プログラム実行時の入力ファイルの与え方について

東京大学情報基盤センター
黒田 久泰

プログラム実行時に入力ファイルを与えるには、①リダイレクトで与える方法、②環境変数で与える方法、③実行時に引数として与える方法、の３通りがあります。ここでは、それぞれの方法について紹介します。

1. リダイレクトで与える方法

次のようにリダイレクトでファイル名を指定する方法です。

```
% ./a.out < input.txt
```

ここでは、input.txt には数値データのみが格納されているものとして、C 言語と Fortran で記述したプログラム例を紹介します。

(C 言語の場合)

```
#include <stdio.h>
#define MAX_DATA_SIZE 1000000
double a[MAX_DATA_SIZE];

int main(int argc,char *argv[])
{
  int i,ret,size;
  char str[80];

  for(i=0;i<MAX_DATA_SIZE;i++){
    if(fgets(str,80,stdin)==NULL) break;
    ret=sscanf(str,"%lf",&a[i]);
    if(ret<1){
      printf("Format error¥n");
      return 0;
    }
  }
  size=i;
  ・・・・・・
```

(Fortran の場合)

```
      program main
      parameter(MAX_DATA_SIZE=1000000)
      integer i,io,size
      real*8 a(MAX_DATA_SIZE)

      do i=1,MAX_DATA_SIZE
        read(5,*,iostat=io) a(i)
        if(io .lt. 0) exit
      end do
      size=i-1

※あるいは次のような書き方もできます

      do i=1,MAX_DATA_SIZE
        read(5,*,end=999) a(i)
      end do
999   size=i-1

      ・・・・・・
```

C 言語では fgets と sscanf 関数、Fortran では read 文により入力ファイルを最後まで読み込みます。読み込みが終了するとループを抜け出します。そして最終的には、配列 a に input.txt 内の全ての数値データ、変数 size にそのデータの個数が入ります。

ところで、C 言語では for 文の中を scanf 関数を使って下記のように書くこともできます。

```
  ret=scanf("%lf",&a[i]);
  if( ret==-1 ) break;
```

しかし、この例では数値以外のデータが入っている場合に、scanf 関数では入力バッファにそのデータを残したままにするため、無限にそのデータを処理しようとしてしまい先に進まなくなってしまいます。そのため、scanf 関数の利用は避けた方がいいでしょう。

mpirun コマンドとリダイレクトを合わせて利用する場合には、システムによって動作が異なるので注意が必要です。具体的には、リダイレクトで指定した場合に入力データを 1 プロセスだけに与える場合、入力データを全プロセスに与える場合、入力データをどのプロセスにも与えない場合などがあり、システムによって異なります。例えば、MPI のフリーの実装系の 1 つである MPICH ではプロセス番号 0 番のみに与えられます。

本センターにおいても、標準の環境では、SR8000/MPP では標準入力がプロセス 0 番のみに与えられ、SR11000/J1 では標準入力が全てのプロセスに与えられるといった違いがあります。

ただし、SR11000/J1 では環境変数 MP_STDINMODE の設定により、標準入力を与えるプロセスを指定することができます。環境変数 MP_STDINMODE には次の値が設定できます。

all	全てのプロセスに同じ標準入力を与える
n	プロセス番号 *n* にのみ標準入力を与える
none	全てのプロセスについて標準入力を与えない

環境変数 MP_STDINMODE が設定されていない場合には、all が設定されているものとみなされます。

使用例

・全てのプロセスに標準入力を与える場合

```
#@$-q parallel
#@$-N 2
setenv MP_STDINMODE all
mpirun ./a.out < input.txt
```

・プロセス 0 番にのみ標準入力を与える場合

```
#@$-q parallel
#@$-N 2
setenv MP_STDINMODE 0
mpirun ./a.out < input.txt
```

2. 環境変数で与える方法

まず Fortran での方法を説明します。Fortran では環境変数 FTxxFyyy (xx は装置番号、yyy は FORTRAN 順序番号)にファイル名を設定します。

```
% setenv FT10F001 input.txt
% ./a.out
```

(Fortran の場合)

```
      do i=1,MAX_DATA_SIZE
        read(10,*,iostat=io) a(i)
        if(io .lt. 0) exit
      end do
```

上記の例では、装置番号 10 番に input.txt という名前のファイルを接続しています。あとは単純に「read(10,*,iostat=io) a(i)」で読み込むだけです。

環境変数 FTxxFyyy の意味

xx は装置番号で 0～2,147,483,647 の 1～10 桁の値を指定します。10 桁以下であれば数値の前に 0 がいくつあっても構いません(例：FT0000000001F001)。

yyy は FORTRAN 順序番号で、同一の装置番号を使用して異なるファイルを順次処理する場合に利用します。プログラムの実行開始時点では 001 です。ファイルの読み込みが完了すると、現在の順序番号に 1 を加えたものが次の読み込みの対象となります。

なお、環境変数の英字は大文字でなければなりません。

次に C 言語および Fortran でも使える方法を紹介します。ここでは、次のように独自に設定した環境変数 INPUT_FILENAME に入力ファイルを設定する場合を例に説明します。

```
% setenv INPUT_FILENAME input.txt
% ./a.out
```

C 言語の場合は getenv 関数で、Fortran では getenv サービスルーチンで環境変数に設定されている値を取り出すことができます。

(C 言語の場合)

