MSP ユーザのための UNIX 入門

– SPP で Fortran –

赤坂 浩一 * 平野 彰雄 * 金澤 正憲 *

1 はじめに

本センターの計算機システムをオペレーティングシステム (OS) で、大別すると UNIX と MSP に分けることができます。

MSP システム (汎用コンピュータ:kuma) は、永年多くの方に利用されてきましたが、近年、ワークステーション (WS) やパーソナルコンピュータ (PC) の普及にともない、新規に利用する方が少なくなっています。

今後、その傾向は強くなることが予想され、近い将来、MSP システムのサービスが終了した場合は、UNIX システムをお使い頂くことになるでしょう。

本稿では、現在、MSP システムで Fortran プログラムを実行されている方が、本センターの UNIX システム (計算サーバ:spp) で実行するための方法を簡単に紹介します。

2 データセットとファイルのお話

MSP システムでは、磁気ディスク装置に保存されている利用者のソースプログラムや入出力データをデータセットと呼んでいます。UNIX システムでは、これらはファイルと呼ばれています。

MSP システムで実行していた Fortran プログラムを UNIX システムで実行するためには、ソースプログラムや入出力データ等のデータセットを UNIX システムのファイルとして、作成しなければなりません。

一からソースプログラムや入出力データを作成することは、非常に手間がかかりますので、MSP システムから UNIX システムにファイル転送することが良いでしょう。

ここでは、ファイル転送する前に、MSP システムのデータセットと UNIX システムのファイルについて、簡単にまとめておきます。

2.1 レコード形式

MSP システムのデータセットには、データセット毎に必ず、レコード形式があり、また、1 レコード (1 行) の長さ、レコード長が決まっています。

一般的にお使いになるデータセットのレコード形式は、全てのレコードの長さが同一の固定長レコード形式、レコード毎に長さが異なる可変長レコード形式のどちらかでしょう。

固定長レコード形のレコード長は、各レコードの長さであり、実際のデータが満たないレコードは、空白文字で満たしています。

可変長レコード形式のレコード長は、使用できる最大レコードの長さであり、1 レコード毎の長さは、実際にデータが入っている部分の長さとなります。

UNIX システムのファイルは、全て可変長レコード形式です。

2.2 行番号

データセットには、1 レコード毎に 8 バイト (8 文字分) の行番号を付加することができ、固定長レコード形式ではレコードの末尾の 8 バイト、可変長レコード形式ではレコードの先頭の 8 バイトを行番号として扱います。

この行番号、MSP システムでは、存在して当たり前、逆に無いと不自由だったりするものですが、UNIX システムでは、かえって邪魔な存在だったりします。

* あかさか ひろかず , ひらの あきお , かなざわ まさのり (京都大学大型計算機センター)

2.3 データセット編成

MSP システムには、データを保持する方法として一般的に良く使われるのが、順編成 (PS) や区分編成 (PO) と言ったデータセット編成です。

順編成データセットは、一つのデータセットに一つのデータが格納されており、区分編成データセットは、一つのデータセットに複数のデータが格納されています。

UNIX システムでは、ファイルは階層的に管理されていますので、順編成データセットは一つのファイル、区分編成データセットは、あるディレクトリ上にある複数のファイルと考えることができます。

2.4 データセット名

MSP システムのデータセット名は、いくつかの制約があります。例えば、英文字で始まる 8 文字以内の文字列をピリオド (.) でつないで合計 44 文字以内の文字列でなければなりません。また、区分編成データセットで使用するメンバ名は、英文字から始まる 8 文字以内の文字列でなければなりません。

UNIX システムでのファイル名やディレクトリ名は、全く制約が無いわけではありませんが、 MSP システムよりも柔軟です。

なお、 MSP システムでは、データセット名は英文字は必ず、大文字として扱われますが、 UNIX システムでは、英小文字と英大文字を区別して扱われます。

では、以上のことを踏まえて、 MSP システムのデータセットを UNIX システムにファイル転送しましょう。

3 ファイル転送

ファイル転送には、いくつかの方法がありますが、ここでは、 MSP システムの RCP コマンドを利用して、 MSP システムからリモートホストとなる UNIX システムにファイル転送します。

