ベクトル並列型スーパーコンピューター SR8000 利用ガイド
～　FORTRAN によるコンパイル、実行入門（1）

システム運用掛　宮 嵜　　洋
佐々木 一孝

「どのコマンドを使用したら良いか」、「オプションは何が必要か」、「数値計算ライブラリーを使用したい」、「具体的なプログラムのコンパイル例を見たい」等、本センターのベクトル並列型スーパーコンピューター SR8000 をこれから利用していこうとする方々からの声をよく耳にします。詳細情報を豊富に含んだマニュアルはプログラムの最適化には不可欠ですが、初心者が必要な情報を探そうとすると困難なものです。そこで、本稿では SR8000 で FORTRAN プログラムをコンパイル、実行する上で初歩的な例題を列挙しましたので御覧下さい。なお、コマンドやオプションの動作、機能に関する詳細につきましては必ずメーカー発行のマニュアルで御確認下さい。

1. コンパイルとリンク

コマンド

FORTRAN プログラムのコンパイル、リンクには以下のコマンドを使用します。

最適化 FORTRAN77	***f77***
最適化 FORTRAN90	***f90***

コマンドとオプション、ファイルの指定は以下のようになります。

% f77 〔 –オプション . . 〕 〔ファイル名 . . 〕 〔 –オプション . . 〕

オプション

オプションは空白で区切り複数指定できます。左から順に処理していきますので肯定／否定の関係にあるオプションは後に指定したものが有効になります。特に、ライブラリー名を指定する–l オプション、リンケージオプションはオプションの位置がコンパイル、リンクに影響するので注意が必要です。なお、オプションの指定方法には –W0,'コンパイルオプション', –Wl,'リンケージオプション'と記述する方法がありますが、本稿では –オプションの形式を使用します。

ファイル名

ファイル名にソースプログラム名（.f）を指定してコマンドを実行するとコンパイラーはコンパイル及びリンクを行ない実行ファイル（a.out）を作成します。このときオプション –c

を同時に指定するとリンクは行われずオブジェクトモジュール（.o）を作成します。また、ファイル名がオブジェクトモジュール名（.o）の場合にはリンクが行われ、オブジェクトモジュールから実行ファイル（a.out）を作成することができます。なお、実行ファイル名は –o オプションで指定することができます。

コンパイル、リンクの例

```
% f77  main.f   sub.f                          main.f, sub.f をコンパイルして実行ファイル
main.f:                                        を作成
f77: compile start : main.f

*OFORT77 V01-01-A entered.
*program name = MAIN                           （main.f のコンパイルメッセージ）
*end of compilation : MAIN
*program  units  =  0001,  no  diagnostics
generated.
sub.f:
f77: compile start : sub.f

*OFORT77 V01-01-A entered.
*program name = SUB                            （sub.f のコンパイルメッセージ）
*end of compilation : SUB
*program  units  =  0001,  no  diagnostics
generated.                                     （この後リンクが行われる）
% ls
a.out   main.f   sub.f                         実行ファイル a.out ができる
%
```

オブジェクトモジュール作成後、リンクする例

```
% f77  -c  main.f   sub.f                      main.f, sub.f をコンパイルしてオブジェク
（略）                                         トモジュールを作成
% ls                                           オブジェクトモジュール main.o, sub.o ができる
main.f  main.o  sub.f  sub.o
%
% f77 -o  program  main.o  sub.o               main.o, sub.o をリンクして実行ファイルを
% ls                                           作成
main.f  main.o  program  sub.f  sub.o          実行ファイル program ができる
%
```

並列アプリケーションではコンパイル時に必要なライブラリー、インクルードファイル、オプション等を自動的に設定する専用のコマンドを用意しているものがあります。これらのコマンドは専用のオプションを解釈した後、必要なオプション、ファイル名を f77 又は f90 コマンドに渡してコンパイルします。

MPI	***mpif77***、 ***mpif90***
PARALLELWARE	***srf77***、 ***srf90***

また、ソースプログラム中に記述した指示文（ディレクティブ）に従って、並列化変換した FORTRAN ソースプログラム（f90 のみ）を生成するトランスレーターもあります。

