OUKA
The University of Osaka
Institutional Knowledge Archive

Title	特集　11年度「速報」の主な記事
Author(s)	
Citation	大阪大学大型計算機センターニュース. 2000, 114, p. 3-35
Version Type	VoR
URL	https://hdl.handle.net/11094/66361
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

The University of Osaka

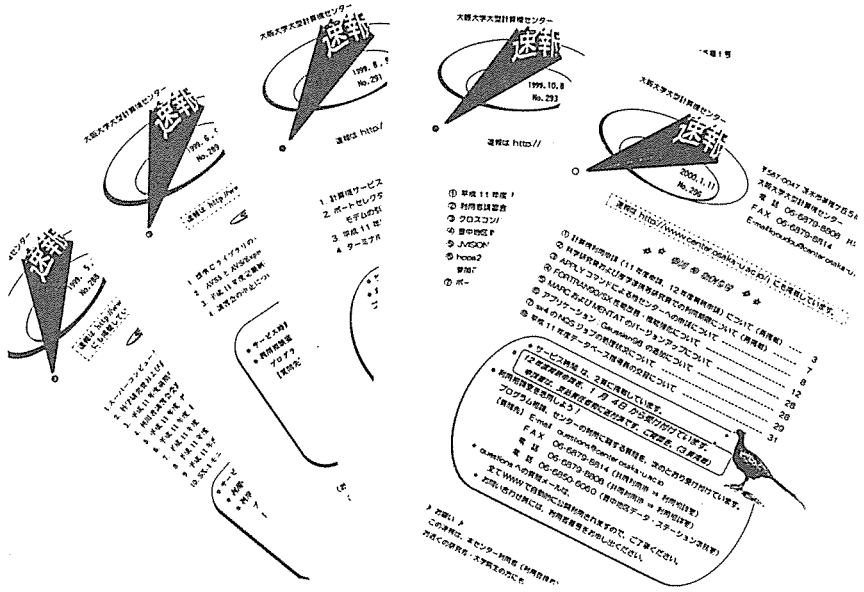
特集

平成11年度「速報」の主な記事

今月号は、本センターが平成１１年度に発行しました「速報」から、計算機システム利用に関する主な記事を特集としてまとめました。ご利用に際して、ご参考にしていただければ幸いです

（カッコ内は、発行日とNoです）。

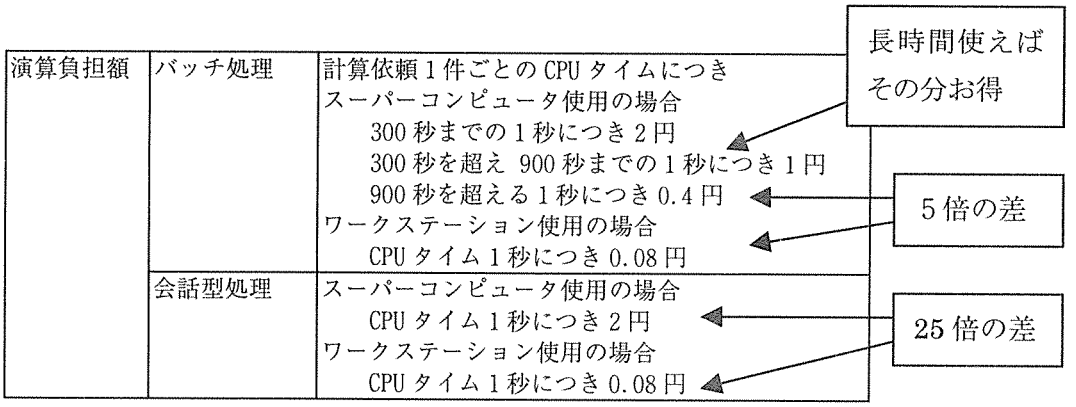
① スーパーコンピュータの効率の良い使い方　（1999. 5.11 No.288 ）　4頁
② 標準Cライブラリのprintf（3S）系関数における障害について
（1999. 6. 9　No.289）　7
③ AVS5とAVS/Express Vizの機能比較と移行について　（1999. 6. 9　No.289）　11
④ 各種媒体の変換について　（1999. 8 .9　No.291）　18
⑤ クロスコンパイラ用ASLライブラリについて　（1999.10. 8　No.293）　18
⑥ FORTRAN90/SX性能改善･機能強化について　（2000. 1.11　No.296）　19
⑦ MARCおよびMENTATのバージョンアップについて　（2000. 1.11　No.296）　35
⑧ アプリケーション：Gaussian98の追加について　（2000. 1.11　No.296）　35



① スーパーコンピュータの効率の良い使い方 (1999.5.11 No.288)

1．スーパーコンピュータの課金体系

スーパーコンピュータもその周辺ワークステーションを使うのも、CPU 資源を使うと課金が発生しますが、スーパーコンピュータとワークステーションでは課金について差別化をしてあります。



ここで言うバッチ処理とは、長時間におよぶ計算を行うときにする処理です。この処理は処理する一連の手順を書いたスクリプトとそれを投入するコマンド(NQS:詳しくはマニュアル参照)でジョブをスーパーコンピューターに投入して使います。このコマンド以外でスーパーコンピュータで作業すると、会話型処理扱いとなります。

会話型処理というのは、対話的にキーボードから入力して処理する事です。具体的にはエディタでの編集、コンパイル、デバッグ・・・などの処理のことです。しかし、この会話型処理ですが、一般ワークステーションとの課金差が25倍もあります。例えば、コンパイルに5分かかるとして、スーパーコンピュータで行うと、2×60×5=600円、その他のワークステーションだと0.08×60×5＝24円です。なぜ25倍も課金差があるのか？それは会話型処理は限られた計算機資源の多くを占有してしまう(当センターのスーパーコンピュータはベクトル型並列計算機なのでスカラー計算が苦手で資源を占有してしまいます)から割高なのです。ですから、このような会話型処理はスーパーコンピュータでやるのは非効率的で、周辺ワークステーションで行うのが理想的なのです。

2．スーパーコンピュータと周辺ワークステーションとの分業

前述の課金体系から、スーパーコンピュータは主計算のみをメインに使うと、課金は最小限に押さる事がわかります。しかし、スーパーコンピュータで動くプログラムを作るのに必要なコンパイル作業はスーパーコンピュータでしか出来ない・・・やはりスーパーコンピュータでの会話型処理の作業は必要不可欠である、と思われがちですが、周辺ワークステーションの中の 画像処理端末(vis) と、スーパーコンピュータ用フロントエンドコンピュータ(up02) にはクロスコンパイラ(*1)を用意しておりますので、スーパーコンピュータにログインして会話処理でコンパイルする必要はありません。さらに、コンパイル、デバッグ、プロファイルなど GUI(*2)で処理できる統合開発環境ソフトPSUITE(*3)もあります。プログラム開発をされる場合はぜひPSUITEをお使いください。

スーパーコンピュータにジョブを投入する場合は、これらのワークステーションに NQS コマンドを用意してありますので、一連のプログラム編集→コンパイル→ジョブ投入→デバッグという流れをスーパーコンピュータにログインすることなく作業することができます。

*1. 現在 f77 コマンドに相当する Fortran77 のクロスコンパイラはありません、ただし sxf90 コマンドは Fortran77 の上位互換を持っています。

*2.GUI(Graphical User Interface) 文字主体の操作方法でなく、アイコン等を使った操作方法。ネットワーク経由で使うには X ウィンドウシステムが必要です。

*3. 詳しくはマニュアルもしくは http://www.center.osaka-u.ac.jp/j/manual/psuite/ 参照してください。ネットワーク経由の利用には X ウインドウシステムの環境が必要です。

3．実際の効率の良い計算作業の一例

スーパーコンピュータを実際に効率よく(課金を少なく)使うためには、スーパーコンピュータでの作業を必要最小限にします。以下にその例を示します。

1. 画像処理端末(vis??) と、スーパーコンピュータ用フロントエンドコンピュータ(up02)のどちらかに接続します。

区分／マシン	ホスト名	ＩＰアドレス	備考
画像処理端末 HP Visualize C200	vis01 vis02 vis03 vis04 vis05 vis06 vis07 vis08 vis09 vis10 visd01 visd02	133.1.4.162 133.1.4.163 133.1.4.164 133.1.4.165 133.1.4.166 133.1.4.167 133.1.4.168 133.1.4.169 133.1.4.170 133.1.4.171 133.1.8.130 133.1.8.131	 豊中 DS 豊中 DS
スーパーコンピュータ用 フロントエンドコンピュータ NEC UP4800/760	up02	133.1.4.246	

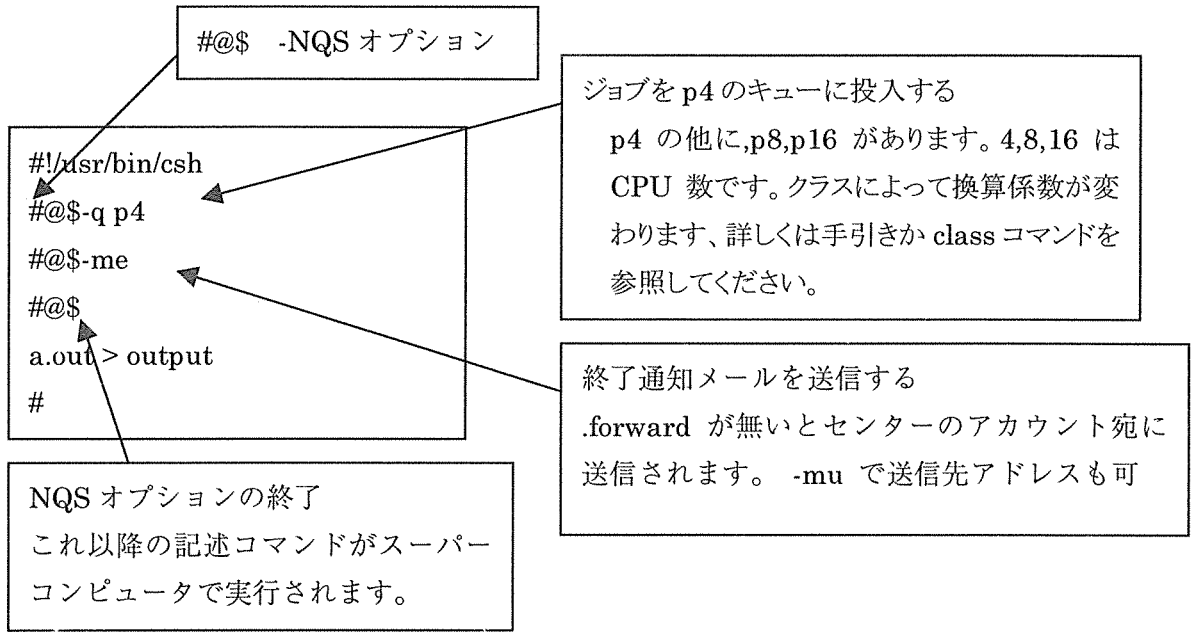
＊すべて計算機センター外から利用の場合、ホスト名.center.osaka-u.ac.jp が必要