3.1 RCP コマンドの使い方

RCP コマンドは、次のように転送したいデーターセット名 (ds 名) と転送先のホスト名 (host) とファイル名 (file 名) を指定して実行します。

```
#  RCP [-R] ds 名  "@host:file 名"
```

-R オプションは、区分編成データセットを転送する場合に指定します。また、ファイル名 (file 名) は省略することができ、省略するとデータセット名 (ds 名) が使用されます。区分編成データセットの場合は、データセット名がディレクトリ名となり、そのディレクトリ配下にメンバ名がファイル名として転送されます。

転送するデータセットに漢字コードを含む場合は、データセット名 (ds 名) の直前に変換したい漢字コードを指定することができます。指定できる漢字コードは次のものです。省略した場合は、 EUC となります。

- S-: シフト JIS
- 7-: 78 年度版 JIS
- 8-: 83 年度版 JIS

RCP コマンドの詳しい使い方は、 MSP システムの HELP コマンドでご覧ください。

3.2 下準備

転送したいデータセットが、行番号付きの場合、そのまま転送すると、 UNIX システムでは、不要な行番号も付加されて転送してしまいます。

不要な行番号は、 UNIX システムに転送してからでも取り除くことができますが、 MSP システムを使い熟れた方は、予め MSP システムで加工してから、ファイル転送を行なうのが良いでしょう。

以下に、 PFD エディタを使った行番号の除去の手順を示します。この例では、固定長レコード形式で行番号付きのデータセット FB.FORT の行番号を除去します。

図 1 のように、 FB.FORT を PFD エディタで開きます。画面の左に表示されている行番号は、 PFD エディタが使用する行番号であり、実際のデータセットの行番号は、固定長レコード形式のデータセットの場合、行の末尾に付加されていますので、画面では丁度隠れています。

