3．実行回数計数ツールの実現に関する総括

九州大学 工学部　牛島和夫，藤村直美
Kazuo Ushijima　Naomi Fujimura

1．はじめに

プログラム中の各実行文の実行回数に関する情報は，プログラムの作成や改善，保守等に役立つ。筆者らは，1975年以来，Fortran，Cobol，Pascal，Ratfor，Snobol4で書かれたプログラムに対して，各文の実行回数を自動的に計数して，ソースプログラムとともに編集して出力するツール（輪郭作成装置と名付ける）を必要の都度作成し，使用してきた。このうち前三者に対するものは，多くの計算機システムに移し換えられている[1－7]。

最初Fortranに対して作成したこのようなツールを使用してみると，その効用は期待を大巾に超えるものであつた。その後Cobol，Pascal，Ratfor，Snobol4等の言語でプログラムを書いたり，それらを教育用に使用したりする必要が生じるたびに，それぞれの言語に対する最初のプログラミングツールとして，この装置を個々に実現して，現在に至つている。ここで五つの輪郭作成装置の実現において経験した問題点（とその解決策）等をまとめてみたい。それらは主として次のようなものである：

(1) 使い易さのための作成上の配慮，
(2) 対象となる言語固有の問題，言語に関わらぬ問題，
(3) 他計算機システムへの移し換えの問題。

プログラミングツールは使用されなければ意味がない。使用の実態等にも言及する。

2．実現の経過

2．1　Fordap

プログラム中の基本ブロックを節点とし，制御の移動を枝とするグラフと見たてて，プログラムを静的あるいは動的に解析する手法はよく知られている。Russellらは Fortran プログラムの基本ブロック毎に

```
CALL EMIT (i)
```

という文を自動的に挿入する方法を与えた[8]。iは節点の番号である。EMITルーチンは直前に呼ばれた時の節点番号を憶えていて，節点iから節点jへの推移回数が計数できる。この手法を用いて，プログラムの動作解析を試みたことがある[9]。しかしEMITルーチンの呼出し回数が多くなると，計測のための実行時間が計測をしない場合の10～20倍もかかるようなこともあつて実用にならず，その仕事は中断されていた。

Knuthの研究[10]は，節点間の推移回数ではなくて，節点の実行回数を数えるだけで，十分有用な情報が得られることを教えている。すなわち，実行回数を計数するだけであれば，

```
COUNT (i) = COUNT (i) ＋ 1
```

というカウンタを基本ブロックごとに挿入すればよい。枝（節点から節点へ）に関する情報が収集できないことも生じるが，CALL文に比べて計測によるオーバヘッドは軽くなる。1974年度の卒業研究で，カウンタ挿入方式による実行回数計数装置を試作してもらうことにした。試作品を試用してみると，プログラミングツールとして予期以上の効用があり，早速学生演習で使用するとともにプログラミングツール（Fordapと名付ける）として本格的に作成することにした[1]。

Knuthの論文[10]には，Fortran文の静的及び動的使用頻度の調査結果が報告されており，この種のデータがほとんどまとめられていなかつた当時にあつて，我国では，コンパイラ作成者やハードウエア設計者によつてこの論文はしばしば引用されていた。しかし，個々のプログラム作成を援助するツールとしての実行回数（Knuthはこれをprofile＝輪郭と名付けた）計数装置への注目はほとんどなされてなかつたように思う。

プログラミングツールとして本格的に作成するに当つて次のような設計方針をたてた：

(1) 計測情報はわかりやすい形式に編集して出力すること，
(2) デバッグ支援のため，異常終了時にも可能な限りそれまで得られた計測情報を上記にならつて出力すること，
(3) 他の計算機システムに容易に移し換えできるようにプログラムを作成すること，そのため，記述は可能な限りJIS FORTRAN水準7000の範

囲で行うこと，

(4) プログラムを簡潔にまとめること，そのために，実行コストの算定を容易にすること，計測の対象となるプログラムは，一旦コンパイラによつて文法的誤りのないことが確認されたものとすること。

東大大型計算機センターの発足以来，(Fortranで書かれた) プログラムの移し換えの問題に，九大中央計数施設の一員として，その後センターのユーザとして関心を持つており，移換性の高いプログラムを作成するためのデータやノーハウを種々蓄積することに努めていた。Fordapは，図1に示すように，多段のジョブステップからなり，その間で多数のファイルの受け渡しを要するシステムであり，このようなシステムを全体として，複数の計算機システムに移し換えて実現するという計画を遂行することによつて，移し換えに関する知見をさらに加えようと目論んだ[2] 。上記方針(4)は，方針(3)に鑑みて個々の手続きをできる限り簡単にしておこうとしたものである。

Fordapを実際に使用して，これが，プログラムの各実行文の回数を計数するという極めて単純な原理に基くものでありながら，このシステムがプログラムの能率改善ばかりでなく，デバッグやテストを助ける手段として，あるいは，プログラムの読み易さを助ける補助手段として有効なのは，主として，解くべき問題に関りなく実行回数を通して，プログラムを客観的に眺められること，解析結果の出力情報が多すぎず，ソースプログラムと得られた計測情報との対応付けが容易であることのためと思われる（図2参照）。

2．2 Coboldap

Cobolでプログラムを作る必要に迫られたときに，Cobolプログラミングの最初のツールとして，Cobolプログラム輪郭作成システム（Coboldapと名付ける）を先ず作ることにしたのは，Fordapの使用経験による。Fordapの場合と同じく，設計方針の一つとして，多くの計算機システムの上に移し換え可能とすることを掲げた。

Coboldap全体の構成はFordapとほとんど同一とした。手続きの大部分をFortranで記述したFordapの移し換えにおける問題は，大部分OSとの整合にあつた。手続きの大部分をCobolで書いたCoboldapを移し換えることによつて，移し換えの問題を言語に依存する部分と依存しない部分に分離できることを期待した。依存する部分を最小

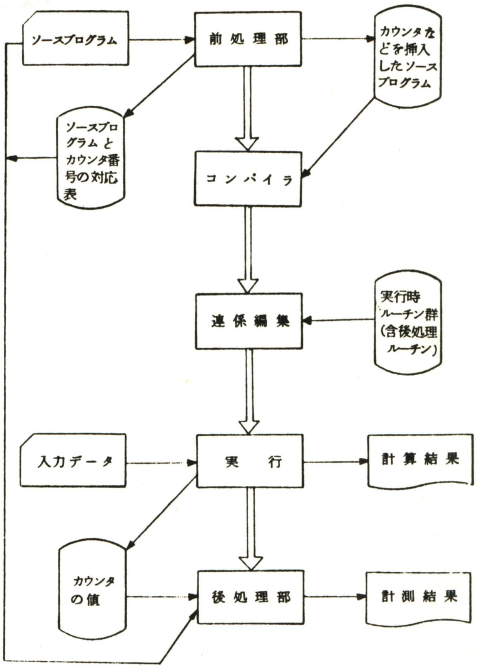

図1 システム構成の概要

限にとどめるために，当初移し換えを予定した四つの計算機システムのCobolマニュアルで予約語の調査を行い，前処理部で必要な予約語表には，四つの処理系の予約語のうち必要なものを全て登録しておいた。しかし，Cobolは，手続き部以外にプログラム中に使用環境を書き込む部分が多く，移し換えに際して，Fortranに比べ，修正箇所が多くなる[3，4]。

