SENAC, v 24, No. 2, 1991.4, pp 26-34

Fortran ツール PCOMP について

中島映至(東北大学理学部附属大気海洋変動観測研究センター)

金田康正(東京大学大型計算機センター)

１．はじめに

科学知識の蓄積につれて、解析・モデリングのためのコンピュータプログラムも年々、多様化し複雑化している。それにつれて、データベース言語、論理構造解析言語、図表作成言語などの多くの支援プログラムが用意されるようになった。しかし、大部分の科学計算は依然として Fortran 言語に頼っているのが現状である。良く知られるように、Fortran 言語は局所変数に関する制約のために、サブルーチンや関数などのサブプログラムをモジュール化するのが大変困難である。これらの欠点は Fortran90 では改善される予定であるが、当分はこれを利用できない。そこで、Fortran コードに対するトランスレーターを開発し、これらの欠点のいくつかを改善することにした。また、このトランスレーターには、多数の副手続きをソースコードのままで管理する機能を持たせた。この様な機能は、パーソナルコンピュータやワークステーションでプログラム開発をしておいて、最終計算を大型機でおこなうという最近の利用形態にとって大変役立つと思われる。なぜならば、ライブラリアンによる副手続きの管理法では、オブジェクトコードをライブラリに登録するために、保守の仕方が個々のコンピュータシステムに強く依存してしまうからである。その結果、複数のコンピュータシステム上で多くのライブラリプログラムを管理するのが大変繁雑になってしまう。幸い、著者の一人は東北大学大型計算機センターの公募研究[開発承認番号: 90-21(L)、Fortran プログラムの組織化に関する研究 - 大気物理学にかかわるプログラムの整備。中島映 至、早坂忠裕(以上 大気海洋変動観測研究センター)、高橋哲夫(大型計算機センター)] を採用していただいた。そこでここでは今回開発した Fortran ツール PCOMP について、その概要と開発結果を紹介したい。

２．PCOMP とは？

PCOMP は、それぞれ独立なソースファイルに収納された多数のユーザー・サブプログラムを次の点で管理するツールである:

(a) メインプログラムを実行するために必要なサブプログラムを異なったファイルから収集して、一つのソースファイルにまとめる。ユーザーはこの出力ソースファイルをコンパイルし実行すれば良い。この機能を使えば、プログラムを機能別に分割して別個のファイルとして管理できるので、プログラムの維持が楽になる。また、修正が必要になった場合でも、修正したサブプログラムのみを転送すれば良いので、異なるコンピューターを使用する環境にとっても便利である。

(b) 動作(a)を行う間に PCOMP は、配列の大きさを決めるパラメータのサイズをプログラム間で整合するように設定する。従って、エディターを使った複雑なパラメータサイズの変更手続きに頼らずに、ユーザーは適正なメモリーで計算を実行できる。これは、まずマイクロコンピューターでテストした後で、メインフレームで本計算をする時に便利である。

この様な機能を持つ PCOMP はそれ自体 Fortran77 で書かれているため、Fortran コンパイラーのある計算機システムならば、ほぼ共通に同じ要領で使用できる。実際に PCOMP を使用するためには、ユーザーがサブプログラムをどのようにリンクして欲しいかとか、パラメーターサイズをどのようにコントロールするかを PCOMP に指示する命令文を Fortran プログラムに書く必要がある。この命令について以下の 3 章と 4 章で説明する。

３．サブプログラムの管理

大きな Fortran プログラムを作る際、全体をサブプログラムの集まりに分解した方が良い。サブプログラム名によってプログラムの各部分の機能をシンボル化できるので、何をしようとしているのかを追跡しやすくなるからである。オブジェクトコードを管理するライブラリアンによることなく、PCOMP はソースコードの形でサブプログラムを管理する。コンピューターの能力、メモリーの増大に伴って、ソースコードによってサブプログラムを管理し、それらを全部集めたコードを毎回コンパイルする方がユーザーにとって便利なことが多い。また、同一サブルーチンをあちこち複数のメインプログラムと一緒に置いて置くのは、これらのサブルーチンの管理のためにも良くない。あるアプリケーションに含まれるサブルーチンを修正したのに、他の場所では忘れたと言うことがおこりうるからである。

サブプログラムをどのようにリンクするかを PCOMP に指示するためにはプログラム内に PCOMP に対する命令を書く。これは Fortran プログラム行の第一カラムから C$ で始まる文によって行われる。第一カラムが C なので Fortran コンパイラー自身はこれを注釈行と解するため、PCOMP を使用しないユーザーにとってはこの様な PCOMP 命令は全く無害である。

PCOMP を使ってどの様にサブプログラムを管理するのかを以下ステップごとに説明する:

(1)　分割されたプログラム単位をそれぞれ異なるファイルに格納する。プログラム単位はそれ以上分割する必要の無い複数のサブプログラムを含んでもかまわない。お互いに関連のあるファイル同志は同一のディレクトリー（カタログ）に格納する。このディレクトリーをライブラリーと呼ぶことにする。

(2)　それぞれのファイルに含まれるサブプログラムは NAME 命令によって PCOMP にその名前を知らせる必要がある:

```
C$NAME   SUB1
      SUBROUTINE   SUB1
                     ...
      RETURN
      END
```

上の例では、SUB1 と言うサブルーチンに対してその名前と同じ名前を NAME 命令で PCOMP に通知している。PCOMP は NAME 命令によってサブプログラムを認識するので、名前はサブルーチン名や関数名と一致しなくても良い。もし一つのファイルに複数のサブプログラムが含まれる場合でも、PCOMP に通知する NAME 命令はファイルの先頭に一個代表してあれば良い。もちろん、同一ファイルに含まれるサブプログラム一個ごとに NAME 命令を付けても良い。しかし混乱をさけるため、特殊な場合を除いてサブプログラム一個ごとにサブプログラム名と同一の名前を持つ NAME 命令を付け、独立したファイルに格納して管理した方が良いだろう。

(3)　　考えているサブプログラムが他のどのサブプログラムを呼び出すかを PCOMP に指定するために LINK　命令を使う：

```
C$NAME　SUB1
　　SUBROUTINE　SUB1
C$LINK　SUB2　SUB3 …
　　…
　　CALL　SUB2
　　…
　　CALL　SUB3
　　…
　　END
```

この例ではサブルーチン SUB1 が SUB2 と SUB3 を呼び出しており、これらのサブルーチンをライブラリーからリンクしたい意志を PCOMP に伝えるために LINK 命令で SUB2 と SUB3 を指定している。名前の区切りは一つ以上の空白である。一行で収まらない時は複数の LINK 命令を並べて書く。たとえ SUB2 がさらにその内部で他のサブプログラムを使用している場合でも、孫引きのサブプログラムを LINK 命令に指定する必要はない。

この様にして、サブプログラムを作成する時、それがどんなサブプログラムを呼び出しているかを指示しておけば、以後そのことを忘れてしまっても良い。自分が呼び出そうとするサブプログラムが他のサブプログラムを孫引きする場合は、PCOMP が自動的にその孫サブプログラムを収集してくれるからである。従って、今作っているサブプログラムがその中でどのサブプログラムを呼び出しているかのみに注意を集中しておけば良い。

同一ファイルに複数のサブプログラムが含まれている場合は､そのファイルの先頭にある NAME 命令で与えた名前を LINK 命令に指定しておけば、そのファイルの全内容をリンクしてくれる。

(4)　　この様にしてライブラリーを作った後に、各ライブラリーごとにライブラリー情報を含む A-file と言うのを作る。それには普通のアスキー型のエディターで次の形式の情報ファイルを作れば良い：

```
SUB1　　SUB1file　　メモ
SUB2　　SUB2file　　メモ
SUB3　　SUB3file　　メモ
…
```

ここで SUB1, SUB2, SUB3 … は NAME 命令で指定したサブプログラム名；SUB1file, SUB2file, SUB3file … はそれを含むファイル名である。ライブラリーのディレクトリー名が暗黙に仮定されるのでこれらのファイル名はディレクトリー名を含まなくて良い。NAME 命令で指定したサブプログラム名とファイル名は一つ以上の空白で区切る。残りの空欄は意味を持たないのでメモのために使える。

この様にして作られた A-file のファイル名は自由であるが、ファイルリストを見る時に先頭に来る様に A と言う名前にしておくと便利である（これが A-file の名前の由来である）。

(5)　　メインプログラム（これをルートと呼ぶことにする）を PCOMP に指定すると、PCOMP

は NAME 命令と LINK 命令を A-file に照らし合わせながら、ルートが最終的に使用するすべてのサブプログラムをライブラリーから収集してきてルートにリンクする。そして、この全部のソースプログラムを出力する。出力するソースファイル名は、ルートの先頭に OUTF 命令を書くことによって次の例の様に指定する。また、使用するライブラリーの A-file 名を LIBF 命令によって指定する:

```
C$OUTF    MAINOUT
C$LIBF      LIB1.A
C$LIBF      LIB2.A
C$NAME    MAIN
C$LINK    SUB1 ...
      DIMENSION   ...
       ...
      END
```