PF11 キー等で、画面を右方向にスクロールすると、図 2 のように行の末尾の 8 バイトに行番号が付加されているのが確認できます。

データセットの行番号を除去するためには、エディタのサブコマンドである UNNUM コマンドを使用します。

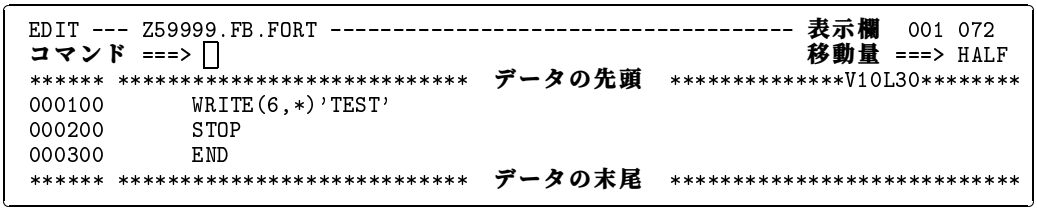

図 1. PFD のデータセット編集画面

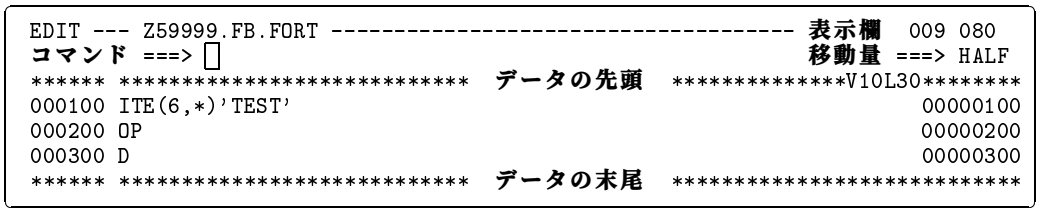

図 2. スクロールして末尾を表示

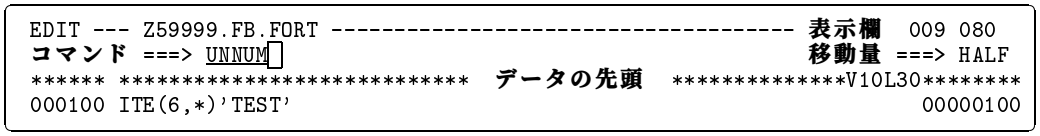

図 3. UNNUM コマンドの指定

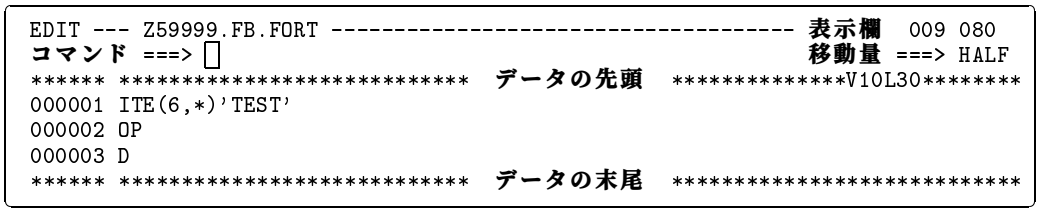

図 4. UNNUM コマンドの実行結果、行番号が除去される

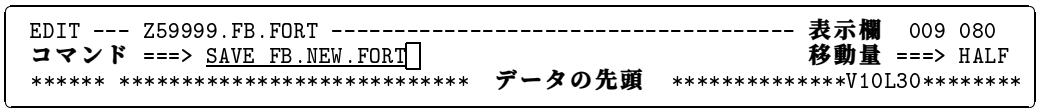

図 5. 別名で保存する

UNNUM **コマンドは、コマンド入力フィールド** (**コマンド** ===>) **に指定し、実行します。図** 3 **のように、** UNNUM **とタイプして** ENTER **キーを押し実行すると、図** 4**のように、データセットの行番号が除去されます。**

このまま、 PF3 **キーを押して、データセットを保存してもかまいませんが、ここでは、図** 5 **ように、別のデータセット** (FB.NEW.FORT) **に保存してから、** PF3 **キーを押して終了します。**

この例では、固定長レコード形式のデータセットで説明しましたが、可変長レコード形式の場合も同様の操作で行番号を除去することができます。なお、可変長レコード形式の行番号は先頭の 8 **バイトにありますので、画面をスクロールして確認する場合は、** PF10 **キー等で左方向にスクロールさせてください。**

転送したいデータセットが順編成データセットで、少数の場合は、一つずつ上記の操作で行番号を除去していけば良いかもしれませんが、例えば、区分編成データセットでメンバが沢山あるような時は、転送してから UNIX **システムで行番号を除去する方が良いかもしれません。**

しかし、 UNIX **システムで行番号を除去する場合、行番号付きの固定長レコード形式のデータセットをそのまま転送すると行番号とともに不要な空白文字も除去する作業が発生するので、図** 6 **ように、** COPY **コマンドで行番号付きの可変長レコード形式に変換して、転送するのが良いでしょう。**

```
#  COPY FB.PO.FORT VB.PO.FORT RECFM(VB) LRECL(255) BLKSIZE(6144)
```

図 6. COPY コマンドで可変長レコード形式に変換

この例では、行番号付き固定長レコード形式の区分編成データセット FB.PO.FORT を新たに行番号付き可変長レコード形式の区部編成データセット VB.PO.FORT として作成します。オペランドとして指定しているのは、レコード形式の RECFM(VB) とレコード長の LRECL(255)、そしてブロックサイズの BLKSIZE(6144) です。

UNIX システムでの、行番号と行末の不要な空白文字の除去方法は、後ほど説明します。

3.3 RCP コマンドの実行例

下準備が完了したら、RCP コマンドを利用して、データセットを UNIX システムにファイル転送しましょう。この例では、転送先の UNIX システムは、計算サーバ (spp) となっています。

【例 1】

順編成データセット FB.NEW.FORT を転送する。ファイル名は省略する。

```
#  RCP FB.NEW.FORT "@spp:"
```

spp のホームディレクトリ配下に、"利用番号.FB.NEW.FORT" という名前でファイルが転送される。

【例 2】

順編成データセット FB.NEW.FORT を fb.new.f というファイル名で転送する。

```
#  RCP FB.NEW.FORT "@spp:fb.new.f"
```

spp のホームディレクトリ配下に、"fb.new.f" という名前でファイルが転送される。

【例 3】

漢字コードを含む順編成データセット KANJI.FORT を kanji.f というファイル名で転送する。kanji.f は 83 年度版 JIS に変換する。

```
#  RCP 8-KANJI.FORT "@spp:kanji.f"
```

spp のホームディレクトリ配下に、"kanji.f" という名前でファイルが転送される。漢字コードは 83 年度版 JIS。

【例 4】

区分編成データセット VB.PO.FORT を転送する。ファイル名 (ディレクトリ名) は省略する。

```
#  RCP -R VB.PO.FORT "@spp:"
```

spp のホームディレクトリ配下に、"利用番号.VB.PO.FORT" という名前でディレクトリが作成され、そのディレクトリ配下にメンバがファイルとして転送される。

