PASCAL プログラム輪郭作成システムの実現とその移し換えについて†

牛島和夫†† 江嶋博††

プログラム中の各文の実行回数を知ることは，プログラムの開発や改善にとって有用である．このことを自動的に行うシステムが幾つか実現されている．PASCAL に対して同様なシステム（PASDAP と称する）を九州大学情報工学科の FACOM 230-45 S OS II 上に作成し，その後 FACOM M 190 OS IV/F 4（九州大学大型計算機センター），HITAC 8800/8700 OS 7（東京大学大型計算機センター），HITAC 8350 EDOS/MSO（電気通信大学情報数理工学科）に移し換えた．ここでは，PASDAP システムの実現法を簡単に述べ，移し換えの際に直面した問題とその解決法について詳細に述べる．それらは，言語仕様，ファイル仕様，コンパイラへの入力形式，ジョブ制御文の差異などである．

1. はじめに

実行回数計数ツールは，プログラムの各実行文の実行回数を計数するというきわめて簡単な原理に基づくものでありながら，プログラムの作成から保守に至る各段階で有用な道具となりうる．しかしながらその使用が必ずしも一般的になっているとはいいがたい．このような道具の使用を普及させるためには，これを多くの計算機システムの上で実際に使用可能にするとともに，これを効果的に使用した例を多く提示してその有用性を周知させる必要がある．とくにプログラミング教育の段階から使用させプログラム作成における道具の役割を体得させることは意義がある．先にわれわれの手許で実現された FORTRAN プログラム動的解析システム (FORDAP) や COBOL プログラム輪郭作成システム (COBOLDAP) はさまざまな計算機システムに移し換えられて一般の利用に供されているが[1),2)]，プログラミング教育を第一義として設計開発された PASCAL に対しても同様な目的を持った輪郭作成システム（PASDAP と呼ぶ）を九州大学情報工学科の FACOM 230-45 S OS II（以後 OS II と略す）のもとで稼動中の PASCAL 230 処理系[3)]に対して作成した．その後これを，FACOM M 190 OS IV/F 4（九州大学大型計算機センター，以後 OS IV と略す），HITAC 8800/8700 OS 7（東京大学大型計算機センター，以後 OS 7 と略す），HITAC 8350 EDOS/MSO（電気通信大学情報数理工学科，以後 EDOS と略す）などへ移し換えた[4),5)]．移し換えの対象となったこれらの計算機システムで稼動中の PASCAL は，すべて PASCAL 8000 と称せられ，東京大学大型計算機センターで実現され[6)]，各計算機へ移植されたものである．

ここでは，PASDAP の実現法を簡単に述べ，移し換えの際に直面したいくつかの問題とその解決法について詳しく述べる．

2. PASDAP システム

2.1 システムの機能と構成

PASDAP の主な機能は次の 2 点である．

（1） PASCAL プログラムの各実行文の実行回数を計数し，実行所要時間を計測する．

（2） 上記の結果をわかりやすい形で出力する（**図 1** 参照）．

図 1 において，EXECUTION PROFILE 欄が各実行文の実行回数を示している．1 行に複数個の実行文を含む場合にその実行回数は横に並べて表示される．なお，出力リスト第 2 行目の EXECUTION TIME は，プログラム全体の実行に要した時間を示すもので，実行回数計数に要するオーバヘッドも含んでいる．

システムの構成を**図 2** に示す．このシステムの主要部は前処理部（図 2 中(1)）と後処理部（図 2 中(4)）であり，共に PASCAL で記述されている．前処理部は，計測に必要なカウンタなどをソースプログラム（図 2 中(A)）に挿入し（図 2 中（D)），ソースプログラ

† Design and Implementation of a PASCAL Program Profiler in Various Environments by KAZUO USHIJIMA and HIROSHI ESHIMA (Department of Computer Science and Communication Engineering, Faculty of Engineering, Kyushu University).

