EC77－44

網向きジョブ管理システム

NETWORK ORIENTED JOB MANAGEMENT SYSTEM

田中英彦　元岡達　山内長承　黒羽法男　堀口真志　小田正美+
Hidehiko Tanaka　Tohru Motooka　Nagatugu Yamanouchi　Norio Kurobane　Masashi Horiguchi　Masami Oda
東京大学工学部　+富士通
Faculty of Engineering, University of Tokyo　Fujitsu Limited

(1) はじめに

計算機網用オペレーティングシステム(NOS: Network Operating System)として，研究開発したTECNETシステムに，今回報告する網向きジョブ管理システムNETJM[1]を実装した。

NOSの設計目標およびNOS上に実装したNETJMの位置について簡単に触れておくことにする。NOSの設計目標は

- 全てのホストが平等な権利を持つ分散管理方式を採用し、特定のホストの障害が網上の他のホストに与える影響を最小にする。
- 計算機網を構成する各ホストにオペレーティングシステムOSを置き，それぞれのOSが協力し合って，計算機網全体を1つのシステムとして管理するオペレーティングシステムをユーザに提供する。

以上である。前者は、集中管理方式と対となる管理方式であり、特に説明の必要がない点である。後者は、単独計算機用オペレーティングシステムと同様に、計算機網全体を1つのシステムと見なすオペレーティングシステムを提供することである。

よって、計算機網全体を1つのシステムとして管理するNOSの構造は，単独計算機用OSで提案されている階層構造[2]や核によるアプローチ[3]を採用した。また，NOSの拡張性を考慮して，システムの各部分をモジュール単位に分割して，モジュール間のインタフェースが明確になるようにした。NOSでは，モジュールとして，プロセスを考えた。

さて，NOSの構造は、

- プロセス群とプロセスが動くことを可能にする核からなる。
- プロセス群は
  - システムプロセス群
    - 一般のプロセスに仮想計算機を提供する為の核の補助をするプロセス
    - 入出力制御プロセス
    - 計算機網ファイルシステムのプロセス群
    - 網向きジョブ管理システムのプロセス群
  - ユーザプロセス群

から成り立っている。

これを図示したものが図1である。

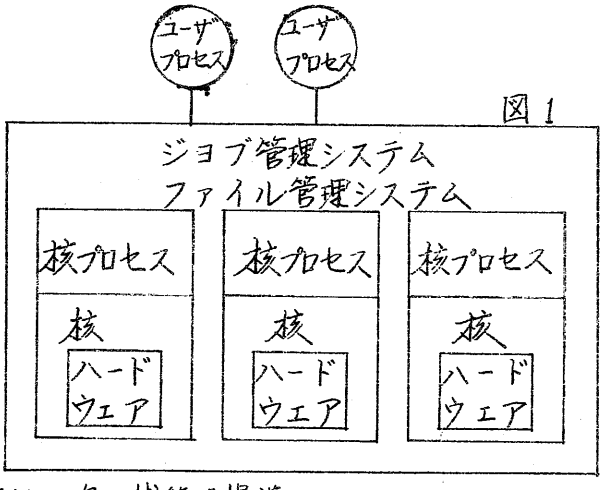


図1

核は、次の機能を提供している。

- マルチプログラミング機構
  複数のプロセスがハードウェアを共用するためのメカニズムである。

・プロセス管理
プロセスを管理する為の機能であり、CREAT（プロセスの生成），START（実行開始），STOP（実行停止），REMOVE（プロセスの抹消）の4種類のプリミティブを用意してある。
・プロセス間通信機構(4)
複数の網上のプロセスが互いにインタラクションしながら，1つの仕事をする時，プロセス間の同期とデータの交換を行なう機能を提供する。ここで提供するSEND/RECVプリミティブは，リモートホスト上のプロセスに対する通信（リモート通信）とローカルホスト上のプロセスに対する通信（ローカル通信）の手続きは，利用者にとって全く同じである。よって，プロセス間通信に関しては，網を意識する必要が無くなった。

また，プロセス群で重要な機能は計算機網ファイルシステムである。

・計算機網ファイルシステム機構(6)
ファイルシステムは，網上のすべてのファイルを同じ手続きでアクセスできる"仮想ファイル"とした為に，入出力装置の種類やファイル形式の違いにより，別々の利用手続きを覚える必要がなくなった。よってプログラムの作成および保守が容易である。また，プロセス間通信と同様に，網上に散在するファイルをアクセスする手続きは，リモートホストにあるファイルも，ローカルホストにあるファイルも全く同じである為に，網を意識する必要がない。
このファイルシステムに関しては(5)で説明する。

以上，説明したように，NETJMを作成するにあたり，NOSでは
・マルチプログラミング機構
・プロセス管理
が開発されており，かつ網上のいかなるプロセスやファイルとも通信，アクセスできる
・プロセス間通信機構
・計算機網ファイルシステム機構
が具備されていた。