Cobolには，SEARCH，SORTなど大型の命令があり，簡単な代入文も，このような命令も重みなしに一律に1と評価するのは問題がないわけではない。Knuthがいうように「I/Oバウンドでないプログラムでは，実行時間の大きな部分が書かれたプログラムテキストのわずか数%に集中している。」したがつて，その集中箇所を発見するというのが，このツールを導入した大きな理由であつた。Cobolプログラムは，さまざまなファイルを対象とする入出力を多く伴う処理を得意とし，むしろI/Oバウンドのプログラムの方が多いといつてもよい。実行回数情報は，能率改善よりはむしろ，作成中のプログラムに対しては，テスト援助用として，本番用のプログラムに対しては，その

```
* SOURCE STATEMENT *                EXECUTION TIME =        290 MSEC          EXECUTIONS          TRUE

      SUBROUTINE BUFCLR                                                00000750          28
C                                                                      00000760
      INTEGER   TXTBUF(56,121),SEQBUF(121)                             00000770
      INTEGER   BUFCNT,BUFLMT,SEQCNT,LINLMT,PAGCNT                     00000780
      INTEGER   SPACE,NUM(10)                                          00000790
C                                                                      00000800
      COMMON /BUFFER/TXTBUF,SEQBUF                                     00000810
      COMMON /CLUNTR/BUFCNT,BUFLMT,SEQCNT,LINLMT,PAGCNT                00000820
      COMMON /CHAR  /SPACE,NUM                                         00000830
C                                                                      00000840
      DO 200 I=1,BUFLMT                                                00000850          28
         SEQBUF(I)=0                                                   00000860        3388
         DO 100 J=1,56                                                 00000870        3388
            TXTBUF(J,I)=SPACE                                          00000880      189728
  100    CONTINUE                                                      00000890      189728
  200 CONTINUE                                                         00000900        3388
C                                                                      00000910
      RETURN                                                           00000920          28
C                                                                      00000930
      END                                                              00000940

* SOURCE STATEMENT *                EXECUTION TIME =        481 MSEC          EXECUTIONS          TRUE

      SUBROUTINE MOVE(IN,OUT,N)                                        00000950        3192
C                                                                      00000960
      INTEGER   IN(N),OUT(N)                                           00000970
C                                                                      00000980
      DO 100 I=1,N                                                     00000990        3192
         OUT(I)=IN(I)                                                  00001000      159992
  100 CONTINUE                                                         00001010      159992
C                                                                      00001020
      RETURN                                                           00001030        3192
C                                                                      00001040
      END                                                              00001050
```

図２　Fordap 出力例

```
     FACOM    OSIV/F4       COBOLDAP    V-04  L-01    -791009-     DATE 79.11.06    TIME 10:02:12   PAGE 0001
LINE   SEQ.  A.  B.                                                          IDENT.    EXECUTIONS

0001  000010*    THIS IS A PROGRAM WHICH PRINTS A SOURCE-FILE
0002  000020*    IN A STINGY FORMAT.
0003  000030 IDENTIFICATION    DIVISION.                                                      1
0004  000040 PROGRAM-ID.       PRINT.
0005  000050 AUTHOR.           NAOMI FUJIMURA.
                                    (Omitted)
0084  000840 PROCEDURE         DIVISION.                                                      1
0085  000850 MAIN              SECTION.                                                       1
0086  000860*
0087  000870 BEGIN.                                                                           1
0088  000880     PERFORM SHOKI-SETTEI.                                                       1
0089  000890     PERFORM TEXT-EDIT UNTIL EOF-SW = 1.                                         1
0090  000900     PERFORM ATO-SHIMATSU.                                                       1
0091  000910*
0092  000920 SUBR              SECTION.                                                       0
                                    (Omitted)
0103  001030 TEXT-EDIT.                                                                    2724
0104  001040     READ SOURCE-FILE INTO CARD AT END MOVE 1 TO EOF-SW.                        2724          1  (AT END)
0105  001050     IF   EOF-SW NOT = 1  PERFORM TEXT-PRINT.                                   2724       2723
0106  001060*
0107  001070 TEXT-PRINT.                                                                   2723
0108  001080     IF   BUF-COUNT > BUF-LIMIT - 2  PERFORM LP-1-PAGE-PRINT.                   2723         26
0109  001090     ADD 1 TO BUF-COUNT.                                                        2723
0110  001100     ADD 1 TO LINE-COUNT.                                                       2723
0111  001110     MOVE LINE-COUNT TO LINE-NO (BUF-COUNT).                                    2723
0112  001120     MOVE TEXT-LEFT TO TEXT-56 (BUF-COUNT).                                     2723
0113  001130     IF   TEXT-RIGHT NOT = SPACE                                                2723
0114  001140          ADD 1 TO BUF-COUNT                                                     469
0115  001150          MOVE TEXT-RIGHT TO TEXT-56 (BUF-COUNT).                                469
0116  001160*
                                    (Omitted)
0131  001310 LP-1-GYO-PRINT.                                                               1647
0132  001320     MOVE SPACE TO LP-1-GYO.                                                    1647
0133  001330     COMPUTE SOEJIR = SOEJIL + 60.                                              1647
0134  001340     IF   SOEJIL < 61                                                           1647
0135  001350          MOVE TEXT-56 (SOEJIL) TO TEXT-L                                       1620
0136  001360          IF   LINE-NO (SOEJIL) IS POSITIVE                                     1620
0137  001370               MOVE LINE-NO (SOEJIL) TO LINE-L.                                 1374
0138  001380     MOVE TEXT-56 (SOEJIR) TO TEXT-R.                                           1647
0139  001390     IF   LINE-NO (SOEJIR) IS POSITIVE                                          1647
0140  001400          MOVE LINE-NO (SOEJIR) TO LINE-R.                                      1349
0141  001410     WRITE LP FROM LP-1-GYO AFTER ADVANCING 1 LINES.                            1647
0142  001420*
0143  001430 LP-1-PAGE-CLEAR.                                                              3388
0144  001440     MOVE ZERO TO LINE-NO (BUF-COUNT).                                          3388
0145  001450     MOVE SPACE TO TEXT-56 (BUF-COUNT).                                         3388
0146  001460*
0147  001470 ATO-SHIMATSU.                                                                    1
0148  001480     IF   BUF-COUNT IS POSITIVE  PERFORM LP-1-PAGE-PRINT.                        1          1
0149  001490     CLOSE SOURCE-FILE.                                                         1
0150  001500     CLOSE PRINT-FILE.                                                          1
0151  001510     STOP RUN.                                                                  1
```

図３　Coboldap 出力例

入力データや出力ファイルの性質を記述したメモとして，非常に便利であることを実感している（図３参照）。これについては，第４節で再述しよう。

２．３　Pasdap

1977年度から，筆者の担当するプログラミング入門の講義で，対象言語をFortranからPascalに切換えた。それに合せて，Pascalプログラム輪郭作成システム（Pasdapと称する）をFACOM 230−45S上に実現した。設計に当つて考慮した点は，Fordap，Coboldapの場合と同じである。実行回数計数機能は，できれば，コンパイラが標準としてもつていてほしい。メーカから提供されるFortranやCobolのコンパイラの中に，使用者が立入ることは，ほとんど不可能であり，これらの場合に，前処理，後処理方式（図１参照）を採つたのは自然の成行きであつた。Pascalのコンパイラは Pascalによつて書かれており，その中を見ることができるので，コンパイラに，実行回数計数機能を追加することを考えた。しかし，コンパイラの成立ちの解明を待つて作成にとりかかつていたのでは，使用に間に合わないこと，計数機能を組込んだコンパイラを他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こと，の二つの理由から，前二者と同様の前処理，後処理方式を採用することにした。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してよいので，作成すべきプログラムの大きさを小さくでき，したがつて開発工程も短くてよい。