†† 九州大学工学部情報工学科

```
FACOM   230-45S   OS2              PASDAP      -781122-     V-02   L-05          DATE  79-04-10   TIME  15:10:38
  SOURCE STATEMENT                    EXECUTION TIME =      7304 MSEC.                EXECUTION PROFILE

 1  PROGRAM FREQENCY(INPUT,OUTPUT);    (*VW-  *)                                  00000010 |
 2    CONST AL = 8;   (* LENGTH OF WORD *)                                       00000020 |
 3    TYPE WORD = @WORDREF;                                                      00000030 |
 4         WORDREF = RECORD NAME: ALFA; FREQ: INTEGER;                           00000040 |
 5                          GREATER,LESS: WORD                                   00000050 |
 6                  END;                                                         00000060 |
 7    VAR ROOT: WORD;  (* ROOT OF BINARY TREE *)                                 00000070 |
 8        WD: ALFA;    (* LAST WORD READ *)                                      00000080 |
 9        CH: CHAR;    (* LAST CHARACTER READ *)                                 00000090 |
10        I,J: INTEGER;                                                         00000100 |
11        A: ARRAY (.1..AL.) OF CHAR;                                           00000110 |
12                                                                              00000120 |
13    PROCEDURE SEARCH(VAR W: WORD);                                            00000130 |
14      VAR W1: WORD;                                                           00000140 |
15    BEGIN W1 := W;                                                            00000150 |  20980 |        |        |
16      IF W1 = NIL THEN                          (* NOT FIND AND ENTRY *)     00000160 |  20980 |        |        |
17      BEGIN NEW(W1); W := W1;                                                 00000170 |    301 |    301 |        |
18        WITH W1@ DO BEGIN NAME := WD;                                         00000180 |    301 |    301 |        |
19          FREQ := 1; GREATER := NIL; LESS := NIL END                          00000190 |    301 |    301 |    301 |
20      END                                                                     00000200 |
21      ELSE IF WD > W1@.NAME THEN SEARCH(W1@.GREATER)   (* SEARCH *)         00000210 |  20679 |   8437 |        |
22      ELSE IF WD < W1@.NAME THEN SEARCH(W1@.LESS)                             00000220 |  12242 |  10550 |        |
23      ELSE W1@.FREQ := W1@.FREQ + 1                    (* FIND *)            00000230 |   1692 |        |        |
24    END;  (* SEARCH *)                                                        00000240 |  20980 |        |        |

                    (OMITTED)

33                                                                              00000330 |
34  BEGIN ROOT := NIL; CH := ' ';                                               00000340 |      1 |      1 |        |
35    WHILE NOT EOF(INPUT) DO                                                   00000350 |      1 |        |        |
36      IF CH IN (.'A'..'Z'.) THEN                                              00000360 |  36694 |        |        |
37      BEGIN I := 1;                                                           00000370 |   1993 |        |        |
38        REPEAT IF I <= AL THEN A(.I.) := CH;                                  00000380 |   1993 |  10254 |  10002 |
39          READ(CH); I := I + 1                                                00000390 |  10254 |  10254 |        |
40        UNTIL NOT (CH IN (.'A'..'Z','0'..'9'.));                              00000400 |  10254 |        |        |
41        FOR J := I TO AL DO A(.J.) := ' ';                                    00000410 |   1993 |   5942 |        |
42        PACK(A,1,WD); SEARCH(ROOT)                                            00000420 |   1993 |   1993 |        |
43      END ELSE READ(CH);                                                      00000430 |  34701 |        |        |
44    PAGE(OUTPUT); PRINT(ROOT)                                                 00000440 |      1 |      1 |        |
45  END.                                                                        00000450 |      1 |        |        |
```

図 1 PASDAP リストの例
Fig. 1 Example of a PASDAP list.

ムとカウンタ番号の対応表（図2中(C)）を作成する．編集された PASCAL プログラムがコンパイルの後，実行される．後処理部では，前処理部で作成された上記の対応表をもとに計測結果を編集，出力する（図2中（E））．これが図1のリストに相当する．

なおシステムを記述する言語は必ずしも PASCAL である必要はない．しかしほかの計算機システムにもできるだけ容易に移し換えられること，PASDAP を道具として使用するからには，移し換え先に PASCAL 処理系が必ずあると想定したことなどにより，手続きを標準 PASCAL[7]に準拠して記述した．

