プログラミング教育を支援するシステム環境の構築

広　内　哲　夫

The Construction of a System Environment for Programming Education

Tetsuo　HIROUCHI

A system called "PROGRESS", which supports programming education was designed and developed from the CAI's viewpoint.

This system was designed to assist a variety of students, from beginners to experts in the field of programming, in receiving a programming education and to realize an stepwise education.

This system gives only basic commands to students. Beginners will not be annoyed with many commands. Following these commands, they are able to carry out file management as well as program correction and execution according to each student's learning style. On the other hand, experts are able to utilize resources of the system jointly with others by expanding the command function.

1．はじめに

最近，多くの教育機関で，プログラミング実習が従来のカードを基本としたバッチ形式から，端末機を用いたTSS形式に移行しつつある．TSSは実習者に端末機を介して，直接コンピュータと対話させる形式なので，バッチ形式に比較して，4～5倍のスピードでプログラミング技術を修得させることができるといわれている．

しかし，TSSの機能を支援するコンピュータ・メーカー開発の基本ソフトウェア（以降，TSS支援ソフトウェアと呼ぶ）は一般に，汎用性や柔軟性の実現に重点が置かれて設計されている．それ故，それらを〝裸のまま〟の形式で実習の現場に導入することは，コンピュータの知識を充分に持ち合わせていない初心者に，使用上の困難を強いることになってしまう．

その対応策として，次の2つの方法が考えられる．その一つは，TSS支援ソフトウェアの利用範囲を制限して，その機能の一部分の

みを実習者に解放することである．もう一つは，実習者とTSS支援ソフトウェアの間にインターフェースを介在させ，その使用上の困難さをインターフェースに肩替りさせることである．

前者の制限方式の場合，実践の立場からは実現容易であるが，しかし実習者がTSS支援ソフトウェアの機能を充分に利用することができない．実習者にとって制限されたシステムは，一貫性のない歪なシステムとして映るであろう．この方式は実習者から不評を買い，長期的な正規の実習システムとして採用できるものではない．

後者のインターフェース方式の場合は，新たにインターフェース・ソフトウェアを開発しなければならないという問題がある．しかし，実習者には，一貫性のある実習システムを提供することができるといえる．

筆者は，文教大学電子計算機センターにTSS支援ソフトウェアを導入するに際し，後者のインターフェース方式を採用し，PROGRESS（Programming Education Support System：プログラミング教育支援システムの英語表現の略称）を開発した．これは次の三つの点を実現すべく設計されたプログラミング教育支援システムである．

(1) 実習者に個人の流儀や作業ペースに従ったプログラミング実習を可能ならしめること（個人学習への対応）．
(2) 実習者に自由に駆使し得るコマンド体系を提供すること（システム操作への対応）．
(3) 実習者に継続的・発展的な学習を可能ならしめること（発展的学習への対応）．

PROGRESSはFORTRAN言語やCOBOL言語を実習者に習得させることを目的とするものではなく，プログラミング実習を容易に行なえるシステム環境を実習者に提供するものである．

本稿では，前半でPROGRESSの設計思想を述べ，後半でその実現の方法を示すことにする．

2．設計思想

2．1　個人学習への対応

一般に，初心者を相手とするプログラミング実習では，TSSを利用するに当たり，実習者全員に同一の作業手順を課すか，あるいは多少の選択の幅のある作業規則を定め，彼等にそれを順守させることが行われる．

これらの手順や規則は，実習中に発生するかもしれないプログラムの破壊，プログラム保存期限の超過，プログラム名の重複，あるいは実行可能プログラムの不完全なリンケージなどの問題を，事前に防止するために定められる．実習者が手順や規則を完全に守らないと，このような悪影響を他の実習者に与え，大きな混乱を引き起してしまうかもしれない．

一方，デバッグ作業におけるバグを発見するためのトレーサーの挿入，あるいはバグを確定するためのテスト・データの作成などは，思考錯誤的技術が要求される作業といえる．この種の思考錯誤の伴なう作業は，全く属人的な流儀によって遂行されるのが普通である．その作業形態は，実習者の力量や習熟度などによって大きな影響を受けるものである．

このようなデバッグ作業を実習者に積極的に体験させるには，実習者同士がお互いに影響を及ぼし合う作業環境ではなく，各実習者に作業上の自由度を完全に保障するシステム的な作業環境が必要となる．そして，実習者を効果的に支援するプログラミング実習のためのシステムは，本質的に各個人の異なる力量あるいは流儀をそれぞれ等しく受容できる方式，すなわち，個人学習を可能とする方式としなければならない．

筆者は，個人学習に対処する方策は，コンピュータ・システム中に，個人専用の作業空間を多数創り出して，これを各実習者にそれぞれ提供することであると考える．以降，この作業空間をパーソナル空間と呼ぶ。このパ