この例ではルートであるメインプログラムに NAME 命令によって MAIN と言う名前が与えられており、このメインプログラムでは SUB1 等のサブプログラムが呼び出されていることが、LINK 命令によってわかる。すべてのユーザー・サブプログラムを含むメインプログラムを MAINOUT と言う名前のソースファイルに出力することを OUTF 命令で指示している。 MAINOUT はディレクトリーを含む出力ファイル名である。また、上の例ではライブラリーとして、LIB1 と LIB2 を使用することを LIBF 命令で指定している。これらのライブラリーに含まれる A-file の名前は LIB1.A と LIB2.A である(これもディレクトリーを含まなければならない)。LIBF 命令はいくつあっても良い。

(6)　　開発中のサブルーチンをテストする時などの特殊な場合として、ライブラリー以外の場所に格納されている特定のプログラムをリンクしたいことが起こりうる。この時は次の様な PRGF 命令を使ってそのファイルを指定する:

```
C$OUTF    MAIN.OUT
C$PRGF    SUB1        SUB1file
C$LIBF      LIB1.A
C$LIBF      LIB2.A
C$NAME    MAIN
C$LINK    SUB1 ...
      DIMENSION   ...
       ...
      END
```

この例で、SUB1 は NAME 命令、LINK 命令で使用されるサブプログラム名; SUB1file はそれを格納するファイル名(ディレクトリーを含む) である。同一サブプログラム名がライブラリーにあった場合は、PRGF 命令で指定されたファイルの中身が優先されてリンクされるので、ライブラリーに登録されているサブプログラムの改良版などを作る時に役立つ。

(7)　　PCOMP を実行する。PCOMP 自体 Fortran プログラムなので通常のプログラムと同様にコンパイルと実行をすれば良い。実行されると PCOMP はコンソールを通して次の質問をしてくる:

(a)　　ルートの格納されているファイル名。PCOMP がルートと同一のディレクトリーにあればファイル名のみで良い。

(b)　　出力ソースプログラムに注釈行をそのまま書くか、削除するかを指定する。元のプログラムに忠実に注釈行を書く場合は 1 を、削除する場合は 0 を入力する。出力ソースプログラムは、コンパイル・実行をするための一時的なものなのでコンパイラーにとって意味の無い注釈行を省略しておいても良い。

４．パラメーターサイズの管理

(1)　　なぜパラメーターサイズの管理が必要か？

Fortran はその歴史上形成されて来た性格のため、基本的には静的なメモリーアロケーションを想定している。従って、ダイナミックアロケーションによって局所変数のメモリー領域を実行時に取る様には作られていない（これができるコンピューターシステムもあるが）。従って次の様なプログラミングは文法違反である：

```
PARAMETER     (N=10)
REAL      A (N, N ),   B(N )
M = 1
CALL   SUB (M, N)
            ...
SUBROUTINE   SUB (M, N )
REAL      A (N, N )
            ...
```

従って、サブルーチンの外側から配列の大きさをコントロールするためには、

```
PARAMETER     (N=10)
REAL      A (N, N ),   B(N )
M = 1
CALL   SUB (M, N, A )
                ...
SUBROUTINE   SUB (M, N, A )
REAL      A (N, N )
                        ...
```

のようなプログラミングを余儀なくされる。この例では配列 A を整合配列としてサブルーチンの引数にしたため、本来サブルーチンの中だけでしか意味の無い配列 A をメインプログラムで宣言しなければならなくなった。小さなプログラムの場合はこれでも良いが、大規模なものだとメインプログラムには意味を知る必要のない多くの作業配列が並ぶことになる。これはプログラミングの効率を落すのと同時に、プログラム全体の見通しを大変悪くする。また、メインプログラムで直接使用しない配列をメインプログラムで宣言するために、ダイナミックアロケーションによってメモリー節約を行っているミニコンピューターやマイクロコンピューターの Fortran にとっては大変なメモリーの無駄使いになる。そこで別の解決法として次の様にパラメーター文を使い局所配列の大きさを指定し、エディターで必要に応じてパラメーターのサイズを変更する方法もある：

```
      PARAMETER     (N=10)
      REAL      B (N )
      M = 1
      CALL   SUB (M )
                  ...
      SUBROUTINE   SUB (M )
      PARAMETER     (N=10)
      REAL      A (N, N )
                        ...
```

この様にするとメインプログラムで宣言する配列やサブプログラムの引数の数は減るが、パラメーターのサイズの変更が煩雑になって誤りのもとになる。

(2)　　PCOMP によるパラメーターサイズの管理

前節で示した様な問題点を解決するために、PCOMP は最良とは言えないが比較的簡単なトリックでこれを乗り切ることにした。それを次の例で説明する:

```
      ...
C$PSET   N = 30;     SUB.N = N
C$PRPC
      PARAMETER     (N=10)
      REAL      B (N )
      M = 1
      CALL   SUB (M )
      ...
C$NAME   SUB
      SUBROUTINE   SUB (M )
C$PRPC
      PARAMETER     (N=10)
      REAL      A (N, N )
                  ...
```