2.2 変換規則

前処理部による被計測プログラムの変換は1パスで行う．その際の変換規則を以下に示す．ここで，対象となるソースプログラムは，PASCAL コンパイラを一度通ってエラーが検出されなかったものとする．

（1） プログラム頭部に，PASDAP が使用するファイルの宣言を追加する．

（2） プログラムの変数宣言部に計測に必要な変数の宣言を追加する（**表1参照**）．

（3） プログラムの変数宣言部の直後に後処理部の手続き宣言部を挿入する．

（4） プログラムの実行文部の最初に

```
for sys800in:=1 to 2000 do
    sys700ct [sys 800 in]:=0;
sys600cl:=clock;
```

を挿入する．

（5） プログラムの実行文部の最後に

```
sys700ct [1]:=clock−sys 600 cl;
sys500pr;
```

を挿入する．

（6） 複合文の **begin** および repeat 文の **repeat** の直後にカウンタ

```
sys700ct [n]:=sys 700 ct [n]+1;
```

を挿入する．

（7） 構造文に含まれる実行文が複合文であれば，それについて（6）を適用する．単文ならその文に対して

```
begin sys700ct [n]:=sys700ct [n]+1;
〈実行文〉
end
```

とする．

（8） goto 文，手続き文および名札の直後にカウンタを挿入する．

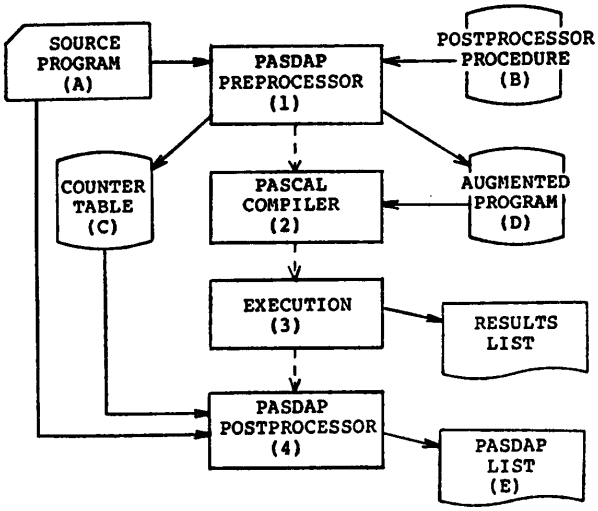

図 2 PASDAP システムの概略の流れ
Fig. 2 General flow of PASDAP system.

これらの変換規則によって，図1のプログラムを変換した例を図3に示す．図3中番号の付されている部分は，それに対応する規則によって挿入されたものである．ただし，このリストは普通は出力されない．

上記規則（3）について若干説明を加える．PASCAL にはプログラム単位の結合という概念がない．したがってソーステキストの形で用意してある後処理部手続きを副手続きとして被計測プログラムの中に複写して挿入することにした．できるだけテキストの行数を減らして，複写に要する手間を軽減させるように配慮してある．なお，OS 7 と OS IV の PASCAL 8000 では FORTRAN 副プログラムをダイナミックリンクにより結合できる[4]．また，外部の PASCAL 副手続きを結合できるよう拡張された処理系もある．しかし移し換えの容易さを考えてそのような拡張機能は使用しないことにした．

3. PASDAP の移し換え

3.1 移し換えの条件

図2で示されるようなシステム構成をもつ PASDAP システムを，全体としてそのままほかの処理系へ移し換えるためには，相手の処理系が次のような条件を満足していることが必要である．

P 1 PASCAL で記述された前処理部および後処理部が正しく動くこと．

P 2 必要なだけのファイルが自由に使用できること．

P 3 前処理部によって作成されたファイル（図2中(D)）が PASCAL コンパイラの入力となり得ること．

P 4 必要なジョブ制御文が使用できること．

以下，これらの条件について個別に検討を行う．

3.2 言語仕様について

ここでは，条件P 1について検討を行う．移し換えの対象となった2つの PASCAL 処理系は，共に標準 PASCAL に準拠してはいるが，個々の計算機システムとの関わりなどのために実現法の相異や制限が存在する．また，拡張された機能も幾つかある．ここでは，移し換えの際直接問題となった相異について述べる．