Pascalには，プログラム単位の結合という概念がない。FordapやCoboldapの場合のように，あらかじめコンパイルした目的プログラム（後処理ルーチン等）を連係編集時に組み込むこと（図１参照）が不可能である。そこで，Pascalで書かれた後処理手続きをソーステキストの形でファイル上に用意しておき，被計測プログラムの宣言部の次に複写して挿入することにした。したがつて，後処理手続きは計測のたびにコンパイルされる。このため後処理部をできるだけ簡潔に構成することに留意した。

このPasdapは，東工大情報科学科で使用してもらつたが，コンパイラの詳細に通じている佐渡一広氏によつて，ユーザの指定によつて，カウンタを要所に機械命令として挿入するようにコンパイラ（Pascal 230）に拡張が加えられた。この場合でも，実行回数とソースプログラムの対応をとつて編集するために後処理部が必要である。一方，東大および九大の大型計算機センターに移し換えられたPasdap（いずれもPascal 8000）は，前処理，後処理方式のままで動いている[5]（図４参照）。

２．４　Ratdap

Ratforのような前処理型言語によるプログラムは，ホスト言語（Fortran）に変換されたあと，コンパイル時や実行時にエラーが発見されても，変換されたあとのFortranプログラム（の行番号）に対して，エラーメッセージが出される。したがつて，もとのRatforプログラムとの対応をとるのが意外に面倒である。OSとの整合は，Fordapと同じなので，異常終了時に出力されるRatdapによる実行回数情報を調べて，異常終了箇所をRatforで書いたプログラムの上で発見することが可能になる[6]。

Ratforの前処理部はRatforで書かれており，Pascalコンパイラよりもずつと単純である。Ratfor文をFortran文に変換する過程でついでにカウンタを挿入することが容易かもしれない。それにもかかわらず，さらにRatdap前処理部を付け加えることにした。これも開発工程を単純にしたいためであつた。Ratforにもプログラミング支援ツールが出来たので，この助けを借りて必要ならば，Ratfor前処理部に計数機能を追加することを考えてもよいだろう。

２．５　Snobol４

原著書からテープの形で入手したOS/360用のSnobol４ macro implementationを，九大大型計算機センターのFACOM M−190 OS Ⅳ/F4のもとに移植する作業と並行して，Snobol４に実行回数計数機能を追加する作業を行つた[7]。Snobol４プログラミングを援助するツールが一つもないことのほか，角田博保氏（東工大）がFACOM 230−45S用に作成したSnobol３（Profile機能付）[11]をよく使つていて，このような言語にも，この機能が有用であることを痛感していたことによる。

このSnobol４処理系[12]は，トランスレータとインタプリータからなり，Snobol４で書かれたソースプログラムを前者がプレフイクスコードに変換し，それを後者が解釈実行する仕組となつている。システムの記述言語はSIL（Snobol Implementation Language）と呼ばれる，マクロ定義の集合をもとにした言語である。計算機システ

```
FACOM    OSIV/F4                PASDAP       -790320-      V-03   L-02              DATE  79-11-06   TIME  11:40:09

 SOURCE STATEMENT                        EXECUTION TIME =      1195 MSEC.                    EXECUTION PROFILE

 1  (*$S+   STINGY PRINTER  *)                                                    00000010 |
 2  PROGRAM STINGY(SOURCE,OUTPUT);                                               00000020 |
 3    TYPE ALFA4 = ARRAY (.1..4.) OF CHAR;                                       00000030 |
 4         ALFA64 = PACKED ARRAY (.1..64.) OF CHAR;                              00000040 |
 5         ALFA80 = PACKED ARRAY (.1..80.) OF CHAR;                              00000050 |
 6    VAR BUFF, BB: ARRAY (.1..122.) OF ALFA64;                                  00000060 |
 7        DT: PACKED ARRAY (.1..32.) OF CHAR;                                    00000070 |
 8        A: ALFA80;                                                             00000080 |
 9        LC, L, P: INTEGER;                                                     00000090 |
10        LE, EB, EC: INTEGER;                                                   00000100 |
11        F: BOOLEAN;                                                            00000110 |
12        LCH: ALFA4;                                                            00000120 |
13        I, J: INTEGER;                                                         00000130 |
14        DC, TC: ARRAY (.1..8.) OF CHAR;                                        00000140 |
15        SOURCE: FILE OF ALFA80;                                                00000150 |
16    VALUE DC := (£' ',' ',' ',' ','D','A','T','E',' '£);                      00000160 |
17          TC := (£' ',' ',' ',' ','T','I','M','E',' '£);                      00000170 |

                                    (Omitted)

54  BEGIN P := 1; LC := 0; DATETIME;                                            00000540 |      1 |      1 |      1 |        |
55    FOR I := 1 TO 122 DO                                                      00000550 |      1 |        |        |        |
56      FOR J := 1 TO 64 DO BB(.I,J.) := ' ';                                   00000560 |    122 |   7808 |        |        |
57    RESET(SOURCE);                                                            00000570 |      1 |        |        |        |
58    WHILE NOT EOF(SOURCE) DO                                                  00000580 |      1 |        |        |        |
59    BEGIN L := 0; BUFF := BB;                                                 00000590 |     27 |     27 |        |        |
60      WHILE (L < 121) AND NOT EOF(SOURCE) DO                                  00000600 |     27 |        |        |        |
61      BEGIN  A := SOURCE&; GET(SOURCE);                                       00000610 |   2723 |   2723 |        |        |
62        IF L <> 60 THEN L := L + 1 ELSE L := L + 2;                           00000620 |   2723 |   2700 |     23 |        |
63        LC := LC + 1; INTTOCH(LC, LCH);                                       00000630 |   2723 |   2723 |        |        |
64        FOR I := 2 TO 5 DO BUFF(.L,I.) := LCH(.I-1.);                         00000640 |   2723 |  10892 |        |        |
65        EB := 72; F := FALSE;                                                 00000650 |   2723 |   2723 |        |        |
66        WHILE NOT F AND (EB > 0) DO                                           00000660 |   2723 |        |        |        |
67          IF A(.EB.) = ' ' THEN EB := PRED(EB) ELSE F := TRUE;                00000670 | 120422 | 117699 |   2723 |        |
68        IF EB > 58 THEN                                                       00000680 |   2723 |        |        |        |
69        BEGIN EC := 58;                                                       00000690 |    433 |        |        |        |
70          WHILE ((A(.EC.) <> ' ') AND (A(.EC.) <> ',')                        00000700 |    433 |        |        |        |
71            AND (A(.EC.) <> ';')) AND ((EB-EC) < 58) DO EC := PRED(EC);       00000710 |    866 |        |        |        |
72          IF (EB-EC) >= 58 THEN EC := 58;                                     00000720 |    433 |      0 |        |        |
73          FOR I := 1 TO EC DO BUFF(.L,I+6.) := A(.I.);                        00000730 |    433 |  24248 |        |        |
74          L := L + 1; J := 64;                                                00000740 |    433 |    433 |        |        |
75          FOR I:= EB DOWNTO EC+1 DO                                           00000750 |    433 |        |        |        |
76          BEGIN BUFF(.L,J.) := A(.I.); J := PRED(J)                           00000760 |   5143 |   5143 |        |        |
77          END                                                                 00000770 |
78        END                                                                   00000780 |
79        ELSE FOR I := 1 TO EB DO BUFF(.L,I+6.) := A(.I.)                      00000790 |   2290 |  48966 |        |        |
80      END;                                                                    0000080  |
81      HEADOUT;                                                                00000810 |     27 |        |        |        |
82      FOR I := 1 TO 61 DO WRITELN(BUFF(.I.), ' ':4, BUFF(.I+61.));           00000820 |     27 |   1647 |        |        |
83    END                                                                       00000830 |
84  END.                                                                        00000840 |      1 |        |        |        |
```