ーソナル空間とは，他に同様に設定されたパーソナル空間から全く影響を受けずに，独立して存在する作業空間である．

このようなパーソナル空間を利用することにより，実習者は他の実習者から完全に切り離された自己の世界で，コンピュータと対話しながら，自己の力量および自己の作業の進展状況に応じて，プログラミング実習を遂行することが可能となろう．作業の遅い実習者は，速い実習者に全く影響されるし，どのようなプログラムにどのような名前を付け，どのように保存してもよい．パーソナル空間とは，このような自己管理可能な作業空間でもある．

2．2　システム操作への対応

システムとの対話は，一般的にコマンドと呼ばれるものを仲介して行われる．コマンドとはシステムの持つ機能を操作するために用意されたコンピュータへの命令語であり，そのコマンド群の集合は，通常，機能および操作の上から一つの体系を成している．

TSS 支援ソフトウェアは，一般に汎用性および柔軟性に重点を置いて設計されているため，精緻なコマンドを数多く持っている．そして，そのコマンド体系は，使用者が当該コマンドを有機的に組み合わせることにより，複雑な作業を行える体系となっている．

これらのコマンドを使い慣れた熟練者は，コンピュータの機能および操作の一般概念を熟知している．彼等は新しいシステムを利用するに当たり，システムの持つ機能の微細なレベルに焦点を合わせながら，コマンドを理解し，それを直ちに実際の細かな作業に容易に適用していく技能を持っている．しかし，初心者は，コンピュータの機能および操作の概念を充分理解しているわけではない．熟練者と初心者の作業上の発想の相違を，バグ・シューティング（プログラム修正）の作業を例にとって述べてみよう．

熟練者は，デバッグの過程に横たわる種々の障害を経験的に予見し得るので，彼等は全体のデバッグ手順の流れを考慮し，最初は，その障害を越えるべく，思考錯誤を働かせながら，微細なレベルのコマンドを用いて，間接的にプログラムの修正をおこなって行く．そして，バグ対策の可能性をすべて追求した後，彼等は，「これで首尾よく解決されたであろう」という信念に達した段階で，始めて，プログラム全体を直接的に修正するのである．

しかし，初心者はこのような全体的な流れを把握し，先を見透す力量を持っていないので，熟練者のような微細で巧妙な作業を行うことは不可能といえる．彼等のプログラムの修正は，一般的に問題が発生する都度，修正を行うといった場当たり的作業である．しかも，作業に当たっては，帳簿の文字や数値を直ちにインク消しを用いて訂正するという日常の作業感覚を頼りに行われる．それ故，彼等は，熟練者の用いるような微細なレベルのコマンドではなく，機能の集約された包括的なコマンドを用いようとする．

このような熟練者と初心者との作業上の発想の相違から，筆者は，初心者のためのプログラミング教育支援システムは，大方の TSS 支援ソフトウェアが持つような精緻なコマンドを豊富に持つよりも，使い易い包括的なコマンドを適度な個数持つ方が重要である，と考える．そして，そのコマンド体系的，初心者が短時間でその全貌を明確に把握できる程度に集約化されており，しかも，それ自身で自己充足の体系となっていることが重要である．

2．3　発展的学習への対応

プログラミング教育支援システムは，単にプログラミングの実習を容易かつ便利にする機能だけに止まっていてはならない．教授する側が実習者の能力に応じて，サンプル・プログラム等を提供し，実習者側がそれらのプログラムを教材として利用できる学習形態が重要である．プログラミング教育支援システ

ムには，この種の提示機構を設ける必要があろう．これは，コースウェアを取り込むフレームワークをシステムに設定することである．

また，プログラミング実習を進めている初心者は，徐々にプログラミングに習熟した熟練者となっていく．完全にプログラミングを身に付けた彼等は，コンピュータの持つ本来的な機能，例えば，図形処理，データベース，あるいは高度なサブルーチン・パッケージを利用したいため，バッチ・システムのジョブ制御言語（以降，JCL と略記する）を直接用いたいと思うかもしれない．そこで，このような機会を与えるシステム的機構をプログラミング教育支援システムに用意する必要があろう．

このシステム的機構とは，閉じられたパーソナル空間から誰もが JCL を利用できる空間へ，プログラムやデータを移し替える機構である．JCL を利用できる空間は，熟練者の利用する空間であるので，開かれた共同利用の空間となってもよい．それ故，この空間をパブリック空間と呼ぶことにする．この種の空間を設定できれば，熟練者はパーソナル空間からパブリック空間にプログラムやデータを転送でき，パブリック空間で発展的なプログラミング実習が可能となる．

さらに，実習者の学習状況を把握することも重要であろう．例えば，実習者がどの程度の頻度でプログラミング実習を行なっているか，あるいはどのようなプログラムをどのような状況で利用しているかといったことを把握することにより，実習の改善が行なえる．そこで，実習者各個人をモニタリングする機能も必要となろう．

以上述べた副次的な機能を積極的に付加することは，プログラミング教育支援システムの存在価値をより高めることになる．

3．コマンドの設定

3．1　コマンド設定の基準

システムの持つ機能は，コマンドを介してのみ利用できるのであるから，システムの設計者は，システムの目的から導き出せる機能を，コマンド体系に過不足なく反映しなければならない．