網向きジョブ管理システムNETJMは，以上に述べた機構を仮想的なハードウェアと見なして，実装したものである。

(2) 網向きジョブ管理システムの方式

ユーザが計算機システムに仕事をさせる時，ユーザを助けて実行の準備，監視，後始末をするジョブ管理システムは，計算機網に対しても必要である。
計算機網を1つのシステムと見なして，管理，運用を目指しているNOSの設計目標に準じて網向きジョブ管理システムNETJMも，計算機網を1つのシステムとして管理する方式とした。よって，NETJMは，
・各ホストのハードウェアが同じなら，ユーザの作ったプログラムをどこのホストでもプロセス化して実行できる。
・更に，それらのプログラムは，ホスト間にまたがって置かれる場合も許される。
・どのホストからも，ユーザは同じ手続きでジョブを制御できる。
等の機能を持っている。これらの機能はホストという言葉を取り去れば，単独計算機システムと同じである。

NETJMは，網向きジョブ管理システムであるので，単独計算機システムのジョブ管理システムと異なった制御方式を検討する必要がある。

さて，NETJMが管理する内容は，ジョブ制御文の受付け，解釈をし，そのジョブ制御文に従って，
・ジョブの順序制御
・資源の確保，解放の管理
・プログラムのローディングと開始
・ジョブの後始末
等である。これ等を，網向きに管理するのにふさわしい方式を検討することにする。

ジョブの順序制御

1つのジョブに関するジョブ制御文は，網上の1つの端末から入力される。よって，ジョブ制御文に従ってジョブの順序制御をするプログラムは，入力装置の所属するホストに置くのが最も自然であり，又ホストの障害に対しても都合

が良い。

図2

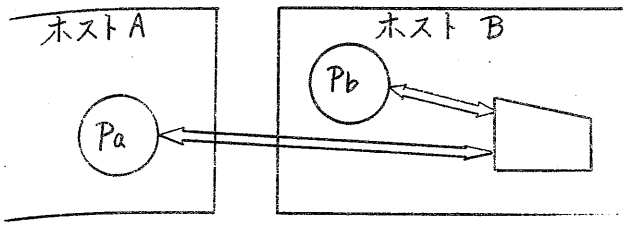
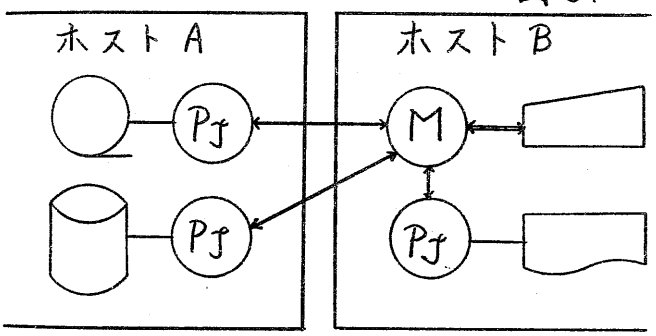


図2で言えば，プログラムPbを，ジョブの順序制御をするプログラムとする。

資源の確保，解放の管理

ファイルシステムは，資源の確保・返却を行なう管理者（プロセス）から，資源ごとに置いた管理プログラムに要求を出す方式である。
ところが，リモート通信とローカル通信の手続きが同じであるプロセス間通信がシステムの核にあるので，資源要求を出すプログラムを置く位置は，特に制約はない。
むしろ，図3のように，資源の確保，返却の要求は，ジョブ制御文と密接に結びついているので，ジョブ制御文の入力端末と同じホストに置く方が良い。

図3.

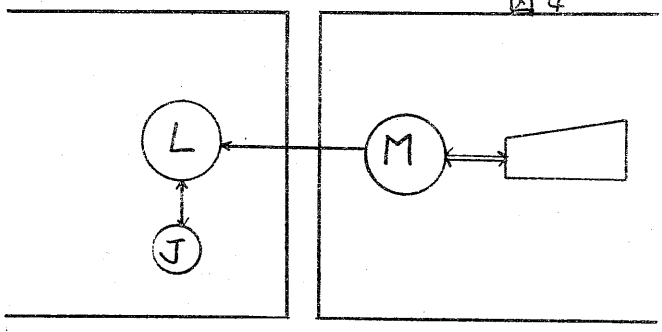


M：資源の確保・返却を管理するプロセス
Pf：資源ごとに置いた管理プロセス

プログラムのローディングと開始

ジョブ制御文に従って，プログラムの登録要求を出すと，プログラムのローディングとローディングしたプログラムをシステムに登録するプロセスがあるものと仮定する。
ローディングしたプログラムをプロセス化し，実行を始めさせるには，そのホストの核に対してプリミティブ（CREAT, START）を出す必要があるが，プリミティブの発信は，他ホストからできないので，そのホストに，ローディングと，プログラムの開始をサービスするプロセスを置いた方が良い。

図4



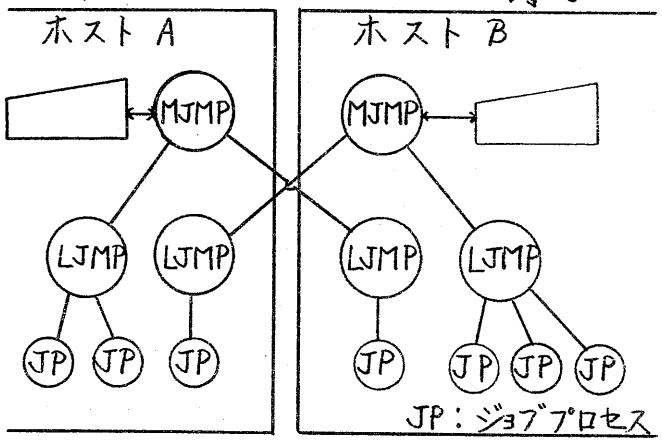
M：ジョブ制御文に従って，プログラムの登録を指示するプロセス
L：Mの指示に従ってプログラムの登録，管理をするプロセス
J：ジョブ制御文の指示により，Lが発生したジョブプロセス

