Vol. 15 No. 10　　情　報　処　理　　Oct. 1974

≪解　説≫

NEAC システム 300/400 ACOS-4 のジョブ制御について

鈴木泰次*　　五島公太郎*

1. はじめに

コンピュータ・システムとその繋りを考えて見ると先ず，利用者は自分の業務システムを設計し，業務に必要なプログラムをコンピュータ・システムで用意されている．プログラミング言語（例えば COBOL, FORTRAN, PL/I, DDL（データ記述言語）等）を使ってプログラミングする．この様にプログラミングされた幾つかのプログラム及びデータとコンピュータ・システムが提供する幾つかのプログラムを合理的に運用するためにジョブ制御言語でジョブの記述をする．一方直接コンピュータ・システムを利用する者（オペレータや一部のプログラマ等）はコンピュータ・システムが用意したオペレータ・コマンド言語を介して，直接コンピュータ・システムと対話をしながら業務を運用する．

大部分のプログラミング言語は既に一般的になった馴染みの深い言葉と共通の文法を持っており，数多くの紹介がなされてきた．これに対しジョブ制御言語は各々のコンピュータ・システムの強い影響を受け，一般的な論議の対象となり得なかったためかプログラミング言語程には採り上げられてはいない．そこで今回は ACOS-4 のジョブ制御についてその考え方，ジョブ制御の流れ，ジョブ制御言語を紹介する．

2. ACOS-4 ジョブ制御の特徴

ACOS-4はバッチ，遠隔処理等を統一的に管理するオペレーティング・システムである．ACOS-4の下でユーザの業務がジョブとしてどの様に処理されるかを記す前に，ジョブ制御の考え方，特徴について触れる．

ジョブ制御言語：ジョブ制御言語（JCL）はシステム生成言語（SGL），システム更新言語（SUL），オペレータ・コマンド言語（OCL）等と共にシステム制御言語（SCL）を構成している．SCL という見方は例えば，JCL がジョブの実行とか制御を記述するための特別なコマンドとしてでなく，一般のプログラミング言語と同一の観点で捉えるという ACOS-4 の設計思想から来ている．ジョブ管理はコンピュータ・システムとユーザの間に介在するオペレーティング・システム（OS）の中で最もユーザに近い部分であり，人間にたとえると SCL 特に JCL は狭い意味での OS の顔であるともいえる．

プログラミング言語として，狭義の OS の顔として JCL は単にジョブを記述するだけでなく JCL カタログ，JCL マクロ定義，スーパー・マクロ記述，ユーティリティ記述等の豊富な機能をユーザに提供する．ACOS-4 JCL の特徴としては特に，JCL マクロ定義が GMP（汎用マクロ・プロセッサ）ベースで処理されることから，標準の JCL セットとは別にユーザ専用の JCLセットが容易に作成出来ることである．JCL セットの定義，JCLパラメータ既定値の規定はシステム生成時に行われる．

ジョブ・スケジューリング：設置システムの業務がセンター・バッチかオン・ライン業務か，業務内容が何であるかによって運用目標が異なり，ジョブのスケジューリングも異なる．設置システムの運用方式，現在の負荷状況，多数のジョブの相互関係及び各ジョブ要求に従ってジョブがスケジュールされる．スケジューリングに対するユーザの要求は次の三つの時点で行われる．

（1）　システム生成時，更新時

システムがどの様な運用方式を採るかは設置システム責任者が設置システムの目標に応じて設計する．従ってユーザは標準的なスケジューリング方式の他に固有のスケジューリング・アルゴリズムをシステム生成時に組込むことが可能になっている．

（2）　運用時の動的介入

システム・オペレータは OCL を用いて運用時点にジョブ投入の禁止，再開やスケジューリング方式を変更することが出来る．

* 日本電気(株)コンピュータ方式技術本部

(3) ジョブ投入時

ユーザは JOB や RUN といった JCL でスケジューリング・パラメータを指定し，スケジューリングに対するジョブ固有の要求を行う．(スケジューリング・パラメータ等についてはクラス，プライオリティといった重要な概念を含めて 3.3 参照)

これらに加えてシステム負荷状態やある種のイベントを動的に把握して特定のジョブを選択して実行したり，アボート（打切り）したり，ブロック(一時停止)したり，ジョブの受入れを制限したり，また，これらの逆を行ったりすることもある．

資源の割当，解放：メモリ，ファイル，デバイスといったコンピュータ・システムが有する資源は物理的に有限であるから，マルチ・ジョブ，マルチ・タスク環境下の資源とその機能を時間的，空間的に有効に利用しなければならない．同時に多数のジョブやステップ間で排他，共有する際の制御特に，システム処理能力の低下やデッドロックといった現象を未然に防ぐ手段も必要である．ACOS-4の資源管理はステップ単位の割当，解放を基本にして静的，動的な資源のスケジューリングが行われる．