筆者は，プログラミング教育支援システムのためのコマンドを設定する基準として，2．2節の考察に従って次の4つの基準を採用した．

(1)　コマンドの個数はなるべく少なくすること．

(2)　コマンドにはコンピュータ特有の精緻な機能よりも，日常の作業感覚により近い具体的な機能を持たせること．

(3)　コマンドには具体的な作業内容を連想できる名前を付けること．

(4)　コマンド同士で相互依存関係を持たせないこと．

コマンド体系を構成する適切なコマンドの個数とは，初心者に短時間でその体系の概念を把握させ得るところの個数といえる．筆者はその個数の目安を，一応20個とした．これは，使い易いといわれている対話型 BASIC 言語の持つコマンドの個数が，多くの場合，20個前後に設定されており，それが一番容易に学習できる個数と考えられるからでもある．

筆者は，上記の4つの基準と，そのコマンドの総和数の目安が20個という制限に従って，学習者のプログラム作成の作業手順の分析から必要とするコマンドの機能を確定し，それらを次節に示す20個のコマンドに集約した．このコマンド体系は，初心者のためのプログラミング教育支援システムにとって，充分，自己充足的であると考えられる．

3．2　コマンドの種類

コマンドは9つの観点から分類されており，四角で囲まれたキーワードがコマンドの名称となる．

3.2.1　システムへのテキストの投入

(1) 端末機からプログラムおよびデータ（以降，テキストと呼ぶ）を1ステートメントづつ[入力]し，名前を付けてパーソナル空間に登録するコマンド．

(2) すでに登録されているテキストに，ステートメントを[追加]入力するコマンド．

追加入力のコマンドは，実習の時間の関係で，途中で打ち切った入力作業を再会させるために必要である．

3.2.2　テキストの修正

(1) 登録されているテキストを端末機を用いて，直接，[修正]するコマンド．

(2) テキストをカードを介して，直接，[更新]するコマンド．

(3) テキストの各ステートメントに順次番号を[付番]するコマンド．

(4) テキストの各ステートメントに付けられた順次番号を[消去]するコマンド．

端末機を用いた修正は，思考錯誤による修正が可能であるが，カードを用いる修正では，それが行えないので，事前にすべての更新作業の手順を定めなければならない．それ故，修正と更新の2つのコマンドを設定する必要がある．付番のコマンドは，カードによる更新が一般にステートメントの順次番号を頼りに行なわれるために必要となる．