図 4　Pasdap 出力例

```
FACOM OSIV/F4 SNOBOL4(BELL TELEPHONE LABORATORIES) -750519-  VERSION 3.11(PROFILE-790327-) DATE 79/10/31    TIME 14:12:30

NO.                  SOURCE  STATEMENT                                                EXECUTION     SUCCESS     FAILURE

1             INPUT('INPUT',27,72)                                          00000010          1           1           0
2             OUTPUT('TITLE',6,'(1H1/1X,136A1)')                            00000020          1           1           0
3             DEFINE('MAKE()')                                              00000030          1           1           0
4             DEFINE('OUT()')                                               00000040          1           1           0
      *                                                                     00000050
5             HEAD = '            FACOM   OSIV/F4     STINGY   PRINTER'     00000060          1           1           0
5     +                '      PAGE '                                        00000070
6             PEAD = '   PAGE '                                             00000080          1           1           0
7             DAEH = '     DATE  ' DATE() '     TIME  ' RTIME()            00000090          1           1           0
8             LINE = ARRAY(61)  ;                                                    (a)      1           1           0
9                                  PAGE = 1  ;                                                1           1           0
10                                            &TRIM = 1                     00000100          1           1           0
11    INIT    MAKE()                                    :F(EOF)             00000110    (b)  27          27          0
12            MAKE()                                    :F(EOF)             00000120         27          26          1
13            OUT()                                     :(INIT)             00000130         26          26          0
14    EOF     OUT()                                     :(END)              00000140          1           1           0
      *                                                                     00000150
15    MAKE    I = 1                                                         00000160    (c)  54          54          0
16    MKLP    TEXT = INPUT                              :F(FRETURN)         00000170        2724        2723          1
17            NO = NO + 1                                                   00000180    (d) 2723        2723          0
18            LE(SIZE(TEXT),56)                         :F(LONG)            00000190        2723        2254        469
19            LINE<I> = LINE<I> DUPL('0',4 - SIZE(NO)) NO ' '               00000200        2254        2254          0
19    +                 TEXT DUPL(' ',66 - SIZE(TEXT))                      00000210
20            I = LT(I,60) I + 1                        :S(MKLP)            00000220        2254        2216         38
21            LINE<61> = LINE<61> DUPL(' ',71)          :(RETURN)           00000230          38          38          0
22    LONG    TEXT LEN(56) . FTX =                                          00000240         469         469          0
23            LINE<I> = LINE<I> DUPL('0',4 - SIZE(NO)) NO ' ' FTX DUPL(' ',10) 00000250       469         469          0
24          LINE<I + 1> = LINE<I + 1> DUPL(' ',61 - SIZE(TEXT)) TEXT DUPL(' ',10) 00000260   469         469          0
25            I = LT(I,59) I + 2                        :S(MKLP)            00000270    (e)  469         454         15
26            LINE<61> = NE(I,60) LINE<61> DUPL(' ',71)  :(RETURN)          00000280         15          10          5
      *                                                                     00000290
27    OUT     TITLE = HEAD PAGE DUPL(' ',8 -  IZE(PAGE)) PEAD PAGE + 1      00000300         27          27          0
27    +               DUPL(' ',4 - SIZE(PAGE + 1)) DAEH                     00000310
28            OUTPUT =                                                      00000320         27          27          0
29            PAGE = PAGE + 2                                               00000330         27          27          0
30            K = 1                                                         00000340         27          27          0
31    OTLP    OUTPUT = TRIM(LINE<K>)                    :F(RETURN)          00000350       1674        1647         27
32            LINE<K> =                                                     00000360       1647        1647          0
33            K = K + 1                                 :(OTLP)             00000370       1647        1647          0
34    END                                                                   00000380          1           1           0
NO ERRORS DETECTED IN SOURCE PROGRAM
```

図 5　Snobol 4 輪郭付き出力例

ムが変われはマクロ定義を変更すればよい。これによつて，Snobol 4処理系をポータブルにしている。しかしこのため実行速度が非常に遅い。実行回数計数機能を実現するために，これまでの場合と同様に前処理方式を採つて，各文ごとに

COUNT (i) = COUNT (i) + 1

のようなカウンタを挿入することにすると，そのカウンタが，トランスレータによつてブレフイクスコードに変換され，それが解釈実行されることになるから，ブレフイクスコードの占有場所の増加もさることながら，計数のための時間の増加が著しい。Snobol 4文は実行の結果，ほとんどの場合，成功・失敗を弁別することができるので，その回数も計数しておきたい。すると1文あたり，カウンタを2個挿入することになつて，時間，容量の増加はさらに大きくなる。

OS/360用Snobol 4をOS IV/F 4のもとで動かすために，システムの内容をかなり詳しく読まねばならなかつたこと，説明文書[12]が完備していたことの二つの理由から，トランスレータとインタプリータの拡張という形をとることにした(SILで記述，アセンブリ語によるマクロ定義を二つ追加)。トランスレータを拡張して，ブレフイクスコードの中に実行カウンタと成功カウンタをブレフイクスコードの形式で挿入する手続きを付け加え，インタプリータ中には，それらのコードを解釈実行する関数を用意した。実行終了後，計測情報とソースプログラムをつき合せて出力する後処理部等を含めて，SIL言語で280行(データ部を含む)程度の追加修正で，実行回数計数機能をSnobol 4処理系に追加することができた。この機能を使用した場合の実行時間の増加は，5～10％程度で，ほとんど問題にならない（図5参照）。

3．実現における問題点

3．1　前処理部

カウンタの挿入　前処理部が文の種類を識別して，それぞれにカウンタを挿入する規則は極めて簡単である。ここでは，if文について，五つのツールを比較してみよう。

Fordapでは，論理IFが真となつた回数を数えることにした。Fortranに複合文を作る道具立てが用意されていれば，

IF（条件）実行文

は

IF（条件）{COUNT(i)=COUNT(i)+1；実行文}

とすればよい。そうはできないので，

B = 条件
IF (B)　COUNT(i)= COUNT(i)+1
IF (B)　実行文

のように変換している[1]。

Cobolでも，複合文を作る道具立てが陽にあるわけではないが，

IF 条件　命令－1　ELSE 命令－2

は

IF 条件　ADD 1 TO COUNT(i) 命令－1
ELSE　ADD 1 TO COUNT(i') 命令－2

また，条件設定を伴う命令

条件命令～{END / SIZE ERROR / 等} 無条件命令.

は，

条件命令～{END / 等} ADD 1 TO COUNT(i)
無条件命令.

と展開している[3]。

Pasdap, Ratdapでは，begin end，あるいは，{ }を利用することができるので特別な配慮はいらない。

Snobol 4の場合，条件文に相当する処置は，ブレフイクスコード中にトランスレータが出力する失敗オフセットをインタプリータが解釈することによつて行われるので，1文ごとに，実行カウンタと成功カウンタとをブレフイクスコードの形で機械的に出力しさえすればよい。

カウンタの能率　Snobol 4を除いて，カウンタをソース言語の形で挿入しているが，Coboldapにおいて，実行能率上の問題が生じた。Coboldapを最初に作成したFACOM 230－45S OS IIにおいて，カウンタの属性を2進数としておいた。これは，計数のための文が，そのまま2進数のload，add，store命令にコンパイルされると考えたからである。しかし実際に使用してみると，実行回数計数によるCPU時間の増加が著しくツールとして使用に耐えない。原因はOS II Cobolでは，2進数の加算を副プログラムで内部10進数に変換して行い結果を再び2進数にもどしているためとわかつた。そこでカウンタの属性を内部10進数に改めて能率の問題を解消できた[3]。

