UNIXワークステーションでのFortran利用法

渡部 善隆*

1　はじめに

本稿では，情報基盤センター教育用システムのUNIXワークステーション (計算機名: “ah.cse.ec.kyushu-u.ac.jp”) におけるプログラム言語Fortranの利用方法を解説します．

なお，記事は，お読みの方が

- UNIXワークステーションへの接続ができる
- muleなどのエディタを用いてFortranプログラムを記述したファイルを作成できる
- UNIXについての基本的な知識がある

ことを前提にして書かれています．UNIXワークステーションへの接続方法，エディタの利用方法などについては『教育用システム・利用の手引』を参照してください．また

http://www.cse.ec.kyushu-u.ac.jp/manual/Tebiki2002/

から参照することもできます．

2　基本的な手順

この章では，Fortranプログラムを実行するまでの基本的な手順を説明します．

2.1　実行までの流れ

ファイルに記述したFortranのソースプログラムを動作させるには，翻訳・結合編集・実行という3つの過程を踏みます．

最初の翻訳処理[1]により，Fortran言語を計算機の理解できる機械語のファイルに変換します．このファイルをオブジェクトファイルと呼びます．7章で紹介する数値計算ライブラリSSL IIはオブジェクトファイルの集合体です．

オブジェクトファイルはそのままでは実行することができません．実行のために必要な手続きをオブジェクトファイルに組込む処理を結合編集[2]と呼びます．結合編集処理により作

*九州大学情報基盤センター　E-mail: watanabe@cc.kyushu-u.ac.jp

[1]「コンパイル」と呼ぶこともあります．

[2]「リンク・エディット」または単に「リンク」と呼ぶこともあります．

成されたファイルを実行可能ファイルと呼びます．実行可能ファイル名をコマンドとして指定することにより，利用者がソースプログラムに記述した処理が実行されます．

通常 frt (f90, f95) コマンドを使用すると，ソースプログラムを翻訳し結合編集により実行可能ファイルを作成するまでの手順が自動的に行われます．実行可能ファイルが作成された時点で中間ファイルであるオブジェクトファイルは消去されます．翻訳時オプション -c を指定することで，翻訳段階で処理を打ち切り，オブジェクトファイルを保存することもできます．保存されたオブジェクトファイルは，結合編集の際に必要に応じてメインプログラムから呼び出されることで，実行ファイルを構成する一部となります．

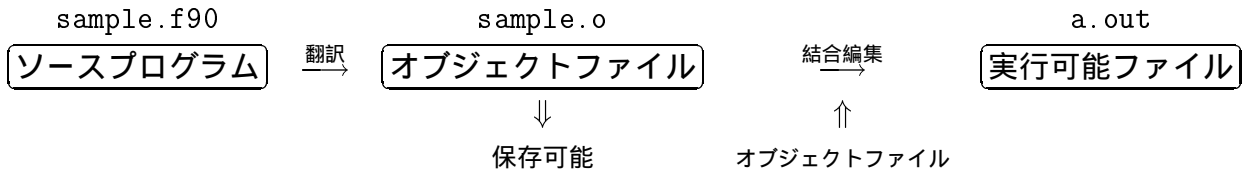

図 1: 実行可能ファイルの作成手順

2.2 コマンド名とファイル拡張子

UNIX ワークステーションで利用できる Fortran システムは，最新の JIS 規格 (JIS X 3000-1) に完全準拠した Fortran 95 コンパイラです．Fortran の翻訳と結合編集のコマンドは frt または f90 または f95 です．3 つのコマンドは等価です[3]．

Fortran ソースファイルの拡張子[4]はプログラムの形式に応じて表 1 のようにしてください．

表 1: Fortran のファイル拡張子

拡張子	形式
.f90 または .f95	自由形式
.f または .for	固定形式

ソース形式の解釈は，翻訳時オプション -Fixed または -Free で変更することもできます．

【注意】

固定形式は FORTRAN 77 以前で使われていた形式で，行の最初の 6 文字分の位置は文番号などを記述し，文は第 7 字目から書くという規則があります．これに対して自由形式のプログラムでは，1 行の長さが 132 文字以内で，1 桁目から 132 桁目のどこからでも書き始めることができます．