2. プログラムに必要なソースファイルなどをエディタで編集します。vis と up02 には mule というエディタがあります。(スーパーコンピュータの emacs と同等の機能)があります。

   ＊その他にもスーパーコンピュータには無い GNU 版 tar や tcsh 、画像関係などの PDS が用意されています。

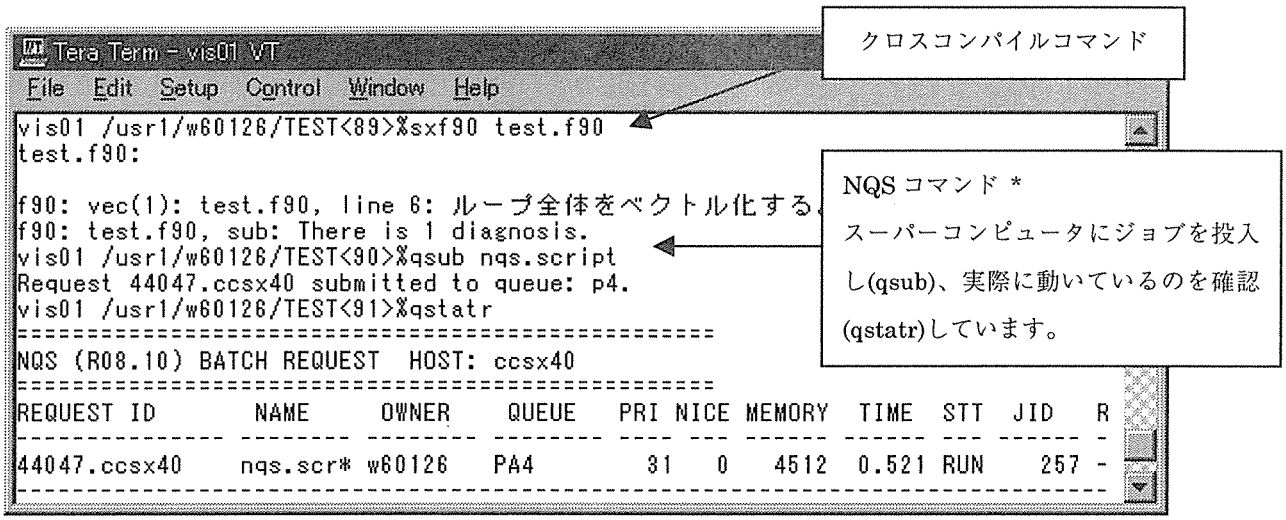
3. コンパイルを行います、ただし通常の f90,cc コマンドでコンパイルを行うとスーパーコンピュータでは実行不能となります。必ず、sxf90,sxcc,sxld のクロスコマンドを実行してください。(注意:IMSL 等特別なライブラリを使用する場合はスーパーコンピュータでコンパイルする。)

4. スーパーコンピュータのジョブ投入用スクリプトを用意します。このスクリプトを NQS コマンドを使って投入します。これがバッチ処理です。以下にスクリプトの例と解説を示しますので参考にしてください。



5. NQS コマンドの投入例

以下の図のように NQS コマンドでジョブをスーパーコンピュータに投入します。あとは計算が終了するのを待ち研究室で終了通知メールを待ちます。



* NQS コマンドの概説

コマンド	解説	使用する場面
qsub *<file>*	バッチリクエストの投入	ジョブを SX4 に投入しバッチ処理させるとき
qstatr	リクエストの状態表示	投入したジョブの状態を知りたいとき
qdel	リクエストの削除	投入したジョブを途中で止めたいとき
qstatq	キューの状態表示	SX4 のキューの混雑状態を見たいとき

コマンドの詳細は SX4マニュアル、もしくはオンラインマニュアルを参照

② 標準Cライブラリの Printf（3S）系関数における障害について

(1999.6.9 No.289)

SX-4 Cプログラムにおいて標準Cライブラリの printf(3S)系関数に以下の障害が発生することが判明いたしました。障害の発生期間、内容と障害の見え方、発生条件および、障害の原因をお知らせいたします。

本障害については、平成１１年４月５日(クロス開発環境については平成１１年４月２７日)に printf(3S)系関数の修正物件を適用済みです。

１．障害発生期間

平成９年１月６日 ～ 平成１１年４月５日

(クロス開発環境版については、平成１１年４月２７日まで)

上記期間中に SX-4 あるいはクロス開発環境でコンパイル・リンクされたCプログラムの実行形式モジュールは以降で説明する条件を満たしている場合に本障害の対象となりますので、再度コンパイル・リンクをお願い致します。

２．障害内容と見え方

float0 モード(IEEE 浮動小数点形式)を利用し、printf(3S)系関数で変換指示子%f(%F), %g(%G)を指定して float 型（単制度浮動小数点), double 型（倍精度浮動小数点）データを出力（変換）する場合に、出力データを浮動小数点データでない整数や不正な文字列として出力することがあります。本障害の対象となる printf(3S)系関数を表１にまとめます。

表１ 障害の発生する printf(3S)系関数一覧

ライブラリ	関数一覧
libc.a	**Fprintf, printf, sprintf, snprintf, vfprintf, vprintf,vsnprintf, vsprintf, fwprintf, swprintf, wprintf, vfwprintf, vswprintf, vwprintf**
libw.a	**Printf, fprintf, sprintf**

次に、本障害が発生した場合の見え方について説明します。ここでは、IEEE 浮動小数点形式の非数(NaN:Not a Number, infinity)と非数以外の浮動小数点データを単に「浮動小数点データ」と呼び区別します。また、printf(3S)系関数の%f(%F)指定で出力される形式を浮動小数点形式、%g(%G)指定で出力される形式を指数形式と呼んで区別します。

障害の見え方には２通りの見え方があります。これは printf(3S)系関数に指定した引数の値に依存します。

(1) 引数の値が浮動小数点データの場合

通常は浮動小数点形式(%f,%F)または、指数形式(%g,%G)で出力されますが、必ず整数として出力されます。図1に例を示します。不正結果の%f, %g の出力結果が異なるのは printf(3S)内の処理の違いによるものです。

図1　浮動小数点データの例

```
d = 0.00005;
printf("value1 = %g¥n", d);
printf("value2 = %f¥n", d);
```

正しい結果

```
value1 = 5e-05
value2 = 0.000050
```

不正結果

```
value1 = 500000
value2 = 50
```

(2) 引数の値が非数の場合

通常は"nan", "inf" (または、"infinity")などの文字列で出力されますが、不正な文字列 "0.nan", "0.inf"として出力されます。図2に例を示します。正しい結果では%f,%g の出力結果は同じとなりますが、障害発生時は%f,%g で結果が異なります。これは、%f, %g の既定値の精度は共に6桁ですが、%g は後尾のゼロを表示しないためです。

図2　非数(NaN, infinity)の障害の見え方

```
d = strtod("-nan", NULL);
printf("value1 = %g¥n", d);
printf("value2 = %f¥n", d);
```

正しい結果

```
value1 = -nan
value2 = -nan
```

不正結果

```
value1 = -0.nan
value2 = -0.nan000
```

```
d = strtod("-infinity", NULL);
printf("value1 = %g¥n", d);
printf("value2 = %f¥n", d);
```

正しい結果

```
value1 = -inf
value2 = -inf
```

不正結果

```
value1 = -0.inf
value2 = -0.inf000
```

本障害は、float0 モードの long double 型データ（4倍精度浮動小数点）や float1 モード, float2 モードを利用するプログラムに影響ありません。また、本障害はプログラムの計算途中に誤りがあるのではなく、計算結果を出力する場合に誤りがあります。計算結果を数値データのままファイルなどに保存されている場合には影響ありません。

3．発生条件

障害発生の条件は次のようにまとめられます。下記条件(a),(b),(c)をともに満足する場合に障害が発生する場合があります。「場合がある」という表現は条件(c)の不正メモリの内容に依存して障害が発生するためです。

(a) float0 モード(IEEE 浮動小数点形式)を利用する場合
(b) 表 1 に述べた printf(3S)系関数を使用し変換指示子%f(%F), %g(%G)を指定して float 型（単精度浮動小数点), double 型（倍精度浮動小数点）データを出力（変換）する場合
(c) 条件(a),(b)を満足した場合、printf(3S)系関数が不正なメモリ内容を参照し、そのメモリ内容によって障害が発生します。

不正なメモリ内容にどのように依存するかについては 4．障害原因をご参照ください。

4．障害原因

障害の原因は、printf(3S)系関数の共通処理の中で出力形式変換を判断する処理に誤りがあり引数の値が浮動小数点データであるのに非数と判断したり、非数を浮動小数点データと判断することがあるためです。図 3 に printf(3S)系関数の処理概要を示します。printf(3S)系関数では先ず、引数の値を文字列展開します（処理①)。次に引数の出力形式変換の有無を決定（処理③）するため引数の非数(NaN, infinity)検査を実施します（処理②)。引数が浮動小数点データの場合（処理④）は、小数点位置の位置決め浮動小数点形式や指数形式への変換処理を実施し、引数が非数の場合（処理⑤）は、再度"nan", "infinity"の文字列を処理します。本障害では、処理②の検査に誤りがあり処理③の判断で引数が浮動小数点データであるのに非数と判断し出力形式変換を実施せずに整数で出力したり、引数が非数であるのに浮動小数点データと判断し文字列に対して不要な小数点位置決めを実施していました。

