談話室

PASCAL はシステム記述言語に適しているか？†

木　下　　恂††

1. はじめに

PASCAL は，今や主要なプログラム用言語の一つとなり，超大型のスーパーコンピュータから超小型のマイクロコンピュータまであらゆる機種の上で利用可能となっている．その適用分野も広く，教育用ばかりでなく応用プログラムやシステムプログラムの記述にも利用可能であると広く喧伝されている．しかし，PASCAL は本当にシステム記述言語に適しているといってよいのであろうか？

筆者は，かねてからこう言い切ることに疑問をもっている．この点について，大型システムのソフトウェア開発に携わる者の立場から見た感想を述べてみたい注1)．

2. PASCAL の設計目標とシステム記述

Niklaus Wirth 教授は，次の二点を PASCAL の設計目標とし，それに適した言語を開発した．

（1） 系統的（システマティック）な作業として，プログラミングを教育する．

（2） 処理系の作成が容易である．

即ち，システマティックにプログラムを作る方法を教える手段を与えることをめざしたのであって，決してシステムプログラムの記述をめざした訳ではない．それではなぜ「システム記述に適した言語」といわれるようになったのであろうか？ たしかに，コンパイラなどのソフトウェア作成に使われていることは事実である．構造的プログラミングということがいわれだしたとき，それを実践できるように最初から設計されていた言語として，各分野でプログラミングに携わっていた人々に注目されるようになったことは記憶に新しい．また，FORTRAN のような原始的(？)な言語が幅を利かしていた時代にあって，それまでの言語に慣れ親しんでいた人々にとっては目の覚めるような思いがあったにちがいない．そのような状況のもとで，PASCAL 言語自体で記述されたコンパイラ注2) が世界中に供給され，そのリスティングの薄さに皆が驚嘆するに及び，PASCAL といえば記述力の優秀性を思い起こすようになった．そして，しだいにシステム記述用とエスカレートして考えてしまう原因となったように思われる．

3. PASCAL 支持の背景

PASCAL をシステム記述言語として支持する立場の人々の発言は，次のような背景のいずれかに基づいているのではないだろうか？

（1） Wirth の標準 PASCAL に対しシステム記述用の機能を付け加え，それを PASCAL と呼んでいる[1]．

（2） システムというものを教育用等の小規模なものに限定して議論している．

（3） 最近の，特に大型のシステム開発の経験がない．あるいは，その現状を知らない．

（3）はともかくとして，（2）が比較的多いようである．たしかに，ミニコンやマイコンのプログラム用言語として広く使われており[2),3)]，小規模なシステムならばシステム記述言語としてすでに十分な実績をあげていることは否定できない事実である．

ここで特に言及したいのは（1）である．PASCAL の設計思想を無視して言語仕様を拡張してゆくことは自由であるけれども，そのようにしてできた方言を PASCAL と呼んでよいかは問題である．「PASCAL

† Does PASCAL Fit for System Programming? by Shun KINOSHITA (NEC-TOSHIBA Information Systems Inc.).

†† 日電東芝情報システム（株）基本ソフトウェア開発部

注1） 一般にシステム記述言語という言葉が広く用いられているが，システム記述 (system description) とシステム作成 (system implementation) とを厳密に区別する人もいる．ここでは，システム記述言語という言葉を広い意味でとらえているので，むしろシステム作成用言語というべきかもしれない．また，システムプログラムとは，ここでは言語処理プログラムや管理プログラム等いわゆる基本ソフトウェアを指している．

注2） PASCAL コンパイラ自体が PASCAL 言語でコンパクトに記述されているからといって，それでシステム記述一般に適しているという証拠にはならない．PASCAL できれいに書きあげられた結果だけの例を見て判断するのではなく，それを作成する過程が問題なのである．

をシステム記述言語として利用し十分な実績をあげた」と聞けば，誰しも Wirth の標準 PASCAL を指していると思うであろう．即ち，どのような背景に基づいて PASCAL をシステム記述言語として支持しているのか見極める必要があろう．

4. システム記述に必要な機能

それでは，システム記述に必要な機能とは何であろうか？ 例えば，次のようなものがよく指摘されている．

（1） 分割コンパイル

大型システムの開発においては，多人数で協力してプログラムを構築するということはごく普通のことである．モジュールに分け，分担して開発したり，一部修正にともなう再コンパイルの影響をローカルにとどめるため，又はライブラリルーチンの作成等を可能にするためには分割コンパイル機能の利用が強力な助けとなる．

（2） 可変長配列

ライブラリルーチン等の利用にあたっては，配列の大きさを実行時に決めることができるようにしておかないと融通性がなくなる．そのために配列の大きさを定数だけでなく式で指定できる必要がある．

（3） ソースライブラリの利用

各モジュールで同一の記述を重複して行う必要がある場合（宣言文等），その手間を省くため，その部分を別ファイルとしてまとめておき，コンパイル時に導入する手法もよく使われる．こうしておくと一部修正の必要が起こってもそのファイルの内容を変えるだけで，あとは再コンパイルすればよい．各モジュールの記述は自動的に修正される．PL/I の %INCLUDE 又は JOVIAL の COMPOOL のような機能である注3)．