2.3 翻訳から実行まで

frt(または f90, f95) コマンドに続けて，ひとつ以上の空白をおき，Fortran プログラムが記述されているファイル名を指定し，Enter キー (または Return キー) を押し下げます．以

[3]言語仕様によってコマンドを使い分ける訳ではありませんので注意してください．すべて同一の処理が行なわれます．

[4]ファイル名の最後に置く文字列で，ファイルの属性や内容を表す目的で使われます．

下の例ではEnterキーの押し下げを ↵ で表記します．この入力により，翻訳および結合編集処理が行なわれ，実行可能ファイルが作成されます．

次は，ファイル名を“ example.f95 ”とした例です．

```
% frt example.f95 ↵          ←翻訳と結合編集により実行可能ファイルを作成
```

処理が重大なエラーを起こすことなく終了したならば，実行可能ファイル“ a.out ”が作成されています．もし既に“ a.out ”が存在する場合は上書きされます．実行可能ファイルの名前は -o オプションで変更することができます (4.3 節)．

実行はファイル名をコマンドとして入力します[5]

```
% a.out ↵                    ←実行
        :
  (実行結果 )
```

3　翻訳・実行時の診断メッセージの見方

この章では，プログラムの翻訳または実行時に出力される診断メッセージの見方と対策を説明します．

3.1　翻訳時に出力される診断メッセージ

以下はメッセージの出力例です．

```
% frt jprmx.f ↵              ←プログラムの翻訳

Fortran diagnostic messages: program name(FMF001)
  jwd2006i-i  "jprmx.f", line 123: この名前は，宣言だけされていて引用されていません.(名前:VC)
  jwd2004i-i  "jprmxss.f", line 700: この変数は，値を設定していますが引用されていません.(名前:GKEY)
                  :
```

“jwd” で始まるメッセージは翻訳時の診断メッセージです．最後の文字によってメッセージのレベルが判定できます．

表 2: 翻訳時の診断メッセージの最後の文字と意味

文字	意味
i	エラーではないが注意を促すメッセージです．
w	軽度のエラーが発生したことを示します．翻訳処理は続行されます．
s	重度のエラーを示すメッセージであり，システムはエラーの文を無視し，その他の文については翻訳を続行します．
u	致命的なエラーを示すメッセージであり，翻訳が打ち切られます．

[5]他の UNIX システムの場合，カレントディレクトリにあるファイルを指定する意味での “./” を実行可能ファイルの前に記述しないと実行が行なわれないことがあります．

3.2 対策

“i” で終わるメッセージは『文法としては間違ってないけど，気をつけてね』という意味です．プログラムがまだ完成していない段階では一時的な変数を使ったりするため，このメッセージが出力されたとしても問題のない場合が多いと思われます．

一方，“w” “s” “u” で終わる診断メッセージが出力した場合には，ソースプログラムの修正が必要になる場合が大半です．6 章を参考にソースプログラムを確認してください．大切なことは，

1. メッセージ中の “line” の後の行数に対応する文を調べる
2. 複数の診断メッセージが表示された場合，始めから順番に対処する

ことです．最初のメッセージから対処する理由は，たとえば，変数の宣言部分で記述に間違いがあった場合，その変数に対応する文が連動してエラーとなる可能性があり，そのため，最初の部分の一箇所の字句を修正するだけでたくさん表示されていたエラー出力がなくなるということがよくあるからです．

3.3 実行時に出力される診断メッセージ

実行時にも，たとえば次のようなメッセージが出力される場合があります．

```
% a.out ↵              ←実行

  jwe0011i-e A floating overflow exception was detected.
   error occurs at g_adxi   loc ff17692c offset 000001a8
   g_adxi (f)   at loc ff176784 called from loc 00035b54 in mpfunmod.mpinrl_
                :
```