Coboldapを他の計算機システムに移し換えるに先立つて，カウンタとして最も速いデータ属性がどれかを知る調査を行つているが，現在のとこ

ろ，移し換えを行つたいずれの処理系でも，内部10進数を採用する結果となつている。

カウンタの上限　処理を簡単にするため各前処理部は1－パス方式をとつている。カウンタ用配列の宣言を，プログラムの先頭で行うので，配列の寸法を超えるような個数のカウンタを挿入せねばならぬような大きなプログラムが入力された場合は，上限を拡大して再試行せねばならぬ。

Snobol 4では，文の数×2個のカウンタを必要とするが，このカウンタはソース言語で書かれているわけではないので個数の制限はない。解釈実行に入る直前にカウンタ用のデータ領域を，別に確保する。

記述言語　Fordapの前処理部は，Cobolで書いてもよいし，Pascal やアセンブリ言語で書いてもよい。しかし，Snobol 4を除いて，他の四つの前処理部は全て，それぞれが処理の対象とするプログラムと同じ言語で書いた。それは，多くの計算機システムへの移し換えを設計方針の一つとしたからであり，これらツールの移し換え先には，それぞれの言語のコンパイラが必ずあるはずだからである。

これらのツールの作成途中の検査あるいは完成検査を行うためのデータ，すなわち，それぞれの言語で書かれたプログラムとそれに対する入力データの数が，Fortranプログラムを除いて，極めて貧弱（あるいは皆無）だつたので，それぞれの言語で書いた前処理部プログラムが，テストデータとして極めて有用であつた。大きさもそれぞれ

Fordap	1639行
Coboldap	933行
Pasdap	494行
Ratdap	1201行

と適当で，テストデータの内容をテスト者が熟知していることも有益であつた。さらにテストの結果，前処理部自身の実行回数情報が得られることによつて未テスト部分が明らかになり，非能率部分も発見できた。それによつて改善した版に同じデータを入力することによつて，改善前の版との対応部分の実行回数の一致を調べることにより改善のための変更の正しさを確認することも容易に行うことができた。

制限　CobolのCOPY命令やRatforのINCLUDE文によつて組み込まれるテキストを，前処理部は走査しない。したがつて，そこに実行文が含まれていても実行回数は計数されない。さらに，もしその中にSTOP命令等が含まれていて，実行がその命令に到達することがあると，計測情報が出力されないまま終つてしまうことになる。

FordapやRatdapでは，プログラム単位ごとに，ソーステキストを計測の対象からはずすことができる。その場合，その中にSTOP文を含み，計測実施時にそこに制御が到達することがあると，計測情報が出力されないのは，上と同じ理由による。

実行時翻訳機能として，Snobol 4に組込まれているCODE関数によつて実行時に生成されたプレフイクスコードにもカウンタは挿入されない。これは実行に入る直前にカウンタの個数を確定してしまうためである。

3．2　使い易さへの配慮

ツールを使い易いものにするために簡単なコマンド言語（ジョブ制御文）で操作可能なこと，出力結果が理解しやすいこと，異常終了時にも結果が得られることがあげられる。

計測結果の表示　ある種のデバッグ用ツールや静的解析ツールでは，解析結果を整理が不充分なまま山のように出力するものがあつて，使用者が本当に必要な情報を探し出すのに苦労する。トレーサなどがその例で，必要箇所に行きつく前に出力過多で打切りということになることもある。適当な出力量を保つことは使いやすさの一つの指針である。

コンパイラの中には，実行回数計数機能をオプションとしてもつものもないわけではない。あるコンパイラについてそのような機能を使用したときの出力例を図6に示す。図2と同じFortranプログラムを同じ入力データで走らせた結果である。ここでLABEL 欄は，コンパイラがソーステキストを走査して，内部的に生成した節点番号と，ソーステキスト中にもともと存在する文番号とを示している。COUNT欄が実行回数である。内部的に生成した番号を，利用者が事前に知つているわけではないから，図6をもとに，プログラムの解析を行おうとすると内部番号とソーステキストの対応表が別に必要になる。利用者は，この対応表と実行回数とのつき合せ作業を余儀なくされる。このような作業は計算機にとつて簡単でも人間には苦痛である。図2～図5のようにソーステキストのすぐ右側に実行回数が編集されて付加されているものに比べ見易さ，使い易さの点で格段の差がある。

最初に作成したFordapの出力表示では，利用

者が与えたソーステキストの原形をそのまま保つことを第一とした。バッチ処理全盛の時代であつたから，出力はラインプリンタを想定し，左右のバランスを考慮して，実行回数情報を対応する実行文のすぐ右側に置くことにした。原則として基本ブロックごとにカウンタを挿入しているので，同じ実行回数が続く一連の実行文を一括して捉える助けとなり，ある種の段付けの効果すらあがることを経験した。学生演習のレポートには，必ず輪郭情報付きのリストを提出することを義務付けたが，教師によるプログラムの点検作業や，試問の際に問題点を指摘する手掛りとして役立つている。

Fortran 以外の言語では，1行に複数個の文が記述できる。それらの実行回数がつねに同一であるとは期待できない。ソーステキストの原形をくずさないという方針で，このような場合には，実行回数欄を横に継ぎ足す形にしている（図3 Coboldap ,図4 Pasdap 参照）。1行が5文以上になるようなことはほとんどないが，そんなときは，次の行に5番目以降を折り返して出力することにしている（Pasdap の場合。Coboldap では3番目以降，Ratdap では4番目以降）。

縦の欄に実行回数が並んでいると人間の眼でチエツクしやすい。Pascal や Ratfor では，1行に複数個の文を書くことがしばしばある。そのような行については，実行回数欄を横に目を走らせるたびにチエツク作業を中断されるような感をもつ。

Snobol4では，実行，成功，失敗の三つの回数を出力することにした，複数個の文に対する回数をさらに右側に並べる紙面の余裕もないこともあり，1行に複数個の文があれば，それを1文1行に分割して表示している（図5(a)参照）。

入力されたソーステキストに多少の位置の移動があつてもこだわらないようになつたのは，一つはTSS環境で計算機が使用できるようになつたことと無関係でないかもしれない。大容量ファイル上にあるソーステキストをエデイタや，プリテイプリンタを介してのぞくことに慣れてくると原形にこだわる必要が少なくなる。

TSS環境下でこれらのツールを使用する場合，本来1行120桁を超えるラインプリンタ用に設計されている出力を，80桁端末に折り曲げて出力せねばならず大変見にくい。そこで Fordap と Coboldap について早速80桁用出力を可能にした（図7，図8参照）。今度は，実行回数を左側に置いている。これによつて，速度の遅い端末でも，有効文字を出力し終ると直ちに改行できて，出力時間もかなり短縮できる。1行におさまらないソーステキストは，次の行に折り曲げて出力する。出力巾が狭いので1行に複数個の文があれば，それは複数行に展開する以外に方法がない。

静的解析ツールとの融合　プログラムが大きくなり，手続きの数が多くなると手続き間あるいは

```
DYNAMIC PROFILE
 ROUTINE NAME      MAIN
  LABEL       COUNT     LABEL       COUNT     LABEL       COUNT     LABEL       COUNT
  100001          1     100003         26       200       36196       400           1
     100       2724     100004       2723    100006        2254    100007           1
  100002       2723     100005      36665       300         469    100008           1
 ROUTINE NAME  ::TIMSET
  LABEL       COUNT     LABEL       COUNT     LABEL       COUNT     LABEL       COUNT
  100001          1     100002          3       100           3    100003           1
 ROUTINE NAME  ::BUFCLR
  LABEL       COUNT     LABEL       COUNT     LABEL       COUNT     LABEL       COUNT
  100001         28     100003     189728        200        3388    100004          28
  100002       3388        100     189728
 ROUTINE NAME  ::::MOVE
  LABEL       COUNT     LABEL       COUNT     LABEL       COUNT     LABEL       COUNT
  100001       3192     100002     159992        100     159992     100003        3192
 ROUTINE NAME  :::PRINT
  LABEL       COUNT     LABEL       COUNT     LABEL       COUNT     LABEL       COUNT
  100001         27     100003      90720     100005      90720        100        1620
  100002       1620     100004       1620     100006       1620     100007          27
 ROUTINE NAME  ::::CONV
  LABEL       COUNT     LABEL       COUNT     LABEL       COUNT     LABEL       COUNT
  100001       3294        100      11000        200        544        300        2176
  100002       2750     100004       2750     100005       2176     100006         544
  100003      11000
```