ジョブの後始末（課金など）

ジョブの後始末は，各ホストでまとめた結果を，1箇所に集計する作業になる。各ホストでまとめるのはサービスプロセス（図4のL）が核の機能を用いて行ない，全体の集計はジョブを開始させたプロセスで行なうのが良い。

以上をまとめると，NETJMの方式は図5のようになる。

図5



網ジョブ全体を管理するMJMP（Master Job Manager Process）―ジョブ制御文を解釈して，資源管理プロセスに，ジョブに必要な資源の確保要求を出し，ジョブプロセスの実行を伴なう

各ホストに置いたサービスプロセス(LJMP)にプログラムのローディング，実行開始，監視，終了，終了情報の報告などを依頼し最後に事後処理をして，ジョブを終了させるプロセス—と，各ホストにばらまかれたサービスプロセスLJMP (Local Job Manager Process)から成る。

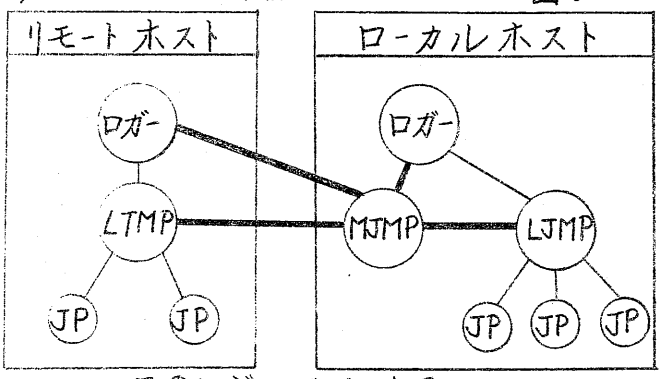
MJMPはジョブの実行をすべて監督するプロセスとして性格付けられ，その意味では集中制御であるが，実際に操作を行なうのは，サービスプロセスLJMPであり，また，ジョブ毎にジョブ入力されたホストにMJMPが作られることから，分散制御の特長である安全性を，持つことになる。

実際に実装したNETJMのMJMP，LJMPは非常駐プログラムなので，端末からログインされたとき，MJMPを起動するLOGGERを各ホストに備え付けした。LOGGERは，MJMPの起動の外に，MJMPの要請によりLJMPの起動・抹消も行なうし，MJMPの終了処理も行なう。

(3) ジョブ制御システムプロトコル(7)

網上のジョブプロセスを管理する為に，MJMPとLJMPの間，また，リモートホストのLJMPの起動と抹消を依頼する為に，ロガーとMJMPとの間のメッセージ授受形態を計算機網内で統一し，プロトコルとして定める必要がある。図6の太線が，ジョブ制御システムプロトコルである。

図6



JP: ジョブプロセス

このジョブ制御システムプロトコルを定めるにあたり，次のことを留意した。すなわち，網上のジョブプロセスを管理する上で，MJMPからLJMPに出すコマンドは詳細に指示できるようにしたが，リモートホスト個有な制御方式—メモリーの割当方式など—をMJMP側に持ち込まないで，リモートホスト上のLJMP側に管理をまかせるようにした。これにより，リモートホストの特殊性(ホストの機種に依存する部分)をローカルホストに持ち込む必要が無くなった為に，計算機網の拡張性に伴うソフトウェアの影響を最小にすることができた。

ジョブ制御システムプロトコルとプロセス間通信プロトコルは，図7のように階層化され，ジョブ制御システムプロトコルは，SEND/RECVプリミティブを利用して，ジョブ管理システム内の通信を行なっている。

図7

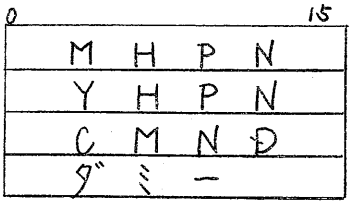
ジョブ制御システムプロトコル
通信プロトコル
ハード

NETJMに実装したジョブ制御システムプロトコルは次の通りである。

ヘッダー

ジョブ管理システム内で交換し合うメッセージには，図8で示すヘッダーをメッセージの先頭に付けることにした。

図8



MHPN：送信元のホスト名．プロセス名
YHPN：受信先のホスト名・プロセス名
CMND：コマンド
ダミー　：拡張用
ヘッダー長は8バイトである。

コマンド

ジョブ管理システム内で通信をする場合は，目的に合うコマンドを指定して相手へメッセージを送り，メッセージを受け取ったプロセスは所定の処理後，リプライ(ヘッダーのCMNDの最終ビット)を立てて送信元へ通知する。コマンドの種類を表1に示す。