図 3　printf(3S)関数の処理概要

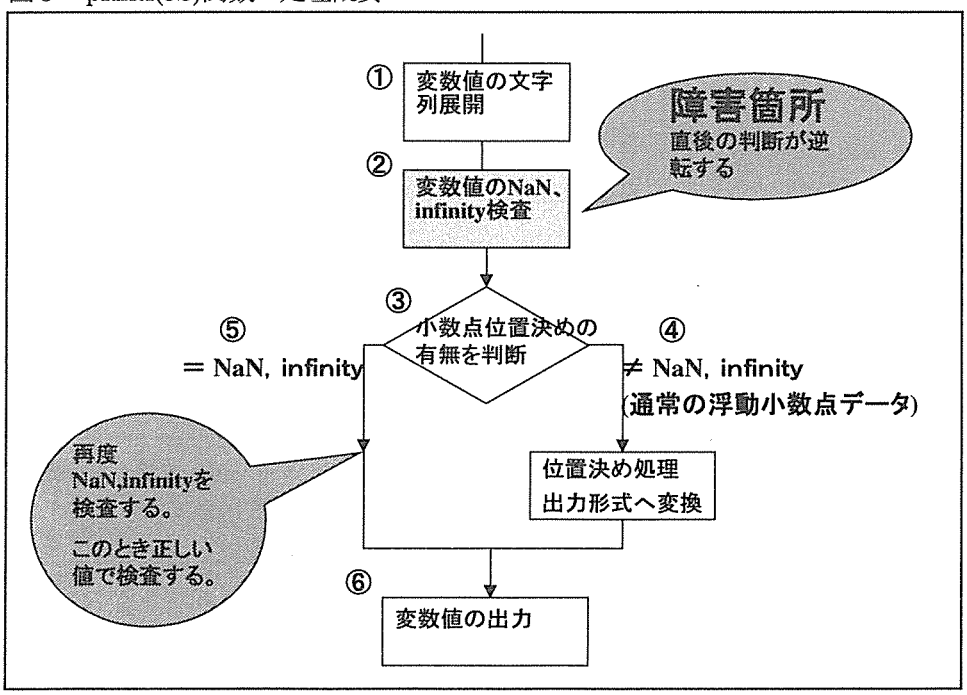


図４にprintf(3S)系関数の不正メモリ参照のイメージを示します。図は、ユーザプログラムがprintf()関数を呼び出し、printf()関数から非数の検査関数 chk()を呼び出した状態を表します。図の右側は関数のスタック割り当てと変数の割り当て状態を示します。printf()関数はユーザプログラムからの引数 z1 を変数ｄに保存し、保存した値を chk()関数に指定しますが、このとき変数ｄの代わりに誤って変数 ld を chk()関数に渡していました。変数ｄと ld はメモリを共有しますが、変数 ld は変数ｄの後８バイト（図中点線の部分）を余分に参照します。つまり double 型データを誤って long double 型データとして chk()関数に指定していました。このとき chk()関数の引数の型は double 型のため関数呼び出し処理の中で不定メモリの内容を含む long double 型データから double 型データへのキャストが発生し、ユーザプログラムから渡された引数の値と異なる値を chk()関数へ渡し非数の検査を実施していました。chk()関数の検査は、検査する浮動小数点データの指数部の値により浮動小数点データ，非数を判定します。指数部(11bit)のビットが全て ON(0x7ff)のとき非数と判断します。従って、ユーザプログラムからの引数 z1 の値と不正に参照するメモリ（8byte）の内容に依存して障害が発生することになります（図５参照)。

図４　不正メモリ参照のイメージ

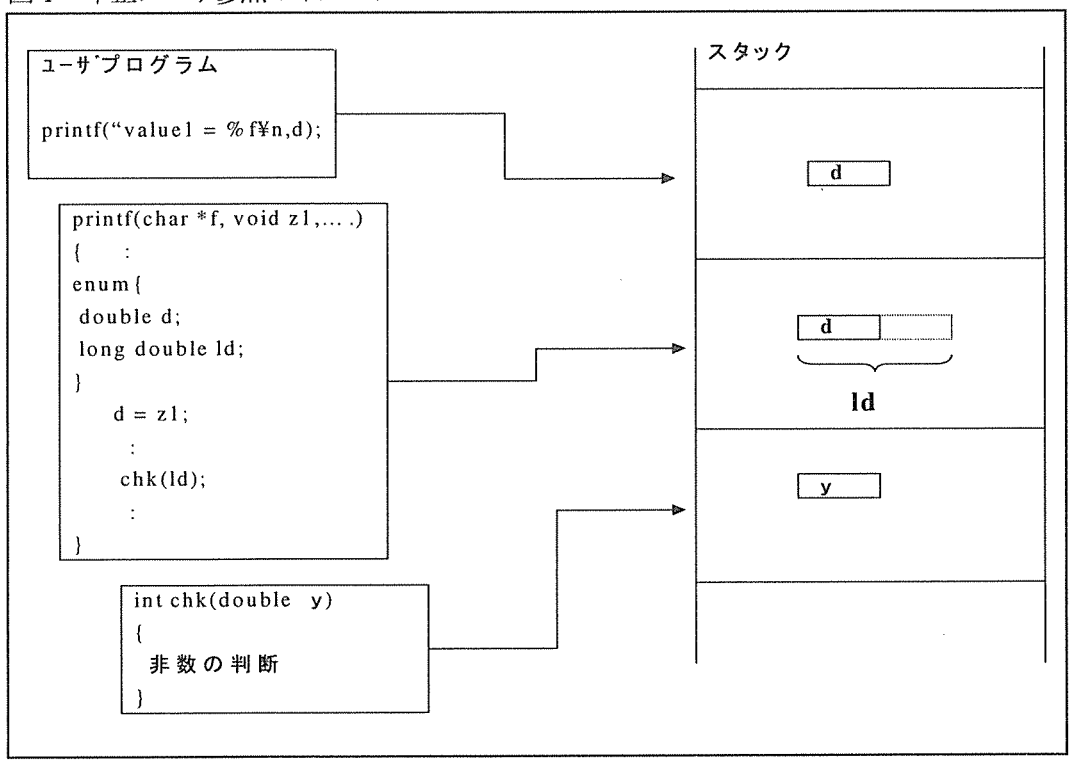
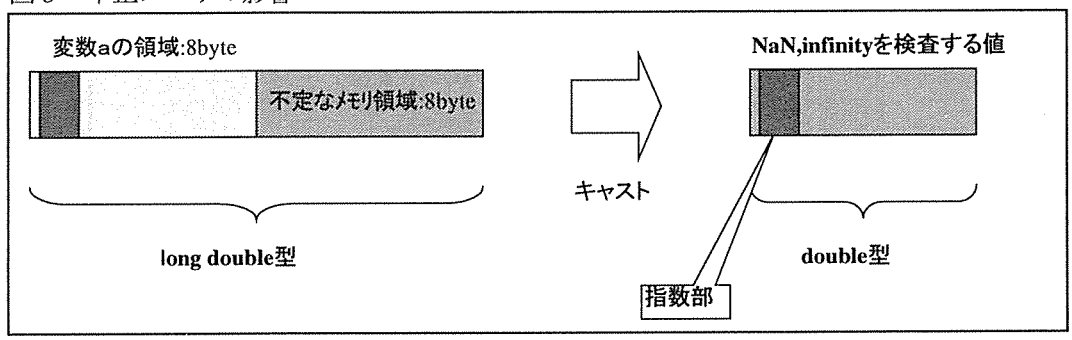


図５　不正メモリの影響



③ AVS5とAVS/Express Vizの機能比較と移行について

(1999.6.9 No.289)

1．はじめに

可視化アプリケーションAVSの後継ソフトであるAVS/Expressが登場してから、5年が経ちました。AVS/Expressは、従来のAVSと共通点も多い反面、アーキテクチャーが異なるため、従来のAVSの資産をそのまま移行することはできません。しかし、一方で、従来のAVSにはなかった様々な機能があります。Windows PCでの利用も可能です。

AVS/Expressには2つのエディションがあります。一つは、Visualization Editionです（通称AVS/Express Viz)。これは、一般エンド・ユーザーのための可視化ツールになっています。もう一つは、Developer Edition(通称AVS/Express Developer)です。これは、本格的なアプリケーションを開発するためのツールです。従来のAVSと同じ位置づけにあるのがAVS/Express Vizです。

ここでは、AVSのバージョン5（AVS 5）とAVS/Express Vizを比較するとともに、どのように移行すればよいかについて説明します。

2．AVS/Express Vizの機能拡張

AVS 5からの主な機能拡張について紹介します。

（1） 時系列データへの対応

フィールド・ファイル・フォーマットとUCDファイル・フォーマットが時系列（複数ステップ）用に拡張されています。読み込みモジュールも拡張され、このファイルを連続的に読み込み、アニメーション表示することが可能です。

（2） フィールド・データとUCDデータの統一

AVS 5では、モジュール間を受け渡しするデータとして、フィールド・データとUCDデータがあり、それぞれ、別々のモジュールが処理していましたが、AVS/Expressでは、両者ともフィールドというデータ・タイプに位置づけられ、前者を構造化フィールド、後者を非構造化フィールドとしています。その結果、いくつかのモジュールは、同じ入力ポー

トから両方のタイプのデータを入力し、同じ処理を行います。

（3） NULL データ

ある値を NULL データとして設定することができ、いくつかのモジュールは NULL データを除外して処理します。例えば、流体計算で障害物がある場合は、そこの格子点の値を特定の値（例えば 0 や-999）にしておき、その値を NULL に設定することにより、障害物を抜いて表示することができます。図１は、タービン翼の回りの流れの可視化で、黒抜きのところがタービン翼です。