【例 5】

区分編成データセット VB.PO.FORT を vb.po というディレクトリ名で転送する。

```
#  RCP -R VB.PO.FORT "@spp:vb.po"
```

spp のホームディレクトリ配下に、vb.po という名前のディレクトリが作成され、そのディレクトリ配下にメンバがファイルとして転送される。既にディレクトリが存在する場合は、そのディレクトリ配下に "利用番号.VB.PO.FORT" という名前でディレクトリが作成され、そのディレクトリ配下にメンバがファイルとして転送される。

これで、MSP システムのデータセットが UNIX システムのファイルとして転送できました。それでは、計算サーバ (spp) で Fortran プログラムを実行しましょう。

4 計算サーバ (spp) の利用

4.1 ログインする

まず、計算サーバ (spp) にログインしましょう。spp へのログインは、研究室や自宅でお使いの WS や PC から Telnet や SSH などを利用して行なったり、本センターの利用者端末 (PC 端末や X 端末) などから利用できますが、ここでは、詳しくは説明しません。

現在、MSP システムの利用を PC や WS から、Telnet で接続されている方は、接続するホスト名

```
Trying 130.54.9.25...
Connected to spp.kudpc.kyoto-u.ac.jp.
Escape character is '^]'.


SunOS 5.8

login:
```

図 7. spp のログイン画面 (接続時)

に spp.kudpc.kyoto-u.ac.jp を指定すれば、図 7 のように、 spp のログイン画面が表示されます。

login: のプロンプトに対して、利用番号を入力し、Enter キーを押すと、次に Password: のプロンプトが表示されますので、UNIXシステムのパスワードを入力してください。

MSP システムと UNIX システムでは、パスワードを別々に管理しているので、 spp や他の UNIX システムを利用する場合は、必ず、 UNIX システム用のパスワードを入力してください。

また、 UNIX システムでは、英大文字と英小文字は区別されます。利用番号の先頭の英字は必ず英小文字で入力し、パスワードに含まれる英字は、英小文字は英小文字で、英大文字は英大文字で入力してください。

なお、 UNIX システムのパスワードがわからない方は、利用承認書に初期パスワードを印字していますのでそちらをご覧ください。

正しい利用番号とパスワードを入力すると、図 8 のようなメッセージが表示され、使用するターミナルタイプを問い合わせてきます。なお、紙面の関係上、画面の右端をカットして表示しています。

Terminal Type (default vt100): には、何も書かずにそのまま Enter キーを押します。

spp のプロンプト、 spp[1]> が表示されれば、ログインは完了です。

```
login: m54966
Password:
Last login: Thu Nov 22 10:41:09 from wink

Last accounting date : Sun Jan 20 00:39:4
SunOS 5.8
Account id : a
       CPU charge       26,025 yen
         ------- 省 略 ------

Terminal Type (default vt100):
spp[1]>
```

図 8. spp のログイン画面 (ログイン完了)

4.2 UNIX のコマンド

UNIX のコマンドは、シェルと呼ばれるコマンドラインインターフェースを通じ、実行されます。 spp では、 spp[1]> の表示がシェルのプロンプトで、このプロンプトに対して、コマンドを記述し実行します。 spp では、 tcsh というシェルが標準となっています。

ここでは表 1 に最低限必要なコマンドだけを簡単に紹介しますが、使用例などは省略します。より詳しく知りたい方は、本センターで発行してる「利用の手引き –UNIX 編–」や市販されている入門書などを参照してください。

UNIX システムでは、 MSP システムの HELP コマンドに相当するコマンドとして、 man コマンドが用意されています。 man コマンドの使い方は次のようになります。

```
spp[1]> man コマンド名
```

また、シェルの環境変数 LANG に ja を設定しておくと、 man コマンドの結果やメッセージに日本語が用意されている場合は、日本語で表示されます。

環境変数の設定には、 setenv コマンドを使用し、次のように設定します。設定した環境変数は、そのシェルを終了するまで有効となります。

```
spp[1]> setenv LANG ja
spp[2]> man コマンド名
```