表1

コマンドの略表	コマンドの送信元	コマンドの受信先	コマンドの意味
SMID	ロガー	MJMP	初期データの送信
RLLS	MJMP	ロガー	LJMPの作成を依頼
RLLP	〃	〃	LJMPの消去を依頼
RLMP	〃	〃	MJMPの消去を依頼
SLID	MJMP	LJMP	初期データの送信
RLJC	〃	〃	JP*のCREATEを依頼
RLJS	〃	〃	JPのSTARTを依頼
RLJP	〃	〃	JPのSTOPを依頼
RLJR	〃	〃	JPのREMOVEを依頼
RLJA	〃	〃	JPのABORTを依頼
JLJQ	〃	〃	JPのQUITを依頼
SMJE	LJMP	MJMP	JPの終了を通知

JP：ジョブプロセス

(4) ジョブ制御言語 (5)

計算機網全体を1つのシステムとして管理するNETJMが，ユーザに対して提供するジョブ制御言語は，次の目的を満たすものとした．

- 網上のどのホストからでも，網に属するすべての計算機が使える．
- 網上にある複数個のジョブプロセス(JP)を並列に走らせることができ，かつ，JPの開始と終了の同期がとれる．
- 網上にあるJPに割込んで，端末にいるユーザとの会話ができる．
- 網上にあるJPを強制終了できる．
- 網上にあるJPの状態や，これから開始するJPの状態を，ユーザが一目でわかる機能を提供する．
- 網上にあるファイルをファイル名だけで指定できる．

NETJMに実装した，ジョブ制御言語の文法は次の通りである．

JOB文

・システムにジョブの生成を指示する．

<JOB文> ::= $JOB

FD文

網上のファイルを定義する．

<FD文> ::= $FD␣<ファイル定義>

<ファイル定義> ::= {<論理ファイル名1>=}{<論理ファイル名2>|<物理ファイル名>}

<論理ファイル名1> ::= 物理ファイル名に対応する識別記号であり，6文字の英数字からなる．

<論理ファイル名2> ::= システムに登録済のファイルを示す論理ファイル名であり，6文字の英数字からなる．これを指定すると，システムは対応する物理ファイル名に変換して，論理ファイル名1で指定した(論理ファイル名1省略時は論理ファイル名2)識別記号を新しい論理ファイル名とする．

<物理ファイル名> ::= <ホスト名>・<デバイス名>[#<枝種通番>]・<ファイル名1>・<ファイル名2>[・<ID1>[・<ID2>[・<属性>]]]

物理ファイル名に関しては，(5)で述べることにする．

LM文

ファイルのカタログ，アンカタログを指示する．

<LM文> ::= $LM␣{␣<ファイル定義>|*<論理ファイル名1>}

␣<ファイル定義>；ファイルのカタログを指示する．

*<論理ファイル名1>；ファイルのアンカタログを指示

EXEC文

ジョブプロセスの起動を指示する．また，ジョブプロセスの起動順序も指定できる．

<EXEC文> ::= $EXEC␣<ファイル定義>[(<PNM>)][@<ホスト名>][<オーダ指定>]

<PNM> ::= 6文字の英数字からなるジョブプロセス名である．省略時は，論理ファイル名1を示える．

<ホスト名>::=ŌKI | PPS
省略時はŌKIとする。この指定はジョブプロセスが実行するホスト名をふえる。
<オーダ指定>::=空|␣|X{<PNM>{,<PNM>}}
空，␣——網上にあるすべてのジョブプロセスの終了時に，ここで指定したジョブプロセスを起動する。
X{<PNM>{,<PNM>}}——PNMで指定したジョブプロセスの終了後に，ここで指定したジョブプロセスを起動する。PNM省略時はすぐ実行。

QUIT文

ジョブプロセスに割込みをかける。
<QUIT文>::=$QUIT␣<PNM>{,<PNM>}

ABŌRT文

ジョブプロセスを強制終了させる。
<ABŌRT文>::=$ABŌRT␣<PNM>{,<PNM>}
ABŌRT文で強制終了したジョブプロセスの終了後に，開始するはずのジョブプロセスは，すべて強制終了する。

ALLŌCATE文

ジョブプロセスを実行するホストにサービスプロセスLJMPを起動することを指示する。ジョブプロセスを起動する前に，ALLŌCATE文を投入しておくこと。
<ALLŌCATE文>::=$ALLŌCATE␣{<パラメタ>}{@<ホスト名>}
<パラメタ>::=CŌRE=<ページ数>
| PRCS=<セグメント数>
実行ホストがŌKiTACの場合はコアサイズ（CŌRE～），FACOM PPS/の場合はセグメント数(PRCS～)を2桁の16進数で指定する。

DISP文

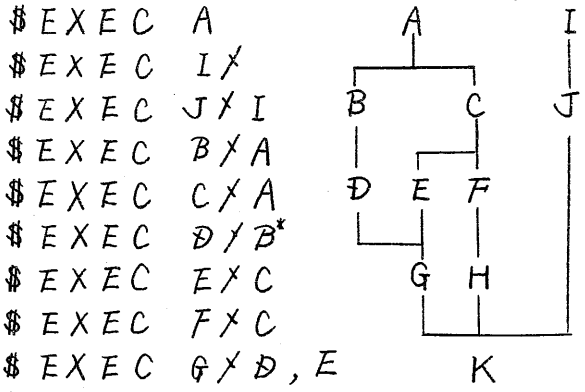
ジョブプロセスやファイルの状態を問い合わせる。
<DISP文>::=$DISP{␣{<出力指定>}}
<出力指定>::=A|L|F|J
L——カタログファイルの論理ファイル名一覧表をリストする。
F——未実行ジョブプロセスにふえられるファイルの論理ファイル名一覧表をリストする。
J——ジョブプロセスの状態をリストする。
␣，空，A——L，F，Jのすべての情報をリストする。