“jwe” で始まるメッセージは実行時の診断メッセージです．こちらも最後の文字によってメッセージのレベルが判定できます．

表 3: 実行時の診断メッセージの最後の文字と意味

文字	意味
i	エラーではないが注意を促すメッセージです．
w	軽度のエラーが発生したことを示します．実行は続行されます．
e	中度のエラーが発生したことを示します．エラーが発生した文は無視されます．10 回 e レベルのエラーが発生すると実行が打ち切られます．
s	重度のエラーが発生したことを示します．実行は打ち切られます．
u	実行が続行できないような致命的なエラーが発生したことを示します．実行は打ち切られます．

3.4 対策

まず，診断メッセージの最後の文字によりメッセージの度合を確認します．その後，メッセージの内容を把握します．上の例は「浮動小数点のオーバーフローが発生しました」とい

う意味で，実行の途中で計算結果が計算機が表現できる数の範囲を超えてしまったというメッセージです．

実行時に出力される診断メッセージにはいろいろな原因が考えられるため，その対策も様々です．6 章で説明するデバッグオプションやソースプログラムを見直しただけでは原因が特定できない場合には，変数の値を WRITE 文を使って出力させたりなどして原因の絞り込みを行なってください．

4 オプション

この章では，よく用いるオプションを紹介します．frt (f90, f95) コマンドには，その他にもたくさんのオプションがあります．詳細は man frt によって参照することができます．

4.1 翻訳時オプション

よく用いる翻訳時オプションを表 4 に示します．

表 4: よく用いる翻訳時オプション

オプション	機能
-c	オブジェクトファイルの作成までを行います．結合編集を行わず実行可能ファイルは作成されません．
-o *filename*	実行可能ファイル名またはオブジェクトファイル名を *filename* に変更します．
-Free	自由形式の Fortran プログラムとして翻訳します．
-Fixed	固定形式の Fortran プログラムとして翻訳します．
-Haesux	プログラムのデバッグのため，引数の妥当性の検査，添字式・部分列範囲の検査，未定義データの引用の検査を行ないます．メッセージは翻訳時，実行時に出力されます．実行時間が増大するため，小規模なプログラムに対しデバッグを行ない，デバッグ終了後は必ず，実行可能ファイル，オブジェクトファイルは作成し直してください．
-Am	モジュールを翻訳する場合に指定します．
-Kfast	翻訳している計算機にあわせて高速に実行させることを指示します．
-X9	言語仕様が Fortran 95 であると解釈して翻訳します．拡張子が .f または .for のファイルに Fortran 90 から新たにサポートされた組込み関数などを記述する場合に必要です．

4.2 オブジェクトファイルの作成と利用方法

翻訳時オプション -c を指定することにより，結合編集を行わず，オブジェクトファイルの作成までを行うこともできます．

```
% frt -c example.f95 ↵          ←オブジェクトファイルの作成
```

オブジェクトファイルの拡張子は .o です．例では“ example.o ”という名前のファイルが作成されます．

オブジェクトファイルは，既にデバックが終了した副プログラムをメインプログラムと切り離して管理する場合によく用いられます．例として，メインプログラムを“ main.f95 ”，FORTRAN 77 で記述された副プログラムを記述したファイルを“ sub.f ”とします．まず，sub.f を -c オプションを付けて翻訳し，オブジェクトファイル sub.o を作成します．

```
% frt -c sub.f ↵                ←オブジェクトファイルの作成
```

次に，メインプログラムを翻訳し，オブジェクトファイルを結合することで実行可能ファイルを作成します．

```
% frt main.f95 sub.o ↵              ←実行可能ファイルの作成
```

このようにすると，メインプログラムを何度も修正する場合でも副プログラムの翻訳は一度だけで済むため，翻訳時間が短縮できます．

frt コマンドには上の例のようにオブジェクトファイル (拡張子 .o のファイル) も指定することができます．また，複数のソースプログラムファイル，オブジェクトファイルも指定することができます．