図6　実行回数表示の1例

```
FACOM  OSIV/F4  FORDAP  -780911-  V-02  L-02  DATE 79-11-06  TIME 09-34-59

EXECUTIONS ** SOURCE STATEMENT **        EXECUTION TIME =       4616 MSEC


          C     THIS IS A PROGRAM WHICH PRINTS A SOURCE-FILE
          C     IN A STINGY FORMAT.
          C
                INTEGER   TXTBUF(56,121),SEQBUF(121)
                INTEGER   BUFCNT,BUFLMT,SEQCNT,LINLMT,PAGCNT
                INTEGER   SYSIN,SYSOUT
                INTEGER   SPACE,NUM(10)
                INTEGER   TM(6)
                INTEGER*4 HIZUKE(2)
                INTEGER   CR(80),LEFT(56),RIGHT(16)
          C
                EQUIVALENCE (CR(1),LEFT(1)),(CR(57),RIGHT(1))
          C
                COMMON /BUFFER/TXTBUF,SEQBUF
                COMMON /COUNTR/BUFCNT,BUFLMT,SEQCNT,LINLMT,PAGCNT
                COMMON /IOUNIT/SYSIN,SYSOUT
                COMMON /CHAR  /SPACE,NUM
                COMMON /HEADER/HIZUKE,TM
          C
        1       CALL TIMSET
        1       CALL BUFCLR
          C
     2724   100 READ(SYSIN,99999,END=400) CR
            99999 FORMAT(80A1)
          C
     2723       IF (BUFCNT .GT. BUFLMT-2) CALL PRINT
       26(  1.0%)
          C
     2723       BUFCNT=BUFCNT+1
     2723       SEQCNT=SEQCNT+1
          C
     2723       CALL MOVE(LEFT,TXTBUF(1,BUFCNT),56)
     2723       SEQBUF(BUFCNT)=SEQCNT
          C
     2723       DO 200 I=1,16
    36665         IF (RIGHT(I) .NE. SPACE) GO TO 300
      469(  1.3%)
    36196   200 CONTINUE
     2254       GO TO 100
          C
      469   300 BUFCNT=BUFCNT+1
      469       CALL MOVE(RIGHT,TXTBUF(1,BUFCNT),16)
      469       GO TO 100
          C
        1   400 IF (BUFCNT .GT. 0) CALL PRINT
        1(100.0%)
          C
        1       STOP
          C
                END



EXECUTIONS ** SOURCE STATEMENT **        EXECUTION TIME =       2970 MSEC

       27       SUBROUTINE PRINT
          C
                INTEGER   TXTBUF(56,121),SEQBUF(121)
                INTEGER   BUFCNT,BUFLMT,SEQCNT,LINLMT,PAGCNT
                INTEGER   SYSIN,SYSOUT
                INTEGER   TM(6)
                INTEGER*4 HIZUKE(2)
                INTEGER   PAGCHR(4),SEQL(4),SEQR(4)
          C
                COMMON /BUFFER/TXTBUF,SEQBUF
                COMMON /COUNTR/BUFCNT,BUFLMT,SEQCNT,LINLMT,PAGCNT
                COMMON /IOUNIT/SYSIN,SYSOUT
                COMMON /HEADER/HIZUKE,TM
          C
       27       PAGCNT=PAGCNT+1
       27       CALL CONV(PAGCNT,PAGCHR)
          C
       27       WRITE(SYSOUT,99999) HIZUKE,TM,PAGCHR
            99999 FORMAT(1H1/8X,32H     FACOM     OSIV/F4     STINGY ,
                +              20HPRINTER     -780101-,14X,5HDATE ,
                +              2A4,5X,5HTIME ,2(2I1,1H:),2I1,5X,
                +                5HPAGE ,4A1/ )
          C
       27       DO 100 I=1,LINLMT
     1620          K=I+LINLMT
     1620          CALL CONV(SEQBUF(I),SEQL)
     1620          CALL CONV(SEQBUF(K),SEQR)
     1620          WRITE(SYSOUT,99998) SEQL,(TXTBUF(J,I),J=1,56),
                +                     SEQR,(TXTBUF(J,K),J=1,56)
            99998 FORMAT(1H ,4A1,2X,56A1,8X,4A1,2X,56A1)
     1620   100 CONTINUE
          C
       27       CALL CONV(SEQBUF(BUFLMT),SEQR)
       27       WRITE(SYSOUT,99997) SEQR,(TXTBUF(J,BUFLMT),J=1,56)
            99997 FORMAT(71X,4A1,2X,56A1)
          C
       27       CALL BUFCLR
       27       BUFCNT=0
       27       RETURN
          C
                END
```