END文

網上のすべてのジョブプロセス終了後に，ジョブの終了を指示する。
<END文>::=$END

以上が，NETJMに実装したジョブ制御言語の文法である。図9のような順序関係にあるジョブプロセスA～KをEXEC文の<オーダ指定>でどのように指定するかを示す。

図9



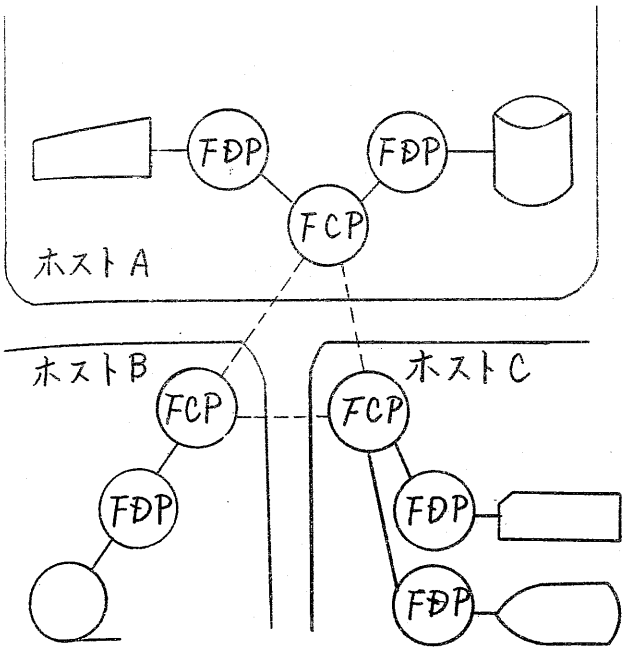
$EXEC A
$EXEC IXA
$EXEC JXI
$EXEC BXA
$EXEC CXA
$EXEC DXB*
$EXEC EXC
$EXEC FXC
$EXEC GXD,E
$EXEC HXF
$EXEC K**

*$EXEC DXB,A でも良い。
**$EXEC KXG,H,J でも良い。

〔5〕網向きファイルシステムとの関係

網向きファイルシステムは，NOSの核上に実装され，その構造は図10の通りである。

図10



網上にあるそれぞれのファイルを制御する為に，ファイルが所属するホストにFDP（File Device Process）を，それぞれのファイルごとに置いておき，各ホストには，網上の全てのFDPを管理するFCP（File Control Process）がある。

この網向ファイルシステムの方式は，〔2〕で述べた網向きジョブ管理システムの方式と同様に，それぞれのファイルを制御するFDPから見れば，分散制御方式となり，システム障害に強い構造となっている。また，各ホストには，FCPがおかれ，1つのFCPから見れば，網上にあるすべてのファイル（FDP）を管理する集中制御方式となっている。

ファイルを利用する使用者（NETJM，ジョブプロセスなど）は，ファイルシステムで定めたプロトコルに従って，ファイルにアクセスすれば，網上のすべてのファイルを同一手順で取り扱うことができる。

NETJMは，ユーザの指定に従って，特定のホスト上にプログラムをローディングし，ジョブプロセスとしてシステムに登録，実行させることができるが，プログラムのローディングを行なうのに，NETJMは網向きファイルシステムを利用している。

NETJMと同様に，ジョブプロセスがファイルを利用する場合には，ジョブ制御言語のFD文を投入すれば，ファイルシステムを利用するのに必要な情報が，FDT（ファイル定義表）に格納され，ジョブプロセスに渡される。