3.2.3　テキストの表示

(1) 登録されているテキストを端末機の画面に[表示]するコマンド．

(2) テキストを印書装置に[印書]するコマンド．

印書のコマンドは，多量のテキストを一度に印書する際に利用する．

3.2.4　プログラムの翻訳および実行

(1) 登録されているプログラムを，FORTRAN，COBOLなどの言語プロセッサーで[翻訳]するコマンド．

(2) 翻訳されたプログラム群を自動的にリンケージして，[実行]するコマンド．

実行のコマンドにおいては，翻訳，結合，実行の3フェーズがすべて自動的に行なわれる．

3.2.5　テキストの生成

(1) 登録されているテキストを，[複成]するコマンド．

(2) 2つのテキストを[併合]して，より大きな1つのテキストを生成するコマンド．

複成と併合の各コマンドは，類似のテキストを作成する場合，またはデバッグ作業を効果的に行う場合に利用する．

3.2.6　パーソナル空間の管理

(1) パーソナル空間に登録されているテキストを[参照]し，その名称や作成年月日などを表示するコマンド．

(2) テキストの名前を[再命令]するコマンド．

(3) パーソナル空間から不要のテキストを[削除]するコマンド．

参照，再命令，削除のコマンドは，実習者が自己のパーソナル空間を管理するために用いる．

3.2.7　システム機能の拡張

(1) コースウェアを取り込むため，それが格納されている空間を，実習者のパーソナル空間の一部として[連結]するコマンド．

(2) プログラムの実行条件等を[拡張]するコマンド．

拡張のコマンドは，初心者向きに設定されたCPU実行時間や印書ページ数の枠を越えて，熟練者が実行する際に利用するものである．

3.2.8　テキストの空間移動

(1) テキストをパーソナル空間からパブリッ

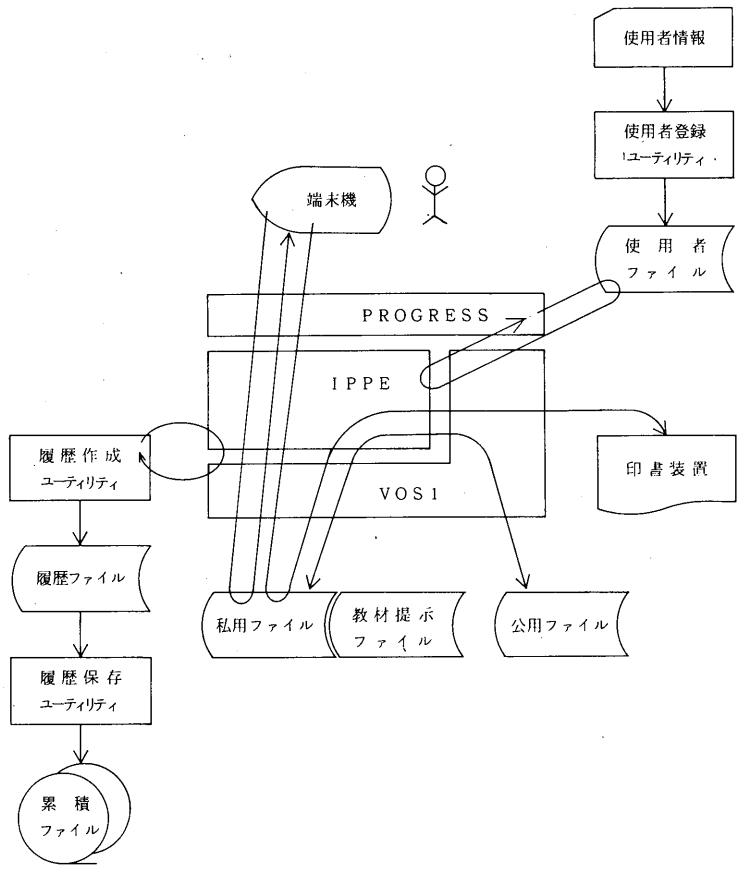

図1　PROGRESS のシステム環境

ク空間へ 転出 させるコマンド.

(2)　テキストをパブリック空間からパーソナル空間へ 転入 させるコマンド.

転出と転入のコマンドは，パーソナル空間に穴をあけ，パブリック空間に連絡をとる働きをする.

3.2.9　コマンド仕様の教唆

(1)　コマンドの仕様および体系を実習者に 教唆 するコマンド.

教唆のコマンドは，コマンド一般の使い方をおぼろげにしか思い出せない初心者にとって有効となる.

4. 実現方法

4.1　実現化の前提

文教大学電子計算機センターに導入されているコンピュータは，中型汎用の HITAC-M 150 である. そのハードウェア構成は，主メモリーが1MB，磁気ディスク・ファイルが70MB×6台である. 端末機は教育用に10台であるが，ハード・コピー装置は付属されていない. 基本ソフトウェアは，オペレーティング・システム（以降，OS と略記する）として，バッチ専用の VOS1，TSS 支援ソフトウェアとして IPPF が導入されている.

IPPF はそれ自身で完全に端末機による対話処理を支援できるソフトウェアであるが，汎用性，柔軟性を指向して設計されているため，200個以上の精緻なコマンドを持っている. 初

心者がそれらの様なコマンドを使いこなすのは難しい．それ故，IPPFを実習の場に，"裸のまま"導入するのは賢明とはいえない．

しかし，IPPFには，パーソナル空間を確保するのに有効であろうと思われる「ファイル設定機能」，精緻なミクロ・レベルのコマンドの機能を集約し，包括的なコマンドを創り出せる「マクロ機能」，およびTSSによる対話処理をバッチ処理に移管する「サブミット機能」が備っている．

そこで，筆者は，2章，3章で示した初心者のためのプログラミング教育支援システムの設計思想を，上記の3つの基本機能を利用して実現化しようと考えた．これは図1に示すように，IPPFを基礎として，使用者とIPPFの間にインターフェースを設定する方式で，PROGRESSを開発しようとするものである．以下に，その実現化の方法を示す．

4．2　私用ファイルの導入

個人学習のためのパーソナル空間を設定することは，ソフトウェア技術の観点からすれば，パスワードで保護された作業用ファイルを，実習者に見合う数だけコンピュータ中に確保することに対応する．しかし，一般のコンピュータシステムでは，それらの条件を満たした非常に多くのファイルを，相互独立の形式で確保することは，ファイルの管理運営およびOSの効率の観点から，困難に近い．そこで，通常は，あるまとまった実習者グループに対して，1つのファイルを貸与することが行われる．しかし，この場合，ファイルを共同で利用する実習者の間で，お互いに利用上の約束を守らねばならないことになり，各実習者に対する完全なパーソナル空間の提供は不可能となる．

IPPFは，VOS1の管理下の1つの物理ファイルを多数のサブファイルに分割し，それらのサブファイルを論理的に独立したファイルに再度設定し直すファイル設定機能を持っている．この機能を用いること，VOS1から見た1つの物理ファイルは，IPPFから見ると，完全に相互独立の多数の論理ファイルとして実現されることになる．そして，これらの論理ファイルは，使用者コードとパスワードで完全に保護される．

