを外部記憶に追い出して空いた領域）に転送し，この動作に合わせてブロックテーブルの内容を修正します。その後，①の動作を行ないます。

このようにして不要不急のブロックは外部記憶に格納され，必要なブロックは主記憶に格納されることになります。ところで，②で，どのブロックが〃不要不急〃であると判定するのか，などといった問題がありますが，これについては，２．３節で述べることにします。

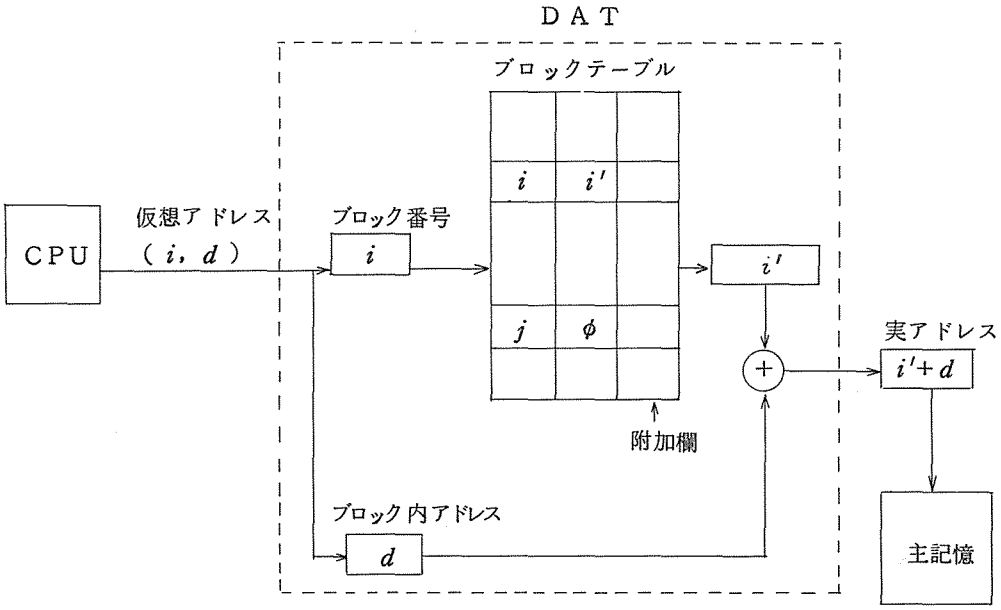


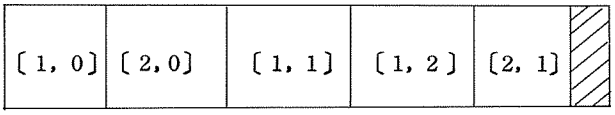
図２　動的アドレス変換機構（ＤＡＴ）

2.2 各方式の比較

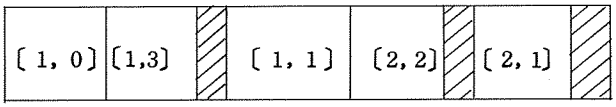
ＶＳの方式には，前節で触れたセグメンテーション方式，ページング方式のほかに，セグメンテーション&ページング方式があります。

(1) セグメンテーション方式

セグメントの大きさは一般にまちまちです。実空間に最初は隙間なく格納されていても（図3 (a) ），やがてはセグメントが格納されている（使用中の）領域と空き領域が縞模様（checkerboard）のようになります（図3 (b)）。このため，空き領域の総和としては十分大きくても，バラバラのために次のセグメントを格納できないことがあるという欠点があります。



(a) 初期状態



(b) "checkerboarding"

〔a,b〕はプログラムaのセグメントbを表わす。斜線部分は空き領域。

図3　セグメンテーション方式

一方，セグメント・テーブルに欄を追加して，その欄に特別の印がついていないときはそのセグメントを他のユーザが読み書きできないように制御する機構や，別の欄を設けてそのセグメントの大きさを記入しておき，仮想アドレス（i，d）が与えられたときdがその大きさを越えていないかをチェックする機構*など，いわゆる情報管理の機能を持たせることができます。図2のブロック・テーブルで，これらの欄を"附加欄"と名付けておきました。