ファイルシステムでのファイルアクセストリーは図11のようになっている。

図11

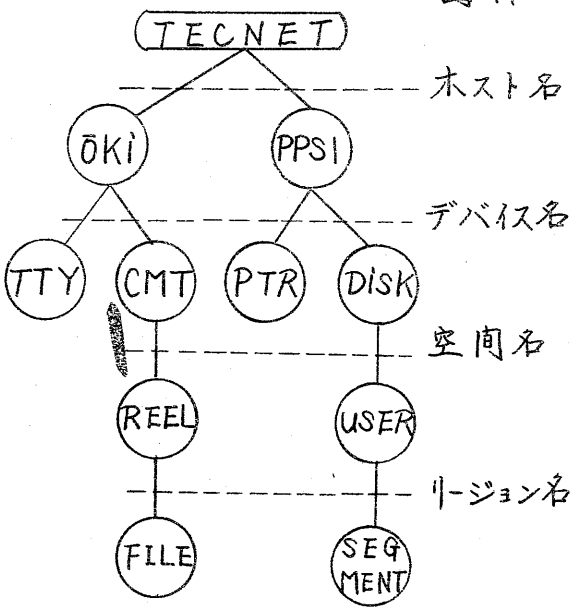


図11はTECNETシステムの1部のファイルアクセストリーである。

ジョブ制御言語のFD文でファイルを指定するのに，物理ファイル名があるが図11とどのように関係しているを示す。

〈物理ファイル名〉::=〈ホスト名〉・〈デバイス名〉〔#〈枝種通番〉〕・〈ファイル名1〉・〈ファイル名2〉〔・〈ID1〉〔・〈ID2〉〔:〈属性〉〕〕〕

〈物理ファイル〉と図11を比較すると次のようになる。

〈ホスト名〉＝図11のホスト名
〈デバイス名〉＝図11のデバイス名
〈枝種通番〉＝同一装置種類が複数台あったときに通番を与え，一意に装置を決める。
〈ファイル名1〉＝図11の空間名
〈ファイル名2〉＝図11のリージョン名
〈ID1〉＝〈ファイル名1〉を扱う為のパスワード
〈ID2〉＝〈ファイル名2〉を扱う為のパスワード
〈属性〉＝ファイルを扱うモードを指定

さて，網上のファイルにアクセスする為には，ファイルシステムで用意した次のコマンドを，使用する。

OPENコマンド

要求コード(1)
ホスト名
デバイス名・装置通番
ファイル名1 *

CLOSEコマンド

要求コード(2)
ホスト名
デバイス名・装置通番
ファイル名2 *

CREATEコマンド*

要求コード(3)
ファイル名2
ID1
ファイル属性
論理レコード長
0

READコマンド

要求コード(4)
ファイル名2 *
ID2 *
ファイル内相対レコード長
論理レコード長
0/データ格納アドバイス

WRITEコマンド

要求コード(5)
ファイル名2 *
ID2 *
ファイル内相対レコードアドレス *
論理レコード長
0/データ格納アドレス

DELETEコマンド*

要求コード(6)
ファイル名2
ID2

*; TTYやPTRのように空間名やリージョン名がないファイルは指定不可

ファイルのオープン処理やクローズ処理に関して，ユーザは，FCPにOPENコマンドやCLOSコマンドを送信してそれぞれの処理を依頼する．また，ファイルの創成（CREATEコマンド），ファイルのアクセス（READ/WRITEコマンド），ファイルの消去（DELETEコマンド）に関しては，OPENコマンドの通知情報として，FCPから通知された．FDPにそれぞれの処理を依頼する方式をとっている．

(6) 網向きジョブ管理システムの実装と機能概略

ポリプロセッサシステムFACOM-PPS1は，3台のプロセッサが主記憶を共有しているマルチプロセッサであり，各プロセッサは1ワード24ビットの書換え可能な制御記憶を持つマイクロプログラム制御方式である．NOSは，マイクロプログラムで記述されていて，(1)で述べた機能の他に，仮想記憶を実現している．

また，NETJMの実装にあたり，PLANという高級言語を使用した．またPLANの目的プログラム（1ワード16ビットの仮想計算機の機械語，TMコードと呼ばれる）を解釈実行するルーチンが備わっている．

OKITAC 4300Cは1ワード16ビットのミニコンで，使用できる言語はアセンブラーより用意していない．この為にプログラムの作成およびデバッグ期間が，FACOM-PPS1に比較して長期にわたった．

この2台の計算機に実装したNETJMのプログラムサイズを表2に示す．

表2

	FACOM-PPS1	OKITAC
LOGGER	マイクロプログラム 500語	500語
MJMP	TMコード 5000語	4000語
LJMP	TMコード 1800語	500語
LOADER	マイクロプログラム 250語	1500語
MMM	———	500語

次にNEJMを構成する各プログラムの，機能概略を示す．

LOGGER：JOB文，ALLOCATE文の投入により，MJMP，LJMPの生成をする．また，END文が投入されると実行するジョブプロセスがなくなり次第，MJMPの消去を行なう．

PPS1では，ジョブプロセスに与える，メモリの管理もしている．

LOGGERは必ず各ホストに1つある．

MJMP：端末から投入するジョブ制御言語に従って，ジョブの生成，割込み，消去などの指示を，網上に散在するLJMPに与え，ジョブプロセスの管理をする．また，ユーザに必要な情報を表示して，網上のジョブプロセスの状態を容易に把握できる機能も備えている．

MJMPはユーザ対応に1つ作られる．

LJMP：MJMPの指示に従って，ホスト内のジョブプロセスの管理をする．ジョブプロセスが存在するホストに，MJMPに対応して1つ作られる．

LOADER：LOGGER，LJMPの指

示に従って，プログラムのロードを行なう．
LOADERに対しては，ローカルなプロトコルが用意されている．

MMM：OKuTACのメモリ管理を行なうプログラムである．利用者は，LOADERであり，ローカルなプロトコルが用意されている．

尚，LOGGER，LOADER，MMMは，常駐プロセスであり，MJMP，LJMPは，非常駐プロセスとして実装した．

(7) 使用例

FACOM-PPS1にJIS7コードを入力とするコンパイラがあり，コードの異なる媒体をJuSコードに変換するプログラムをOKuTACにおくことで，負荷分散し，コンパイルが完了した時点で，実行プログラムを作成する編集プログラムが動く場合の例を，ジョブ制御言語で記述してみる．図12参照，この時のジョブ構造は，図12-1である．