ジョブ・スポーニング：オペレータがインプット・リーダを起動し，ジョブ記述を読込み，実行するのが通常のジョブ処理の手順である．ACOS-4 では更に，実行中のジョブが新たに他のジョブの実行を要求するジョブ・スポーニング機能を持っている．即ち実行中のジョブの JCL 順列中に準備される JCL かまたは，プログラム内のマクロ命令の実行が別のジョブの実行を促すことに成る．JCL またはマクロ命令で示されるファイル中の JCL 順列がソース形式であればインプット・リーダの開始を，既に内部形式の JCFI（インプットリーダの項参照）であればスケジューラの処理を要求することになる．

JCL マシーン：(JCL の解釈，実行) ソース形式の JCL はインプット・リーダによってソース形式及び内部形式のジョブ記述に翻訳，変換される．ジョブがスケジュールされ実行に必要な資源が割当てられると，コマンド・インタプリタと呼ばれるプログラムが内部形式の JCL を一枚づつ読み，解析し，実行することでジョブの処理が進められてゆく．このコマンド・インタプリタはジョブに対応して割当てられている様に見える．CPU がプログラムを実行するとき，機械語命令が一命令づつデコードされ，実行される様にオペレーティング・システムがジョブの実行を恰も JCL 文を命令の様に読み，解析し，実行してジョブを処理してゆくことからコマンド・インタプリタをJCL マシーンという言葉になぞらえることが出来る．

3. ジョブの投入から終了まで

ここではジョブが投入されてからどの様にスケジュールされ，資源の割当，解放が成され，実行されるのかを記述する．以下ではジョブ制御という OS 内部の側から記す．

3.1. ジョブの投入

ユーザはジョブを構成するステップ，ステップが利用する資源情報とかステップ間相互の受渡し情報（制御情報）といったジョブの実行に必要な記述を JCL やデータと一緒に準備する．準備されたジョブ記述は入力ストリームとしてインプットリーダ (IR) に依りシステムに読込まれる．

IR は入力デバイスごとに存在し，次の任意のデバイスを扱うことが出来る．

- カード・リーダ
- ディスク装置
- 端末装置
- その他（磁気テープ）

ジョブの投入：

ジョブの投入は OCL に依る IR の起動で始まる．ソース形式の入力ストリームが準備されるとオペレータは RUN コマンド及び入力デバイスと入力メディアが JCL ソースであることを示すコマンドを入力して IR に入力ストリームの読込みを開始させる．

ACOS-4では上記の通常の手段に加えて，ジョブスポーニングと呼ばれる動的なジョブの開始要求に伴って IR が自動的に起動されることは前に述べた．

IR のタイプ：

ACOS-4 はバッチ，遠隔処理等の複数のジョブ処理形態を統一的に管理する．各形態に於ける JCL はほぼ同一でありその差異はジョブ処理要求及び出力がセンターであるか遠隔地であるか或るいは，JCLが一括して処理されるか，対話的に成されるかの違いでしかない．処理形態に応じてバッチ・インプットリーダ，リモートインプットリーダのいずれかが働く．

IR の処理：

IR は大きく三つの仕事を行う．先ず入力ストリーム上に現れる各ジョブがシステムの正しい使用権利を持っているかどうかを検査する．正当であればそこで始めてジョブをシステムに受入れることになり，JCL

表-1 ACOS-4 JOB PROCESSING MODE & TYPE OF INPUT READER

ジョブ処理形態	説明	IR のタイプ
バッチ	入力ストリームはジョブ単位に読込まれ，検査される．ジョブが認められ，読込まれた後に実行される．ジョブ記述の誤りは再入力を必要とする．	バッチ インプット リーダ
リモート バッチ	入力デバイスが遠隔端末である点を除いてはバッチと全く同じである．	リモート インプット リーダ
リモート ジョブ エントリ	ジョブ全体が入力された後，ジョブの実行前にジョブ記述の訂正，編集が行える．他はリモートバッチと同じである．	

順列が読込まれる．(ジョブ・サブミッション，ジョブ・イントロダクション)