Parallel FORTRAN	***pf90***

これらのコマンドの詳細は各並列アプリケーションのマニュアルを御覧下さい。使用例は次号スーパーコンピューティングニュースでも紹介します。

2. オプションと機能

最適化 FORTRAN77（または最適化 FORTRAN90）で使用される主なオプションとその機能について記述します。

最適化オプション

最適化とは演算子の変更、ループ構造の変換、演算順序の変更などをコンパイラーが自動的に行ない、プログラムの実行速度を向上させる機能です。各種オプションが最適化機能ごとに用意されておりますが、以下に示すレベルに従って最適化することができます。

最適化レベル

レベル０	原始プログラム通りのコンパイル、文の実行順序に影響を与えない一文単位の最適化 （べき演算の乗算化、文関数のインライン化、局所的なレジスタの割り当て等）
レベル３	演算順序の変更はせず、プログラム全体での大域的な最適化 （分岐命令の最適化、命令の実行順序変更、演算子の変更、不変式のループ外への移動、ユーザー関数のインライン展開等）
レベル４	制御構造、演算順序の変更を含むプログラム全体の最適化 （演算順序の変更、除算の乗算化、ループ展開・分配・融合等）
レベル S	実行速度が最も速くなるようなコンパイルオプションの自動設定 （擬似ベクトル化、要素並列化を含む各種最適化オプションの設定）
レベル SS	演算精度を落としても実行速度が速くなるようなコンパイルオプションの自動設定 （レベル S に加えてライブラリーコード使用、引数チェックの抑止等）

高いレベルの最適化は、それより低いレベルの機能を含んでいます。また、一般にレベルが高い最適化ほどコンパイルに時間がかかります。さらに場合によっては最適化機能による副作用のため、計算結果が異なる場合がありますので御注意下さい。

−opt=最適化レベル ***最適化オプション***

% f77 program.f	最適化レベル S でコンパイル
% f77 -opt=3 program.f	最適化レベル３でコンパイル
% f77 -opt=ss program.f	最適化レベル SS でコンパイル

オプション無指定時の最適化レベルは f77 が S（ただし要素並列化を含まない）、f90 が３。

−loopdiag ***最適化ループ診断メッセージ***

```
% f77 -loopdiag program.f                    program.f を最適化レベル S でコンパイルしたときのループ構造診断メッセージを出力する
f77: compile start : program.f

*OFORT77 V01-01-A entered.
*program name = MAIN
*end of compilation : MAIN
  KCHF1805C
      the do 10 loop is fuzed with the      （DO10 ループと後続のループを融合）
following one. line=3
  KCHF1809C
       the do 10 loop is unrolled 2 times.  （DO10 ループを 2 回展開）
line=3
*program units = 0001, no diagnostics
generated.
```

オプションの詳細及び個別の最適化オプションにつきましてはマニュアル「最適化 FORTRAN77 使用の手引（6A30-3-314）」又は「最適化 FORTRAN90 使用の手引（6A30-3-311）」を御覧下さい。

擬似ベクトル化オプション

擬似ベクトル化とはメモリー上のデーターを先読みすることで演算をパイプライン的に処理するオブジェクトを生成する機能です。データー転送のオーバーヘッド（主記憶アクセスレイテンシー）を隠蔽できるので命令間の依存による性能低下を抑えることができます。擬似ベクトル化が適用される主な条件は以下の通りです。

- DOループに以下の文を含まない
  - ループ脱出文(GOTO文、EXIT文など)
  - 割り当てGOTO文、計算型GOTO文、SELECT文
  - 関数、手続き呼び出し（擬似ベクトル化数学関数を除く）
  - 入出力文
  - 終了・停止文(STOP文、PAUSE文、RETURN文など)
- DOループ内でIF文による分岐命令を生成しない最適化（PREDICATE）が適用できる
- DOループ内に別名参照（EQUIVALENCE等）を含まない
- DOループの実行性能向上が見込める

-pvec ***擬似ベクトル化オプション***

```
% f77  program.f                    擬似ベクトル化する
% f77  -pvec    program.f           擬似ベクトル化する
% f77  -nopvec  program.f           擬似ベクトル化を抑止する
```