図12

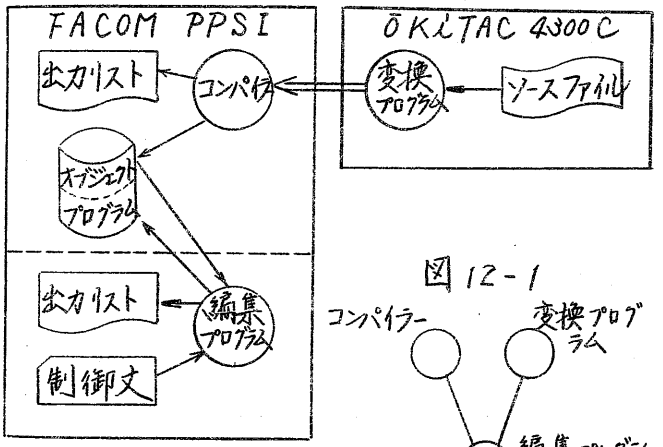


図12-1

変換プログラムは，紙テープリーダからソースファイルを読み取り，コード変換後に，コンパイラーの入力データとする．コンパイラーはディスパックにオブジェクトを掃き出すとともに，コンパイルリストをラインプリンタに出力する．コンパイルが完了した時点で，編集プログラムが起動され，制御データをカードリーダから読み取り，コンパイラーのオブジェクトから実行プログラムをディスクパックに作成する．

以上の場合のジョブ制御言語の使用例を以下に示す．ユーザ端末は，OKuTAC 4300Cにあるタイプライタを使用した．

```
$JOB ①
===> ②
$ALLOCATE PRCS=01@PPS ③
ALLOCATE EC=00 ④
===>
$ALLOCATE CORE=10@OKu ⑤
ALLOCATE EC=00
===>
$FD PTROKu ⑥
===>
$EXEC CONT ⑦
JP<CONT>STARTED EC=X00 ⑧
===>
$FD LP ⑨
===>
$LM OUT,PPS・PACK#00・A・B ⑩
===>
$FD OUT ⑪
===>
$EXEC PLAN@PPSX ⑫
JP<PLAN>STARTED EC=X00
===>
$FD iN,OUT ⑬
===>
$FD EB,PPS・PACK#00・EB・OBJ ⑭
===>
¥FD LP ⑮
===>
¥FD CR ⑯
===>
¥EXEC LIED@PPS ⑰
===>
JP<CONT> ENDED EC=X00 ⑱
JP<PLAN> ENDED EC=X00 ⑲
JP<LiED> STARTED EC=X00 ⑳
===>
JP<LIED> ENDED EC=X00 ㉑
===>
$END ㉒
```

ジョブ制御文の説明

①MJMPをOKiTAC4300CKで作る。
②入力促進文字
③FACOM-PPS1にLJMPを作る(ユーザプログラム域64KB)
④LJMPの作成終了メッセージ
⑤OKiTAC4300CKLJMPを作る(ユーザプログラム域10ページ,512B/1ページ)
⑥変換プログラムの入力ファイルを論理ファイル名Zで示す。
⑦変換プログラムCONVを起動する。
⑧ジョブプロセスCONVのイニシエート完了メッセージ
⑨コンパイラの出力リストファイル
⑩ファイル登録簿にコンパイラのオブジェクトファイルを登録しておく。
⑪コンパイルのオブジェクトファイル
⑫ジョブプロセスPLANをすぐに起動する。
⑬コンパイルのオブジェクトファイルを編集プログラムの入力ファイルとする。
⑭編集プログラムの出力ファイルを物理ファイル名で指定する。
⑮編集プログラムの出力リストファイル
⑯編集プログラムの制御用データファイル
⑰ジョブプロセスCONV,PLANが正常終了したら,編集プログラムLIEDを起動する。
⑱ジョブプロセスCONVの終了
⑲ジョブプロセスPLANの終了
⑳ジョブプロセスLIEDの開始
㉑ジョブプロセスLIEDの終了
㉒ジョブの終了を指示する。

NETJMに実装した網向きジョブ制御システムプロトコルの使用例は,図5を参照

〔8〕NETJMの評価と今後の課題

計算機網全体を1つのシステムとして管理するオペレーティングシステムNOS上に,網向きジョブ管理システムNETJMを実装し,一応のテストが完了したが,当初の設計目標は達成出来た。今後は効率測定などを行ない改良していくと供に,データベース等を組込み,使い易いシステムを実現していきたい。

今迄NETJMの特徴を列記すると,次のようになる。

・計算機網全体を1つのシステムと見なす
NETJMは,網中のジョブプロセスを管理するオーバヘットやジョブプロセス間の通信や同期のオーバヘッドが妥当な値になると思われる。

・NETJMを機能単位(MJMP,LJMP等)にモジュール化した為に,プログラムの生産性が向上した。

・TECNETシステムのプロセス間通信は,ローカル通信,リモート通信とも,同一手順でアクセスできる為に,NETJMのようなサブシステムを作成する上で有効であった。特にプログラムのデバッグをする上で,ローカル/リモートホストの区別がいらない為に,プログラムの検査期間を短縮できた。