図７　８０桁端末への Fordap 出力例

```
FACOM  OSIV/F4  COBOLDAP  V-04  L-01  DATE 79.11.06  TIME 09:55:24  PAGE 01
EXECUTIONS   SEQ.  A.  B.

          000010*    THIS IS A PROGRAM WHICH PRINTS A SOURCE-FILE
          000020*    IN A STINGY FORMAT.
        1 000030 IDENTIFICATION    DIVISION.
          000040 PROGRAM-ID.       PRINT.
          000050 AUTHOR.           NAOMI FUJIMURA.
          000060*
          000070 ENVIRONMENT       DIVISION.
          000080 CONFIGURATION     SECTION.
          000090 SOURCE-COMPUTER.  FACOM-M190.
          000100 OBJECT-COMPUTER.  FACOM-M190.
          000110 SPECIAL-NAMES.    C01 IS PAGE-CHANGE.
          000120*
          000130 INPUT-OUTPUT      SECTION.
          000140 FILE-CONTROL.
          000150     SELECT SOURCE-FILE ASSIGN TO S-SYSIN.
          000160     SELECT PRINT-FILE  ASSIGN TO S-SYSPRINT.
          000170*
          000180 DATA              DIVISION.
          000190*

                              (Omitted)

          000830*
        1 000840 PROCEDURE         DIVISION.
        1 000850 MAIN              SECTION.
          000860*
        1 000870 BEGIN.
        1 000880     PERFORM SHOKI-SETTEI.
        1 000890     PERFORM TEXT-EDIT UNTIL EOF-SW = 1.
        1 000900     PERFORM ATO-SHIMATSU.
          000910*
        0 000920 SUBR              SECTION.
          000930*
        1 000940 SHOKI-SETTEI.
        1 000950     OPEN INPUT SOURCE-FILE, OUTPUT PRINT-FILE.
        1 000960     PERFORM LP-1-PAGE-CLEAR VARYING BUF-COUNT FROM 1 BY 1
          000970             UNTIL BUF-COUNT > BUF-LIMIT.
        1 000980     MOVE ZERO TO PAGE-COUNT, LINE-COUNT, BUF-COUNT, EOF-SW.
        1 000990     MOVE CURRENT-DATE TO PDATE.
        1 001000     MOVE TIME-OF-DAY TO TIMEW.
        1 001010     MOVE CORR TIMED TO HEADER.
          001020*
     2724 001030 TEXT-EDIT.
     2724 001040     READ SOURCE-FILE INTO CARD AT END MOVE 1 TO EOF-SW.
        1 (AT END)
     2724 001050     IF   EOF-SW NOT = 1  PERFORM TEXT-PRINT.
     2723
          001060*
     2723 001070 TEXT-PRINT.
     2723 001080     IF   BUF-COUNT > BUF-LIMIT - 2  PERFORM LP-1-PAGE-PRINT.
       26
     2723 001090     ADD 1 TO BUF-COUNT.
     2723 001100     ADD 1 TO LINE-COUNT.
     2723 001110     MOVE LINE-COUNT TO LINE-NO (BUF-COUNT).
     2723 001120     MOVE TEXT-LEFT TO TEXT-56 (BUF-COUNT).
     2723 001130     IF   TEXT-RIGHT NOT = SPACE
      469 001140          ADD 1 TO BUF-COUNT
      469 001150          MOVE TEXT-RIGHT TO TEXT-56 (BUF-COUNT).
          001160*
       27 001170 LP-1-PAGE-PRINT.
       27 001180     MOVE SPACE TO LP.
       27 001190     WRITE LP AFTER PAGE-CHANGE.
       27 001200     ADD 1 TO PAGE-COUNT.
       27 001210     MOVE PAGE-COUNT TO PAGE-NO.
       27 001220     WRITE LP FROM HEADER AFTER ADVANCING 1 LINES.
       27 001230     MOVE SPACE TO LP.
       27 001240     WRITE LP AFTER ADVANCING 1 LINES.
       27 001250     PERFORM LP-1-GYO-PRINT VARYING SOEJIL FROM 1 BY 1
          001260             UNTIL SOEJIL > LINE-LIMIT.
       27 001270     PERFORM LP-1-PAGE-CLEAR VARYING BUF-COUNT FROM 1 BY 1
          001280             UNTIL BUF-COUNT > BUF-LIMIT.
       27 001290     MOVE 0 TO BUF-COUNT.
          001300*
     1647 001310 LP-1-GYO-PRINT.
     1647 001320     MOVE SPACE TO LP-1-GYO.
     1647 001330     COMPUTE SOEJIR = SOEJIL + 60.
     1647 001340     IF   SOEJIL < 61
     1620 001350          MOVE TEXT-56 (SOEJIL) TO TEXT-L
     1620 001360          IF   LINE-NO (SOEJIL) IS POSITIVE
     1374 001370               MOVE LINE-NO (SOEJIL) TO LINE-L.
     1647 001380     MOVE TEXT-56 (SOEJIR) TO TEXT-R.
     1647 001390     IF   LINE-NO (SOEJIR) IS POSITIVE
     1349 001400          MOVE LINE-NO (SOEJIR) TO LINE-R.
     1647 001410     WRITE LP FROM LP-1-GYO AFTER ADVANCING 1 LINES.
          001420*
     3388 001430 LP-1-PAGE-CLEAR.
     3388 001440     MOVE ZERO TO LINE-NO (BUF-COUNT).
     3388 001450     MOVE SPACE TO TEXT-56 (BUF-COUNT).
          001460*
        1 001470 ATO-SHIMATSU.
        1 001480     IF   BUF-COUNT IS POSITIVE  PERFORM LP-1-PAGE-PRINT.
        1
        1 001490     CLOSE SOURCE-FILE.
        1 001500     CLOSE PRINT-FILE.
        1 001510     STOP RUN.
```

図８　８０桁端末への Coboldap 出力例

各手続きとデータ間の関係が複雑になり，動的解析結果の解釈が難しくなりがちである。Fortranプログラムの，ソーステキストを解析して，手続き間の呼出し関係，手続きとコモンブロックの相互関係を調べる静的解析ツールを作つた。それによつて得られるプログラムの静的構造情報の上に実行回数情報を重ねてみると，プログラムの動的振舞いを詳しく把握する助けが得られる。この情報をもとに重要と判定した部分の実行回数付きリストのみを会話的に出力させたりする手だてを作成中である。

このように，特定のTSS環境に適合するように，ツールを作り変えてゆくと，移し換えの容易さが失われるおそれがあり，計算機システムに依存する部分としない部分をさらに明確に切り分けるよう心掛けねばならぬ。

異常終了時の処理　作成途中にあるプログラムの実行が異常終了するのは，常に起こりうると考えてよかろう。このような場合に，それまで集めた実行回数情報は，異常の原因追求に極めて有用であり，このツールがデバッグ用ツールとして使える所以である。正常終了した時だけしか輪郭情報が得られないのではツールの価値は半減するといつてよい。しかし異常終了時の処置は，各計算機システムに依存する度合が強く，各計算機システムに移し換えたそれぞれのツールが全て異常終了時の処置を備えることができているわけではない。

プログラム実行中に発生するエラーは，その処置の方法の相違から，およそ次の三つに分類される：

(1)　ライブラリレベルのエラー，例えば，FortranのSQRTで引数が負になつた場合等，

(2)　プログラム割込みによるエラー，例えば，固定小数点オーバフロー，記憶保護例外等，

(3)　OSがエラーとするもの，例えば，実行時間や出力用紙の予定超過，ジョブのキャンセル等。

これらが，計算機システムによつては，必ずしも明確に区別されているわけではない。FACOM 230-45Sでは，(2)と(3)のエラーが生じた場合，アセンブリ言語による簡単なルーチンを作成することで対応できるが，(1)に対しては，各言語のライブラリルーチンの中で閉じていて（例えばSTOP文で終つてしまう），ツールの側からは処置がとり得ない。

九大大型計算機センターのFACOM OSIV/F4や，東大大型計算機センターのHITAC OS7では，Fortranの場合，エラーモニターが用意されており，(1)に対応できる。また(2)もエラーモニターで管理している部分があつて，それらは当然対応できる。しかし，その他の言語では，(1)のエラーには対処できない。

OSIVでは，プログラムの全ての異常終了（ABEND）に対して，ユーザプログラム側で後始末を可能にする手段をOSが提供しているので，アセンブリ語によるプログラムを仲介とすることによつて，全てのエラーにツール側で対応できる。

OS7の場合には，現在のところ(3)のエラーについては，対応策を施していない。これは，遠隔地にあるため，OSの機能を調べてそれをテストする時間的余裕がないことによる。

3．3　移し換えにおける問題点

この問題については，文献[2, 4]に具体的に述べておいたので，ここでは，問題点の指摘にとどめる。ここで述べているような前処理部と後処理部を有するシステムを全体として，他の計算機システムに移し換える際に問題になる点は，言語に依存した部分と，依存しない部分に分けることができる。言語に依存するのは次のとおり：

(1)　言語仕様：　各計算機システムごとに，言語仕様に微妙な差があるので，可能な限り，JIS規格があればその範囲内で記述することに努める。

(2)　ファイル：　各計算機システムでファイルの仕様が異なるので，できるだけ単純な構造のファイルを採用することが望ましい。ここでは，全て順編成ファイルを使用している。FortranとCobolのファイルは限られた範囲でしか互換性がない。CobolやPascalのプログラムでは，ソースプログラム中にファイルの仕様を記述せねばならぬため，移し換えに際して，書き換えを余儀なくされる。

言語に依存しないものには次のようなものがある：

(3)　ジョブ制御言語：　ジョブ制御文は，解釈実行型と翻訳型の二つの型がある。それによつてファイルの取扱いが異つてくる。特に，ジョブステップ間でのファイルの受渡しの方法に相違がでる。一時ファイルと保存ファイルで扱う方法が異るものもある。また，ジョブ制御文のカタログ化（マクロ化）機能の有無やその構成も，このツールを使いやすくする上で必須であるが，計算機システムごとにマクロジョブ制御文を個別に作成せねば