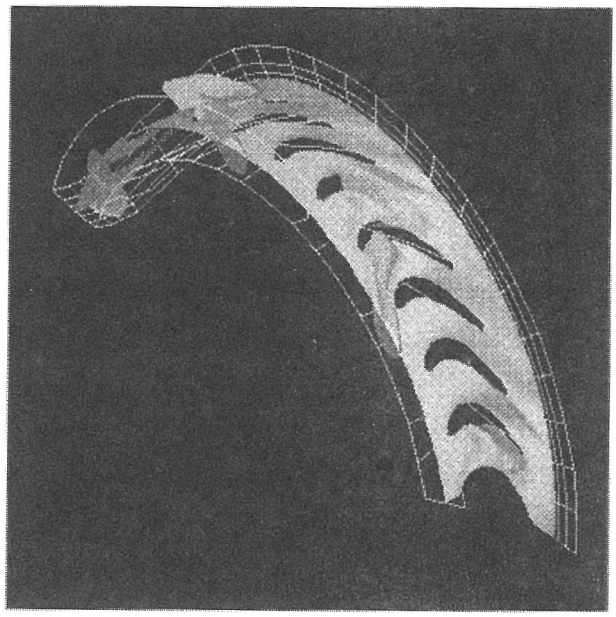


図１ ＮＵＬＬデータを使った障害物の処理

（4） ２次元データに対する処理の強化、モジュールの組み合わせの多様化

AVS5 では、多くの可視化モジュールが３次元データのみを対象としていましたが、AVS/Express では３次元と２次元の両方のデータを扱うモジュールが増えました。また、モジュールの出力に関しても、AVS5 では、幾何形状データを出力するか、フィールド・データ（または UCD データ）を出力するかのどちらかでしたが、AVS/Express はその両方を出力します。その結果、モジュールの組み合わせが多様になりました。

例えば、図２のような任意断面のライン・コンター(等数値線)表示が可能です。AVS5 の arbitrary slicer モジュールは幾何形状データのみを出力するため、断面のシェーディング・コンターのみを表示するだけでしたが、AVS/Express の slicer モジュールは断面を 2 次元のフィールド・データとして出力し、それを入力した isoline モジュールが断面上にライン・コンターを表示します。

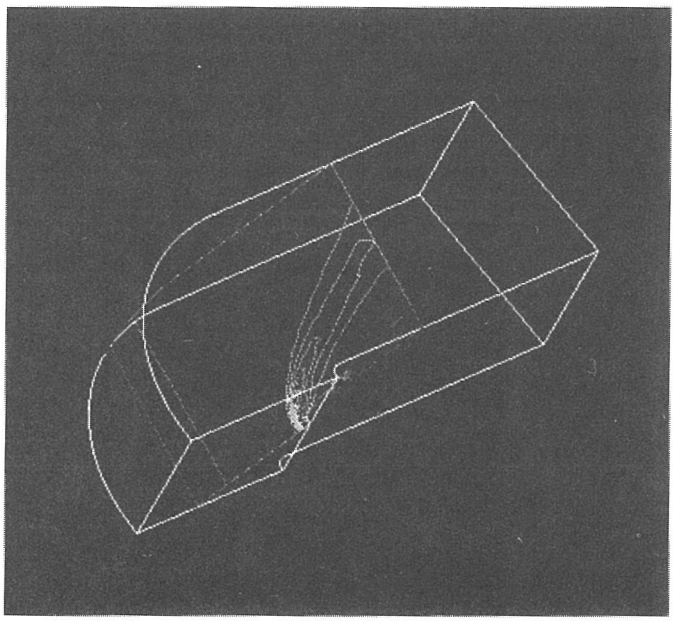


図2 任意断面のライン・コンター表示

（5） グラフ・モジュールの強化

グラフの部品モジュールが提供されています。これを組み合わせることにより、多様なグラフ表示が可能です。

（6） イメージ・フォーマットの強化

AVS 独自のイメージ・フォーマット以外に以下のイメージ・フォーマットに対応しています。

Bitmap, GIF, JPEG, PBM, SGI Image, SUN Raster, Tiff, PostScript

ベクターPostScript にも対応しています。また、動画ファイルとして、MPEG や AVI を作成することも可能です。

（7） 2次要素のサポート

UCD データで2次要素をサポートしています。シェーディング・コンターやライン・コンターは中間節点の値を反映して表示されます。

（8） GUI 部品の提供

ボタンやスライダーなどの GUI（グラフィカル・ユーザー・インターフェース）を作成するためのモジュールが提供されています。これらを使えば、様々な GUI を簡単に構築することができます。

（9）　その他の新しいモジュール

新しく追加されたモジュールとして以下のようなものがあります。

- ソリッド・コンター

  ノード間を滑らかに補間するのではなく、一定の範囲の値で色を塗り分けたコンター表示をします。（図３）

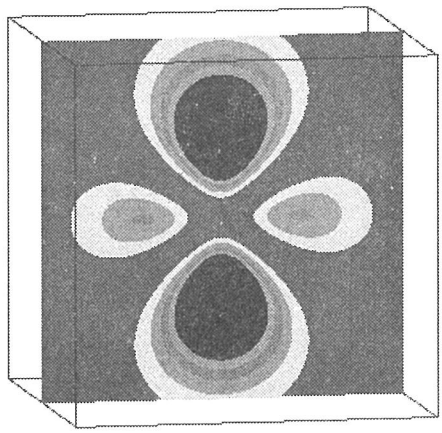


図３　ソリッド・コンター表示

- isovolume

  isovolume は等数値面（isosurface）と似ていますが、特定の値の領域だけでなく、ある値以上（または以下）のすべてボクセルを表示します。図４は水素分子の電子密度分布の表示ですが、外部境界に接触するような等数値面では、そこで穴が空いた状態になるのに対して、isovolume は蓋をします。

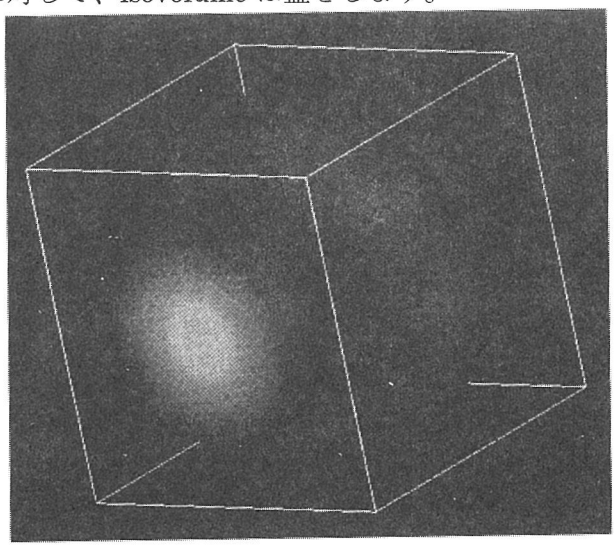


図４　isovolume 表示

- GIS モジュール

  地図情報処理を行ういくつかのモジュールがあります。世界の地図データを生成することのできる GISWorldDataBank や、それを投影するための GISMapTransform などがあります。

- data math, coordinate math モジュール

  フィールド・データの各データ成分や座標値の各成分に対する四則演算を行うことができます。例えば、data math では、パラメータとして次のように数式を記述できます。

  #1 + #2 * 5

  これは、1番目の入力ポートのデータに2番目の入力ポートのデータを5倍したものを足して出力することを意味します。

- VRML 出力

  幾何形状データを VRML フォーマットで出力することができます。

3．機能比較表

以下は、AVS/Express Viz と AVS5 を機能を大まかに比較したものです。機能に関しては差のある項目のみリストアップしています。

	AVS5	AVS/Express Viz
プラットフォーム		
Super Computer	◎	○ (NEC のみ)
UNIX ワークステーション	◎	◎
Windows PC	×	◎
Windows Linux	○ (Intel のみ)	×
機能		
リモート・モジュール機能	◎	×
時系列データ	×	◎
NULL データ	×	◎
2次元データの可視化	△	◎
グラフ	△	○
イメージ・ファイル	△	◎
動画ファイル	×	◎
2次要素データの可視化	△	○
GUI 構築	○	◎
ソリッド・コンター	×	◎

isovolume	×	◎
GIS	×	○
データ値、座標値の四則演算	×	◎
VRML 出力	×	○
パーティクル・トレース	◎	○ (流線表示に問題あり)
カラムデータの読み込み	○	◎ (ツールあり)
日本語メニュー	×	○

＊ ◎優れている ○機能あり △問題あり ×機能なし

4．AVS5 から AVS/Express への移行

AVS5 のユーザーが AVS/Express Viz を利用するにあたって、いままでの資産をどのように移行すればよいかについて説明します。ここでの内容は、AVS/Express Viz と Developer の両者に共通ですので、AVS/Express への移行として説明します。

(1) データ・ファイル

AVS5 のファイル・フォーマットは AVS/Express と共通です。AVS5 で読み込むために AVS のフィールド・ファイル、UCD ファイル、ジオメトリ・ファイル、およびイメージ・ファイルを作成していれば、そのまま、AVS/Express Viz で読み込むことができます。

(2) モジュール

UNIX 版に限り、AVS/Express 上でほとんどの AVS5 のモジュールが動作します (AVS 5 のモジュールがそのマシンにインストールされている必要があります)。自作のモジュールについても同様です。ただし、CLI など一部の機能は使用できません。また、AVS 5 上で実行するのに比べてパフォーマンスが落ちます。

UNIX 版に限り、AVS 5 で保存したネットワーク・ファイルをそのまま読み込むことができます。このときは、AVS_Compat のメニューを使用します。各モジュールがインストールされていれば、多くの場合、AVS/Express 上で再現することができます。

AVS 5 のモジュールを AVS/Express 上で利用したいケースとしては、AVS5 での可視化に AVS/Express の新しいモジュールを加えたいケースが考えられます。AVS 5 のデータ構造と AVS/Express のデータ構造が異なるため、AVS 5 のモジュールと AVS/Express のモジュールを混在して使用する場合は、データ変換のためのモジュールを使用する必要があります。

基本的には、AVS/Expressを利用する際は、AVS 5のモジュールではなく、AVS/Expressのモジュールでネットワークを組み直すことをお勧めします。AVS/Express には、仕様が多少異なりますが、一部を除き、AVS 5で提供されているものに対応するものが用意されています。これを組み合わせて、ネットワークを作り直すことになります。