オプション無指定時でもf77では擬似ベクトル化オプションが有効（デフォルト）、f90では指定が必要。

-pvdiag ***擬似ベクトル化診断メッセージ***

```
% f77  -pvdiag  program.f           program.f を擬似ベクトル化してコンパイ
f77: compile start : program.f     ルしたときの診断メッセージを出力する

*OFORT77 V01-01-A entered.
*program name = MAIN
*end of compilation : MAIN
   KCHF1700C
       the do 20 loop is pseudo-vectorized.   （DO20 ループを擬似ベクトル化）
line=7
*program  units  =  0001,  no  diagnostics
generated.
%
```

オプションの詳細及び個別の擬似ベクトル化オプションにつきましてはマニュアル「最適化 FORTRAN77 使用の手引（6A30-3-314）」又は「最適化 FORTRAN90 使用の手引（6A30-3-311）」を御覧下さい。

要素並列化オプション

要素並列化とはプログラムを並列処理単位に分割して、ノード内の複数のプロセッサーで並列実行するためのオブジェクトを生成する機能です。メッセージ通信を行う並列化とは異なり、利用者が並列処理自体をコーディングすることなく、単一ノード内でプログラムを高速化することができます。以下に示す要素並列化レベルが設定されています。

要素並列化レベル

レベル 0	要素並列化をしない
レベル 1	SECTION 型要素並列化、強制 DO 型要素並列化
レベル 2	変数・配列のプライベート化、リダクション変数要素並列化
レベル 3	ループ分配、ループ分割、ループの一重化
レベル 4	パイプライン要素並列化、圧縮・拡張ループの要素並列化

高いレベルの最適化は、それより低いレベルの機能を含んでいます。

要素並列化変換にはオプション指示で DO ループを自動的に変換する DO 型要素並列化とパラメーター指示（POPTION）で文の集まりをプロセッサーに分配する SECTION 型要素並列化があります。DO 型要素並列化が適用される主な条件は以下の通りです。

- ✍✍ SECTION型要素並列化指示された範囲内にあるDOループでない
- ✍✍ DOループに以下の文を含まない
  - ループ脱出文(GOTO文、EXIT文など)
  - 割り当てGOTO文
  - 関数、手続き呼び出し
  - 入出力文
  - 終了・停止文(STOP文、PAUSE文、RETURN文など)
- ✍✍ DOループにNOPARALLEL指示がない
- ✍✍ DOループ内に同期制御又は排他制御指示がない
- ✍✍ DOループの実行性能向上が見込める

–parallel=要素並列化レベル ***要素並列化オプション***

% f77 -parallel program.f	要素並列化（レベル２）する
% f77 -parallel=4 program.f	要素並列化（レベル４）する
% f77 -noparallel program.f	要素並列化を抑止
% f77 -parallel main.o sub.o	要素並列オブジェクトをリンクする場合

–parallelオプションにレベル指定がない場合はレベル2（–parallel=2）を仮定。
–parallel リンケージオプションにはレベル指定は不要。

–pardiag ***要素並列化診断メッセージ***

```
% f77 -pardiag program.f                  program.f を要素並列化してコンパイルし
f77: compile start : program.f           たときの診断メッセージを出力する

*OFORT77 V01-01-A entered.
*program name = MAIN
*end of compilation : MAIN
*end of compilation : _parallel_func_1_MAIN
(diagnosis for loop structure)
   KCHF2000C
     the do 10 loop is parallelized.line=3    （DO10 ループを要素並列化）
   KCHF2011C
    the variable(s) or array(s) in do 10 loop  （DO10 ループ内の変数を TLOCAL 化）
is applied to tlocal transformation.name=I
line=3
*program units = 0001, no diagnostics
generated.
%
```

オプションの詳細及び個別の要素並列化オプションにつきましてはマニュアル「最適化 FORTRAN77 使用の手引（6A30-3-314）」又は「最適化 FORTRAN90 使用の手引（6A30-3-311）」を御覧下さい。

64 ビットアドレッシングモード

プログラムで使用するメモリーサイズが 2GB を超える場合には、64 ビットアドレッシングモードのオブジェクトモジュールを作成する必要があります。以下のオプションを指定してコンパイルします。また、オブジェクトモジュールをリンクする際にも指定が必要です。

–64 *64 ビットアドレッシングモードオプション*

% f77 -64 program.f	64 ビットモードでコンパイルする
% f77 program.f	32 ビットモードでコンパイルする
% f77 -32 program.f	32 ビットモードでコンパイルする
% f77 -64 main.o sub.o	64 ビットモードのオブジェクトをリンクする