ならぬ。バッチ処理用とTSS用をそれぞれ分けて作らねばならぬシステムもある。

(4) 時間計測： cpu時間はプログラムの性能評価にとつて直観的な目安を与えてくれるが，これを計測しようとすると，その精度，計測に伴うオーバヘッド等が問題になる。FACOM OSIIのような比較的単純なOSでは，良い値を得ることができるが，超大型システムのOSでは，特に時計がしばしば呼び出されるような場合に，それが重荷になつて望ましい結果を与えないことがある。

4．使用の実態

Snobol4は，文字列処理とパタン合せを得意とする高水準言語である。TSS環境で仕事をするようになると，ファイルを覗いたり変更したりという仕事が頻繁に生ずるが，それを行うのに必要なツールが常に用意されているとは限らない。こんなとき，Snobol4を，便利に使うことができる[13]。そのときの実行回数情報は，入力ファイルや出力ファイルに対するメモとして重宝である。図5のプログラムに沿つて説明しよう。

このプログラムは，1行72桁からなる入力ファイルを136桁の出力用紙の1ページに左右2列に分けて出力するものである。図5中(b)により，このプログラムによつて出力された用紙が27ページだつたこと，(c)により，入力テキストが2723行だつたこと，(d)から2723行中1行の有効桁が56桁を超えるものが469行だつたことがわかる（57桁から先の有効文字は次の行に折り返して出力する）。1ページの標準は60行としているが，折返した行が61行目にかかれば，それをそのままその行に印刷する。(e)はそのような行が10行あつたことを示している。

図2，3，4，7，8のFordap，Coboldap，Pasdapによる出力例中のプログラムは，実は，全て上のSnobol4プログラムと同じ目的をもつたプログラムである。プログラム作成者が異るので，それぞれ出力仕様に多少の違いがあるが，同じ入力テキストを処理しているので実行回数情報の対応をとることができる。図2のFortranサブルーチンBUFCLRやMOVEは，Snobol4やCobolで1文で表現可能だ。したがつて，図3(Coboldap)や図5(Snobol4)では，18万回や15万回のような回数は，外に現れない。これはシステムの内側で行われているものだ。このような場合，実行回数情報が直ちに能率改善の手掛りを与える可能性は，Fortranの場合より小さくなる。

実行回数情報を手掛りにして，Snobol4プログラムの能率改善を行つた例がないわけではない。約150文からなるプログラムで，全文の総実行回数92285回のうち，唯一つの文だけで37513回も実行されていることを見付け出し，たつた1文を書き直すだけで，実行時間を32秒から23秒（FACOM M-190）に短縮したような例もある。また，Coboldapの前処理部をCoboldapで計測し，その結果から表引きの方法を改めることによつて処理時間を最初の版の約半分に減少させたこともある。

このように実行回数情報を，能率改善の手掛りとするか，プログラムテストの援助に使うか，処理手続きやデータに対する操作へのメモとして使うか，といつたことは，それぞれの言語の特性によるところが大きく，抽象化のレベルと無関係ではない。図2－8にあげたプログラムは，全て同じ目的を持ち，同じ入力データを処理しているのだから，例えばFortran(図2，7)の場合でも，Snobol4で説明したような，メモとなりうる回数が各所に現れている。Fortranは，文字処理やパタン合せを得意としているわけではないから，そのような処理が必要となれば，一つ一つ詳細に記述せねばならぬ。我々の例ではそこに実行が集中してしまつた。このことは，n×nの行列の積を計算する例を用いるともつとわかりやすいかもしれない。二つの行列の名前を演算子をはさんで並べるだけで行列の積の計算を命じることができる言語では，積の実行回数は1回であるのに対し，要素ごとの積和を計算する3重ループを用いるFortranのような言語では，それがn^3回となつて現れる。

使用実績　このようなツールが実際どの位使われているか興味がある。表1は，九大情報工学科における使用回数である。学生演習を通じて，このようなツールの使用を奨めているので，ホスト言語コンパイラの使用回数に対する比率が非常に高い。1978年度Pasdap/Pascalの比は534/1690にもなつている。一方，これらのツールの移し換え先の一つである九大大型計算機センターの使用実績を表2に示す。Fordapは，Fortran Hあるいは，Fortran HEコンパイラに制御を渡すようにしているので，FACOM 230-75時代にはおよそ1/20，FACOM M-190時代になつてからは，およそ1/15の使用比率である。FortranD

表1　九大情報工学科（FACOM 230-45S）における各処理系の使用状況

処理系＼年度	1974	1975	1976	1977	1978
Fortran	16911	17712	16288	16573	15513
Fordap	204	2204	3116	3481	2616
Cobol	187	1013	404	368	30
Coboldap	—	—	222	183	0
Pascal	—	—	216	639	1690
Pasdap	—	—	—	97	534
Snobol3	—	700	1425	1608	1105
Fasp	3217	2082	1119	525	1487

表2　九大大型計算機センターにおける各処理系の使用状況

1976.4 ~ 1977.3 (FACOM 230-75)*		1977.11 ~ 1979.3 (FACOM M-190)**	
Fortran D	87606	Fortran GE	230946
Fortran H	11769	Fortran HE	95895
Fordap	456	Fordap	651
		Cobol	142
		Coboldap	32
		Pascal & Pasdap	1710

* 九大大型計算機センター広報 Vol.10,No.2, p. 104 (1977)

** 同　上　Vol.12,No.2, p. 163 (1979)

（Fortran GE）コンパイラの使用数まで考慮すれば，その比率は非常に小さい。大型計算機センターがこのようなツールの使用を積極的に奨めているわけではない。何かのきつかけで，一度使つて味をしめた比較的限られた数の利用者だけがかなり積極的に使つているようである。

ツールの使用を説明する場合，能率改善の効果は定量的なので説得力がある。一方その他の効用は，実際に使つてみて始めて体得できるといつたもので，頭では理解しにくい。実行回数計数ツールに限らず，計算機を専門としない計算センター利用者に，いわゆるツール類を使用して，楽に仕事をする方法を普及させることを考えるのは，もう少し先のことなのかもしれない。

参考文献

[1] 藤村，牛島：プログラムの実行解析システムの作成と使用について，九大工学集報，Vol. 48, No. 4, pp. 95—100, 1975.

[2] 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.

[3] 藤村，牛島：COBOLプログラム輪郭作成システムの実現と使用について，九大工学集報，Vol. 50, No. 3, pp. 209—214, 1977.

[4] 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.

[5] 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，九大大型計算機センター広報，Vol. 11, No. 4, pp. 289—292, 1978,東大大型計算機センターニュース，Vol. 11, No. 2, pp. 20—24, 1979.

[6] 藤村，鶴田，牛島：Ratforプログラム動的解析システムの実現と使用について，情報処理学会第20回全国大会講演論文集，pp. 243—244, 1979.

[7] 牛島，高比良：SNOBOL4 輪郭作成機能の使用について，九大大型計算機センター広報，Vol. 12, No. 2, pp. 110—113, 1979.

[8] Russell, E.C. and Estrin, G.: Measurement based automatic analysis of Fortran programs, SJCC, Vol.34, pp. 723-732, 1969.

[9] 牛島，津田：Fortranプログラム実行解析のためのプリコンパイラについて，第20回電気四学会九州支部連大，pp. 341—342, 1970.
[10] Knuth, D.E.: An empirical study of FORTRAN programs, Software-Pract. and Exper., Vol.1, pp.105-133, 1971.
[11] 角田：SNOBOL 3 使用手引書，東工大情報科学科，1977.
[12] Griswold, R.E.: The Macro Implementation of SNOBOL4, W.H. Freeman and Co., San Francisco, 1972.
[13] 木村：Software Toolとは何か，情報処理，Vol. 20, No. 8, pp. 673—680, 1979.