PROGRESSにおいては，IPPFのファイル設定機能を用いて，実習者各個人にIPPFの1つの論理ファイルを与えることとした．このファイルが2．1節で述べたパーソナル空間を具体化したものとなる．それ故，以降，この論理ファイルを「私用ファイル」と呼ぶことにする．私用ファイルの設定により，実習者は，他の実習者から全く影響を受けずに自己の閉じられた世界でコンピュータを使用することができる．

しかし，この方式の欠点は，コンピュータの運用管理者の立場からすると，使用者コードやパスワードの管理（登録や削除）が厄介なことである．IPPFにおいては，これらの使用者情報は，事前に図1に示す使用者登録ユーティリティを用いて，使用者ファイルに登録される．使用者数を最大700人と見積っているが，この繁雑な作業は，登録事務を年一回集中的に行えば，解決される問題である．実習者の個人学習の実現という観点からすれば，その利点は登録の厄介さをもってしても余りあるといえよう．

4．3　プログレス・コマンドの作成

4．3．1　コマンド作成の前提

IPPFのコマンド体系は非常に精緻に作られており，200個以上の多種多様なコマンドから成っている．コマンドの持つ機能は，コンピュータ特有の基本的な機能から，人間の作業感覚に対応する包括的な機能まで，多岐に渡っている．

筆者は，このコマンド群の仕様および機能を注意深く解析しながら，「如何にそれらのコマンドを組み合わせて集約（マクロ化）すれ

ば，3．2節で述べた抽象的な機能のコマンドを，PROGRESSにおける実用上のコマンドとして実現できるか」を検討した．その結果，充分に実用性のあるPROGRESSのコマンドを作り出せることを見い出した．

IPPFのコマンドは数が多いことから，その組み合せは幾通りも考え出されたが，その中からPROGRESSのコマンド体系として斉合性があり，最良と思われるものが一般の対話処理システムの使用上の筆者の経験をもとに選び出された．PROGRESSのコマンドをIPPFの持つ基礎的なコマンドを区別するために，以降，前者を「プログレス・コマンド」，後者を「IPPFコマンド」と呼ぶことにする．次節以降で，プログレス・コマンドの外部仕様の一般形式およびその作成方法を紹介する．

4．3．2　コマンドの仕様

プログレス・コマンドの外部仕様の一般形式は，次の形式とした．

@ command 〔option〕 △ operand

特殊記号@はコマンドの先頭を示し，commandはコマンド名を表わすようにした．コマンド名には略記号を用いないで，3．2節のコマンド設定規準に従って，具体的な作業の内容を明確に表わす英語の動詞のスペルを採用することとした．operandはcommandの作用（命令）の対象となるもので，そこには私用ファイルを登録されているプログラム名およびデータ名を指定することとした．（コマンドの種類によっては，operand指定の不要なものも存在する．）optionはコマンド名だけで機能を表現できない場合に，英数字の組み合せによってその表現を補強するのに利用することとした．

```
@MACRO
@NOPRINT
/INPUT
/CANCEL
/INPUT
/INSERT &&PARAM1
/INSERT &&PARAM2
/END
/SAVE &&PARAM3
```

図2　@MERGEに対するIPPFコマンド列

プログレス・コマンドの外部表現は，何（operand）を如何に処理するか（command）という単純な形式となっているので，プログレス・コマンドは初心者にとって，理解が容易と思われる．

4．3．3　マクロ機能によるコマンドの作成

プログレス・コマンドの作成方法を説明する意味で，併合のプログレス・コマンドを取り上げる．このコマンドの外部仕様は，前節の一般形式に従って，次のように定められた．

@ MERGE prog1, prog2, prog3

併合のプログレス・コマンドの外部仕様は，「prog1とprog 2の2つのプログラムを併合して，新しいより大きな1つのprog3のプログラムを創り出す」ことを意味する．

プログレス・コマンド@MERGEに対応するIPPFコマンドの最良の組み合せは，図2に示される．この組み合せによる機能は，上記に示した外部仕様と全く等価である．ここではIPPFコマンドの動作を述べるのが目的ではないが，理解を進めるため，IPPFコマンド列の時系列動作を以下に述べておく．

まず，作業のために用意されている一時的な作業領域をクリアーし（／INPUTと／CANCELの働きによる），その領域でprog1とprog2を併合する準備を行う（／INPUTの働きによる）．prog1を私用ファイルから作業領域に入力し（／INSERT △&& param 2の働きによる），そして，作業領域への入力の完了を宣言する（／ENDの働きによる）．最後

に併合されたプログラムに名前を付けて，私用ファイルへ保存し(／SAVE △&& param3の働きによる)，併合作業が完了する．

上記のIPPFコマンド列は，図2に示されるように，その先頭にマクロ化を意味する@MACROと@NOPRINTが付加された後，プログレス・コマンド@MERGEに対応付けされて，IPPFのマクロ機能に組み込まれる．

実習者が端末機から，@MERGEを投入すると，IPPFのマクロ機能が自動的に働き，IPPFコマンド列が時系列的に1ステップづつ解析され，実行されるのである．その際，