（1） 計算機システムとの関わりから，ファイルの仕様が異なる．また，プログラム中のファイルと実際のファイルとを対応付ける方法が異なる．これらについては **3.3** および **3.4** 節で詳述する．

（2） 文字型について，標準関数 ord の値が異なる．PASCAL 8000 では，文字の EBCDIC コードをそのまま ord の値とする．したがって，このコード系で表わせる文字は英小文字も含めてすべて扱えるが，'A' から 'Z' までは必ずしも連続でない．これに対して，PASCAL 230 では，EBCDIC コードを変換して，文字型の要素数を 64 個にしている．そのため，英小文字など扱えない文字があるが，'A' から 'Z' までは連続している．

この相異のため，集合型に対する制限（要素数は 64 以下で，各要素の値は 0 から 63 の間になければならない．）が同じにも関わらず，PASCAL 8000 では文字型を基底型とする集合が扱えない*．

（3） パックされたデータに対する実現法が異なる．PASCAL 230 では，それを扱えない．ただし特別に，

alfa＝**packed array** [1..8] **of** char;

が標準型として設けられており，これに

表 1 変換後のソーステキストで用いる名前の一覧
Table 1 A list of identifiers used in the augmented source text.

IDENTIFIER	FUNCTION	TYPE
sys500pr	postprocessor	procedure
sys600cl	time measurement	integer
sys700ct	counters	array [1..2000] of integer
sys800in	control variable	integer
sys900fl	counter table	file of integer
sys910fl	source program	text
sys920fl	PASDAP list	text