今後のNETJMの課題としては,次のものがある。

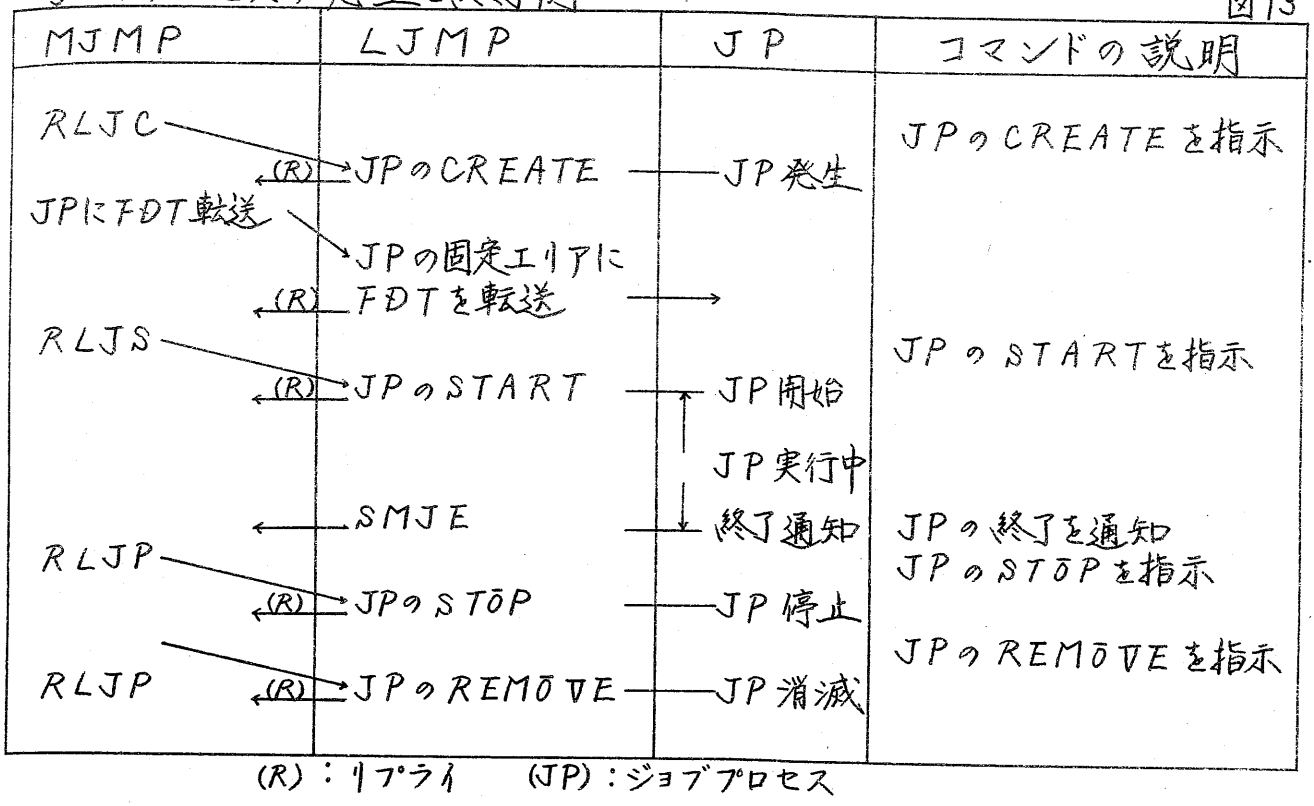
・ユーザに渡す資源(ファイルなど)は,NETJMで確保,返却処理をしていなく,ユーザにすべてをまかせている為に,ジョブプロセス間で,ファイルに関してデッドロックが発生する可能性がある。当然のこととして,NETJMで資源の管理をして,デッドロックを回避する為にも,今後改造をする必要がある。

・NETJMは,プログラム領域などの管理をしているが,プロセス数の管理をしていないので,プロセス管理テーブルのサイズの制約で,ジョブプロセスを開始できない場合もあり,リソース管理と関連して,より細かな制御をする必要がある。

・ファイルの登録簿は,ジョブ単位に管理している為に,ジョブ間にまたがるファイル利用が難かしく,網中の全ファイルを管理するファイルシステムの作成を急ぐ必要がある。

ジョブプロセスの発生と終了例

図13



(R)：リプライ　(JP)：ジョブプロセス

謝辞

NETJMの設計及び製作は，約3年間にわたり，ここに一応の完成を見たが，設計に関しては山内氏，製作は黒羽氏，堀口氏が担当し，総合テストから小田が参加した．設計，製作，テストを通じ，研究室諸氏の協力を深く感謝する．

参考文献

(1) 山内長承，田中英彦：ネットワークジョブ管理システム，情報処理，17回全国大会

(2) Bernstein, A.J "A Computer Architecture for Level Structured Systems" IEEE Trans. on Computers, Vol. C-24, No.8, August 1975

(3) Dijkstra, E.W. "The structure of the "The"- Multiprogramming System," Comm. ACM, Vol. 11, No.5

(4) 和賀井フミ子，田中英彦：TECNETのプロセス間通信システム，情報処理，16全国大会

(5) 堀口真志，田中英彦：計算機網向きジョブ制御言語，情報処理，18回全国大会

(6) 堀田敬志，田中英彦：計算機網ファイルシステム，情報処理，16回全国大会

(7) 小田正美，黒羽法男，田中英彦，網向きジョブ制御システムプロトコル，情報処理，18回全国大会