表1　プログレス・コマンドを構成するIPPFコマンドとJCLのステップ数

機　能	ニモーニック	IPPFコマンド	JCL
教　唆	@HELP	40	
入　力	@INSERT	6	
追　加	@ADD	7	
修　正	@MODIFY	6	
更　新	@UPDATE	8	
参　照	@REFER	2	
表　示	@SHOW	2	
再名命	@RENAME	2	
削　除	@PURGE	2	
複　成	@CREATE	8	
併　合	@MERGE	9	
付　番	@NUMBER	2	
消　去	@BLANK	23	
連　結	@JOIN	2	
拡　張	@EXTEND	19	
転　入	@IMPORT	51	
転　出	@EXPORT	44	
印　書	@LIST	27	8～20
翻　訳	@COMPILE	27	6～16
実　行	@RUN	27	8～28

PROGRESS側のprog$n$とIPPF側の&& param$n$は，マクロ機能により対応付けられる．

印書，翻訳，実行を除く他のプロブレス・コマンドも，上記の@MERGEと同様な方法により具体化された．表1に，プログラス・コマンドが，何個のIPPFコマンド列から構成されているかを，実際に用いたニモーニック（コマンドのシンボリック名称）とともに示す．

入力，追加，修正，更新のプログレス・コマンドに関しては，それらの機能の複雑さからして，単一のコマンド操作で，その作業を遂行することは不可能である．そこで，上記のプログレス・コマンドのもとでは，IPPFの持ついくつかのサブコマンドを解放し，実習者が直接，IPPFの持つ画面編集機能を利用できるようにした．

4.3.4　サブミット機能によるコマンドの作成

実習者にとってハード・コピーが必要とされる作業は，印書，翻訳，および実行に関する作業といえる．しかし，教育用の端末機には，ハード・コピー装置が付設されていない．そこで，筆者は，これらの作業に関して，その結果がすべて中央の印書装置に出力されるように，それらのプログレス・コマンドの内部仕様を設計した．この仕様は，バッチ・ジョブを発生させるIPPFのサブミット機能を用いて具体化された．

その実現の手続きは次の通りである．プログレス・コマンドの外部仕様と等価の働きをするが，未だ完全ではないJCLのイメージが，他のIPPFのコマンド列とともにマクロ化されIPPFに登録される．プログレス・コマンドが実行されると，マクロ機能が自動的に働き，当該プログレス・コマンドのオペランドとオプション部分に指定された情報（処理対象のプログラム名など)を用いて，完全なJCLのイメージを作り出す．そして，そのJCLイメージはサブミット機能により，バッチ・ジョブとしてVOS1に引き渡される．これ以後，JCLはIPPFの支配を離れ，完全なバッチ・ジョブとして処理され，結果は印書装置に出力される．以上が実現化の概要である．

この具体例を，実行のプログレス・コマン

ドについて示す．このコマンドの外部仕様は以下のように決められた．

> @RUN*n*D　language, prog1,…,
> prog*n*, data

コマンドの外部仕様は，「プログラム言語 language を用いて書かれた *n* 個のプログラム prog1，～，prog*n* を翻訳した後，実行可能プログラムを作り出し，それをデータ data を用いて実行する」ことを意味する．コマンド名 RUN の後に付けられている *n*D はオプションであり，*n* はプログラムが *n* 個から成り，D は実行に際しデータを用いることを意味する．

@RUN に対応する JCL と IPPF コマンドの列を図 3 に示す．四角で囲まれたものが JCL の未完のイメージであり，そのイメージを挟む形で上下に，完全な JCL のイメージを作り出し VOS1 に引き渡す働きをする IPPF コマンド列が存在する．JCL と IPPF コマンドの列は，VOS1 と IPPF の機能の解析から決定されたものである．これらの列はプログレス・コマンド @RUN に対応付けて，IPPF に登録される．

実習者が端末機から，@RUN1D を投入すると，図 3 に示す不完全な JCL イメージが，マクロ機能により完全な JCL イメージに仕立て上げられ，サブミット機能により VOS1 に引に渡される．その後，その JCL イメージはバッチ・ジョブとしてスケジュールされ，プログラムの翻訳（／／EXEC △&& param1），データの印書（／／EXEC △ PGM＝PRGRSPRRT），プログラムの結合（／／EXEC △ LNKEDT），および実行（／／EXEC）の各過程を経て，結果は印書装置に出力される．