* PASCAL 230 の最近の改訂版（東京工業大学情報科学科木村研究室による）では，文字型が PASCAL 8000 と同じものになったので，文字型を基底型とする集合が扱えない．

```
 1  PROGRAM FREQENCY(
 2                  SYS900FL:U18,SYS910FL:U17,SYS920FL:U16,
 3                  INPUT,OUTPUT);    (*W-  *)
 4    CONST AL = 8;   (* LENGTH OF WORD *)
 5    TYPE WORD = @WORDREF;
 6         WORDREF = RECORD NAME: ALFA; FREQ: INTEGER;
 7                          GREATER,LESS: WORD
 8                  END;
 9    VAR
10       SYS600CL: INTEGER; SYS910FL,SYS920FL: TEXT;
11           SYS700CT: ARRAY (.1.. 2000.) OF INTEGER;
12           SYS800IN: INTEGER; SYS900FL: FILE OF INTEGER;
13       ROOT: WORD;  (* ROOT OF BINARY TREE *)
14       WD: ALFA;    (* LAST WORD READ *)
15       CH: CHAR;    (* LAST CHARACTER READ *)
16       I,J: INTEGER;
17       A: ARRAY (.1..AL.) OF CHAR;
18
19
20  PROCEDURE SYS500PR(V,L,Y: INTEGER);  (* POSTPROCESSOR V=02,L=05 *)
21   VAR I,IX,LC,CDN,CNT: INTEGER; OV: BOOLEAN; CH: CHAR;
22   PROCEDURE HEADOUT;  (* HEADING PRINT *)
23    BEGIN PAGE(SYS920FL); WRITELN(SYS920FL);
24     WRITELN(SYS920FL,' ':4,'FACOM  230-45S  OS2',' ':15,'PASDAP',
25      ' ':7,'-',Y:6,'-',' ':5,'V=',(V DIV 10):1,(V MOD 10):1,'   L=',
26      (L DIV 10):1,(L MOD 10):1,' ':12,'DATE  ',DATE,'   TIME  ',TIME);
27     WRITELN(SYS920FL);
28     WRITELN(SYS920FL,' ':6,'SOURCE STATEMENT',' ':20,'EXECUTION TIME =',
29      SYS700CT(.1.):10,' MSEC.',' ':20,'EXECUTION PROFILE');
30     WRITELN(SYS920FL); WRITELN(SYS920FL) END;  (* HEADOUT *)
31   PROCEDURE COUNTOUT(IX: INTEGER);  (* COUNTER PRINT *)
32    BEGIN CNT := CNT + 1;
33     IF CNT > 4 THEN BEGIN WRITELN(SYS920FL,' I'); CNT := 1;
34       WRITE(SYS920FL,' ':80,'------->') END;
35     WRITE(SYS920FL,' I',SYS700CT(.IX.):8) END;  (* COUNTOUT *)
36  BEGIN RESET(SYS900FL); RESET(SYS910FL); REWRITE(SYS920FL,8,136,TRUE);
37   HEADOUT; CDN := 0; OV := FALSE;
38   REPEAT CDN := CDN + 1; CNT := 0; WRITE(SYS920FL,CDN:6,'  ');
39    FOR I := 1 TO 80 DO BEGIN READ(SYS910FL,CH); WRITE(SYS920FL,CH) END;
40    READLN(SYS910FL);
41    REPEAT IF EOF(SYS900FL) THEN IX := 0
42           ELSE BEGIN IX := SYS900FL@; GET(SYS900FL);
43            IF IX > 2000 THEN BEGIN
44              IF NOT OV THEN BEGIN OV := TRUE; LC := CDN END;
45              REPEAT IX := SYS900FL@; GET(SYS900FL) UNTIL IX = 0 END END;
46     IF IX > 0 THEN COUNTOUT(IX)
47    UNTIL IX = 0;
48    IF CNT > 0 THEN FOR I := CNT TO 3 DO WRITE(SYS920FL,' I',' ':8);
49    WRITELN(SYS920FL,' I');
50   UNTIL EOF(SYS910FL);
51   IF OV THEN BEGIN WRITELN(SYS920FL); WRITELN(SYS920FL);
52    WRITELN(SYS920FL,' *****  WARNING   COUNTER NUMBER OVERFLOW AT ',
53                     LC:5,'TH LINE, YOUR PROGRAM IS TOO LONG,') END;
54   WRITELN(SYS920FL); WRITELN(SYS920FL)
55  END;   (* POSTPROCESSOR FOR FACOM 230-45S OSII *)
56    PROCEDURE SEARCH(VAR W: WORD);
57      VAR W1: WORD;

     (OMITTED)

109  BEGIN
110       ; FOR SYS800IN := 1 TO  2000 DO SYS700CT(.SYS800IN.) := 0;
111       ; SYS600CL := CLOCK;
112
113                 ; SYS700CT(.  13.) := SYS700CT(.  13.) + 1;
114       ROOT := NIL; CH := ' ';
115    WHILE NOT EOF(INPUT) DO
116
117              BEGIN SYS700CT(.  14.) := SYS700CT(.  14.) + 1;
118      IF CH IN (.'A'..'Z'.) THEN
119      BEGIN
120                 ; SYS700CT(.  15.) := SYS700CT(.  15.) + 1;
121           I := 1;
122        REPEAT
123                 ; SYS700CT(.  16.) := SYS700CT(.  16.) + 1;
124              IF I <= AL THEN
125              BEGIN SYS700CT(.  17.) := SYS700CT(.  17.) + 1;
126                                A(.I.) := CH
127                 END
128                                    ;
129          READ(CH); I := I + 1
130        UNTIL NOT (CH IN (.'A'..'Z','0'..'9',.));

     (OMITTED)

150       ; SYS700CT(.1.) := (CLOCK - SYS600CL) DIV 1000;
151       ; SYS500PR( 2, 5, 781122);
152  END.
```

図 3 変換後の PASCAL ソーステキストの例
Fig. 3 An augmented PASCAL program list.

対して標準手続き pack, unpack がある．一方，PASCAL 8000 では，パックされたデータの使用が容易になっており，その要素に直接アクセスできる．このため pack, unpack は標準手続きとなっていない．

（4） 計測結果に後処理部で日付けや時刻を入れることにした(図1第1行目参照)．これらを得る手段の標準名は，どちらの処理系でも date と time であるが，PASCAL 230 ではこれらが関数であり，PASCAL 8000 では手続きとなっている． また，時間を測定する標準関数 clock の単位も異なっており，PASCAL 230 ではマイクロ秒，PASCAL 8000 ではミリ秒単位である．さらに，EDOSではシステムから提供される時間が秒単位である．