(2) ページング方式

実空間も，仮想空間のページと同じ大きさに区切る（これをページ枠といいます）と，仮想空間のどのページも任意のページ枠に格納できます。また実空間上で無駄になる空き領域はプログラムの最終ページ（一般には途中までしかない）が格納されたページ枠の後の方だけです（図4）。

このように実空間の管理もしやすく，効率も良い方式ですが，セグメンテーション方式のよ

---

*　この機構により，セグメントが唯一の配列から成る場合，配列外のデータ（またはプログラムのコード）にアクセスしようとするプログラム・ミスを容易にチェックできます。

うな情報管理機能を持たせ難いのが欠点です。

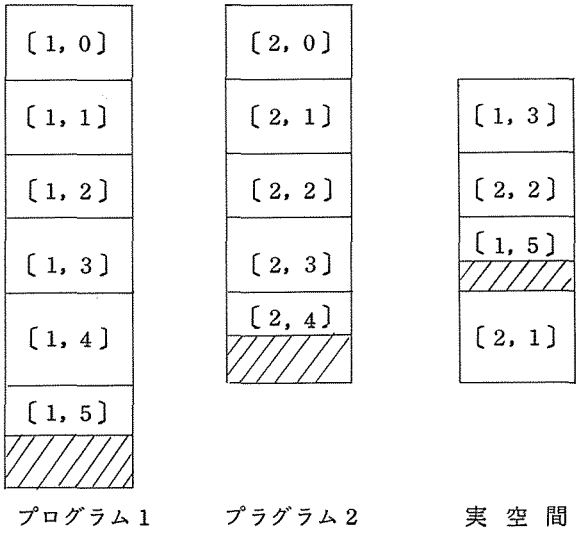


図 4　　ページング方式

(3) セグメンテーションページング方式

これは両者の長所を活かした方式で，情報管理はセグメンテーション方式を，実空間管理はページング方式を採用したものです。ACOS－6のVSはこの方式です（4章参照）。

2.3 主記憶管理の3方針

VSシステムでは，前節に述べた方式のほか，主記憶（実空間）の管理に関する次の三つの方針の決め方でそのシステムが特徴づけられます。

(1) 位置選択の方針（Placement Policy）

セグメンテーション方式では，セグメントを主記憶に格納するとき，そのセグメントの大きさよりも大きい連続した空き領域が必要です。そのような領域が複数個あるとき，どのような方針で空き領域を選ぶかが問題になります。候補になる空き領域のうちから最小のものを選ぶbest fit アルゴリズムや，最下位の番地から探していき最初に見つかった候補者を選ぶfirst fit アルゴリズムなどがあります。また，主記憶上のセグメントを一方へ詰めて隙間をなくし，複数の小さい空き領域を大きなものにまとめるmemory compaction を行うかどうか，行うとすればどのタイミングで行うか，などの問題もこれに含まれます。

(2) 格納時選択の方針（Fetch Policy）

ブロック（セグメントまたはページ）をいつ主記憶に格納するかが問題です。前節で述べたように，ブロックが必要になった時点（ブロック不在フォルトが発生した時点）でのみそのブロックの格納動作を開始する demand rule と，そのプログラムの履歴などから必要になりそうなブロックを予測して事前に格納する機能も持たせた anticipatory rule があります。