他の 印書 と 翻訳 のプログレス・コマンドも，上記の実行のプログレス・コマンドと同様な方法によって作成された．表 1 に，印書，翻訳，実行 の各プログレス・コマンドを構成する JCL と IPPF コマンドの個数を，実際に設定されたニモーニックと共に示す．

```
@MACRO
/ECHO  ***  SUBMIT-JOB  RUNNING  ( RUN )  ***
@NOPRINT
/INPUT NO
// EXEC   &&PARAM1
/INCLUDE &&PARAM2
/*
// FILE IJSYSCL,'LSYSCLB FILE VOS1',,,SYSCLB,VOL=BUCC03,DISP=OLD
// ASSGN SYSCLB,DISK,VOL=BUCC03,SHR
// EXEC PGM=PRGRSPRT
&&PARAM3@@MEMBER
/INCLUDE &&PARAM3
/*
// EXEC  XLNKEDT
// EXEC
/INCLUDE &&PARAM3
/*                                    JCL のイメージの枠組
/&
/END
/SAVE PIMGSBJB                        ←── JCL イメージの IPPF への登録
@PRINT
PIMGSBMT PIMGSBJB                     ←── JCL イメージの VOSI への引渡し
@NOPRINT
/PURGE PIMGSBJB                       ←── JCL イメージの IPPF からの登録解除
@PRINT
/ECHO  ***  SUBMIT-JOB  ENDED  ( RUN )  ***
```

図 3　@RUNID に対応する IPPF コマンドと JCL の列

4．4　発展的学習への対応

4．4．1　教材提示ファイルの設定

コースウェアを取り込むファイルは，IPPFのファイル設定機能を用いて設定された．ファイルの設定時に“共用”と宣言すれば，当該ファイルは，各実習者のもつ私用ファイルのもとで読み出し専用で利用可能となる．図4の一部には，コースウェアを取り込むファイルが教材提示ファイルとして示されており，矢印は読み出し方向が示されている．

なお，このファイルは，IPPFのファイル連結機能を用いて作成された[連結]のプログレス・コマンド@JOINを使用することにより，各実習者の私用ファイルに連結される．

4．4．2　公用ファイルの設定

PROGRESSを離れてJCLの直接利用を可能とするパブリック空間は，VOS 1が直接管理する物理ファイルを利用して実現された．（図4参照）．このファイルを，以降，「公用ファイル」と呼ぶ．公用ファイルは，システムにとっては，ただ一つ設定されるものである．公用ファイルと私用ファイルの間のプログラムやデータの転送は，[転入]，[転出]の，プログレス・コマンド@IMPORT, @EXPERTによって行われる．

公用ファイルは多数の実習者から共同で利用されるため，個人の勝手な流儀で使用してはならないファイルと決められた．公用ファイルの利用に際しては，各自が次の使用規則を守らねばならない．それは，①プログラムやデータの名前には，使用者のイニシャルを付ける，②保存期間は1日で，1日以降は自動削除，③プログラムやデータの破壊に対しては，誰も責任を負わない，等である．

なお，私用ファイルと公用ファイルの間でのプログラムやデータの転送のアクセス権は，私用ファイル側に設定されているので，公用ファイル側から私用ファイル側へ介入することは不可能である．

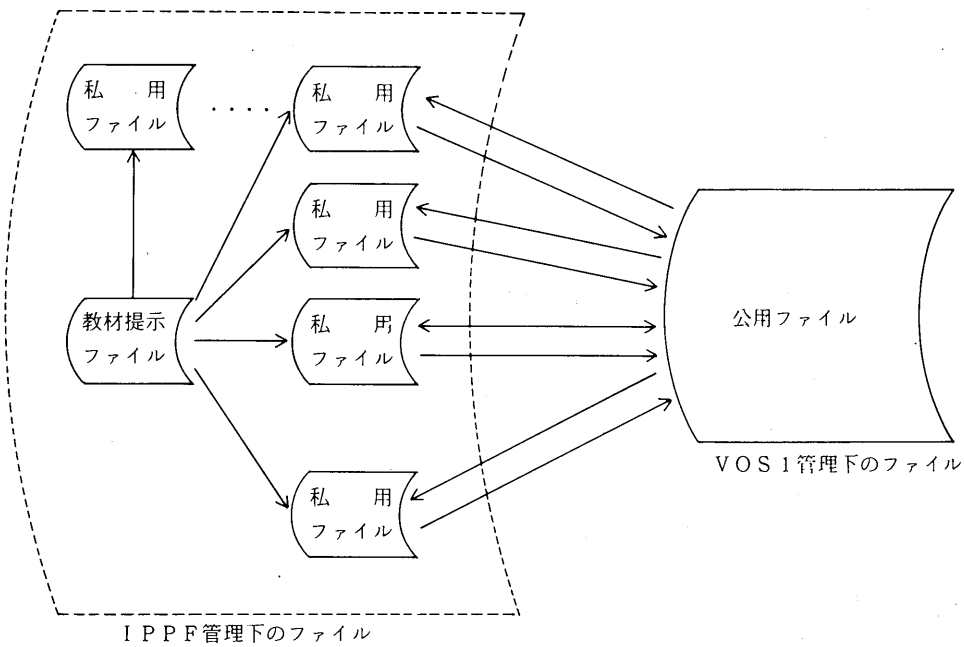

図4　PROGRESSにおけるファイル

4.4.3　モニタリング機能の設定

PROGRESSはそのシステム内部で，IPPFとVOS1機能を利用している．PROGRESSの使用者の学習状況や使用状況をモニタリングするには，この内部のIPPFとVOS1のロギング・システムを活用する必要があるが，それぞれのロギング・システムは，システム的に断絶しており，個別に稼動するようになっている．そこで，この2つのロギング・システムを統合化して処理できる履歴作成ユーティリティを開発した（図1参照）．

