64 ビットモード標準化における注意点

東京大学情報基盤センター
黒田 久泰

2007 年 4 月以降、本センターの SR11000 において標準のオブジェクトモードを 32 ビットモードから 64 ビットモードに変更します。具体的には、各ユーザーがログインしたときに環境変数 OBJECT_MODE が 64 に設定されるという仕組みで実現されます。64 ビットモードでコンパイルすることにより、2GB 以上のデータを扱うことができるようになります。また、8 バイト整数を利用しているプログラムでは速度向上が期待できるといった利点があります。そして、64 ビットモードを標準にすることで、コマンド実行時に毎回 64 ビットモードを指定するオプションを指定しなくてもよくなります。

32 ビットモードを標準にしたい場合には、「setenv OBJECT_MODE 32」のように環境変数を設定、または 「unsetenv OBJECT_MODE」で環境変数を解除してください。csh 用設定ファイル ~/.cshrc に登録しておくと、ログイン時に自動的に反映されます。

64 ビットモードが標準になる主要なコマンド、及び、32 ビットモードを明示的に指定するオプションは次の通りです。

コマンド名	コマンドの説明	32 ビットモード指定オプション
f77	最適化 FORTRAN77 コンパイラ	-32
mpif77	最適化 FORTRAN77 コンパイラ(MPI プログラム用)	
f90	最適化 FORTRAN90 コンパイラ	
mpif90	最適化 FORTRAN90 コンパイラ(MPI プログラム用)	
cc	最適化 C コンパイラ	
mpicc	最適化 C コンパイラ(MPI プログラム用)	
CC (sCC)	最適化標準 C++コンパイラ	
mpiCC	最適化標準 C++コンパイラ(MPI プログラム用)	
ld	リンケージエディタ (リンカ)	-b32
size	オブジェクトのセクションサイズ情報表示	-X32
nm	オブジェクトファイル内のシンボル表示	
ar	アーカイブの管理	
strip	オブジェクトファイルからシンボルを削除	
dump	オブジェクトファイルのダンプ	
ranlib	アーカイブ・ライブラリの変換	
lorder	オブジェクト・ライブラリの順序関係の検出	
as	アセンブラ	-a32

GNU コンパイラをご利用の方

gcc 、g++ コマンドでは、環境変数 OBJECT_MODE を参照しないためリンク時にコンパイルが失敗します。コンパイルオプションで **-maix64** により 64 ビットモードを指定してください。

64 ビットモード標準化における注意点

1. C 言語の場合

long 型とポインタ型のサイズが 4 バイトから 8 バイトになります。

「long」「unsigned long」「long int」「unsigned long int」により型宣言をしている場合には、変数のサイズが 8 バイトとなりますのでご注意ください。32 ビットモードと同じ 4 バイトにする場合には、「int」または「unsigned int」に変更が必要です。

「void *」「int *」「double *」などのポインタ型も 8 バイトになりますが、多くの場合、問題はありません。

また、「long long」「unsigned long long」は 8 バイト整数のままで変更はありません。

2. FORTRAN 言語の場合

暗黙の型宣言による頭文字が i から n までの変数、及び、「integer i」のように宣言した変数は 4 バイト整数のままになります。逆に、これらを 8 バイト整数としたい場合には、コンパイルオプションとして「-intexp=basic」を指定してください。また、「integer*8 i」のように明示的に宣言することも可能です。

64 ビットモードにおける速度向上について

8 バイト整数を利用していないプログラムの場合、実行速度にほとんど違いはありません。8 バイト整数を利用しているプログラムでは下記のように実行速度に差が出ます。sum.f は 8 バイト整数同士の加算、div.f は 8 バイト整数同士の割り算をそれぞれ 10 億回行うプログラムです。

sum.f （8 バイト整数同士の足し算）

```
program main
integer*8 i,sum
real*8 tt
call xclock(tt,7)
do i=1,1000000000
   sum=sum+i
end do
call xclock(tt,8)
write(*,*) 'sum=',sum
write(*,*) 'Elapsed time=',tt
```

div.f （8 バイト整数同士の割り算）

```
program main
integer*8 i,sum
real*8 tt
call xclock(tt,7)
do i=1,1000000000
   sum=sum+i/30
end do
call xclock(tt,8)
write(*,*) 'sum=',sum
write(*,*) 'Elapsed time=',tt
```

32 ビットモード(f90 -32 -Oss -noparallel) 及び 64 ビットモード(f90 -64 -Oss -noparallel) でコンパイルしたときの 1CPU による実行時間は下記の通りです。

プログラム名	32 ビットモード	64 ビットモード	速度向上
sum.f	2.820s	0.489s	5.8 倍
div.f	183.740s	2.605s	70.5 倍

8 バイト整数同士の加算では 5.8 倍、8 バイト整数同士の割り算では 70.5 倍という速度向上が得られることがわかります。