（3） モジュール開発

UNIX 版であれば、前述のように、多くの場合 AVS 5での自作モジュールをそのまま実行することはできますが、パフォーマンス向上や、AVS/Express の機能を有効に使うためには、AVS/Express のモジュールとして作り直すことをお勧めします。残念ながら、モジュールのプログラミング方法は、AVS5 と AVS/Express では異なりますが、AVS 5で実現できたことはAVS/Expressでも実現可能です。

（AVS 5での形状出力のための GEOM 関数に相当するものが AVS/Express ではありません。しかし、１１月頃にリリース予定の AVS/Express 4.3 では提供される予定です。）

5．日本語マニュアル

日本語マニュアルとして、以下のものがCD-ROM（PDF ファイル）で無償提供されています。

- チュートリアル・ガイド

  基本操作や代表的なモジュールの利用例が紹介されています。

  （AVS/Express 4.2 からは新規購入には製本版がバンドル提供されています。）
- ユーザーズ・ガイド

  ファイル・フォーマットや操作方法の詳細な説明があります。
- デベロッパーズ・ガイド

  モジュール開発に関する説明があります。（AVS/Express 4.2 から登場）

（（株）ケイ・ジー・ティー　吉川 慈人）

④ 各種媒体の変換について

(1999.8.9 No.291)

本センター本館2階のワークステーション室には、各種媒体の変換装置を設置してあります。これらの装置を用いることにより色々な媒体からの入出力が行え、データをセンターのunix機より利用することが可能です。また、フォーマットには時間のかかる場合があるので、なるべくフォーマット済みの媒体をお持ちください。

	Win95	MacOS	unix	備考
4mm DAT (DDS データ カートリッジ)			1.3GB	DDS
			2GB	DDS
			4GB	DDS2
			12GB	DDS3
8mm データカートリッジ			5GB	112m
			7GB	160m
CD	○	○	○	
CD−R	○			
CD−RW	○			
1/4インチCGMT			525MB	
F D 3.5インチ	640KB		640KB	
	720KB	720KB		
	1MB		1MB	unix:5.25インチも対応
	1.44MB	1.44MB	1.44MB	
MO	128MB	128MB		Win95:読みのみ MacOS:他機互換保証なし
	230MB	230MB		
	540MB	540MB		
	640MB	640MB		
OpenMT			○	IBMフォーマット
PD	640MB			
ZIP	100MB	100MB		

〈システム管理掛〉

⑤ クロスコンパイラ用ASLライブラリについて

(1999.10.8 No.293)

ASLライブラリがSX4のクロスコンパイラから利用できるようになりました。これにより従来はSX4で行っていたASLライブラリのリンク処理もスカラー処理の得意なワークステーションで行えます。なお、クロスコンパイラが用意されているホストはvis01～vis10,visd01,visd02,ccup02です。

```
/usr1/local/sx/lib0   float0
/usr1/local/sx/lib1   float1
/usr1/local/sx/lib2   float2
/usr1/local/sx/lib    lib0 へのリンク
```

コンパイル例 (float0)

```
sxf90 -o  samp  samp.f   /usr1/local/sx/lib/asl
```

(システム管理掛)

⑥ FORTRAN90/SX性能改善・機能強化について (2000.1.11 No.296)

FORTRAN90/SX のバグ修正を含むコンパイラのリビジョンアップに伴い、以下の性能改善及び機能強化を併せて行いましたのでお知らせします。

FORTRAN90/SX コンパイラにおいて強化した内容とそれ以前に改善・強化した部分を含んで説明します。

まず、以下の改善を行っています。

・コンパイル性能改善
・行列積ライブラリ呼び出しを強化
・文字型比較および文字型代入の実行性能改善
・ベクトル化の解析を強化
・組込み関数 MODULO、RESHAPE、TRANSFER および SPREAD の最適化を強化

その他、以下の機能強化を行っております。

(1) 最適化診断メッセージの追加

ループ変形などの最適化や自動並列化が行われなかったときの理由を表示する、以下の最適化診断メッセージが追加されました。

No.	メッセージ
1011	Too large to optimize -- reduce program or loop size. ループまたはプログラム単位が大きすぎる。
1017	Subroutine call prevents optimization. サブルーチン呼出しがあるため最適化できない。
1019	Feedback of scalar value from one loop pass to another. スカラ変数が異る繰り返しで定義した値を参照している。
1025	Reference to this function inhibits optimization. 最適化を阻害する関数呼出しがある。
1033	Potential multiple store conflict -- use directive if OK. 同一の配列要素に対して定義が複数回行われる可能性がある。 (nosync/nodep を指定すれば最適化を行う)
1034	Multiple store conflict. 同一の配列要素に対して定義が複数回行われる。
1036	Potential feedback - use directive or switch if OK. 異る繰り返しで定義された値を参照している可能性がある。 (nosync/nodep を指定すれば最適化を行う)
1037	Feedback of array elements. 異る繰り返しで同一の配列要素を定義／参照している。
1038	Loop too complex -- optimization of this loop halted. ループが複雑なため最適化できない。
1041	Internal error detected -- please report. 最適化の内部エラーが発生した。
1056	Loop nest too deep for optimization. ループのネストが深すぎるため最適化できない。
1057	Complicated use of variable inhibits loop optimization. 変数の定義／参照が複雑で最適化できない。
1059	Unable to determine last value of scalar temporary. スカラ変数の終値が確定できない。
1061	Use of scalar under different condition causes feedback. スカラ変数が異る条件下で参照されている。
1062	Too many data dependency problems.

データ依存が多すぎるため最適化できない。

1082 Backward transfers inhibit loop optimization.
逆方向分岐があるため最適化できない。

1083 Last value of promoted scalar required.
作業配列化されたスカラ変数の終値保証が必要である。

1084 Branch out of the loop inhibits optimization.
ループからの飛び出しがあるため最適化できない。

1097 This statement prevents loop optimization.
最適化を阻害する文である。

1117 Indirect branch inhibits optimization of loop.
間接分岐があるため最適化できない。

1118 This I/O statement inhibits optimization of loop.
最適化を阻害する入出力文がある。

1128 Branching too complex to optimize at this optimization level.
分岐が複雑で最適化できない。

1130 Conditional scalar inhibits optimization of outer loop.
条件下で定義されるスカラ変数が外側ループの最適化を阻害している

1131 User function references in iteration count inhibits optimization.
利用者定義の関数参照が最適化を阻害している。

1166 Equivalence of scalar inhibits optimization -- use directive if ok.
EQUIVALENCE されたスカラ変数が最適化を阻害している。
(NODEP を指定すれば最適化を行う)

1268 Use of pointer variable inhibits optimization.
ポインタ変数があるため最適化できない。

1282 This store into array inhibits optimization of outer loop.
配列への代入が外側ループの最適化を阻害している。

1285 Not enough work to justify concurrency optimization.
ループ中の作業量が少ないため自動並列化しない。

1298 Use of loop index outside the loop inhibits optimization.
ループの指標変数がループ外で参照されているため最適化できない。

1299 Redefinition of loop index in loop inhibits optimization.
ループの指標変数がループ内で再定義されているため最適化できない。

1300 Assumed-size private arrays inhibit concurrency.
大きさ引継ぎ配列があるため並列化できない。

1339 User parallel directives inhibit optimization.
並列化制御オプションが指定されているため最適化しない。

1376 User function reference inhibits optimization.
利用者定義の関数参照があるため最適化できない。

1377 Must synchronize to preserve order of accesses.
同期制御が必要である。

1378 Too many synchronizations needed.
同期制御が多数必要なため並列化しない。

1380 User function references not ok without CNCALL directive.
利用者定義の関数参照があるため並列化できない。

1382 Subroutine calls are handled only when CNCALL directive is used.
サブルーチン呼出しがあるため並列化できない。

1387 Overlapping EQUIVALENCEd variables inhibit concurrency.
EQUIVALENCE された変数間に重なりがあるため並列化できない。

(2) f90 詳細オプション-P に次の b/nb サブオプションを追加しました。
-P b 実引数の文字定数の末尾に 0x20(空白文字)を追加する。
-P nb 実引数の文字定数の末尾に 0x0(ヌル文字)を追加する(既定値)。

(3) 最適化診断メッセージ追加

配列代入を融合したことを示す次の最適化診断メッセージを追加しました。

メッセージ番号： 11
和文メッセージ： 配列代入を融合した。 :line /1/ - /2/
英文メッセージ： Fused array assignments. :line /1/ - /2/
説明： 連続した配列代入を融合した。 /1/は開始行、/2/は終了行である。

(4) 実行時におけるエラーのメッセージ変更

実行時に出力される、50番のエラーメッセージを以下のように変更しました。

(和文) 入出力文の実行中、システムルーチンの実行でエラーが検出された
UNIT=nn [FILE=name] REC=rrrrrr
message(system#xxxxxx)
(英文) Error detected in system routine UNIT=nn [FILE=name] REC=rrrrrr
message (system#xxxxxx)

外部ファイル装置nnに割り当てられた外部ファイル(name)のrrrrrr番目の記録を入出力しているとき、システムルーチンがエラーを検出した。
ここで、message は、検出されたエラー(errno)の内容
system は、エラーを検出したシステムコール名
xxxxxx は、エラーを検出したシステムコールを呼び出している関数の内部番号である。

(5) SYSTEM(3F)の機能強化

SYSTEM組み込みサブルーチンを、組込み関数としても使用できるようにしました。

関数名	引用方法	関数の型	機能概要
system	system(cl)	基本整数型	文字列clをシェルに入力として渡して実行する。 関数の返却値は、clの返却値である。

(6) 使用可能なCPU数を返す関数追加

使用可能なCPU数を返す以下の関数が追加されました。

関数名	引用方法	関数の型	機能概要
maxcpu_num	maxcpu_num()	整数型	使用可能な最大CPU数を返す
icpu_num	icpu_num()	整数型	優先的に使用可能なCPU数を返す