(3) 置換えの方針（Replacement Policy）

主記憶上に空き領域を作る必要が生じたとき，主記憶上のブロックを追い出す必要がありますが，どのブロックを追い出すかが問題となります。セグメンテーション方式の場合は優先度がより低いプログラムがあればそれを追い出し，なければ空き領域ができるのを待つといった方針がとられます。ページング方式では，代表的なものとして次のものが挙げられます。

a）FIFO（first－in first－out）：最も長く主記憶に入っているページを追い出す。

b）LRU（least recently used）：最近の一定時間の間に使われた回数が最小のページを追い出す。

c）RANDOM：全くランダムに選んだページを追い出す。

また，今回主記憶に格納された時点から現在までそのページの内容が変更されていないものは，そのコピーが外部記憶上にそのままあるはずですから，本当に〝追い出す〟必要はなく，そのページ枠に新しいページを上書きするだけで済みます。これにより，外部記憶装置へのアクセス回数を減らすことができます。上記の方針とこれを組合わせることもよく行われています。

以上の3方針はいずれもシステム全体の効率や個々の利用者の結果待ち時間や応答時間に直接影響を与えるだけに，さまざまの工夫がなされています。

3．VSの効率的な使い方

3.1 のたうちまわるシステム

前章で述べたように，VSシステムでは，プログラムの実体は外部記憶にあり，そのうちの必要になったブロックが主記憶に転送され，〝不要〟になった部分が（もし内容に変更があれば）外部記憶に戻されます。したがって，主記憶に格納されたブロックを集中的に参照するような構造のプログラムは効率が良く，逆に仮想空間上のあちこちを散発的に繰り返して参照するような構造のプログラムは，外部記憶への入出力ばかりで，極めて効率が悪くなります。（主記憶と，ドラムやディスクとでは，アクセス時間が10000倍～100000倍程も違います。）後者のようなプログラ

ムが原因となってＣＰＵ（頭）はほとんど働かずにディスクなどの入出力装置（手足）ばかりが動作するシステムの現象をスラッシング（thrashing――〝のたうちまわる〟の意）と呼びます。この現象は図５のような，極く〝普通〟のプログラムでも起り得ます。その理由と対策は次節に述べます。

```
      DIMENSION A(4,3)
      DO 10 I=1,4
      DO 20 J=1,3
      A(I,J)=0.0
   20 CONTINUE
   10 CONTINUE
      ⋮
```

図５　悪いプログラムの例

プログラムは，命令コードから成る手続き部分と，配列などのデータ部分に分けられます。手続き部分について，ある短い時間の間に参照される〝番地〟の集合を調べてみると手続き部分のごく一部分であり，相当な時間にわたってそのメンバーは安定していて入れ換りが少ない，という一般的傾向があります。（これはプログラムの制御の流れが，直線型かＤＯループ型かのいずれかで，時間的には比較的小さいループ内に留っていることが多いからであると考えられます。）このような性質を，プログラムの参照の局所性（locality of reference）といいます。したがって，データ部分，特に配列の構造が問題となります。

3.2　配列の使い方

ＦＯＲＴＲＡＮ言語に関する限り，ＡＣＯＳはもちろん，ＩＢＭを始めとした内外の主なシステムはほとんどそうですが，二次元以上の配列は，第１要素から順に変化するcolumn-wiseな配列として仮想空間に割付けられます。例えば，図５のプログラムの配列Ａは図６のような構造に展開されます。このプログラムが１ページの大きさが４語のページング方式のＶＳシステムで動作したとすると，Ａ（１，１），Ａ（１，２），Ａ（１，３），Ａ（２，１）……の順に０．０を代入するのに，毎回異なるページにアクセスすることになります。ページ枠が１つしか与えられていないとすると，毎回，ページの追い出し（page-out）と次のページの読み込み（page-in）の2回の入出力動作が行われることになります。

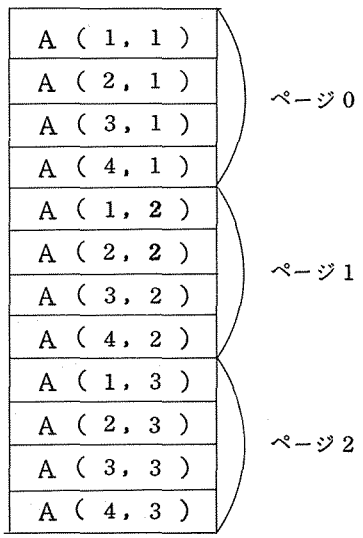


図６　配列Ａの割付け

一方，図７のようにプログラムを書き変え

```
      DIMENSION A(4,3)
      DO 10 J=1,3
      DO 20 I=1,4
      A(I,J)=0.0
20    CONTINUE
10    CONTINUE
        ⋮
```