4.3 実行可能ファイル名前の変更

翻訳時オプション -o の後，ひとつ以上の空白に続けて任意のファイル名を指定することにより“ a.out ”以外の実行可能ファイル名に変更することができます．

```
% frt -o test.ex example.f90 ↵      ←実行可能ファイル名を変更
% test.ex ↵                         ←実行
```

上の例では“ test.ex ”という名前に変更しています．

【注意】

-o オプションに続けて指定するファイル名は任意です．したがって，もし，既存のファイル名を指定してしまった場合にはファイルの内容が失われてしまいます．

4.4 高速化オプションの指定例

以下は -Kfast オプションの指定例です．プログラムによってはかなりの高速化が得られる場合があります．ただし，翻訳時間は一般に増大します．

```
% frt -Kfast main.f90 ↵         ←最適化オプションの指定
% a.out ↵                       ←実行
```

◇ 最大限の高速化オプション

メーカーに確認した最大限の高速化オプションは以下の通りです．ただし，実行結果に副作用が生じる可能性があります．また，他の SPARC マシン (UNIX ワークステーションとソフトウェアの互換性がある計算機) への移植もできないため，指定する場合は十分に注意してください．

```
% frt -Kfast,eval,V8PFMADD,gs,prefetch main.f90 ↵      ←最大限の高速化
```

4.5 実行時オプション

実行時オプションは実行可能ファイル名の後に-Wl,(" l"は英小文字) を指定し，続いてオプションを指定します．複数の実行時オプションはカンマ (,) で区切って続けます．

よく用いる実行時オプションを表 5 に示します．

表 5: よく用いる実行時オプション

オプション	機能
-u	浮動小数点アンダーフローが発生した場合，エラーメッセージを出力します．
-C*number*	書式なし入出力文において，装置番号 *number* から IBM370 形式浮動小数点データのファイルを入出力するときに指定します．*number*を省略した場合すべての装置番号に対して有効となります．
-M	-C の指定により行なわれる IBM370 形式から IEEE 形式への浮動小数点データの変換によって仮数部のビットが損失したときに，診断メッセージを出力します．
-T*number*	書式なし入出力文において，装置番号 *number* からリトルエンディアンデータのファイルを入出力するときに指定します．*number* を省略した場合すべての装置番号に対して有効となります．

以下は，実行時オプション -T を指定して装置番号 10 番からリトルエンディアンデータのファイル入出力を行なう例です．

```
% a.out -Wl,-T10 ↵          ←リトルエンディアンデータのファイル入出力
```

【注意】

多バイトの数値をメモリに格納する方式は，計算機の CPU の仕様に応じてビッグエンディアン (big endian) とリトルエンディアン (little endian) に分かれます [1]．UNIX ワークステーションはビッグエンディアンのサポートです．一方，Intel や DEC Alpha ではリトルエンディアンが採用されています．したがって，リトルエンディアンデータを UNIX ワークステーションにおいて Fortran の書式なし WRITE 文で作成したデータを読み込む場合には，-T オプションが必要になります．

5　ファイル入出力

この章では，ファイル入出力について説明します．

5.1　OPEN 文

OPEN 文を用いてファイルの入出力を行なう場合には，次の点に注意してください．

- ファイル名は ’　’ または "　" で括って指定します．

```
OPEN(5,FILE='EXAMPLE.DATA')    ! ファイルの指定
```

- ファイル名は大文字 / 小文字を区別します．したがって

EXAMPLE.DATA,　example.data,　EXAMPLE.data

はすべて異なったファイル として認識されます．

- UNIX の相対パスを使って，実行するディレクトリ以外のファイルをアクセスすることも可能です．

- Fortran 95 の仕様から，書式なし入出力文に FORM='UNFORMATTED' を指定することが必須になりました．

```
REAL(KIND(1D0)),DIMENSION(100) :: X                  ! 配列の宣言
      :
OPEN(1,FILE='EXAMPLE.DATA',FORM='UNFORMATTED')       ! 書式なしデータを開く
READ(1) X                                            ! 書式なし入力
CLOSE(1)                                             ! ファイルを閉じる
```