このとき JCL 順列はジョブの実行，制御を記述する JCL 部分と，実行時に利用されるデータ部分に分離されて SYSIN と呼ばれるディスク・ボリューム上のグローバル・エクステント（ファイルをアロケート出来る様なディスク・ボリューム上の領域）に格納される．JCL 部分を含むファイルは JCFS (ソース形式ジョブ制御ファイル) と呼ばれる．次いで IR は JCFS 内の JCL 文をシステムがジョブの実行に都合のよい内部形式に翻訳，変換して同じ SYSIN 上の JCFI (内部形式ジョブ制御ファイル) に格納する．これは JCL シンタックス・チェック，JCL マクロ展開や JCL カタログ等の処理を含めてトランスレータと呼ばれる IR 内のプログラムが入力ストリームの読込みと並行して行う．最後に翻訳，変換が済むと IR はジョブの選択を行うスケジューラに通知を行う．

IR は入力ストリームの終りを示すレコードを検出すると待状態に入るかまたは，終了する．いずれの場合も割当てられていた入力デバイスは解放される．

IR の扱う JCL 文についてはジョブ制御言語 (JCL) の項を参照のこと．

3.2. ジョブの出力

システムへのジョブ入力が一旦 SYSIN グローバル・エクステントを用いて処理されると同様に，実行中のジョブ出力データやシステム・レポートといったデリバリ情報は SYSOUT と呼ばれるグローバル・エクステント上のファイルを利用して行われる．もち論ユーザがジョブの出力装置としてプリンタやカードパンチ等を直接利用することも出来るがマルチ・プログラミングの下でのシステム全体の処理能力や経済性の観点から，一般には一括して出力を制御するスプーリング処理が普通である．

出力処理：

アウトプット・ライタ (OW) は IR と同様に割当てられた最終出力装置ごとに存在する．出力装置としては次がある．

- プリンタ
- カード パンチ
- 端末装置
- その他

出力装置及び出力ファイルに与えられるデリバリ・クラス名がシステム生成時に定義されており，この対応はシステム生成または運用時の OW への OCL に依って成される．OW は SYSOUT 上に貯えられているジョブの出力データやシステム・レポートをジョブ終了時，ステップ終了時またはオペレータの指示に従って最終出力装置に出力する．

ジョブの出力情報をどの時点で出力するのかあるいは，ユーザのファイルに格納して欲しいとかは JCL で要求する．

更に OW を制御する豊富なオペレータ・コマンドも用意されている．

3.3. ジョブ・スケジューリング

スケジューリング・パラメータ：

スケジューリング要求は所与の SGL, OCL, JCL を通して与えられるスケジューリング・パラメータに依って行われる．スケジューリング・パラメータの内二つの重要な概念，クラスとプライオリティについて説明し他は概略を表記する．

(1) ジョブ・クラス

ACOS-4 ではメモリ，デバイス，ファイル等の資源についての操作をジョブ・スケジューリングの段階では適用しない．資源については予めクラスの概念を導入してそれに依るジョブ・スケジューリングを行う．

クラスは具体的に A～P の文字で表現される．これは個々のジョブの性格と設置システムが持っている目標とを組合せて，どの様にシステムを運用するかを表わすユーザの概念である．即ちクラスは設置システムの使用に関するジョブの要求する仕事のタイプを示し例えば，入出力処理が頻繁であるとか，内部処理中心である大量の主メモリを要求するといったタイプに依る分類，あるいは遠隔処理に於ける入力デバイスに依存する分類づけの様に，ユーザはジョブ・クラスに設置システム固有の意味を与えることができる．

(2) ジョブ・プライオリティ

ジョブはクラスによる分類が成されるとジョブ・スケジュールの優先度としてのプライオリティが考慮さ

表-2 SCHEDULING PARAMETERS

スケジューリングの時機	スケジューリングパラメータ	説明	要求
システム生成時	スケジューリング方式	FIFO クラス，プライオリティによるスケジューリングアルゴリズム 実行開始無限待ちを回避する定期的なプライオリティ更新処理	SGL SUL
	投入最大ジョブ数		
	JOB文，RUN文に現れる各種パラメータ既定値	ジョブ処理制限時間 ジョブ処理経過時間 出力穿孔カード制限枚数 出力プリント制限ライン数	
	入力デバイス	入力デバイスはスケジューリングアルゴリズムと関係する	
	ジョブクラス	使用するクラス名 クラス毎の投入可能な最大ジョブ数 クラス内ジョブが実行されるときの CPU ディスパッチング プライオリティの範囲	
	プリエンプティブ ジョブ	対象ジョブをアボートする 対象ジョブをブロックする	
ジョブ投入時	JOB文，RUN文のパラメータ	クラス プライオリティ CPTIME LAPSED TIME CARD LINE HOLD EVERY TIME	JOB 文 RUN文
システム運用時	オペレータコマンド	ジョブ入力の禁止，再開 スケジューリング方式の変更 ジョブ情報の表示 ジョブのブロック，アボート HOLD	OCL
	システム負荷	スラッシング発生，解消の通知及びそれに伴うジョブのアボート，ブロック，投入の禁止，再開	システムプログラム
	タイマ	TIME, EVERY	
	システム停止		
	ISL	ISL (初期システム ロード) 時 TIME ジョブをチェックする	