この他に、シェルの機能として、コマンドの入出力にファイルを利用するリダイレクションやコマンドの出力を別のコマンドの入力とするパイプなどがあります。また、ファイルを操作するコマンドを使用する際に、ファイル名にワイルドカードを使用することができます。

<	ファイルの中身をコマンドの入力にする
>	コマンドの出力をファイルに書き出す
>>	コマンドの出力をファイルに追加書きする
|	コマンドの出力を他のコマンドの入力につなぐ
*	任意の 0 文字以上の文字列
?	任意の 1 文字

それでは、 MSP システムからファイル転送した Fortran プログラムを例にファイルの編集などを行なってみましょう。

表 1. 必要最小限の UNIX コマンド

コマンド名	機　　能
cat	ファイルの中身を表示する。自動的にスクロールする、つまり、垂れ流し
less	ファイルの中身を表示する。一画面で表示が停止
ls	ディレクトリの内容 (一覧) を表示する。"-l" オプションで詳細表示
cd	カレント (現在の) ディレクトリを変更する。
pwd	カレントディレクトリを表示する。
cp	ファイルをコピーする。"-r" オプションでディレクトリのコピー
mv	ファイルやディレクトリを移動 (名前を変更) する。
rm	ファイルを削除する。"-r" オプションでディレクトリの削除
mkdir	ディレクトリを作成する。
je	ファイルを編集する。PFD エディタのようなスクリーンエディタ
setenv	環境変数を設定する。
unsetenv	設定した環境変数を取り消す。
logout	ログインのセッションを終了する。
passwd	パスワードを変更する。

4.3　je の使い方

ファイルの中身を表示するだけならば、`cat` や `less` コマンドを使えば良いでしょうが、中身を編集したい場合は、エディタが必要です。

UNIX システムでは、最も標準的な vi エディタや高機能なエディタ (mule,xemacs) がありますが、永年、MSP システムを使い熟れてきた方々にとって、新しいエディタの操作を覚えることは悪戯に時間を費やすだけですので、ここでは、MSP システムの PFD エディタのような je を紹介します。

je は、日本原子力研究所の谷田部茂さんが、作成されたエディタで、本センターの UNIX 系システム (sakura,spp,vpp) にインストールしています。

なお、本腰を入れて UNIX システムを使いこなすのであれば、mule や xemacs といった高機能なエディタをお使いになるのが良いと思います。

je は、次のように起動します。

```
spp[1]> je [-b] [ファイル名]
```

ファイル名を省略すると、コマンドを起動したカレントディレクトリのファイル一覧が表示されるので、その一覧から編集したいファイルを選択します。"-b" オプションを指定するとブラウズ (閲覧) モードで起動します。

それでは、`fb.new.f` を編集してみましょう。

```
spp[3]> je fb.new.f
```

のように入力して、Enter キーを押すと、図 9 のように、je が起動します。

je は、ほぼ PFD エディタと同じような操作で使用できます。例えば、画面の上下のスクロールは、F7 , F8 キーで行ないます。

カーソルの移動は、→ , ↑ , ↓ , ← キーで行ない、Delete キーによる文字の削除や行コマンド (`D` 行削除、`I` 行挿入など) もサポートされています。

```
EDIT --- fb.new.f ----------------------------------------------COLUMNS  001 072
COMMAND ===>                                                  SCROLL ===> PAGE
****** ***************************** TOP OF DATA ****************V01B04*******
000010       WRITE(6,*)'TEST'
000020       STOP
000030       END
****** ************************** BOTTOM OF DATA **************************
```

図 9. je のファイル編集画面

je を終了する場合も、PFD エディタと同じように F3 キーを押します。ファイルが更新されている場合は、終了時の処理を問い合わせてくるので、↑, ↓ キーで選択して、Enter キーを押します。

終了時の処理には、次の 4 つが選択できるようになっていますが、現時点では、Save & end. 以外は、全て No save end. つまり、変更が無効となりファイルが更新されませんので、ご注意ください。

- Save & end.
- File name change, save & end.
- No save end.
- (No used)Cancel.