図7　良いプログラムの例

B (1, 1, 1)
B (2, 1, 1)
⋮
B (n_1, 1, 1)
B (1, 2, 1)
B (2, 2, 1)
⋮
B (n_1, 2, 1)
⋮
B (n_1, n_2, 1)
B (1, 1, 2)
⋮
B (n_1, n_2, n_3)

図8　3次元配列

ると，ページインしたページの各語への操作が全部済んでから次のページに移ることになり，入出力動作の回数は激減します。

このように，二次元の配列に対する操作は，まず列方向に行うことが肝要です。三次元以上の配列に対しても同様で，例えば三次元配列B（n_1，n_2，n_3）は図8のように割付けられます。

くどいようですが，図5のプログラムと図7のプログラムのように，行方向に走査するプログラムと列方向走査のプログラムとでは，前者が後者の何万倍も遅い主記憶を使うことに相当します。VS機能を使う人はもちろん，そうでない人も将来のことを考えて，FORTRAN言語を使う限りは，列方向走査優先のプログラミングをするのが得策です。

なお，COBOLについてはFORTRANの逆で，最後の要素から順に変化するrow－wiseな配列となります。従って，この場合は行方向に走査する方が良いことになります。

4．ACOSのVシステム

最初に述べたように，VシステムにはVS機能をフルに働かせてプログラムを実行するVモードの処理法と，従来の処理法と見掛け上は全く同じRモードの処理法があり，JCLで選べるようになっています。

4.1　Vモード

ACOS－6では，プログラムで指定するアドレスは，セグメント番号sとセグメント内アドレスdの対（s，d）です。各セグメントごとに，セグメント記述子（2章で述べたセグメンテーション方式におけるセグメント・テーブルの行に代るもの）があり，これにはそのセグメントの大きさ，「読み」「書き」「実行」それぞれが可かどうかを示す属性や，仮想空間上のこのセグメントの開始番地$\bar{s}$などが記入されています。（図9）

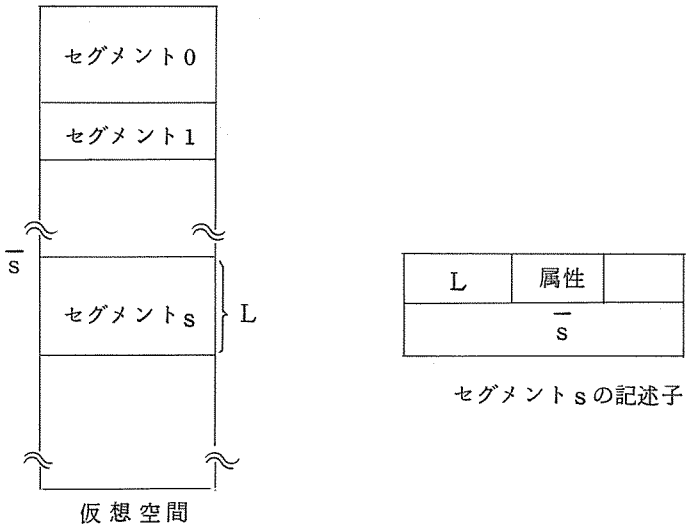


図 9　仮想空間とセグメント記述子

VS機構は，（ s，d ）が与えられたとき，セグメント記述子の属性をチェックし，またd≦Lかどうかをチェックした後，　$l = \overline{s} + d$ から目的のデータの仮想アドレス l を知ります。この後はページング方式と同じで l をページの大きさで割った整数部分をページ番号，余りをページ内アドレスとして2章に述べたページング方式のメカニズムで実アドレスにアクセスします。