(7) 編集リスト

f90 コマンドオプション-R5、あるいは f90 詳細オプション-L のサブオプション fmtlist を指定することにより編集リストを出力することができます。(付録Aを参照)

f90 オプション -R5
f90 詳細オプション -L [fmtlist|nofmtlist]

編集リストにはループ構造とそのループ構造のベクトル化／並列化状況が表示されます。

(8) コンパイラ指示行強化

付録Bを参照。

(a) 範囲指定コンパイラ指示行

範囲指定コンパイラ指示行を使用できます。
この指示行を使用することにより、指示オプションが有効となる範囲を
直後の文だけではなく、手続の終わりまでとすることができます。

範囲指定コンパイラ指示行の形式は、次の通りです。

自由形式:
```
!CDIRR コンパイラ指示オプション並び
```

固定形式:
```
!CDIRR コンパイラ指示オプション並び
```
あるいは、
```
*CDIRR コンパイラ指示オプション並び
*ODIRR コンパイラ指示オプション並び
```

範囲指定コンパイラ指示行には、以下のオプションは指定できません。

```
ASSERT
OVERLAP/NOOVERLAP
PERMUTATION
IEXPAND/NOEXPAND
並列化制御オプション
```

(b) 配列式用コンパイラ指示オプション
配列式用コンパイラ指示オプションを使用することにより、配列式に対する自動ベクトル化と自動並列化を制御できます。

配列式用コンパイラ指示オプションの形式は次の通りです。

```
ARRAY({S|V|P|*},{S|V|P|*},...)
```

{S|V|P|*},{S|V|P|*},... は、次元に対応します。
文字の意味は以下の通りです。

```
S: ベクトル化の対象にも並列化の対象にもしない。
V: ベクトル化の対象にする。
P: 並列化の対象にする。
*: ベクトル化、並列化のどちらを適用するかコンパイラに任せる。
```

配列式用コンパイラ指示オプションは、コンパイラ指示行に単独で指定されなければなりません。

(c) 自動並列化制御用指示オプション
CONCUR オプションを強化します。次の形式です。

```
CONCUR[({FOR[=n]|BY=m})]
```

直後の文に現れる配列式またはDOループを自動並列化する場合、FOR[=n] で並列化するかBY=m で並列化するかを指定できます。n と m はスカラ定数でなければなりません。

(9) サマリリスト及びオプション指示行強化
-pvctl の以下のサブオプションがサマリリストに出力されるようになりました。また、同サブオプションはオプション指示行に指定可能になりました。

```
assoc|noassoc
inner|noinner
parcase|noparcase
```

parthreshold[=n]
for[=n]|by=n
cncall=name[,name[,...]]
res=[whole|parunit|no]
vchg|novchg

(10) 組込み関数のインライン展開強化
次の組込み関数で、実引数の形状が7次元までインライン展開を行います。

ALL, ANY, COUNT, MAXVAL, MINVAL, PRODUCT, SUM, MAXLOC, MINLOC,
CSHIFT, EOSHIFT, PACK, UNPACK, SPREAD

(11) 配列式で使用する作業配列の削減
実引数並びや入出力並び中の配列ポインタ名や形状引継ぎ配列名に対して、次のf90詳細オプション-contを指定することにより、作業配列を使用しないようにします。

-cont　実引数並びや入出力並び中の配列ポインタ名や形状引継ぎ配列名を常に連続として使用していることを指定する。コンパイラは、その変数に対して作業配列を使用しない。
-Ncont　実引数並びや入出力並び中の配列ポインタ名や形状引継ぎ配列名を常に連続としては使用していないことを指定する(既定値)。コンパイラは、その変数に対して作業配列を使用する。
注意)　プログラムの中で配列ポインタを不連続に使用している場合に、-contを指定すると、プログラムの実行は保証されません。

(12) 複素数除算オブジェクトの改善
・複素数/実数の演算で実数の絶対値が非常に大きいか非常に小さい場合、浮動小数点オーバーフローや浮動小数点アンダーフローが発生しないオブジェクトを生成します。
・複素数除算で除数の複素数の実部や虚部の絶対値が非常に大きいか非常に小さい場合、次のf90詳細オプション -P aを指定することにより、浮動小数点オーバーフローや浮動小数点アンダーフローが発生しないオブジェクトを生成します。

-P a　単精度、倍精度の複素数除算に対して精度を優先したコードを生成する。
-P na　単精度、倍精度の複素数除算に対して性能を優先したコードを生成する(既定値)。
注意)　-Pa指定時は、単精度、倍精度の複素数除算に対してライブラリ呼び出しのコードを生成するため、-P na指定時(既定値)に比べ、実行性能が低下します。

(13) モジュールのオブジェクト改善
初期値を持つモジュールを含むアーカイブファイルと、モジュールの引用を含むオブジェクトファイルをリンクする際、モジュールの引用側に初期値を反映します。

(14) 2Gバイト越えデータのシンボリックデバッグ
2Gバイトを越える大きさの共通ブロック、構造型、および配列に対して、シンボリックデバッグ機能が利用できます。

(15) 条件式がループ内で不変なIFの最適化が強化され、IFがネストしている場合も最適化されるようになりました。

(16) ある条件で実行される除算が、最適化の式の変形によって無条件に実行されるようになり、その結果、ゼロ除算例外が発生するようになった場合、除数が次の条件を満たす場合にかぎり、ゼロ除算例外を発生させないようにします。
- DATA文によってある変数が初期化され、代入等によってその値が変更されない

- 除数の値が上記の変数によって決定し、コンパイル時に、値がゼロであることが静的にわかる

(17) 装置番号の決定処理の改善
プログラムの最初の入出力文における装置番号の決定処理を改善し、命令数を削減しました。

(18) 配列要素引用に指定した添字の数が宣言している配列の次元数より少ないとき、警告メッセージが出力されるようになりました。

(19) 組込み関数 SPREAD の引数 NCOPIES が定数でないときにインライン展開可能にしました。

(20) 拡張形式の最大桁を 132 桁から 2048 桁に拡張しました。

(21) f90 コマンドオプションと入力ファイル名の順序がわかりやすくなるように、man データの記述を修正しました。

(22) 詳細オプション -0 のサブオプション overlap/nooverlap の既定値の変更
-Chopt,-Cvopt,-Csopt の時の詳細サブオプション -0 のサブオプション overlap/nooverlap の既定値を overlap から nooverlap に変更しました。

旧仕様：
-0 overlap　　(既定値)
-0 nooverlap

新仕様：
-0 overlap　　(-Cvsafe,-Cssafe 指定時の既定値)
-0 nooverlap　　(-Chopt,-Cvopt,-Csopt 指定時の既定値)

(23) 強制並列化指示オプションサポート
自動並列化で並列化されないループなどを強制的に並列化を行うための以下の指示オプションをサポートしました。

- PARALLEL DO [PRIVATE(変数[,変数…])]
- PARALLEL SECTIONS [PRIVATE(変数[,変数…])]
SECTION
END PARALLEL SECTIONS
- ATOMIC

尚、FORTRAN77/SX との互換のために以下の形式も使用できます。

- FORCEPARDO
- FORCEPRIVATE
- FORCEPARCASE/FORCECASE/FORCEENDCASE
- FORCEREDUCTION

(24) 簡易性能解析機能の追加
この機能を使用することによって、手続毎の性能情報を簡単に採取することができます。
使用方法以下の通りです。

(a) 測定対象のソースプログラムを f90 コマンドオプション-ftrace を指定してコンパイルします。

例　　f90 -ftrace test.f90

(b) 実行可能プログラムを実行します。測定結果としてカレントディレクトリに ftrace.out が生成されます。

例　　a.out

備考　　実行時オプションとして、環境変数F_FTRACE を値 YES と設定することにより、ftrace コマンドを用いずに、実行可能プログラムの終了時に、解析結果を標準エラー出力ファイルへ出力させることもできます。

(c)ftrace コマンドを実行します。解析結果が標準出力ファイルに出力されます。

例　　ftrace

(25) 最適化時のデバッグ機能追加

-g オプションが指定された場合、最適化、ベクトル化、並列化されたオブジェクトに対して、シンボリックデバッグ機能を利用可能にしました。

(26) 組込み関数 BOOL の追加

組込み関数 BOOL が引用できるようになりました。

(27) 詳細オプション-O の新規サブオプション nomovediv の追加

詳細オプション -O に最適化の副作用のために発生するゼロ除算例外を抑止するためのサブオプション nomovediv が追加され、既定値が変更されました。

-O [{move|nomovediv|nomove}]

個々のサブオプションの意味は以下のとおりです。

move　　ループ内の条件下にある不変式はループ外へ移動する(翻訳モード-Chopt を指定した場合の既定値)。

nomovediv　ループ内の条件下にある除算以外の不変式はループ外へ移動する(翻訳モード-Cvopt および sopt を指定した場合の既定値)。

nomove　　ループ内の条件下にある不変式はループ外へ移動しない(翻訳モード-Cvsafe および ssafe を指定した場合の既定値)。

(28) コンパイラ指示行強化

次に示すコンパイラ指示行のオプションに整定数表現だけではなく、スカラ整数型の初期値式が指定できるようになりました。

```
LOOPCNT=n
VWORKSZ={n[M|m] | (n)[M|m] }
THRESHOLD[(n)]
SHAPE=(n1[,n2[,n3[,n4[,n4[,n6[,n7]]]]]])
UNROLL[=n]
COUNT(n)
CONCUR[({FOR[=n]|BY=n})]
RESERVE[=n]
PARDO [{BY=n|FOR[=N]}]
CRITICAL[=n]
ENDCRITICAL[=n]
POST[=n]
WAIT[=n]
```

n は、スカラ整数初期値式

(29) 自動並列化強化
並列化制御オプション RESERVE/RELEASE が指定されている手続きも自動並列化が行われるようになりました。

(30) 拡張固定形式の拡張
拡張固定形式では一行に 2048 文字まで書けるようになりました。

(31) HP クロス版のメッセージ強化
HP クロスコンパイラでコンパイルする場合、環境変数 LANG または LC_MESSAGES に japanese を設定すれば、診断メッセージをシフト JIS コードの日本語で表示できます。

(32) nX 形編集記述子の仕様拡張
nX 形編集記述子の仕様を以下のように拡張しました。

(a) nX 編集の n の省略を可能とし、省略した場合は 1X と記述されたものとして認識します。
(b) nX 編集の直前のカンマの省略が出来ます。

(33) ループ融合の強化
配列代入をループ融合可能にしました。

(34) -sx4/-sx5 オプションの追加
基本オプション-sx4/-sx5 を追加しました。

-sx4|-sx5

オプションの意味は以下の通りです。

-sx4
SX-4 向けの命令列を生成することを指定します(HW が SX-4 のときの既定値)。
浮動小数点データ形式としては、環境変数 FLMOD の指定がない場合、-float0 が既定値となります。

-sx5
SX-5 向けの命令列を生成することを指定します(HW が SX-5 のときの既定値)。
浮動小数点データ形式としては、-float0 のみが使用できます。-float1、-float2 は使用できません。

(35) 拡張固定形式で 99 行の継続行を指定できるようにしました。

(36) UNSHARED 指示オプションを追加しました。

(37) OpenMP 対応
OpenMP に対応しました。 OpenMP は共有並列における並列処理プログラムの移植性を高めるために 1997 年 10 月に発表された並列化仕様です。
原始プログラムの OpenMP 指示行を有効にする f90 コンパイルオプションは-P openmp です。

OpenMP is trademark of the OpenMP Architecture Review Board. Portions of this product/publication may have been derived from the OpenMP Language Application Program Interface Specification.