れる．プライオリティはジョブの他のジョブと比較しての緊急度を表わすユーザの概念である．

ジョブ・プライオリティは0～15の数値で示し0が最も優先度が高い．

クラスやプライオリティは CPU やデバイス，ファイル，メモリといった資源を競合する際の内部プライオリティを決定する要素となっている．

（3） スケジューリング・パラメータ

スケジューラの処理：

スケジューリング・パラメータ，システム負荷に従ってシステムへのジョブの受入れが可能かどうかの許可，受入れられ読込まれた後は多数のジョブの中からどれが開始に最適なジョブかを選択し，実行を要求するのはスケジューラと呼ばれるプログラムが行う．

IR を通して与えられるジョブは JOB 文，RUN 文に示される要求に従って，あるものはイベント待行列 (WAIT-QUEUE) に登録される．イベントには開始時間 (TIME)，周期的ジョブの開始 (EVRY) やオペレータの指示 (HOLD) がある．他方何も制限が無いとかイベントが発生して制限が解除されたジョブは開始待行列 (PROPOSED-QUEUE) に登録される．PROPOSED-QUEUE 内のジョブはスケジューリング方式に依って FIFO (First-in First-out) またはクラスごとにプライオリティ順に並べられており，システム負荷チェック及び選択アルゴリズムによって実行に最適なジョブを選択するスケジューラは実行ジョブ制御表にその旨を記述してジョブの実行を要求する．

3.4. ジョブの開始，実行，終了

実行ジョブ制御表に登録されたジョブはコマンド・インタプリタと呼ばれるプログラムが内部形式のジョブ記述である JCFI 中の JCL を読出し，解析し，実行する．コマンド・インタプリタは JOB 文に対してはジョブの実行に必要な情報構造を作成し，ジョブ・レベルの初期化（ジョブイニシェーション）を，STEP 文に対しはステップ・レベルの初期化を行う．各種資源要求を示す ASSIGN, ALLOCATE, DEVICE 文が来ると情報構造内に資源要求表を作成する．

ENDSTEP 文が読まれるとステップの開始に必要な資源割当てを行い STEP 文で示されるプログラムを仮想記憶上にロードし実行させる．（ステップ・イニシエーション）

この様に ACOS-4 ではジョブはステップ単位に処理される．プログラムが終了したり，アボートされるとステップは資源を解放し（一部の資源はステップ間で引渡される）コマンド・インタプリタに制御を戻す（ステップ・ターミネーション）．JCFI 中の JCL はこの様に次々と実行される．

ENDJOB 文が現れるとジョブの終了（ジョブ・ターミネーション）となり，スケジューラに通知される．

資源の割当と解放

資源管理の方式：

（1） 静的管理方式

ジョブがステップ単位で実行されるとき，ステップが実行開始に入る以前に必要な資源が割当てられる．

要求はステップ・エンクロージャ内の STEP, ASSIGN, ALLOCATE 文等に指定され，コマンド・インタプリタが ENDSTEP 文の処理としてステップ開始時点（ステップ・イニシェーション）でメモリ，デバイス，ファイル*，スペース等資源レベル毎に割当要求を行う．割当てられた資源の解放はステップの終了時（ステップ・ターミネーション）に行われる．

（2） 動的管理方式

ジョブやステップの開始以前に予め使用を宣言しておいたり，一部の割当てのみを行って実行に入り，資源が実際に使用されるときに要求したり，使用後直ちに解放する方法である．動的要求はシステム資源の有効利用という目的に沿うものではあるが，要求時にフリーな資源が無いときのデッドロックの問題，静的方式と比しての経済性や能率といった見方からシステム・ユーザ，一般ユーザ更に後者ではジョブ処理形態によって資源待合せサービスは異る．例えばバッチ処理に於ける一般ユーザは待合せを受けることなく，資源が無いときはその旨の通知をうけ取るだけかまたはアボートされる．

（3） 資源割当てのレベル

静的方式では資源割当てはメモリ，デバイス，ファイル，ボリューム・スペースといった資源レベルごとに行われる．各レベル内では ALL OR NOTHING（あるレベルで充分な資源が無いときは以前のレベルまでの資源は保持し，そのレベルで一部割当てられたものは解放されてレベル行列に登録される）を原則として，レベル間では進行方向を一定にする割当処理が行われる．