この他にも，ビットストリング処理，静的記憶域指定，直接呼出し入出力機能等，システム記述に必要な機能をあげてゆくとまだまだあるであろう[4]．しかし一応上記三機能が実現されていればあとは言語仕様を拡張しなくても異種言語（アセンブラ等）で記述したサブルーチンを呼び出すとか，ライブラリを充実させることによってある程度実現させることができるであろう．

ここで筆者は，これらの機能が PASCAL には欠けているから拡張せよと主張している訳ではない．むしろ，PASCAL をシステム記述言語として支持する人達が，これらの機能を導入し仕様を拡張しようとするのに反対したいのである．事実，PASCAL にこれらの拡張を施すことは困難である注4)．

システム記述に必要な機能ということで，あれもこれもと導入してゆくと，結局は PL/I のように重い言語になってしまって，PASCAL の良さが失われてしまうのを恐れるのである．

PASCAL を「システム記述言語に適している」と考えるよりも，「あらゆる分野のプログラム作成の参考となる概念やアイディアをもった言語」とみなし，将来のプログラム用言語研究のベースとして守り育てるべきではないだろうか？

5. アルゴリズムの寿命

システム記述言語をもたなかった人達が PASCAL に飛びつくのならまだしも，今まで何らかのシステム記述言語をもってシステム開発に携わってきた人達が一時的な PASCAL 熱に惑わされて今までのシステム記述言語を捨て PASCAL に乗り換えようとする傾向があるとすれば，これは大きな問題である．

良いアルゴリズムは長い生命を保ちうるが，特定のハードウェアに依存した言語で記述されるとそのハードウェアと同じ寿命しかもちえない．ハードウェアに依存しない言語で記述されると，そのアルゴリズムはそれなりの長い寿命を保ちうるものである．ところが，安易に記述言語を取り替えてしまうとそれまで築いてきたアルゴリズムの集積を失うばかりでなく注5)膨大なソフトウェア財産の寿命が尽きてしまうことになりかねない．

むしろ，PASCAL に盛り込まれている優れた概念を今までの個々の記述言語に吸収して，その良さを伸ばしてゆく努力の方が全体としてメリットがあがる場合もあると思うのである．PASCAL 熱に惑わされない冷静な対処が望まれる．

注3) ソースプログラムの利用については，プリプロセッサにより実現する等言語外の問題とすることも可能である．しかし，常に前処理のアクティビティを通すのでは，大きなシステムで沢山のモジュールから成る場合，時間がかかるので利用者の立場からすると使いにくい．言語処理プログラムの中に取り込むことが望ましい．標準 PL/I や ADA でも言語の中に入っており，実際の implementation でもそうなっているようである．

注4) (1)，(2)は厳密には PASCAL の設計思想とは相容れないという意味で困難と述べた．(3)も(1)が前提としてないと意味がない．ただし，(1)を処理系に依存する事項とみなして部分的に取り入れている例は多い．

注5) アルゴリズムの記述や伝達の手段としてだけ用いるのであれば別であるが。

6. おわりに

筆者がいいたかったことは，PASCAL を批判することではない．PASCAL は優れた言語であるが，システム記述言語指向を強調する余り，妙な機能拡張をすべきではない，PASCAL スピリットを守ろうということである．現在 ISO において PASCAL の標準化がすすめられており，仕様的には Wirth の標準 PASCAL とそれほど変らないものとなることが予想されている．しかし，この標準化活動に遅れをとったアメリカは，PASCAL の次の拡張を狙って着々と研究を重ねているという話もある．今後の動向を十分に注視してゆきたい．

システム記述用機能の取り込みについては，むしろその優れた概念を個々の記述言語に取り入れる方向ですすめるべきであると述べたが，同時に分割コンパイル方式[注6]についての研究も必要であろう．即ち，PASCAL のような一括コンパイル方式，FORTRAN や PL/I のような独立コンパイル方式，及び ADA の分離コンパイル方式について，それぞれの利害得失，実現上の問題点等についてもっと議論をたたかわせることを期待したい．

最後に，本稿に対し武市正人先生，星野康夫氏より有益なコメントをいただいた．記して感謝の意を表したい．

注6）独立コンパイル方式と分離コンパイル方式とを合わせてここでは分割コンパイル方式と称した．独立コンパイル (independent compile) 方式とは，プログラム単位ごとに完全に独立にコンパイルする手法である．分離コンパイル (separate compile) 方式とは，内部手続きをその外側の環境とともに，分離してコンパイルする手法とでも考えればよい．手続き自体の記述は変わらなくても，外側の環境が変わると再コンパイルが必要となるのが特徴である．

参考文献

1) Bate, R. et al.: Language extensions, utilities boost Pascal's performance, Electronics pp. 111-121 (June 7, 1979).
2) Conrad, M.: PASCAL: A High-Level Language for Micros and Minis, Datamation pp. 153-156 (July 1979).
3) Posa, J.: Programming Microcomputer Systems with High-level languages, Electronics pp. 105-112 (Jan. 18, 1979).
4) Schneider, G.: Pascal: An Overview, COMPUTER pp. 61-66 (April 1979).

(昭和54年11月5日受付)