以下は–64 オプション指定時の注意事項です。

- ✍✍ 定数（名前付き定数）及び DATA 文等により配列に初期値を与える場合（初期化項目）には、2,147,483,647 バイトを超えるデーターは使用できない。
- ✍✍ ASSIGN 文、割り当て GOTO 文、FMT 指定子の変数は整数型 8 バイトでなければならない。
  - 型変更は利用者自身で行なうか、オプション–intexp=full で全ての整数型 4 バイトの変数、配列、定数を整数型 8 バイトへ拡張（引数の整合性には注意）する必要がある。
- ✍✍ サービスサブルーチンの引数には、整数型 8 バイトは使用できない。
  - サービスサブルーチンを使用している場合には引数を整数型 8 バイトを整数型 4 バイトに変換するオプション–exsrvc を指定する必要がある。
- ✍✍ オプション–hugeary が仮定され、以下の組込み関数は整数型 8 バイトを返す。（f90 のみ）
  - LBOUND,SHAPE,SIZE,UBOUND,COUNT,MAXLOC,MINLOC

なお、2GB を超えるファイルサイズの入出力は非同期入出力文を除いて 32 ビットアドレッシングモードでも可能です。

ログメッセージ

ログメッセージ出力オプションを指定することで、コンパイル診断メッセージをファイルに出力することができます。要素並列化や擬似ベクトル化を適用した様子がソースプログラムに併記されるのでプログラムのチューニングが容易になります。

–loglist *ログメッセージ出力オプション*

% f77 -parallel -loglist program.f f77: compile start : program.f *OFORT77 V01-01-A entered. *program name = MAIN *end of compilation : MAIN *end of compilation : _parallel_func_1_MAIN *program units = 0001, no diagnostics generated % ls a.out program.f program.log	ログメッセージファイルを出力する ログメッセージファイルはソースプログラム毎に 「ソースプログラム名.log 」 というファイル名で出力される。 ログメッセージファイル program.log が作成される

【続く】

```
% cat  program.log                          ログメッセージファイル program.log の内容
       parameter (n=1000)
       real a(n),b(n),c(n)

** Parallel processing starting at loop entry   (並列処理開始)
** Parallel function: _parallel_func_1_MAIN
** Parallel loop                               (並列ループ)
** --- loops merged(DO10 and DO20)---           (DO10、DO20 ループ融合)
** Parallel processing finishing at loop exit
**
**     [DO 10]
**     Innermost loop unrolled (2 times).      (最内側ループ展開)
XX  there  is  no  target  for  pattern        (擬似ベクトル化適用せず)
referencing,PVP not applied.
**
 (略)
```

コンパイルメッセージやログメッセージは日本語で出力することができます。(デフォルトは環境変数 LANG に設定された文字コード種別です。)

–listlang=文字コード種別

```
% f77  -parallel  -loglist  program.f       ログメッセージファイルを出力する
       -listlang=sjis                        (オプション –listlang=sjis  を指定)
f77: compile start : program.f

*OFORT77 V01-01-A 開始                       コンパイルメッセージが日本語で表示される
*プログラム名 = MAIN                          (端末の文字コードを SJIS に設定すること)
*end of compilation : MAIN
*end of compilation : _parallel_func_1_MAIN
*プログラム数 = 0001 , エラーはありません。
% cat program.log                           ログメッセージファイル program.log が日本
       parameter (n=1000)                    語で作成される
       real a(n),b(n),c(n)

** ループ入口で並列処理 開始
** 並列手続き名 : _parallel_func_1_MAIN      文字コード種別の対応
** 並列ループ
** --- ループ融合を行った(DO10 と DO20)---      文字コード種別    言語(文字コード)
** ループ出口で並列処理 終了                       C              英語(ASCII)
**                                                SJIS           日本語(SJIS)
**     [DO 10]                                    EUC            日本語(EUC)
**     最内側ループ展開(2 倍)を行った。
XX    対象となる配列参照が無いので擬似ベクトル化を適用しなかった。
**
```

性能モニター

性能モニター情報出力オプションを指定することで実行時に性能、負荷等の情報を得ることができます。これらのオプションはコマンド行の最後(ファイル名の後)に指定して下さい。また、リンク時にも指定が必要です。