アロケーション・モード：

資源割当ては JCL 文やプログラム中のマクロ命令を通して要求される．このときユーザは資源管理に対する制御情報としてアロケーション・モードを指定出来る．

（1） ノーマル・モード

ユーザは内部プライオリティに従って資源割当ての制御を受ける．静的方式では資源不足が起るとレベル待行列への登録は内部プライオリティに従う．

（2） アージェント・モード

静的方式では資源不足が起っても現レベルで得た資源は解放されず予約したままにしておく．予約された資源がノーマル・モードのユーザに割当てられることはない．

（3） プリエンプティブ・モード

割当要求はシステムが容認する限り充たされる．このため既にあるレベルで資源を確保しているステップや実行中のステップから資源を強制的に奪うこともある．但しプリエンプティブ・モードのユーザが競合するときは強制出来ないので，このときはアージェント・モードの待行列に登録される．

メモリ，デバイス，ファイル，スペース：

（1） メモリ

仮想記憶は主メモリ，ディスク上の二次記憶であるバッキング・ストア (BS) から構成される．BS はステップの開始時に割当てられ終了時に解放される．プログラムの実行中バッファやデータのセグメントが大きくなり新たな BS を要求することもある．

主メモリは STEP 文で記されるプログラムが実行するのに最適な主メモリの容量を示すワーキング・セットが割当てられる．プログラムの実行中は仮想記憶管理が主メモリの必要量を制御するので実際にワーキング・セットを越えたり，以下であるときもある．しかしながらプログラムがそのことで影響をうけたり，プログラマがこれを意識する必要は全く無い．

（2） デバイス

デバイスは論理的，物理的に同じ性格を持つデバイス・クラスの概念を用いて管理される．マルチ・ジョブ処理制御の下で，普通個々のジョブが特定のデバイスを要求することはなく，クラス名で要求する．指定のクラス内でフリーなデバイスがあればそれを割当てる．

（3） ファイル

ファイルはステップにもジョブにも割当てることが出来る．ファイルの割当てはプログラム内に記述される内部ファイル名とボリューム上の外部ファイル名を結びつけ，このためにデータ管理のカタログ機能や装置管理の AVR（自動ボリューム認識機能）が働き，ユーザのファイルへのアクセス権限，排他，共有使用等をチェックする．

（4） ボリューム・スペース

ディスク・スペースはファイルを作成したり，拡張したりするために要求される．スペースの割当，解放は静的に行っても動的に行ってもよい

4. ジョブ制御言語 (JCL)

文法：

SCL 文従って JCL 文は OS が種々のユーザに与え

* ファイルについてはジョブ単位に割当，解放を行うことも出来る．

る基本機能であるマクロ命令（ACOS-4 ではプリミティブともいう）と共に同一のシンタックス，パラメータ，キーワード及びセマンティクスに従っている．

（1） 修飾規則

部分的な識別名で指定された外部ファイルを完全名にする場合に適用される．修飾は PREFIX 文やアカウント名により行われる．

（2） 文脈規則

二つの文脈規則がある．

i 適用範囲（エンクロージャ）

適用範囲と呼ばれる文脈規則……JCLプログラムを記述するときに個々の JCL 文が予め定められている構造に於いてしか現れてはいけない……がある．この特定の構造のことを適用範囲という．適用範囲はいわば COBOL や PL/I プログラムのブロックに対応するもので JCL プログラムの構成を規定する．

四つの型の適用範囲及び相互の関係は次の通りである．

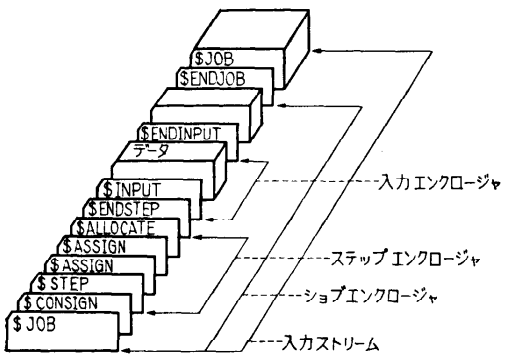

Fig. 1 INPUT STREAM, JOB DESCRIPTION & ENCLOSURE

- ジョブ・エンクロージャ　JOB 文から END-JOB文または，次の JOB 文の直前まで
- ステップ・エンクロージャー　STEP 文から ENDSTEP 文まで
- 入力エンクロージャ　INPUT 文から ENDINPUT 文または，特定の ENDINPUT 文まで
- 定義エンクロージャ　DEF 文から END 文まで