- OPEN 文を使わずに WRITE 文に外部ファイル装置番号のみを指定した場合には，

“ fort. 外部ファイル装置番号 ” というファイルが作成されます．

5.2　標準入出力

外部ファイル装置番号 5 番 (READ(5) または READ(*) に対応，標準入力と呼びます) と 6 番 (WRITE(6) または WRITE(*) に対応，標準出力と呼びます) は通常端末での入出力になります．UNIX の「リダイレクション機能」を用いることで， これらの入出力をファイルに切替えることができます．

ただし，操作に慣れるまでは，リダイレクションの方向 (<, >) に十分注意するようにしてください．

【注意】

リダイレクションはたいへん便利なUNIXの機能です．しかし，方向 (<, >) を間違えると大切なファイルが上書きされてしまう危険があります．危険防止のために，大切な入力ファイルは書き込み権を chmod コマンドで禁止しておくか，コピーをとっておくことをお勧めします．

以下の例では，実行可能ファイルを“ a.out ”，入力ファイルを“ in.data ”，出力ファイルを“ out.data ”としています．

```
% a.out < in.data ↵        ←標準入力を in.data の内容から読み込む
```

```
% a.out > out.data ↵       ←標準出力への出力結果を out.data に保存（上書き）
```

```
% a.out >> out.data ↵      ←標準出力への出力結果を既存の out.data に 追加 書き
```

```
% a.out >& out.data ↵      ←出力結果および実行時メッセージを out.data に保存
```

```
% a.out < in.data > out.data ↵  ← in.data から読み込み，結果を out.data に保存
```

5.3 環境変数による結合

標準入出力以外の外部ファイル装置番号とファイルとの結合は，環境変数“ fuxx ”で行なうこともできます．xx に外部ファイル装置番号を指定します．番号が一桁の場合“ fu03 ”などと指定してください．環境変数を設定せずにファイル出力を行なった場合は“ fort.xx ”というファイルに出力されます．標準入力以外の装置番号に対し環境変数を設定せずにファイル入力を行なった場合には実行時にエラーとなります．

以下の例は，装置番号 10 番とファイル“ example.data ”を結合し，実行後，unsetenv によって結合を解除する例です．

```
% setenv fu10 example.data ↵     ←ファイルと装置番号の結合
% a.out ↵                        ←実行
      :
% printenv fu10 ↵                ← printenv による確認
 example.data
% unsetenv fu10 ↵                ←結合の解除
```

6 デバッグ

プログラムに潜むエラーを「バグ」，バグを取り除くことを「デバッグ」といいます．ここでは，デバッグのためのオプションの指定方法とあわせて，デバッグのポイントを簡単に説明します．

6.1 デバッグオプション

デバッグオプションを指定することで，プログラムの妥当性をチェックすることができます．一見，正しそうに見えるプログラムであっても，思わぬミスが見つかることがあります．

デバッグオプションは -Haesux と指定します．-H に続く小文字 a, e, s, u, x にはそれぞれサブオプションとしての意味があります．しかし，通常はすべて指定することをお勧めします．サブオプションのそれぞれの機能は man frt で参照することができます．

【注意】

デバッグオプションは，翻訳時だけでなく，デバッグオプションを指定して翻訳・結合編集し，実行する段階でも意味を持ちます．したがって，必ず実行可能ファイルを指定してプログラムを実行させてください．また，デバッグオプションを指定して作成した実行可能ファイルは，プログラムの動作のチェック用の特別な機能を組み込んでいるため，通常実行速度が遅くなります．したがってデバッグが完了した後，デバッグオプションを指定せずに，実行可能ファイルを作成し直してください．