–pmfunc　　*性能モニター情報(関数、手続き)出力オプション*

```
% f77 -parallel  program.f  -pmfunc        性能モニター情報を出力する実行ファイルを
f77: compile start : program.f              作成
                                            性能モニター情報ファイルは実行ファイル実
*OFORT77 V01-01-A entered.                  行後に

                                                              【続く】
```

```
*program name = MAIN
*program name = SUB
*end of compilation : MAIN
*end of compilation : _parallel_func_1_MAIN
*end of compilation : SUB
*end of compilation : _parallel_func_2_SUB
*program units = 0002, no diagnostics
generated.
% a.out
  3.00000000        3000.00000
% ls
a.out
program.f
pm_a.out_Aug22_1625_node000_149513
% pmpr  pm_a.out_Aug22_1625_node000_149513
pm_a.out_Aug22_1625_node000_149513:
 (略)
##########################################
##  Function/Subroutine                 ##
##########################################
==========================================
==  Function/Subroutine  Ranking        ==
==========================================
    CPU-Time[%]    Times  Func(File+Line)
------------------------------------------
[1] 0.004126[93.08]  1   MAIN(program.f+3)
[2] 0.000011[ 0.25]  1   SUB(program.f+14)
----------------------
TOTAL  0.004137[93.32]
```

「pm_実行ファイル名_日付_時間
_ノード番号_プロセス番号 」
というファイル名で出力される。

このファイルはバイナリー形式のため、
性能モニター情報出力コマンド pmpr
を使用してテキスト形式で表示する。

実行ファイル a.out を実行する

性能モニター情報ファイルが出来ていること
を確認

pmpr コマンドで情報をテキスト形式で表示
（pmpr –j ファイル名とすると日本語 SJIS
で表示される）

性能モニター情報の内容については
スーパーコンピューティングニュース
Vol.2NO.4(2000.7)「SR8000 チューニング支
援機能を利用したプログラムチューニング」を
参照して下さい。

–p m p a r ***性能モニター情報（要素並列化）出力オプション***

```
% f77  -parallel  program.f  -pmpar
f77: compile start : program.f

*OFORT77 V01-01-A entered.
*program name = MAIN
*program name = SUB
*end of compilation : MAIN
*end of compilation : _parallel_func_1_MAIN
*end of compilation : SUB
*end of compilation : _parallel_func_2_SUB
*program units = 0002, no diagnostics
generated.
% a.out
  3.00000000        3000.00000
% ls
a.out
program.f
pm_a.out_Aug22_1647_node000_149551.
%  pmpr  pm_a.out_Aug22_1647_node000_149551
 (略)
##########################################
##  Element  Parallel  Region
##########################################
==========================================
==  Element Parallel Region Ranking     ==
==========================================
CPU-Time[%] MFLOPS MIPS Times Func[Rank]
------------------------------------------
[1]0.000006[ 0.03] 173.010 963.322 1 SUB[-]
[2] 0.000005[ 0.03] 392.912 1104.777 1 MAIN[-]
------------------------------------------
TOTAL  0.000011[ 0.06]
```

性能モニター情報を出力する実行ファイルを
作成

性能モニター情報ファイルは実行ファイル実
行後に
「pm_実行ファイル名_日付_時間
_ノード番号_プロセス番号」
というファイル名で出力される。
このファイルはバイナリー形式のため、
性能モニター情報出力コマンド pmpr
を使用してテキスト形式で表示する。

実行ファイル a.out を実行する

性能モニター情報ファイルが出来ていること
を確認

pmpr コマンドで情報をテキスト形式で表示
（pmpr –j ファイル名とすると日本語 SJIS
で表示される

性能モニター情報の内容については
スーパーコンピューティングニュース
Vol.2NO.4(2000.7)「SR8000 チューニング支
援機能を利用したプログラムチューニング」を
参照して下さい。

ソース差分部分コンパイル

ソース差分部分コンパイル機能はオブジェクトファイル中にソースプログラム情報を保持し、再度そのオブジェクトを出力先としてコンパイルしたときには、ソースプログラムの変更があったプログラム単位のみコンパイルする機能です。これによりコンパイル時間を短縮することができます。