適用範囲と類似した有効範囲という言葉がある．有効範囲は個々の JCL 文まで示される機能がどこまで有効であるかを規定する．有効範囲は適用範囲とは限らず同じ機能を持つ JCL 文や有効限界を示す JCL 文等によって規定される．

表-3 RELATIONS AMONG ENCLOSURES

内部エンクロージャ ＼ 外部エンクロージャ	ジョブ	ステップ	入力	定義
ジョブ	×	×	△	×
ステップ	○	×	△	○
入力	○	×	△	×
定義	○	○	△	×

○ 入れ子関係を許す
× 入れ子関係を許さない
△ 入れ子関係を許すが解釈されない

ii ある種の JCL 文はその出現順序が決められている．例えば ASSIGN 文，ALLOCATE 文，DEFINE 文等がそうである．

（3） JCL の記法

JCL のシンタックスは GMP のシンタックスであり，ステートメント名は“$”記号で始まり“;”記号で終了する．パラメータは“,”記号で分離される．条件付ブランチを可能にするためラベルを付するときは“ラベル:”がステートメント名に先行する．

```
$ステートメント名　パラメータ，パラメータ，…；

$ラベル：ステートメント名，パラメータ，…；
```

Fig. 2 GENERAL FORMAT OF JCL STATEMENT

ラベル，ステートメント名には単純識別名，修飾名が記される．パラメータは二進，十六進，十進数値，自明値，演算式，リテラルやマクロ・コールが定位置パラメータやキーワード・パラメータまたは，パラメータ・リスト形式で記述される．

JCL 文の例：

ACOS-4 の JCL 文は大きくは基本文，ユーティリティ，スーパー・マクロの三つに分類される．以下ではこれらの代表的なものを列記し，二，三の JCL 標準記述を挙げる．

（1） ACOS-4 標準 JCL（**表—4**）

（2） JCL 記述形式の例 (805 頁参照)

（3） FORTRAN COMPILE, LINK, GO の例

プログラムの実行

これまではジョブ制御の観点からジョブがシステム

表-4 LIST OF ACOS-4 STANDARD JCL

	タイプ	JCL 文	適用範囲	説明
基本文	非実行文	COMMENT	ジョブ ステップ	ジョブ記述の中に注釈を挿入する.
		CONTINUE	同上	ラベルを付けるのが許されていない文, 例へば ENDJOB 文の前に置いて間接的なラベル参照を可能にする. 複数ラベルを付けるときにも利用される.
	処理パラメータセット文	PREFIX	同上	他の JCL 文に現れる外部ファイル名にプレフイックスを付けて完全名にする.
		INSERT	同上	カタログファイルの内容を現在の JCL 順列に挿入する.
		DEF	同上	JCL マクロ定義を行い以後で JCL マクロの使用を可能にする.
		END	同上	DEF 文以後で示すマクロ宣言記述の終了を示す.
		OUTPUT	ジョブ	SYSOUT 内のジョブ出力の時期, 出力先, 出力特性を規定する.
		RUNPAR	ジョブ	ジョブ実行時に修正出来る ランタイム パラメータを定義する.
	入力文	INPUT	ジョブ	入力エンクロージャを開き, エンクロージャ内のデータをファイルとして参照出来る様にする.
		ENDINPUT	ジョブ	入力エンクロージャを閉じる.
	ジョブ文	JOB	ジョブ	ジョブ エンクロージャを開き, ジョブ識別名, クラス, プライオリティ等でジョブを定義する.
		ENDJOB	ジョブ	ジョブの終りを示す.
		CONSIGN	ジョブ	ジョブ レベルでファイルを確保し共有/排他使用を宣言する.
		DECONSIGN	ジョブ	CONSIGN の効果をキャンセルする.
	ステップ文	STEP	ジョブ	ステップを定義する.
		ENDSTEP	ステップ	ステップの実行を要求する.
		ASSIGN	ステップ	内部ファイル名と外部ファイル名を結びつけると共にステップがこのファイルを使用可能にする.
		DEVICE	ステップ	内部ファイル名と装置又は装置クラスを結びつける.
		DEFINE	ステップ	ファイルの定義, 処理に関するパラメータを与える.
		ALLOCATE	ステップ	ファイルに割当てられるスペース量割当方法, ボリューム指定を宣言しスペース確保を行う.
	制御変数文	DCV	ジョブ	制御変数はステップの通信に用いる制御変数を定義し, タイプ及び初期値を与える.
		LET	ジョブ	制御変数に値を代入する.
	流れ制御文	JUMP	ジョブ ステップ	指定のラベルを持つ文までスキップする.
		WHEN	同上	条件付ジャンプを行う.
		RSTNORMAL	ジョブ	JCL 処理をバグモードから処理モードに戻す.
		EXECUTE	ジョブ	指定されるファイル内の JCL 文の実行に入る.
		RUN	ジョブ	指定されるファイル内の JCL 文の実行を要求する. EXECUTE 文と同様にランタイムパラメータの指定が出来る.
	オペレータ交信文	CRCONSOLE	ジョブ	ジョブコンソールを定義する.
		DL CONSOLE	ジョブ	ジョブコンソールを解消する.
		SENDO	ジョブ	オペレータにメッセージを送る.
		SENDOR	ジョブ	オペレータにメッセージを送り, オペレータよりメッセージを受取る.
	カタログ文		ジョブ	カタログに登録する, CATFILE, CATGEXT, CATVOL, CATDIR, CATSYN 文, 登録済のものを変更する MODIFY, SHIFT, DELET, ACLDEF 文, 更には常駐カタログで他システムのカタログを利用出来るためのATTACH DETACH, DECAT 文がある.
	カタログファイル文	CATALOG GENERATION	ジョブ	既に登録されているファイルのある世代のスペース及びボリューム識別名をカタログに登録する.