（5） PASCAL 8000 には，loop 文や forall 文などの拡張機能がある．PASDAP システムもこれらの機能に対処できるように改訂を行った．

前処理部と後処理部とは文字処理を中心としているため，特に（2），（3）のような相異が移し換えの際のプログラムの修正の大きな部分を占めた．

3.3 ファイル仕様について

ここでは条件 P2 について検討を行う．PASCAL では順次ファイルを扱うことができ，そのための標準手続きや標準関数が用意されている．ファイルは，外部ファイルと内部ファイルに分けることができるが，PASDAP で使用するファイル（図2中（A），（C）など）は，このシステムの構成から大記憶上の外部ファイルとした．(もっとも，内部ファイルを扱えるのは OS 7 のみである．)

各処理系における外部ファイルの仕様は，各計算機システムと大きな関わりをもっている．そして，どちらの処理系でも，外部ファイルを使用する際には，それに対応するジョブ制御文を必要とする．

PASCAL では，外部ファイルはプログラム頭部において宣言され，これによってプログラム中のファイルと実際のファイルが対応付けられる． PASCAL 8000 では， プログラム中のファイル変数名がそのままジョブ制御文のファイル定義名となる．一方，PASCAL 230では，ファイル変数名をプログラム頭部で宣言する際，

〈ファイル変数名〉:〈ファイル定義名〉

という形で，ファイル定義名との対応を宣言する必要がある．これは，OS II におけるファイル定義名が，システムが事前に決めた名前（たとえば U01～U50, SLIB など）に制限されていることによる．

実際のファイルの物理的属性（レコード長，ブロック長，レコード形式など）は，プログラムの変数宣言部で宣言されたファイルの型によって，各処理系が設定する．たとえば，PASCAL 8000では，ファイルの要素の大きさがレコード長に， レコード長の倍数で 2000 を超える最小の数がブロック長となる． PASCAL 230 では，1レコード1ブロックとなる．この物理的属性をユーザが自分で指定する方法は，各 OS によって異なる．OS IV, OS 7 や EDOS では，ジョブ制御文中に DCB（データセット制御ブロック） パラメータや FCB（ファイル制御ブロック） パラメータとしてレコード長，ブロック長，レコード形式などの情報を記述することができる．OS 7 や EDOS では，ジョブ制御文中にこれらを指定すると，それが有効となる．もちろん，ジョブ制御文で指定した属性がPASCAL 8000 で許される仕様に合わなければエラーとなる．ところが， OS IV では DCB パラメータをジョブ制御文中で指定しても有効でないので，処理系で設定した値を変更することができない．一方， OS II ではジョブ制御文中にレコード長，ブロック長などの情報を記述することができない． そこで，PASCAL 230 では標準手続き rewrite, reset の機能を拡張して，これらを指定できるようにしてある．すなわち，これらの手続きの引数によって，レコード長，1ブロック中のレコード数，text ファイルにおけるラインプリンタ用制御文字の有無などを指定できる．これらを指定する引数は無くてもよく，そのときは標準値が設定される．

PASDAP で使用するファイルの型は，図2中(C)が **file of** integer であり，ほかは text である． これらのファイルの物理的属性は，図2中（D）を除いてすべて処理系の設定する値で問題は生じなかった．(D)については次の **3.4** 節で述べる．

3.4 コンパイラの入力形式

ここでは条件 P 3 について検討を行う．PASDAP のような前処理部をもつシステムを移し換える際の難関の1つは，前処理部の出力するファイルを次のコンパイラへ渡す方法の問題である[1),2)]．

PASCAL コンパイラの入力となるファイルの型は，どちらの処理系でも text である．そこで，前処理部の出力ファイルの型も text にすればよさそうである．各 OS のもとで，各 PASCAL 処理系が，text 型のファイルに対して設定するレコード長，レコード形式を**表2**に示す．特に，PASCAL 8000では，標準ファ

```
$  JOB  PASDAP
$  PASDAP  *
 *** Source Program ***
$/
 *** Data ***
$ JEND
```

(1) OSII

```
//A123456  JOB
//     EXEC  PASDAP
//PDP.SYSIN  DD  *
 *** Source Program ***
//GO.SYSIN   DD  *
 *** Data ***
//
```

(3) OSIV/F4

```
//PASDAP: JOB
//PSOURCE
 *** Source Program ***
//PDATA
 *** Data ***
//PASDAP
//END
```

(3) OS7

```
//PASDAP  JOB
// CATLG  PASDAP
 *** Source Program ***
// CATLG  GOPASDAP
 *** Data ***
// END
```

(4) EDOS/MSO

図 4 JCL のマクロ化

Fig. 4 Macro JCL for 4 PASDAP systems.