–diffcomp *ソース差分部分コンパイルオプション*

```
% f77  -diffcomp  program.f                  差分コンパイルオプションを指定する
f77: compile start : program.f              通常と同様にコンパイルを開始

*OFORT77 V01-01-A entered.
*program name = MAIN
*program name = SUB
*end of compilation : MAIN                  プログラム MAIN コンパイル終了
*end of compilation : SUB                   プログラム SUB コンパイル終了
*program  units  =  0002,  no  diagnostics
generated.
% ls
a.out      program.f    program.o           オブジェクトファイル program.o および実
                                            ファイル  a.out が作成される

% f77  -diffcomp  program.f                  プログラム SUB の内容を変更後、改めて差分
f77: compile start : program.f              コンパイルする

*OFORT77 V01-01-A entered.                  プログラム MAIN に変更がないので構文チェ
*program name = MAIN                        ックのみ行う
*program name = SUB                         プログラム SUB は変更があったのでコンパイ
*end of compilation : SUB                   ルを行う
*program  units  =  0002,  no  diagnostics  メッセージ「*end of compilation : MAIN」がな
generated.                                  いことに注意
%
```

オブジェクト再コンパイル

オブジェクト再コンパイル機能はオブジェクトファイル中に保持しているソースプログラム情報を使用し、オブジェクトを入力として再コンパイルする機能です。

–recomp *オブジェクト再コンパイルオプション*

```
% f77  -diffcomp  program.f                  差分コンパイルオプションを指定する
f77: compile start : program.f              通常と同様にコンパイルを開始

*OFORT77 V01-01-A entered.
*program name = MAIN
*program name = SUB
*end of compilation : MAIN                  プログラム MAIN コンパイル終了
*end of compilation : SUB                   プログラム SUB コンパイル終了
*program  units  =  0002,  no  diagnostics
generated.                                  オブジェクトファイル program.o および実
% rm  program.f                             ファイル  a.out が作成される
% ls                                        試しにソースプログラム program.f を消し
a.out      program.o                        てみる

% f77  -recomp  -parallel  program.f        再コンパイルオプションを指定してコンパイ
f77: compile start : program.f              ル（参考のため–parallel オプションも追加）

*OFORT77 V01-01-A entered.                  オブジェクトモジュール内のソースプログラ
*program name = MAIN                        ム情報を利用して再コンパイルを行う
*program name = SUB
                                                                  【続く】
```

```
*end of compilation : MAIN                    プログラム MAIN コンパイル終了
*end of compilation : SUB                     プログラム SUB コンパイル終了
*program units = 0002, no diagnostics
generated.
%
```

制限コンパイル

通常コンパイラーは最大実行性能の実行ファイルを作成しようとするために、コンパイル時間やコンパイル時使用メモリー量は無制限に使用して最適化を行いますが、時間がかかり過ぎる場合やコンパイル時のメモリー不足の場合に、これらに制限を設けてコンパイルすることができます。

–limit *制限コンパイルオプション*

```
% f77  -limit    program.f          制限コンパイルを行う
% f77  -nolimit  program.f          制限コンパイルを行わない（デフォルト）
```

並列コンパイル

プログラム単位毎に別々のプロセスでコンパイルするため、ノード内の各プロセッサーを使用した並列コンパイルができます。

–Xpcomp *並列コンパイルオプション*

```
% f77  -Xpcomp  program.f          並列コンパイルを行う
```

言語仕様の拡張

他の FORTRAN 処理系でコンパイルできるプログラムが本センターの SR8000 でコンパイルエラーとなるとき、以下のオプション指定により対応できる場合があります。

–i,U –nosymnchk

変数名、外部手続き名が 8 文字を超える、またはアンダースコア（_）を含む場合は次のオプションを指定する必要があります。（f77 のみ）

–i,P

以下の項目について言語仕様を拡張します。（本センターではこのオプションをデフォルトで設定しています。）

- デバッグ行（D、？）、タブコード、注釈（！）の扱い
- 型編集（$型、NL 型、O 型、Q 型）
- 組込み関数（%VAL、%REF）
- 実定数が上限、下限値を超える場合の仮定処理
- DO　WHILE 文
- DOUBLE COMPLEX 型　等