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_DATA_SIZE 1000000
double a[MAX_DATA_SIZE];

int main(int argc,char *argv[])
{
  int i,ret,size;
  char str[80];
  FILE *fp;

  fp=fopen(getenv("INPUT_FILENAME"),"r");
  for(i=0;i<MAX_DATA_SIZE;i++){
    if(fgets(str,80,fp)==NULL) break;
    ret=sscanf(str,"%lf",&a[i]);
  }
  fclose(fp);
  size=i;
  ・・・・・・
```

(Fortran の場合)　　※SR11000/J1 の場合

```
   program main
   parameter(MAX_DATA_SIZE=1000000)
   integer i,io,size
   real*8 a(MAX_DATA_SIZE)
   character*80 filename

   call getenv('INPUT_FILENAME',filename)
   open(10,FILE=filename)
   do i=1,MAX_DATA_SIZE
      read(10,*,iostat=io) a(i)
      if(io .lt. 0) exit
   end do
   close(10)
   size=i-1
   ・・・・・・
```

※SR8000/MPP では call getenv の 1 行を下記に変更する。

```
call getenv('INPUT_FILENAME',14,filename,80)
```

14 は環境変数 INPUT_FILENAME の文字数、80 は文字型の変数 filename の長さを意味します。

この場合も、mpirun コマンドと合わせて利用する場合には注意が必要です。

SR8000/MPP および SR11000/J1 では setenv で設定した環境変数の値は全てのプロセスから getenv を用いて参照することができます。

```
#@$-q parallel          ※SR11000/J1 の場合
#@$-N 2
setenv INPUT_FILENAME input.txt
mpirun ./a.out
```

(プロセス 0)

```
getenv("INPUT_FILENAME")="input.txt"
```

(プロセス 1)

```
getenv("INPUT_FILENAME")="input.txt"
```

しかし、MPICH の環境のように、プロセス 0 番以外のプロセスからは setenv で設定した環境変数を参照できないものもあります。移植性を高くするには、プロセス番号 0 番でのみ getenv 関数を利用し、他のプロセスがその環境変数に設定された値を参照する必要がある場合には、MPI の送受信関数を用いて与えるようにするなどの工夫が必要です。

3. 実行時に引数として与える方法

次のように実行時にファイル名を引数として与える方法です。

```
% ./a.out input.txt
```

C 言語では main 関数の引数にコマンドライン引数が入ります。Fortran でコマンドライン引数を受け取るには getarg サービスルーチンを利用します。getarg サブルーチンは 1 番目の引数にコマンド引数の位置を与え、2 番目の引数に戻り値を格納するための文字型の変数名を与えます。

(C 言語の場合)

```
#include <stdio.h>
#define MAX_DATA_SIZE 1000000
double a[MAX_DATA_SIZE];

int main(int argc,char *argv[])
{
  int i,ret,size;
  char str[80];
  FILE *fp;

  fp=fopen(argv[1],"r");
  for(i=0;i<MAX_DATA_SIZE;i++){
    if(fgets(str,80,fp)==NULL) break;
    ret=sscanf(str,"%lf",&a[i]);
  }
  fclose(fp);
  size=i;
  ・・・・・・
```

(Fortran の場合)　※SR11000/J1 の場合

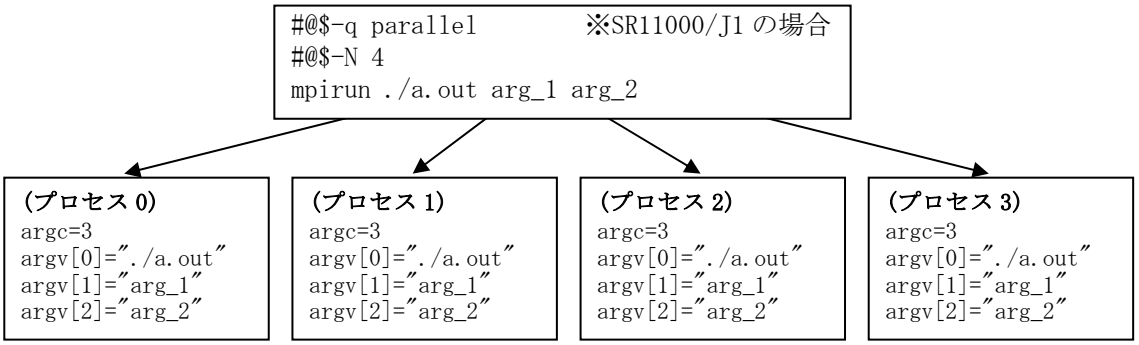
```
  program main
  parameter(MAX_DATA_SIZE=1000000)
  integer i,io,size
  real*8 a(MAX_DATA_SIZE)
  character*256 filename

  call getarg(1,filename)
  open(10,FILE=filename)
  do i=1,MAX_DATA_SIZE
    read(10,*,iostat=io) a(i)
    if(io .lt. 0) exit
  end do
  close(10)
  size=i-1
  ・・・・・・
```

※SR8000/MPP では call getarg の 1 行を下記に変更する。

```
call getarg(2,filename)
```

プログラム名自身も引数としてカウントするためコマンド引数の位置を+1 する必要があります。

mpirun コマンドと合わせて利用する場合、コマンドライン引数は全てのプロセスで同一の内容を受け取ることができます。



4. 並列実行における注意

複数のプロセスが同時に同一のファイルから読み込みを行うことは性能低下に繋がります。

1. 1 つのプロセスでファイルを読み込み、その後、他のプロセスに必要なデータを MPI の送受信関数を用いて与える。
2. 実行するプロセス個数分のファイルを作成しておき、各プロセスがそれぞれ異なるファイルから読み込むようにする。

といった方法で高速化してください。