若干、動作に不具合がありますが、とりあえず編集作業は可能です。

4.4 行番号と不要な空白文字の除去方法

ここでは、行番号付き可変長レコード形式のデータセットとしてファイル転送した場合の不要な行番号と行末の空白文字の除去方法を紹介します。

ファイル vb.fort の中身は、次のようになっているとします。ここでは、行末の不要な空白文字を □ として表しています。

```
spp[4]> cat vb.fort

00000100      WRITE(6,*)'TEST'□□□□□□□□
00000200      STOP□□□□□□□□□□□□□□□□□□□□
00000300      END□□□□□□□□□□□□□□□□□□□□□□
```

不要な行番号と行末の空白文字を除去するには、複数のコマンドを組み合わせて行ないます。不要な行番号は、各行の 1 〜 8 カラムにありますので、cut コマンドを使って、9 カラム目以降のデータを取り出し、パイプ (|) でつなぎ、sed コマンドをフィルタをして使って、行末の不要な空白文字を取り除きます。そして、結果をリダイレクション (>) で別のファイルに保存します。

cut コマンドで指定している -c9-255 は、各行の 9 文字目から 255 文字目の取り出しを指示しています。sed コマンドに指定している 's/ *$//' は、各行の末尾から空白文字以外が現れるまで空白文字の除去を指示しています。

この一連の処理は、次のように入力します。なお、紙面の関係上、"→" で改行していますが、コマンドは続けて入力します。

```
spp[5]> cut -c9-255 vb.fort | →
          sed 's/ *$//' > vb.new.f
```

ファイル vb.new.f の中身は、次のようになります。

```
spp[6]> cat vb.new.f

    WRITE(6,*)'TEST'
    STOP
    END
```

以上のようにして、不要な行番号と行末の空白文字を除去することができます。

区分編成データセットをファイル転送した場合は、ディレクトリ配下に複数のファイルが作成されますので、ファイル数が多くなると一つ一つのファイルに対して上記の操作を行なうのは手間がかかります。

そこで、ディレクトリ配下のファイルの不要な行番号と空白文字を一括して除去する方法を紹介します。

既にお気づきの方もおられるかも知れませんが、UNIX システムでは、Fortran プログラムのファイル名には、.f の拡張子を使用するようになっています。

区分編成データセットをファイル転送した場合は、メンバ名がファイル名となっていますので、別のファイルに落とすときにファイル名に.f の拡張子を付けて保存するのが良いでしょう。

ディレクトリ vb.po の中には、次のようなファイルが存在しているものとします。

```
spp[7]> ls vb.po

EXAM0001  EXAM0003  EXAM0005  EXAM0007
EXAM0002  EXAM0004  EXAM0006  EXAM0008
```

ディレクトリ配下の全ファイルに対して、同じ処理を繰り返し実行することになるので、foreach コマンドを使った例を紹介します。

それでは、処理を行なうディレクトリ vb.po に cd コマンドで移動し、次のように foreach コマンドを入力します。

```
spp[8]> cd vb.po
spp[9]> foreach file (*)
```

foreach コマンドのオペランド"file (*)"は、このディレクトリにある全ファイルを対象に処理を行なうという意味で、foreach コマンド内部で $file という変数にファイル名が渡されます。

コマンドを実行すると、foreach? のプロンプトが表示されるので、このプロンプトに対し、繰り返し処理したいコマンドを入力します。foreach? のプロンプトに end を入力すると繰り返し処理したいコマンドが実行されます。

また、foreach コマンドは、end を入力するまでは、複数のコマンドを記述し、実行することができますので、例えば、行番号と空白文字を除去し、別のファイルに落とした後で、元のファイルを削除したりすることも可能です。

次の例は、行番号と空白文字を除去し、結果を元のファイル名に.f を付加した別のファイルに落とし、元のファイルを削除しています。

```
spp[9]> foreach file (*)
foreach? cut -c9-255 $file | →
            sed 's/ *$//' > $file.f
foreach? \rm $file
foreach? end
spp[10]>
```

ファイルの削除を行なう rm コマンドを \rm としているのは、削除して良いかの問い合わせを行なわないようにするためです。

ディレクトリ vb.po の中のファイルは、次のようになりました。

```
spp[10]> ls

EXAM0001.f EXAM0003.f EXAM0005.f EXAM0007.f
EXAM0002.f EXAM0004.f EXAM0006.f EXAM0008.f
```