–i,LT

ダブルクォーテーション（”）で囲まれた文字列を定数とみなします。（無指定時には ” 以降、行の最後までを注釈とみなします。）

デフォルトオプション

本センターでは幾つかのオプションをデフォルトで設定しています。オプション無指定時にも以下のオプションが仮定されますのでご注意ください。

f77: –i,P –opt=s –pvec –noparallel –symnchk

f90: –i,P

デフォルトオプションは環境変数 F77OPTS（又は F90OPTS）で利用者ごとに変更することができます。（特に必要がない場合には変更しないで下さい。）

```
% setenv  F77OPTS  " "                          デフォルトオプションを設定しない
% setenv  F77OPTS  " -pvec  -parallel "         デフォルトオプションの設定例
```

3. 実行

コンパイルして作成した実行ファイル a.out（または実行ファイル名）はコマンドとして実行することができます。

```
% a.out                                         プログラムの実行
```

sr8000-s では要素並列化したプログラムは実行できません。

バッチジョブ（NQS）で実行する場合にはスクリプトファイルを作成し、バッチシステムにサブミットします。

```
% cat  a.csh                                    スクリプトファイル例
#@$-q  single                                       要素並列プログラムのときは
#@$-lT  1:00:00                                       #@$-q  parallel
#@$-lM  2GB                                           #@$-N  1
cd  sample                                            #@$-lT  1:00:00
a.out                                                 #@$-lM  2GB
% qsub  a.csh                                   ジョブのサブミット
Request  6180.sr8000-s.cc.u-tokyo.ac.jp
  submited to queue:  single
% qstat  -h  sr8000-bt                          ジョブの確認
REQUEST     NAME      OWNER    QUEUE  PRI          （RUNNING は実行中）
6180.sr8000-s  a.csh   a30000  C       0
 NICE  CPU    MEM  STATE
 20    28800  7168  RUNNING
% ls                                            ジョブの結果
a.csh   a.csh.e6180   a.csh.o6180
% cat  a.csh.o6180                              ジョブが終了していれば標準出力ファイル(結
hello.                                          果）と標準エラー出力ファイル（エラー）の 2
% cat  a.csh.e6180                              つのファイルが作成されます。
%
```

複数のノードを使用した並列プログラムの場合には各アプリケーションによって実行方法が異なります。実行例は次号スーパーコンピューティングニュースでも紹介します。

資源制限値

ログインした環境（TSS）で使用できる計算機資源はデフォルト値によって制限されています。このため、プログラムのサイズによっては資源不足となる場合がありますので必要に応じて制限値を変更して下さい。ただし、ログイン環境自体がこれらの制限値の影響を受けますので無闇に大きく（または小さく）設定しないよう心掛けて下さい。

```
% limit                                  limit 資源制限（sr8000-p の例）

cputime         1:00:00                  CPU 時間制限値
filesize        unlimited
datasize        131072 kbytes            データーサイズ
stacksize       1024 kbytes              スタックサイズ
coredumpsize    0 kbytes
memoryuse       unlimited
descriptors     128 files
64datasize      262144 kbytes            データーサイズ（64 ビット使用時）
64stacksize     1024 kbytes              スタックサイズ（64 ビット使用時）
pgrpcputime     unlimited
addressspace    128 Mbytes               メモリー使用量
pgrpadrspace    unlimited
essize          512 Mbytes               （TSS で ES は使用できない）
pgrpessize      unlimited
threadcount     9 threads

% limit  addressspace  unlimited         メモリー使用量を指定できる最大値を設定
% limit  datasize   2048                 データーサイズに 2048MB を設定
% limit  cputime    7200                 CPU 時間制限値を 2 時間を設定
% limit  64datasize    3072              64 ビットモードの場合（3072MB を設定）
```

バッチジョブ（NQS）で使用する資源制限値はスクリプトファイルのオプション（#@$-）で設定します。

```
#@$-N   4                    #@$-N ノード数
#@$-lT  1:00:00              #@$-lT 実行時間制限（ETIME）
#@$-lM  2048MB               #@$-lM メモリー使用量
#@$-lV  1GB                  #@$-lV ES 使用量
#@$-lt  00:30:00             #@$-lt プロセス毎の実行制限時間（CTIME）
#@$-ld  1024MB               #@$-ld プロセス毎のデーターサイズ
#@$-ls  1024KB               #@$-ls プロセス毎のスタックサイズ
```

次号スーパーコンピューティングニュースでは複数ノードを使用した並列プログラムのコンパイルや実行の例、ライブラリー使用例、実行時オプション等を紹介する予定です。