このように，前半はセグメンテーション方式による情報管理機能，後半はページング方式による主記憶の管理機能を用いています。したがって，〃主記憶管理の3方針〃のうち，位置選択の方針は関係がありません。格納時選択の方針は，いわゆるdemand−paging で完全なdemand rule に従っています。（プログラムの実行開始時にのみ，プログラムの〃前半〃の複数ページを，割当てられた主記憶いっぱいに一度に格納するようなバリエーションもありますが，ACOSでは採用していません）。また，置き換えの方針としては，基本的にはLRUで，内容に変化がなかったページのみを対象にするなどのバリエーションを採用しています。

なお，1セグメントの大きさは256KW以内であり，1ページの大きさは1KW（1024語）となっています。また，一つのプログラムには一つの仮想空間が割当てられますが，この大きさは最大で約16,000KWです。

つぎに，ACOS−6では，仮想空間上のプログラムの実体は，ディスク・ファイルに格納されるのですが，2種類のファイルを用います。プログラムをリンカによって結合編集した結果は，まずランユニット（RU）ファイルに蓄えられます。ここから必要に応じてページが主記憶に転

送されますが，主記憶上でページの内容が変更されかつ外部記憶へ追い出されるときは，ＲＵファイルではなく，バッキングストア（ＢＳ）ファイルへ転送され，以後このページを参照するときは，ＢＳファイルのものが用いられます。

ところで，プログラマは，自分が使いたい仮想空間の大きさをＪＣＬのＶＳＰＡＣＥというパラメータで指定することができますが，この値は上記のファイルの領域を確保するために用いられます。従って，必要以上に大きな値を指定しますと，ファイルの空き領域がなかなか確保できずに待ち時間が長くなるだけでなく，一旦確保した領域はそのプログラムが終了するまで他のプログラムで使用できません。自分の為にも，他人の為にも，むやみに大きな値を指定しないで欲しいものです。（近い将来，実際に要る分だけが確保されるように改良される予定です。）

なお，ＡＣＯＳ－６では，各プログラムに割当てることのできる主記憶の大きさを制限できるようになっていますから，システム全体がスラッシングに陥る機会は極めて少ないはずです。しかし個々のプログラムについては，前章に述べたようにはなはだ効率の良くない動作をすることはいくらでもあり得ますので，くれぐれも御注意下さい。

4.2 Ｒモード

前にも述べたように，主記憶を同じ大きさのページ枠に区切ると主記憶の利用効率がよくなります。プログラム全体を同時に主記憶に入れるけれども，主記憶上で占める位置は連続領域ではなく，ページ単位に空いているところへ格納されるという方式は，ＶＳ機構の一部を用いるだけで実現できます。即ち，プログラム全体を入れるだけのページ枠（連続している必要はない）を確保し，ページ単位にプログラム全部を格納します。そして，ページテーブルに設けた欄に印をつけ，プログラムが終了するまで ｐａｇｅｏｕｔされないようにしておけばよいのです。これにより，ＤＡＴが動作したとき，ページ不在フォルトは起りませんから，従来のＲシステムと同じ性能（ＤＡＴの動作時間はほとんど無視できます）が得られます。この動作モードをＲモードといいます。

したがって，センター・ニュース本号の速報集録にあるジョブ区分で示すように２４０ＫＷ（従来は１８０ＫＷでした）以下のプログラムは，このＲモードで動作させるとよいでしょう。

参　考　文　献

(1) P.J.Denning, "Virtual Memory", Computing Surveys, vol.2, No.3, pp.153～189(1970)

(2) 元岡編 "計算機システム技術" ，第４章，pp.41～66，オーム社（1973）

(3) S.E.Madnick and J.J. Donovan, "Operating Systems", McGraw－Hill(1974).
（訳書あり，池田訳 "オペレーティング・システム"， 日本コンピューター協会（1976）

(4) "ＡＣＯＳ－６ 仮想記憶解説書"，マニュアルＦＣＺ０２－１，日本電気（1978）