これは、オマケですが、例えば、行番号と空白文字を除去した結果を落とすファイル名を英小文字にしたい場合は、次のようになります。

```
spp[9]> foreach file (*)
foreach? set a = "`echo $file | →
          nawk '{print tolower($1)}'`"
foreach? mv $file $a
foreach? cut -c9-255 $a | →
              sed 's/ *$//' > $a.f
foreach? \rm $a
foreach? end
spp[10]>
```

nawk コマンドで、元のファイル名の英小文字を調べ、mv コマンドでそのファイル名に名前を変更してから、行番号や空白文字を除去しています。

それでは、いよいよ spp で Fortran プログラムを実行してみましょう。

4.5 Fortran プログラムの実行

MSP システムで、Fortran プログラムを実行されていた方は、会話型 (TSS) での実行よりバッチジョブ (SUBMIT ジョブ) として実行されていた方が、多いのではないでしょうか?

UNIX システムでは、会話型処理での実行が一般的ですが、本センターのように大規模な計算を処理するシステムでは、バッチ型 (NQS ジョブ) での利用が可能な場合もあります。

今回は、バッチ型 (NQS ジョブ) での利用方法は省略しますが、興味のある方は、本センターのホームページの「計算サーバ」をご覧ください。

http://www.kudpc.kyoto-u.ac.jp/SPP/

4.5.1 翻訳・結合

MSP システムの FORT コマンドは、オペランドに Fortran ソースプログラムを指定すると、翻訳、結合、実行の一連の処理を行ない実行結果を得ることができました。

spp では、はじめに frt コマンドで、Fortran ソースプログラムを翻訳、結合を行ない実行可能ファイルを作成し、その後、実行可能ファイルを実行して結果を得ることになります。

spp の Fortran コンパイラは、ISO/IEC 1439-1:1997,JIS X 3001-1:1998(Fortran95) に準拠しています。コマンド名は frt です。

ソースファイルの拡張子が、.f95 または.f90 であるファイルは、自由形式の Fortran95 プログラムと解釈され、拡張子が.f または.for であるファイルは、固定形式の Fortran77 プログラムと解釈されます。

以下に、コマンドの記述形式を示します。

コマンド名	オペランド
frt	[オプションの並び] ファイルの並び

frt コマンドの詳細は、man コマンドをご覧ください。以下に主なオプションを紹介します。

【主なオプション】

- -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 *file_name* : 実行可能ファイル、オブジェクトファイルの名前を *file_name* に変更します。省略時は a.out という名前になります。
- -Kfast_GP[=*level*] : 計算サーバ上で高速な実行性能を得るために適した最適化オプションを誘導します。*level* は 0,1,2 が指定でき、2 を指定したときがもっとも強力な最適化を行います。
- -Qp : ソースリスト、翻訳時に有効となったオプション、アンローリング情報、並列化情報をソースファイル名.lst に出力します。
- -NRtrap : 実行時に、オーバーフロー、アンダーフローなどの浮動小数点の演算例外や組込み関数の演算例外が発生した場合に、メッセージを出力します。

それでは、Fortran ソースプログラムを翻訳、結合して実行可能ファイルを作成してみましょう。

【例 1】

fb.new.f を最適化オプションをつけて翻訳し、実行可能ファイル a.out を作成する。

```
spp[11]> frt -Kfast_GP=2 fb.new.f
```

【例 2】

test.f を翻訳し、実行可能ファイル test.exe を作成する。また、翻訳情報をソースファイル名.lst に出力する。

```
spp[12]> frt -Qp -o test.exe test.f
```

spp の Fortran でも、SSLII などの数値計算ライブラリを結合して使用することができます。

【例 3】

samp.f を翻訳し、数値計算ライブラリ (SSLII) を結合して実行可能ファイル samp.exe を作成する。

```
spp[13]> frt -o samp.exe samp.f -SSL2
```

では、実行可能ファイルを実行してみましょう。

4.5.2 実行

作成した実行可能ファイル a.out を実行するには、次のように入力します。

```
spp[14]> ./a.out
```

UNIX システムでは、実行可能なファイル (コマンド) を探し出すのに、環境変数 PATH に設定されているディレクトリから検索を行ないます。

環境変数 PATH には、カレントディレクトリが含まれていないため、a.out を実行するには./ を付け./a.out として、ファイルの在処を指定する必要があります。