以下は，デバッグオプションを指定して実行可能ファイルを作成し，実行によってバグが検出された例です．

```
% frt -Haesux example.f95 ↵              ←デバッグオプションの指定
% a.out ↵                                ←実行を通してデバッグ
                    :
 jwe0320i-w line 7 The subscript or substring M is out of the specified range
  (reference value: 334,1001, specification value: 1:1000,1:1000).
  error occurs at MAIN__   line 7 loc 00010a1c offset 00000180
  MAIN__       at loc 0001089c called from o.s.          ←警告メッセージ
 taken to (standard) corrective action, execution continuing.
```

上の例では，DO ループの中の添字が配列宣言された値を飛び出している旨の警告メッセージが出力されています．

◇ たくさんのメッセージが出力される場合

画面に収まり切らないたくさんのメッセージが出力される場合には，5.2 節のリダイレクションを用いて結果をファイルに保存することをお勧めします．

```
% a.out >& out ↵        ←実行結果をファイル out に出力
% more out ↵            ← more コマンドによってファイルの中身を表示する例
```

6.2 初心者に見られるバグ

以下，Fortran プログラムに慣れていない人によく見られるミスを列挙します．すべてプログラムの入力ミスです．多くの場合，翻訳時に

```
  jwd1302i-s  "a.f90", line *: この文は,FORTRAN の文とはみなせません.
```

```
jwd1035i-s  "a.f90", line 3, column 4: 正しい演算子が指定されていません.
```

などの診断メッセージが出力されます．

- 0 と O

数字の 0(ゼロ) と 英大文字の O の打ち間違いは，なかなか気がつきません．

```
    A=1.2DO    ! 0 ではく O と入力している
```

- 全角スペースの混入

半角スペースではなく全角スペースが入ってしまっている場合，画面を見ただけではまったくわかりません．エラーメッセージが出力される行の空白部分にカーソルを合わせて全角スペースでないか確認してみてください．

- **WRITE** 文

WRITE 文で括弧の対応がとれていないことも多く見受けます．

```
    WRITE(*,'(1X,I3,E20.13)' I,X(I)   ! 右括弧が不足
```

- 名前の指定

プログラム名，変数名の最初が数字になっていると，文法の制約からエラーとなります．最初は英字の必要があります．

```
  REAL 1X      ! 最初が数字のため文法違反
```

6.3 よく見られるバグ

以下，よく遭遇すると思われるバグ (とは言えないものも含まれています) とデバッグの方法を列挙します．

- ファイル入出力時のエラー

実行時にファイル入出力に関する以下のエラー:

```
jwe0113i-e line 6 A(an) unformatted I/O statement cannot be executed for
          a unit connected to formatted (unit= 1).
```

が表示される場合は，5.1 節で説明した通り，書式なし入出力文の FORM 接続指定子が記述されていない可能性があります．OPEN 文を調べてみてください．

◆ 反応がない

実行時に，たとえば，

```
% a.out ↵          ←実行
                   ←このまま先に進まない
```

となって，この状態のままいつまで待っても先に進まない場合には，READ 文によってキーボードはたは標準入力からデータを入力するプログラムを記述していることが考えられます．その場合は，指定の数値を入力して Enter キーを押し下げます．

その他，実行が無限ループなどに入っている可能性もあります．実行を強制的に打ち切るには，Ctrl(コントロール) キーを押しながら C キーを押し下げます．

◆ π

円周率 π は組込み関数がないため，自身で値を作成する必要があります．その際，数字を間違って入力したり，少ない桁数しか入力しなかったために予想した実行結果が得られないことがあります．

```
REAL :: PI              ! π の宣言
PI=3.1515926535         ! 入力ミス．正しくは 3.1415926535...
```

対策として，次のように π を作成することをお勧めします．

```
REAL :: PI              ! π の宣言
PI=4*ATAN(1.0)          ! π の作成 (単精度)
```

倍精度では (たとえば) 次のようになります．

```
REAL(KIND(1D0)) :: PI   ! π の宣言
PI=4*ATAN(1D0)          ! π の作成 (倍精度)
```

◆ **Fortran 90 からサポートされた関数が使用できない**

拡張子が .f または .for である Fortran ソースプログラムに Fortran 90 から新しくサポートされた組込み関数を記述した場合，翻訳時にエラーとなりことがあります．翻訳時オプション -X9 を指定してください．