このユーティリティは，その管理下の履歴ファイルに，使用者の個人レベルの記録，すなわち，PROGRESSの利用回数，使用時間数，印書装置使用回数，使用プログラム名，プログラム終了状態などの情報を，日々集計し，蓄積するものである．なお，この履歴ファイルの内容は，月末にテープに累積保存される．このファイルを解析することにより，実習者各個人の学習状況を把握でき，今後の実習の改善に役立てることができる．

5．プログレス・コマンドの使用例

私用ファイルは，実習者各個人に使用者コード（それには，他人からの流用を防ぐためのパスワードが付いている）とともに貸与される．私用ファイルの貸出し期間は，年度始めから年度末までの1年間で，その間，システム運用管理者からアクセス（介入）されることはない．私用ファイルの容量は，初心者に対しては，最大500ステップの原始プログラムが11本迄，熟練者に対しては，最大2,000ステップの原始プログラムが29本迄，それぞれ保存可能である．私用ファイルの管理は，その一切が実習者各個人に任される．

次に，プログレス・コマンドを使った作業例を紹介しよう．作業の概要は次の通りである．目的のFORTRANプログラムが何というプログラム名であったかを忘れたので，私用ファイルに登録されているプログラムの名前を［参照］する．画面に表示された一覧リストから，それがOLDであることを知ったので，次にそれを［複成］し，新たに生じたものにNEWと名付ける．次にNEWを改良するために［修正］を施す．NEWを走らせるためのテスト・データを私用ファイルに［入力］し，DATAという名で登録する．そして，NEWを［実行］する．実行リストを確認した後，OLDを不要なものとして［削除］する．

この作業に関するプログレス・コマンドの実行順序は次の通りである．

手順1 ［参照］ @REFER
手順2 ［複成］ @CREATE OLD, NEW
手順3 ［修正］ @MODIFY NEW
　　［修正作業］
手順4 ［入力］ @INSERT
　　［テスト・データの入力とDATAの登録作業］
手順5 ［実行］ @RUN1D FORTRAN, NEW, DATA
手順6 ［削除］ @PURGE OLD

以上のように，非常に簡単な手順で，一連の作業を行なうことが可能である．

6．おわりに

最近，マイコンの対話型BASIC言語が，比較的簡単に習得できるということで，初心者から評判が良い．筆者はこれを次の2つの理由によるものと考える．

(1) 大型コンピュータと異なり，マイコンは他人に影響されずに，自分のペースで利用することができる．

(2) BASICのスラートメントの分り易さもさることながら，BBSIC自身の持つコマンド体系が非常に簡単である．そのコマンドを用いると，コンピュータを自由に使いこなしているような満足感を味わえる．

これは，マイコンがまさしくmy computer

といわれる所以である.

従来の対話処理による実習システムでは, FORTRAN や COBOL などの実用言語を, マイコンによる BASIC のようなパーソナルな感覚で学習できない状況にあった.

そこで, 筆者は, パーソナル感覚を多少なりとも実現することを目指して, プログラミング教育支援システム PROGRESS を開発した. その設計に際しては, 前に掲げたマイコンの評判の理由(1), (2)を設計上の拠り所とした.

PROGRESS は昭和56年度の秋から本格的な利用を開始した. 学生に使わせていると, 非常に興味を示し, 3時間のトレーニングを受けて, 大半の学生はその概要を理解したようである. これで FORTRAN や COBOL の実習が, マイコンによる BASIC の学習のように, 多少なりともパーソナルな感覚で行なえるようになったといえるかもしれない.

システムを設計することは, 建築設計と同じように, 技芸の領域に属する性質のものであるから, その成果は, 一般に, 設計者の個性, 趣味, センスなどから, 非常に大きな影響を受ける. この点, PROGRESS も筆者の個人的な趣味の影響を受けているのを否めない. 今後, 使用者の意見を取り入れて, 更に改良して行く予定である.

なお, PROGRESS は, 端末機を用いたコマンド操作と全く同じ操作をカードを用いて行なうことも可能である. これについては, 日立製作所に IPPF の改造をお願いして可能になったことを付け加えておく.

最後に PROGRESS の開発に当たり, 献身的な努力をして下さったファコム・ハイタック㈱の綾部秀二, 岡山克守, 大本和彦の各氏, および文教大学電算室の海老沢信一氏に深く感謝の意を表します.

参考文献

1. 広内哲夫著　電子計算機利用の手引き(改訂版)　昭和57年4月
2. 日立製作所編　IPPF 解説　昭和55年10月
3. 日立製作所編　VOS 1 制御プログラム解説　昭和55年3月

(1982年9月22日受付)