Fortran プログラムでは、READ 文や WRITE 文で装置番号を指定して、計算に必要なデータを読み込んだり、計算した結果を書き出したりします。

使用する装置番号にファイルを割り当てるには、ソースプログラム中に OPEN 文で指定するか、プログラムを実行する前に環境変数 fu?? に使用するファイルを設定します。?? には、使用する装置番号 (00 〜 99) を指定します。

しかし、標準エラー出力 (装置番号 0)、標準入力 (装置番号 5)、標準出力 (装置番号 6) に関しては、環境変数 fu?? に割り当てることができませんので、実行時にリダイレクションを使って、ファイルの入出力を行ないます。

それでは、実行前に環境変数 fu?? にファイルを割り当てた実行と実行時にリダイレクションを使った実行を行なってみましょう。

【例 1】

装置番号 8 の READ 文で使用する入力データのファイル名が in.dat、装置番号 10 の WRITE 文で使用する出力データのファイル名が out.dat の実行可能ファイル samp.exe を実行する。

```
spp[15]> setenv fu08 in.dat
spp[16]> setenv fu10 out.dat
spp[17]> ./samp.exe
```

【例 2】

実行可能ファイル a.out の実行時にリダイレクションを使って、標準入力から in.dat を読み込み、標準出力を out.dat に格納する。

```
spp[18]> ./a.out < in.dat > out.dat
```

【例 3】

実行可能ファイル a.out の実行時にリダイレクションを使って、標準入力から in.dat を読み込み、標準出力と標準エラー出力を out.dat に格納する。

```
spp[19]> ./a.out < in.dat >& out.dat
```

spp での Fortran プログラムの実行は、以上のような手順で行ないます。次に、spp を使うための注意点をいくつか紹介しておきます。

4.6 spp を使うための注意点

spp など、UNIX システムのプログラムの実行単位は、プロセスと呼ばれています。プロセスには制限値が設定されており、spp の会話型利用では、次のようになっています。

CPU 時間		メモリサイズ	
標準	最大	標準	最大
1 時間	20 時間	500MB	8GB

プロセスの制限値は、limit コマンドで確認することができます。また、limit コマンドはプロセスの制限値を変更することができます。

CPU 時間やメモリサイズの制限値を最大の許可量にする場合は、unlimit コマンドで行ないますが、不用意に最大許可量に設定しておくと、無限ループするようなプログラムを実行すると 20 時間分の CPU 課金が発生したりするので、注意が必要です。

必要な量だけを設定するようにしてください。

4.7 ログアウトする

それでは、spp をログアウトしましょう。ログアウトには logout コマンドを次のように入力します。

```
spp[20]> logout
```

5 おわりに

今回は、これまで MSP システムを中心にお使いいただいた方が、UNIX システムを使い始めるきっかけとなるような解説記事を目標に書き始めたが、少々まとまりが悪くなってしまいました。

あえて、MSP と spp の性能比較は行なっていませんが、spp の方が MSP に比べて、はるかに演算処理が勝っていることは、お使いいただければ実感していただけるのと思います。

使用できるメモリサイズや CPU 時間は、MSP を凌ぐ許可量となっているので、今後、大規模な計算を行なうには、MSP から spp に早く乗り換えられるのが、良いのではないでしょうか。

【附録】浮動小数の表現の違いに注意

MSP は 16 進数表示で、単精度 (4 バイト) でも倍精度 (8 バイト) でも、絶対値が 10^{-78} から 10^{75} の範囲を表現できます。即ち、指数部は同じ 8 ビットで表現されています。有効桁数がそれぞれ 10 進 5 桁、10 進 14 桁となります。

SPP は 2 進数表示で、単精度では絶対値が 10^{-37} から 10^{38}、倍精度では 10^{-307} から 10^{308} となります。有効桁数は、単精度で 10 進 6 桁、倍精度で 10 進 15 桁となります。即ち、指数部も仮数部もビット数が異なります。

サブルーチンの引数で、片方が単精度でもう一方が倍精度でも、MSP ではそれなりの計算をしていましたが、SPP ではとんでもないことになります。十分注意してください。

なお、上の数字はいずれも概数です。