この例では新たに PSET 命令と PRPC 命令が現れた。PSET 命令はユーザーが設定したいパラメーターサイズを PCOMP に知らせる命令である。複数のパラメーターは ; によって区切られる。複数の PSET 命令を書いてもかまわない。上の例の PSET 命令ではパラメーター N のサイズを 30 に設定することを意味している。実際の Fortran プログラムのパラメーター文では N = 10 になっているが、PCOMP が出力する最終的なソースプログラムではメインプログラムの N は 30 になる。SUB.N = N と言う文章の意味は、メインプログラムで使用しているパラメーター N のサイズをサブプログラム SUB で使用しているパラメーター N に割り当てることを意味している。PRPC 命令は PCOMP によってパラメーターサイズを置き換えて欲しいパラメーター文の直前に置く。たとえ PSET 命令でパラメーターサイズの割り当てを命じてあっても、PRPC 命令を置かない限り PCOMP はサイズの変更を行わない。このような方法で、PCOMP はユーザーが予め PSET 命令と PRPC 命令で設定しておいた手順に従って、次の様にパラメーターサイズを調節してくれる:

```
PARAMETER      (N=30)
REAL       B (N )
M = 1
CALL    SUB (M )
                    ...
SUBROUTINE    SUB (M )
PARAMETER      (N=30)
REAL       A (N, N )
                              ...
```

ここで説明した様な要領に従ってライブラリーサブプログラムを作る段階で、サブプログラム及びそれが直接呼び出すサブプログラムで使用しているパラメーターのサイズを PSET と PRPC 命令でコントロールしておけば、以後はそれを記憶にとどめる必要はない。PCOMP がメインプログラムから順次パラメーター同志の関係を解析していって自動的に調整してくれる。従って、メインプログラムの PSET 命令でパラメーターサイズを変更すれば、全プログラムにわたってパラメーターサイズを整合させることができる。

PSET 命令に書く表現としては、整数の四則演算及び次のような最小・最大値がある:

```
C$PSET    M = 10;    N = 20
C$PSET    L 1= MIN ( M    N );      L2= MAX ( M    N )
```

MIN と MAX の引数は一つ以上の空白で区切られれば何個並んでいても良い。

(3)　　一つの問題点

同一サブプログラムを異なるサブプログラムが呼び出す時、PSET 命令による設定がかち合うことがある。この場合 PCOMP は両者の最大値をパラメーターサイズとして与える。この様な操作は多くの場合満足な操作であるが、場合によっては不都合を生じることがある。この場合は、サブプログラムがかち合わない様に同じサブプログラムを異なる名前でライブラリーに登録する必要がある。

５．　出力ファイル

PCOMP を実行すると、二つのファイルが出力される。ひとつは、結果となるソースファイルであり、そのファイル名は先に述べた OUTF 命令によって指定されたものである。もう一つは、実行結果を示す次の様な内容の PCOMPOUT と言う名前のファイルである:

(a)　　ルートファイル名。
(b)　　使用したライブラリーのリスト。
(c)　　PSET 命令によって設定されたパラメーター名とそのサイズ。
(d)　　PRPC 命令によって置き換えられたパラメーター名とそのサイズ。

６．終りに

実際に PCOMP を使用して著者の一人が扱っている大気物理学に関わる多くのプログラムを整理して見た所、プログラムの生産性の増大と保守に費やす労力の大幅な軽減が確認された。使用し

ているコンピューターは ACOS とマッキントッシュであるが、両システム上でライブラリーの管理がスムースに行われる様になった。特に、RS-232C の様な低速の通信回線を使用している場合でも、プログラムの修正が小さな容量のファイルに限られるため、マッキントッシュ側のエディターで修正を行い、転送後 PCOMP によってリンクして実行するなどの芸当が出来るようになった。

読者はすでに気付いたと思われるが、PCOMP がプログラムの Fortran 文法を解析していれば、LINK、PRPC 命令は不用である。しかし今回の開発計画では、PCOMP の負担を減らすためにこの様な文法解析は行わなかった。現在、この様な欠点を取り除き、他の機能を大幅に追加したより使い易いトランスレーターの開発が進行中である。最終的には、このようなトランスレーターを大気海洋系の大循環モデルなどの大規模プログラムに使用できれば良いと思っている。

P/S　PCOMP のソースコードの ACOS 版、Macintosh 版、MS-DOS 版を保有していますので興味のある方は青葉山 3589 中島まで御連絡下さい。3.5 インチのフロッピーディスクを用意してあります。また、ACOS ライブラリーへの登録を準備中です。