イル output と同じ属性を持たせてあり，さらにEDOS では，output および input 以外の text 型のファイルが扱えない．

ところで，PASCAL 230 コンパイラは，入力 text ファイルの行が80桁以上になるとエラーとなるので，前述の rewrite 手続きの引数によって，レコード長を80 バイト，制御文字を無しと指定することにした．

一方 PASCAL 8000コンパイラは 112 桁以後に非空白文字があるとエラーになる．さらに行の先頭の制御文字の問題がある．OS7 では，ジョブ制御文中の FCB パラメータによってレコード形式を制御文字無しと指定した．一方 EDOS では，output ファイルのレコード形式として VBA（表2参照）以外を指定するとエラーとなった．しかし改行のための制御文字が空白であるため，一応そのままでコンパイラの入力となり得る．また OS IV では処理系がきめた標準属性を変更できないが，EDOSと同じ理由でそのままコンパイラの入力となりうる．

3.5 ジョブ制御文について

ここでは条件P 4 について検討を行う． PASDAP を実際に使用するためには，ジョブ制御文を用意せねばならぬ．今回の移し換えは，一般ユーザへの公開を考え，使用を容易にするために，ジョブ制御文のマクロ化も移し換えの一部として考慮した．各処理系において，ユーザは**図4**のような制御文の構成で容易に PASDAP を使用できる．

ジョブ制御文は，計算機システムに全面的に依存し，各 OS で異なっている．PASDAP の制御文を作成するに際しては， 各 OS で PASCAL を使用する場合の基本制御文の構成を参考にした．また，ファイルの受け渡しなどの問題には文献 1) が参考になった．

表 2 テキスト型ファイルの属性

Table 2 Record size and format of text file.

PASCAL SYSTEM	OS	RECORD SIZE	RECORD FORMAT
PASCAL 230	OSII	136	F
PASCAL 8000	OSIV	133	FBA
	OS7	133	VBA
	EDOS	133	VBA

F: fixed, V: variable, B: blocked
A: with control charactor

4. PASDAP の使用の実際

PASDAP を使用してプログラムを実行すると，必ずオーバヘッドが生じる．それは，実行時の実行回数計数のための cpu 時間の増加とプログラムサイズの増加に分けられる．

PASDAP では，プログラム全体の実行に要した時間を計測している．以下のプログラムについて各処理系の PASDAP による計測結果を**表3**に示す．比較のため PASDAP を使用しない通常の場合の数値も併記した．

（1） PASDAP: PASDAP の前処理部プログラム

（2） EXAMPL 1 : text ファイル中の英数字名を計数するプログラム(図1参照)

（3） EXAMPL 2 : PASCAL プログラムの手続きや関数のネスト構造を解析するプログラム

入力データとして，いずれも PASDAP 前処理部ソーステキストを使用した．表3から，計測によるオーバヘッドは 10～25% 程度である．

一方，プログラムサイズの増加の要素は2つあり，1つは実行回数計数用配列と後処理部手続きという固定的増加分である．もう1つは，計測のために挿入される命令で，その数はソースプログラムに依存する．各 PASCAL 処理系は，プログラムを実行する際に，ある固定した領域を確保し，その中にプログラムの領域とデータの領域とを割り当てる．したがって，PASDAPによるプログラムサイズの増加分だけ後者が減少することになる．ただし，プログラムを実行する際に確保する領域は，各処理系ともジョブ制御文で指定できる．

5. あ と が き

本稿では，主として PASDAP のような道具を多くの計算機システムの上で比較的容易に実現できること

表 3 PASDAP システム使用による CPU 時間の増加
Table 3 Comparison between CPU time with and without PASDAP system (unit: sec.).