タイプ		JCL 文	適用範囲	説明
ユーティリティ	ファイル文		ジョブ	スペース確保又は拡張を行う PREALLOCATE 文，ファイルを消去する．DEALLOCATE 文，コピー，退避，回復，比較，プリント，名前の変更，パッチを行う DUPLICATE-FILE,SAVE-FILE,RESTORE-FILE, COMPARE-FILE, PRINT-FILE, RENAME, PATCH-FILE がある．
	ボリューム文		ジョブ	ボリュームに対する各種ユーティリティ PREPARE, PRINT-VOLUME, COMPARE-VOLUME, SAVE-VOLUME, RESTORE-VOLUME, DUPLICATE-VOLUME, RELOCATE-VOLUME, LIST-CONTENTS がある．
	カタログ文	LIST-CATALOG	ジョブ	カタログの内容を報告する．
ロースマークパ			ジョブ	言語プログラム翻訳，リンクを行う COB, FORT, RPG, LINK 文やコンパイル＆リンクを行う COBLK, FORTLK, RPGLK 等がある．

JCL 記述形式の例

```
$JOB user-name [,ID=identifier
                [,ACNT=identifier]] [,PRIORITY=value
                                    [,CLASS=character]]
   [,REPORT=characters]
   [,PROCESSOR-TIME={NLIM|value}]
   [,LAPSED-TIME ={NLIM|value}]
   [,CARD=value]
   [,LINE=value]:
```

```
$STEP member-name [,REPORT=characters]
   [,PROCESSOR-TIME={NLIN|value}]
   [,LAPSED-TIME ={NLIM|value}]
   [,CARD=value]
   [,LINE=value]
   [,MEMORY=value][,{SOCIABLE|UNSOCIABLE}] [,{DUMP|NO-DUMP}]
   [,{CKPT|NO-CKPT}] [,{REPEAT|NO-REPEAT}] [,{DEBUG|NO-DEBUG}]
   [,OPTIONS=(character strings)];
```

```
$FORT source-file, object-file, COMPOPTS;
```

```
$FORTLK source-file, object-file,{CMD|CMDFILE}
        ,COMPOPTS, LINKOPTS;
```

```
$LINK cu-file, lm-file,{CMD|CMDFILE}, LINKOPTS;
```

```
STANDARD FORMAT OF
JOB, STEP, FORT, FORTLK, LINK JCL STATEMENTS
```

FORTRAN COMPILE, LINK, GO の例

```
$JOB NECTOKA, ID=MYID, ACNT=MYACNT;
$FORLK IN, MYPROG/LMLIB, CMD: USERINFO;
$INPUT FORTRAN;
(SOURCE CARARDS)
$ENDINPUT;
$PREALLOCATE FILE=EFN1, SZ=31, FORG=PRTQD,
                LRECSZ=1024, PRCSZ=1024 MEDID
               =(DVC=MS/DSK300, VOLID=D005);
$PREALLOCATE FILE=EFN2, SZ=11, FORG=PRTQD,
                LRECSZ=1024, PRECSZ=1024, MEDID
               =(DVC=MS/DSK300, VOLID=D005);
$STEP MYPROG/LMLIB, CPTIME=20;
$ASSIGN IFN EFN1;
$ASSIGN IFN2, EFN2;
$ENDSTEP;
$ENDJOB;
A SIMPLE EXAMPLE OF COMPILE, LINK,
PREALLOCATE &GO.
```

