京/FX10 を利用する際の注意点

2014 年 11 月 28 日

理化学研究所仁科加速器研究センター 土井琢身

京コンピュータ、富士通 FX10 を利用する上で問題となる可能性のある事象についてまとめる。

行った計算： 格子 QCD による核力の計算
使用言語、コンパイラ： C/C++, Fujitsu C/C++ compiler

可変長配列のメモリ確保のスコープについて

[問題現象] 可変長配列のメモリ確保/解放のタイミングがプログラマの期待と異なる振る舞いをするため、
スタック領域を食い潰してプログラムがクラッシュする。

[発生環境]
京/FX10 における Fujitsu C/C++ compiler
（尚、同様の問題は Hitachi SR16000 上で IBM XL C/C++ compiler for AIX でも発生している。詳しくは当該レポート参照。）

[具体例] 次のような、可変長配列をスタックに取るコードを考える。

```
------------
void func(int n)
{
    for(int i=0; i<1000; i++) { // scope-A
       int array[n];
       // hoge
    }
}
int main()
{
    for(int i=0; i<100; i++) {
       func(1);
    }
```

```
    // fuga
}
```

プログラマの期待は、可変長配列 array[] は、そのスコープ(i.e., scope-A) に対応してスタック領域に確保・解放が行われるというものであろう。実際、GCC on x86 ではそのように動作するようである。この場合、サイズ n があまり大きくない限りはスタックを食い潰すことはない。

しかし、compiler/OS によっては、以下のように、スタックメモリ確保・解放のタイミングが異なるため、問題が起こり得る。

サポートセンターとのやり取りにより、Fujitsu C/C++ compiler は「可変長配列は、宣言された位置でスタックポインタを操作してスタックに領域を獲得します。そして、関数が終了した時点で領域を解放します。」という仕様であることが判明した。従って、上記コードでは、プログラマの期待よりも 1000 倍スタックを異常消費してしまうことになる。

なお、固定長配列では、上記のような問題は無い。

[対策]
たとえサイズが小さくても、可変長配列を自動変数としてスタックに取るのはやめ、malloc/free, new/delete 等でヒープ領域に明示的に確保・解放する。あるいは、可能であれば、自動変数のままでもループの外側で定義すれば問題は軽減される。なお、C++ では(未検証ではあるが) vector コンテナ等を使っても解決するはずである。

Fujitsu compiler について、仕様改善の希望を伝えた(2013/06)ところ、対応を検討するとの回答を頂いた。直近の情報（2014/07）では、2015/04 公開予定の compiler で対応予定とのことである。

以上