◆ どうも結果がおかしい

様々な原因が考えられます．可能性のひとつとして，Fortran では整数と整数の割算が整数となることの考慮洩れが考えられます．

```
   a=1/10       !   0 となる
   b=1.0/10     ! 0.1 となる
```

プログラムによっては，この性質を用いて文を記述することもあるため，文法上はまったく問題ありません．しかし，結果が実数となることを前提としたプログラムの場合，予想と実行結果が異なることになります．

6.4 バグの混入を防ぐには

デバッグの観点からは，翻訳時や実行時に診断メッセージが出力される方があり難いと言えます．なぜなら，デバッグオプションでも検出されず，特定の入力データに対しては妥当な実行結果を出力しながら，実はアルゴリズムとまったく違う処理を行なうプログラムになっていて，それに気がつかずにいるという恐ろしいことも考えられるからです．

結局のところバグの混入を防ぐには《注意深くプログラムを作成することが肝要》という，ごくあたりまえのことになります．以下の方法も何かの役に立つかもしれません．

◆ IMPLICIT NONE の挿入

PROGRAM 文の直後に IMPLICIT NONE と記述することによって，暗黙の型宣言を抑止します．暗黙の型宣言の抑止によって，変数をすべて宣言する手間が生じる反面，スペルミスによって混入する可能性のあるバグを除去することができます．以下に (多少極端な) 例を示します．

```
PROGRAM TEST
   S=0
  IO=200
  DO I=1,I0     ! IO or I0 ?
    S=S+I**2
  END DO
  WRITE(*,*) S
END PROGRAM
```

上のプログラムは 1 から IO=200 までの繰り返し計算を意図して書かれたプログラムとします．しかし，数字の 0 と英字の O を間違えて入力しているため，もし I0 の値がゼロならば DO 文は実行されません．IMPLICIT NONE を記述して，変数宣言を行なうプログラムに変更すると次のようになります．

```
PROGRAM TEST
  IMPLICIT NONE      ! 暗黙の型宣言の抑止
  REAL    :: S       ! 変数の宣言
  INTEGER :: I,IO
   S=0
   IO=200
   DO I=1,IO         ! タイプミスは残ったまま
     S=S+I**2
   END DO
   WRITE(*,*) S
END PROGRAM
```

すると，翻訳時に宣言されていない I0 を検出してエラーを返すため，バグの混入を防ぐことができます．

◆ 精度を変えてみる

計算機で扱うことのできる数 (実数と複素数) は有限の桁数で表現されます．そのため，場合によっては結果に思わぬ誤差が生じることもあります．

計算結果が予想と異なると思われる場合には，このことも疑ってみるべきです．もしプログラムを単精度で作成している場合には，精度を倍精度に変更して結果を比較してみるといいでしょう．

◆ データを変えてみる

入力データをいろいろ変えてみて実行結果を比較することは，簡単でかつ効果的なデバッグの手段です．

7 数値計算ライブラリの利用

UNIX ワークステーションの Fortran では，数値計算ライブラリ SSL II を利用することができます．この章では SSL II の概要と利用方法を紹介します．

7.1 SSL II

SSL II(Scientific Subroutine Library II for Vector Processor) は，連立 1 次方程式や微分方程式などの数値計算を行う約 280 種類のサブルーチンからなる汎用数値計算ライブラリです．各サブルーチンは Fortran で記述されており，利用者プログラムから CALL 文で使用できます．表 6 に SSL II の概要を示します．