に読込まれてから終了して最終出力に至るまで，どの様に制御されるかについて述べて来た．ここではジョブのプログラムが準備されてから仮想記憶上にロードされ，実行に入る過程について説明する．

コンパイル：(プログラムの翻訳)

ユーザは COBOL や FORTRAN や PL/I といったコンパイラ言語でソース・プログラムを記述し，ファイルまたはソース・プログラム・ライブラリに格納する．コンパイラはファイルやライブラリ内のソース・プログラムを入力として翻訳を行う．

ソース・プログラム中にコンパイラ言語と共に記述される各種のマクロ・ステートメントはコンパイラがコールする GMP (汎用マクロ・プロセッサ) に依って処理される．GMP はマクロ・プロセッサ言語で書かれた (どのマクロ・ライブラリをサーチするのか，対応するマクロ展開といった要求を示す) マクロ・ステートメントを実行する．コンパイラの処理結果はコンパイル・ユニット (CU) と呼ばれ，CU ライブラリに出力される．CU はプログラム情報をセグメント単位で有している．セグメンテーションは各コンパイラのアルゴリズムとソース言語ステートメントによって決まる．例えば COBOL の場合 CU は少くとも DATA DIVISION と PROCEDURE DIVISION に対応する二つのセグメントから成る．これは実行権を持つコードとアクセス権が異るセグメントにデータを分離するセグメンテーションに基く仮想記憶の一般的原則に従っている．

CU には他の CU を / から参照する / されるデータやプロシージャの外部参照 (SYMREF) / 外部定義 (SYDEF) の様な情報も入っている．

(2) リンク

コンパイラの出力である CU を纏めて一つのプログラムを作成するのがリンカである．リンカは沢山の CU が持つ多数のセグメントをハードウェアが認識出

来るアドレス空間として構成する．アドレス空間の生成とはユーザが指定するタスク構成に従って

・タスクの物理的実体であるプロセスとの対応づけ
・タスクの下で実行される CU を結びつけ，CU 内外の参照を解決してセグメント・アドレスを割付けることである．

リンカの出力は実行可能な形式をしており，ロードモジュール (LM) と呼ばれ LM ライブラリに格納される．

(3) プログラムの実行

ジョブはステップ単位に実行され，具体的に STEP 文で示されるプログラムが LM ライブラリから仮想記憶上にロードされステップの主タスクを開始させる (ハードウェアの START 命令) ことで実行される．

(注) リンクの項で述べた様にプログラムは一つ以上のタスクで構成され，対応するプロセスは関連のあるプロセス群としてステップが実行される．ACOS-4 の CPU はステップに対応するプロセス群及び，タスクに対応するプロセス群内のプロセスを夫々 J 番号*と P 番号というハードウェア上の実体として認識し，従来ソフトウェアが行っていたプロセス・デスパッチングやプロセス間の同期，通信といった機能はハードウェア／ファームウェアが行っている．

従ってアドレスは (J 番号，P 番号，セグメント・アドレス) の組で示され，ハードウェアがセグメント情報をプロセス群のプロセスごと従ってステップ，タスクごとにその参照，アクセス時の保護を行っている．

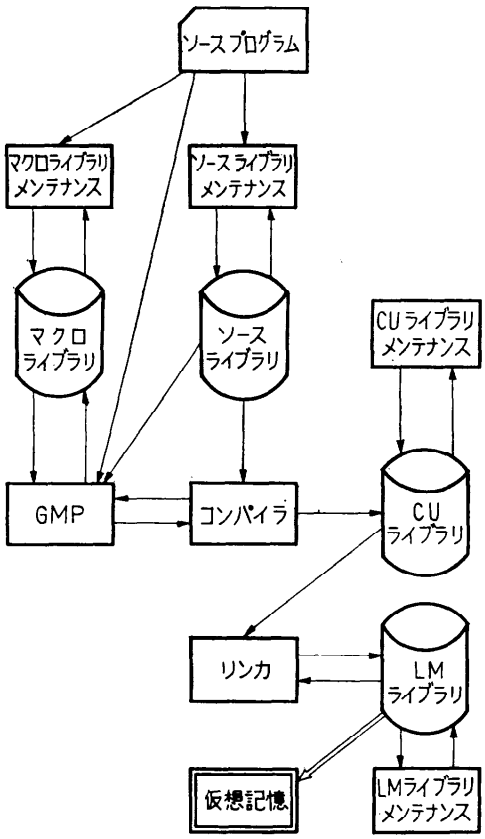

Fig. 3 PROGRAM PREPARATION & EXECUTION

* ここでいう J 番号は資源としてステップの開始，終了ごとに割当，解放が行われる．

(昭和 49 年 5 月 10 日受付)