OS		PASDAP	EXAMPL1	EXAMPL2
OSII	PASCAL	11.84	5.70	7.10
	PASDAP	13.91	7.18	8.81
OSIV	PASCAL	1.82	0.69	0.67
	PASDAP	1.98	0.73	0.77
OS7	PASCAL	4.18	1.35	1.37
	PASDAP	4.57	1.47	1.64
EDOS	PASCAL	29.00	14.00	*
	PASDAP	33.00	16.00	*

を示した．この作業に当ってFORDAP, COBOLDAPの移し換えの経験[1),2)]が役立った．これによって移し換えにおける問題点を3.1節に示した諸点に絞ることが可能となり，主として言語に特有な問題を解決することでPASDAPを移し換えることができた．冒頭に述べたように，移し換えの対象となった4つのOSのうち，OS Ⅳ, OS 7，EDOSで稼動中のPASCAL処理系はすべてPASCAL 8000である．そのため，一旦OS IIからOS IVへ移し換えた後，OS IVからOS 7やEDOSへの移し換えの問題は，ほとんどファイルの仕様とジョブ制御文のそれに限られた．

幾つかの問題も残っている．たとえば，OS IV上のPASDAPはプログラムの異常終了時にも計測結果を出力することができるので，デバッグなどにも有効なツールとなっている．ほかの処理系についてもこの機能を検討中であるが，これを実現するためにはPASCALモニタにある程度立入ることが必要となる．

実行回数計数機能はできればコンパイラの追加機能としたい．しかしメーカから提供されたFORTRANやCOBOLのコンパイラの中に使用者側が立入ることはほとんど不可能であり，FORDAPやCOBOLDAPの場合に前処理，後処理方式を採ったのは自然の成行きであった．PASCALコンパイラはPASCALで書いてあり，その中を見ることができる．しかし計数機能を追加したコンパイラをほかの処理系に移し換えることがそれほど容易とは考えられないので，前二者と同様に，前処理，後処理方式を採用した．この方式は現に必要としている機能だけに注目し，コンパイラの詳細を無視できるので作成すべきプログラムを小さくでき，したがって開発工程も短くてよい．なおこの機能をコンパイラに組込んだ場合でも，見やすい計測結果を作るために後処理部を追加する必要がある．

謝辞 移し換え作業全体を通じて，かつてFORDAPやCOBOLDAPなどの移し換えで得られた経験をもとにわれわれの研究室の藤村直美氏から多くの援助を得た．OS 7への移し換えでは，同氏のほかに東京大学情報科学科石畑清氏にお世話になった．EDOSでは，電気通信大学情報数理工学科の沼田一道，花田孝郎，木村友則の各氏にお世話になった．ここに記して謝意を表す．なお，PASCAL 230とその改訂版は，それぞれ東京大学和田英一教授，東京工業大学木村泉助教授のご好意により使用させていただいている．本研究の一部に日本情報処理開発協会第7回研究奨励金を使用した．

参考文献

1) 藤村，牛島：FORTRANプログラム動的解析システムの移し換えについて，情報処理，Vol. 17, No. 11, pp. 1048–1055 (1976).
2) 藤村，牛島：COBOLプログラム輪郭作成システムの移し換えについて，情報処理，Vol. 19, No. 9, pp. 853–859 (1978).
3) 武市，高橋：PASCAL処理系の概要，東京大学情報工学専門課程 (1976).
4) 牛島，江嶋：PASCALプログラム輪郭作成システム—PASDAP—の使用について，九州大学大型計算機センター広報，Vol. 11, No. 4, pp. 289–292 (1978).
5) 牛島，江嶋：同上，東京大学大型計算機センターニュース，Vol. 11, No. 2, pp. 20–24 (1979).
6) 疋田，安村，石畑：PASCAL入門—PASCAL 8000に沿って—，東京大学大型計算機センター (1977).
7) Jensen, K. and Wirth, W.: PASCAL user manual and report 2nd ed., Lecture Notes in Computer Science, No. 18, Springer Verlag, Berlin (1978).

(昭和54年4月23日受付)
(昭和55年9月18日採録)