FORTRAN90/SX の機能拡張のプロダクトとして、サポートしました。

(38) 実行時エラーメッセージの追加

F_UFMTADJUST[nn] オプションのみを指定して、いづれのデータ変換オプション(F_UFMTIEEE, F_UFMTFLOAT1, F_UFMTFLOAT2)も指定していない場合、実行時に以下の warning メッセージを出力するようにしました。

(和文) F_UFMTADJUST オプションのみ指定された UNIT=nn
(英文) Only the F_UFMTADJUST option is specified UNIT=nn

(39) 編集記述子の改善
編集記述子中の ",” の省略を可能にしました。

(40) 実行時エラーメッセージの変更(英文のみ)
実行時に出力される、114 番のエラーメッセージ(英文のみ)を以下のように変更しました。

File already exists UNIT=nn [FILE=name]

(41) tremain(3F) の追加
NQS ジョブの残り CPU 時間を返却する tremain という FORTRAN ライブラリを追加しました。

(42) #220 エラーの扱いの変更
#220 エラーはデフォルトではエラーとしてカウントしないように変更しました。

(43) -c が有効で、かつサフィックスが ".c”, ".s”, ".f”, ".f90”, ".F”, ".F90” 以外の場合に出力されるメッセージを変更しました。

(44) -Nc が有効な場合、ソースファイルとオブジェクトファイルが指定されていなくてもリンカを起動します。

(45) たとえソースファイルが指定されていなくても-V オプションが指定されていたら、コンパイラのバージョンを出力します。

(46) 単項演算子 '-' を持つ整数型の最大値を式に書けるようにしました。

(47) 入力ソースファイルが空のファイルである場合、警告エラーを出力します。

(48) 文関数の右辺に明示的引用仕様を持つ手続引用を含むことができます。

(49) 文関数の右辺で構造型である仮引数の成分を引用できるようにします。

(50) 組込み関数 SIGN の引用において、第二引数の型が第一引数の型と一致していない場合、型変換せずに致命的エラーを出力します。
(51) 命令のスケジューリングを基本ブロック内でのみ行う詳細オプション "-0 reorderrange=basic” を追加しました。
(52) 詳細オプション "-0 extendreorder” を否定するオプション指示行 "-0 noextendreorder” をサポートしました。

(53) 配列にメモリのかわりにベクトルレジスタを割り当てることを指定する VREG 指示行を追加しました。

仕様は以下のとおりです。

VREG(array1[,array2...])

VREG は並びに指定された配列にメモリのかわりにくベクトルレジスタを割り当てることを指定する。

コンパイラはVREG指示行を含む手続内のすべてのVREG指定された配列の参照をロード／ストアではなく、ベクトルレジスタの参照として翻訳する。

VREG指定される配列は以下の条件を満たしていなければならない。
- 1次元の明形状配列でなければならない。
- 大きさは最大ベクトルレジスタ長以下でなければならない。
- I*4, I*8, R*4, R*8, L*4, L*8 のいずれかの型でなければならない。
- 仮配列でない。
- SAVE属性、ポインタ属性、ALLOCATABLE属性、TARGET属性のいずれも持っていてはならない。
- EQUIVALENCE結合、親子結合してはならない。
- ベクトル化可能な組み込み関数以外の手続きの実引数として現れてはならない。
- 入出力並びに現れてはならない。
- ベクトル化されるループ中でのみ定義／参照されている。
- 各々の定義／参照は、同じ繰り返しにおける添字の値が同じでなければいけない。

(54) 文関数定義文の右辺でFORTRAN77/SXのすべての非標準の組込み関数が引用できるようになりました。

(55) 以下の詳細オプションを追加しました。

-pvctl ifopt
　ループ内不変のIF構造をループネストの外側に移動する最適化を行う

-pvctl noifopt
　ループ内不変のIF構造をループネストの外側に移動する最適化を行わない

既定値は以下の通りです。
　-C hopt の指定が有効な場合: -pvctl ifopt
　-C hopt の指定が無効な場合: -pvctl noifopt

(56) 多次元配列の添字式が複雑な場合のベクトルオブジェクトコードを改善しました。

(57) 基本オプション -Cvsafe における詳細オプション -0 のサブオプション darg/nodarg の既定値を nodarg から darg に変更しました。

既定値は、以下のようになります。

-Chopt,-Cvopt,-Cvsafe,-Csopt が有効な場合: -0 darg(仮引数は最適化の対象とする)
-Cssafe が有効な場合: -0 nodarg(仮引数は最適化の対象外とする)

(58) FORTRAN77互換形式の宣言文で配列を宣言するとき、上下限に配列要素が記述できるようになりました。
例)

```
subroutine sub(u,d)
integer d(3)
double complex u(d(1),d(2),d(3))
```

(59) 非10進定数が全ての式に記述できるようになりました。
この機能を有効／無効にするための詳細オプション -P x/-P nx を追加し、-P x が指定されているときこの機能は有効になります。

-P x/-P nx の意味は以下の通りです。

-P x ： 非10進定数表現を式の中に記述できる。
-P nx ： 非10進定数表現を式の中に記述できない。(既定値)

(60) -R2 オプション指定時に、変形リストに加えて編集リストも出力されるようになりました。

(61) 組み込み関数の変更[強化]

以下に示すベクトル版組み込み関数 SQRT は、性能改善のため計算方法を変更しました。従来の関数値とは誤差の範囲内で差異が生じることがあります。

```
-float0
  SQRT,DSQRT,RSQRT,DRSQRT
```

以下のベクトル版組み込み関数は、SQRT を呼び出しているため従来の関数値とは誤差の範囲内で差異が生じることがあります。

```
-float0
  CSQRT,CDSQRT,ASIN,ACOS,DASIN,DACOS,ASINH,ACOSH,DASINH,DACOSH
```

変更前の組み込み関数は以下に保存してありますので、必要な時はこれをご利用ください。

```
/usr/lib/libv90sxe_sv.a          (セルフ)
/SX/usr/lib/libv90sxe_sv.a       (クロス)
```

使用する際は以下のオプションを指定して下さい。

```
-lv90sxe_sv
```

付録A： 編集リスト

f90 コマンドオプション
  -R5
または f90 詳細オプション
  -Wf"-L fmtlist"
を指定した場合に、以下の情報がソースプログラムと共に出力されます。

・ループと配列式のベクトル化情報
・手続インライン展開情報
・ループと配列式の並列化情報

備考
・致命的エラーが検出されたプログラム単位には、編集リストは出力されない。
・編集リスト中の FORTRAN STATEMENT には INCLUDE 文による展開行は出力されない。
・編集リストは原始プログラム中の EJECT 文の影響を受けない。

編集リストの形式を以下に示します。

図　編集リストの形式

```
FILE NAME: test.f90
PROGRAM NAME: main
FORMAT LIST

  LINE    LOOP      FORTRAN STATEMENT
```

```
  1:              program main
  2:              real,dimension(100) :: a,b,c
  3:              real(kind=16),dimension(100) :: q
  4:
  5:            I call init(s,n)
  6: V======      b=100.0; c=100.0
  7:
  8: P------>     do i=1,10
  9: |V----->        do j=1,100
 10: ||                a(i)=a(i)+s*b(j)
 11: |V-----         end do
 12: P------      end do
 13:
 14: V------>     do i=1,100
 15: |              a(i)=a(i)+c(i)
 16: |         S    q(i)=a(i)+1.0
 17: V------      end do
 18:
 19:              call sub2(q)
 20:              end
```

各桁の説明
1- 6桁：外部行番号
7-16桁：ループの範囲、ベクトル化／並列化状況
17-18桁：ベクトル化不可部分、手続インラインの表示
20-80桁：原始プログラム

以下に主なループ情報、スカラ情報、手続インライン情報の出力イメージを示します。

1. ループ全体がベクトル化された場合

```
V------>     do i=1,100
|               a(i)=b(i)+c(i)
V------      enddo
```

2. ループが部分ベクトル化された場合

```
V------>     do i=1,n
|               a(i)=b(i)+c(i)
|         S     print *, a(i)
V------      enddo
```

ベクトル化不可の処理がある行には "S" が表示される。

3. ベクトル化されなかった場合。

```
+------>     do i=1,10
|               print *,a(i)
+------      enddo
```

4. ループが並列化された場合

```
P------>     do i=1,n
|               :
```

```
P------      enddo
```

5. 配列式をベクトル化した場合

```
             real,dimension(100):: a,b,c
V=======     a=b+c
```

ループの先頭と最後が同じ行である場合、そのループ構造は'='で表示される。

6. 多重ループが一重化された場合

```
W------>     do i=1,10
|*----->        do j=1,20
||                :
|*-----          enddo
W------      enddo
```