表 6: SSL II の概要

線形計算	連立 1 次方程式, 逆行列, 最小自乗解, 特異値分解
固有値問題	固有値, 固有ベクトル
非線形方程式	代数方程式, 超越方程式, 連立非線形方程式
極値問題	関数の極小化, 線形計画問題, 非線形計画問題
補間・近似	補間式, 近似式, 平滑化, 級数展開
変換	フーリエ変換, ラプラス変換
数値微積分	離散点, 関数入力, 有限区間, 無限領域, 1 次元, 2 次元
微分方程式	連立 1 階常微分方程式, 連立 1 階スティフ常微分方程式
特殊関数	ベッセル関数, 楕円積分, 指数積分, 正弦・余弦積分
擬似乱数	乱数生成 (一様/正規/指数/ポアソン/二項), 乱数検定

7.2 オンラインマニュアル

SSL II の利用方法は man コマンド経由で検索することができます．機能一覧は man ssl2 です．個別のサブルーチンについては，man ssl2 で確認した後，例えば man dvlax，man vmggm などで機能・引数等が検索できます．

```
% man dvlax ↵            ←サブルーチン dvlax の検索

Scientific Subroutine Library                              vlax(3F)
【名前】
     vlax, dvlax - 実係数の連立 1 次方程式 ( ブロッキング L U 分解法 )

【形式】
       CALL VLAX(A,K,N,B,EPSZ,ISW,IS,VW,IP,ICON)

【機能説明】
   (1) 機能
       実係数連立 1 次方程式
            A x = b
              :
```

また，冊子体のマニュアル [3], [4], [5] はセンター 4 階図書室で参照することができます．

7.3 SSL II の利用方法

frt(または f90, f95) コマンドのオプション -l に続けてライブラリ名 fssl2 を指定し，実行可能ファイルを作成します．-l は結合編集時オプションのため，次の例のように，コマンド並びの 最後 に指定してください．

```
% frt main.f90 -lfssl2 ↵        ← SSL II の結合
```

8 便利なコマンド

8.1 時間計測コマンド

timex コマンドにより，翻訳・結合編集・実行に要する経過時間，CPU 時間を計測することができます．

```
% timex frt example.f95 ↵              ←翻訳・結合編集時間の計測
 real        4.01                      ←経過時間 (4 秒 01)
 user        1.70                      ←ユーザー CPU 時間
 sys         0.80                      ←システム CPU 時間

% timex ./a.out ↵                      ←実行時間の計測
 real       10.30                      ←経過時間 (10 秒 30)
 user       10.24                      ←ユーザー CPU 時間
 sys         0.03                      ←システム CPU 時間
```

また，Fortran 95 標準の組込みサブルーチンとして，経過時間を計測する SYSTEM_CLOCK，

CPU 時間を計測する CPU_TIME が利用できます．これらのサブルーチンを拡張子 .f または .for のついたファイルに記述した場合には，翻訳時オプション -X9 が必要です．利用方法はコマンドラインから man system_clock または man cpu_time で調べることができます．

8.2 file コマンド

手元のファイル名に拡張子などがなく，何のファイルだったかわからなくなった時には，ファイルコマンドでタイプを判定することができます．

実行可能ファイルの場合，次のように表示されます．

```
% file step2 ↵        ←ファイルタイプの判定

step2:          ELF 32-ビット MSB 実行可能 SPARC バージョン 1[動的にリンクされています][取り除かれていません]
```

9 おわりに

本稿では UNIX ワークステーションにおける Fortran の基本的な利用方法について説明しました．UNIX ワークステーションの利用方法に限らず，教育用システムに関する質問・要望は

uketuke@cse.ec.kyushu-u.ac.jp

で受け付けています．

参考文献

[1] アスキー デジタル用語辞典 http://yougo.ascii24.com/

[2] FUJITSU Fortran 使用手引書 V4 用, 富士通株式会社, J2X0-3350, 1998.

[3] 富士通 SSL II 使用手引書 (科学用サブルーチンライブラリ), 富士通株式会社, 99SP-4020, 1987.

[4] FUJITSU SSL II 拡張機能使用手引書 (科学用サブルーチンライブラリ), 富士通株式会社, 99SP-4070, 1991．

[5] FUJITSU SSL II 拡張機能使用手引書 II(科学用サブルーチンライブラリ) V4.0 用, 富士通株式会社, J2X0-1366, 1997．