7. 多重ループにおいて、ループの入れ換えが行われた場合

```
X------>     do i=1,100
|+----->        do j=1,10
||                :
|+-----          enddo
X------      enddo
```

8. 手続呼び出しがインライン展開された場合

```
I call sub(a,b,c)
```

インライン展開された手続がある行には "I" が表示される。

付録B： コンパイラ指示行

コンパイラ指示行に対して以下のような機能が強化されました。

- 指示オプションの有効範囲を手続の終わりまでにするCDIRR指示行の追加
- 配列のベクトル化／自動並列化を制御するための並列化指示オプションARRAYの追加
- ループの自動並列化の制御を行うCONCURのサブオプションBY=n|FOR[=m]の追加

以下に、強化された機能の説明を記述します。

1. コンパイラ指示行の形式

コンパイラ指示行は特別な注釈行の形式をしており、コンパイラオプション-Wf"-dir"(既定値)指定時に有効となる。
-Wf"-Ndir"指定時は通常の注釈行として扱われる。

コンパイラ指示行が書ける場所は注釈行と同じである。

開始行の第1桁から第6桁までの文字が"!CDIR ", "!CDIRR", "*CDIR ", "*CDIRR"のいずれかである行をコンパイラ指示行とする。

・自由形式の場合

コンパイラ指示行　：！{CDIR|CDIRR}　コンパイラ指示オプション並び

・固定形式の場合

コンパイラ指示行　：！{CDIR|CDIRR}　コンパイラ指示オプション並びあるいは *{CDIR|CDIRR}　コンパイラ指示オプション並び

制約：

・ARRAY 指示オプションおよび並列化制御オプションは、コンパイラ指示オプション並びの中に、他のコンパイラ指示オプションと一緒に指定してはならない。

・以下のコンパイラ指示オプションは、"!CDIRR"指示行のコンパイラ指示オプション並びに現われてはならない。

ASSERT
OVERLAP/NOOVERLAP
PERMUTATION
IEXPAND/NEXPAND
並列化制御オプション

- コンパイラ指示行の例

```
!CDIR VECTOR
!CDIRR VECTOR,CONCUR
```

備考

FORTRAN77/SX との互換のために、以下の形式は"！CDIR コンパイラ指示オプション並び"と同等であるとみなされる。

*VDIR コンパイラ指示オプション並び
*PDIR コンパイラ指示オプション並び
*ODIR コンパイラ指示オプション並び
*ODIRL コンパイラ指示オプション並び

以下の形式は "!CDIRR コンパイラ指示オプション並び" と同等であるとみなされる。

*ODIRR コンパイラ指示オプション並び

2. コンパイラ指示オプションの有効範囲

・CDIR 指示行に指定されたコンパイラ指示オプションの有効範囲

1) 以下のコンパイラ指示オプションは直後に現われる最初の実行文に有効となる。

ASSUME/NOASSUME, LOOPCNT, SHAPE,UNROLL/NOUNROLL, ZCHECK/NOZCHECK,
手続名の指定がない IEXPAND, 手続名の指定がない NEXPAND,
ALTCODE/NOALTCODE, ASSOC/NOASSOC, COMPRESS/NOCOMPRESS, DIVLOOP/NODIVLOOP,
LOOPCHG/NOLOOPCHG, LSTVAL/NOLSTVAL, NODEP, SHORTLOOP, SPARSE/NOSPARSE,
SPLIT/NOSPLIT, VECTOR/NOVECTOR, VERRCHK/NOVERRCHK, VLCHK/NOVLCHK,
VWORK, VWORKSZ,
ARRAY, CNCALL, CONCUR/NOCONCUR, INNER/NOINNER, SELECT, SKIP,
SYNC/NOSYNC, THRESHOLD/NOTHRESHOLD

2) 以下のコンパイラ指示オプションは、そのコンパイラ指示オプションが指定された有効域内で有効である。

OVERLAP/NOOVERLAP, PERMUTATION, IEXPAND(手続名), NEXPAND(手続名)

3) ASSERT 指示オプションは、有効域内で、ASSERT 指示オプションの指定された位置より後に現われるすべての実行文に対して有効である。

・CDIRR 指示行に指定されたコンパイラ指示オプションの有効範囲

CDIRR 指示行に指定されたコンパイラ指示オプションは、指示行が指定された有効域内(手続からそこに含まれる内部手続を除いた部分)で、その CDIRR 指示行より後に現われる全ての実行文の直前に指定されたものと見なされる。

CDIRR 指示行で指定されたコンパイラ指示オプションと、CDIR 指示行で指定されたコンパイラ指示オプションが競合する場合、CDIRR 指示行で指定されたものが有効となる。

ある CDIRR 指示行で指定されたコンパイラ指示オプションが他の CDIRR 指示行で指定されたものと競合している場合、最後に現われた CDIRR 指示行で指定されているものが有効となる。

CDIRR 指示行の例

```
!CDIRR NOVECTOR, NOUNROLL
      A(:)=B(:)+C(:)          ! NOVECTER と NOUNROLL が有効となる
      DO I=1,N                ! NOVECTER と NOUNROLL が有効となる
        D(I)=E(I)+1.0
      ENDDO
!CDIRR VECTOR
      X(:)=Y(:)+Z(:)          ! VECTOR と NOUNROLL が有効となる
```

3. コンパイラ指示オプション

・ARRAY(c1[,c2...])

直後の文に現れる配列式に対して、ベクトル化または並列化を次元をどのように選択するかをコンパイラに指示する。

cn は以下のいずれかである。
S: n次元目はベクトル化も並列化も行わないことを指定する。
V: n次元目はベクトル化を選択することを優先する。
P: n次元目は並列化を選択することを優先する。
*: n次元目に対してはベクトル化／並列化の優先の指定はしない。

括弧内の項目数がN個の ARRAY オプション、すなわち ARRAY(c1,c2,...cN) は、N次元の配列式に対してだけ有効となる。

このオプションは実行文の現われることのできる場所に指定されたコンパイラ指示行
内にのみ指定できる。

ARRAY は、単独で指定されなければならない。
すなわち、一つの指示行内に、他のコンパイラ指示オプションと同時に指定したり、
複数の ARRAY オプションを並べて指定したりはできない。

直後に現れる最初の実行文中に配列式が現われない場合、この指定は無効となる。

ARRAY 指示オプションの例

```
!CDIR ARRAY(*,P,V)
!CDIR ARRAY(S,V)
    CALL SUB(A(:,:,:)+C(:,:,:),X(:,:)*Y(:,:))
```

配列式 A(:,:,:)+C(:,:,:) において、２次元目に対して並列化、３次元目に対してベクトル化を優先して選択することを指定する。
配列式 X(:,:)+Y(:,:) において、１次元目はベクトル化も並列化も行わず、２次元目に対してはベクトル化を優先して選択することを指定する。

・CONCUR({BY=n|FOR[=m]})/NOCONCUR

直後の文に現れる配列式またはDOループに対して、自動並列化を行う(CONCUR)/行わない(NOCONCUR)ことを指定する。

CONCURはループの並列化を許可するが、強制するものではない。
NOCONCURはループの並列化を禁止する。既定値はCONCURである。

CONCURのサブオプション BY=n、FOR[=m] はループの並列化方法を指定する。
BY=n が指定された場合、ループは PARDO BY=n で自動並列化される。
FOR[=m] が指定された場合、ループは PARDO FOR[=m] で自動並列化される。
BY および FOR の指定がない場合、ループの並列化方法はコンパイラオプション -Wf"-pvctl by=n/for[=n]" の指定にしたがう。
(PARDO BY=n および PARDO FOR=m の説明については「FORTRAN90/SX 並列処理機能利用の手引」を参照)

このオプションは実行文の現れることのできる場所に指定されたコンパイラ指示行内にのみ指定できる。

直後に現れる最初の実行文中に配列式が現れず、しかもその文がDO文でもない場合、この指定は無効となる。

CONCUR指示オプションの例

```
!CDIR CONCUR(FOR=4)
      DO I=1,100
        DO J=1,N
          A(J,I)=B(J,I)+C(J,I)*2.0
        ENDDO
      ENDDO

        ↓  自動並列化

!CDIR PARDO FOR=4
      DO I=1,100
        DO J=1,N
          A(J,I)=B(J,I)+C(J,I)*2.0
        ENDDO
      ENDDO
```

(システム管理掛)

⑦ MARCおよびMENTATのバージョンアップについて

(2000.1.11 No.296)

MARC6.2から7.3、及びMENTAT3.1から3.3にバージョンアップしましたのでお知らせします。

MARC7.3の利用方法
　sx40にログイン
　/usr/local/appli/marck73　options

MENTAT3.3の利用方法
　indigo01にログイン
　/usr2/local/appli/mentat330

(システム管理掛)

⑧ アプリケーション：Gaussian98の追加について

(2000.1.11 No.296)

分子軌道設計ツール「Gaussian98」を演算サーバExemplarにインストールしましたのでご利用ください。なお、Gaussian94も今までどおりご利用いただけます。

[利用方法]

1．Visualize（vis01～vis10,visd01,visd02）にログインします。

2．環境設定を行ないます。

```
vis01%  setenv PATH "${PATH}:/apl/g98"
vis01%  setenv GAUSS_EXEDIR /apl/g98
```

3．解析データが存在するディレクトリを環境設定します。

```
setenv GAUSS_SCRDIR ディレクトリのパス名
```

（例）解析データが存在するディレクトリへ移動し環境設定する場合

```
vis01% cd g98-dir      ..... 解析データのあるディレクトリへ移動
vis01% setenv GAUSS_SCRDIR  `pwd`
```

4．実行

```
bsub -q queue_name -i inputfile -o outputfile g98
```

queue_name ： 実行するキュー名を指定する。
hp_p2,hp_p4,hp_p8 のいずれかを指定する。
inputfile ： 入力データファイル名。オプション(-i)を使用。
outputfile ： 解析結果を出力するファイル名。オプション(-o)を使用。

(例)

```
%bsub -q hp_p2 -i testg94 -o list94 g98
```

入力データファイルtest94、出力結果ファイル名をlist94を指定。
キューはhp_p2を使用。

(業 務 掛)