ポインター・エイリアシングとベクトル化

この記事は、インテル® デベロッパー・ゾーンに掲載されている「Pointer Aliasing and Vectorization」の日本語参考訳です。

はじめに

C++ のポインター・エイリアシングは、ベクトル化やその他の最適化の妨げとなるため、パフォーマンスの低下につながります。参照されるメモリー位置が一意であることをコンパイラーに知らせると、ベクトル化を含む多くの最適化を適用できます。

この記事では、ポインター・エイリアシングの規則と違反の解決方法を説明します。

- エイリアシング規則とは?
- エイリアシング規則違反とは?
- エイリアシング規則違反を解決するには?
- コンパイラー・サポートを有効にするには?
- パフォーマンス向上の可能性は?
- まとめ

エイリアシング規則とは?

エイリアシングに関する規則は、C++ 標準 (英語) に記載されています。

1. オブジェクトの保存された値は、次の型の lvalue (ロケーター値) によってアクセスできます。
    - オブジェクトの有効な型と互換性がある型
    - 互換性がある型の修飾された (const、volatile、…) 符号付き/符号なしバージョン
    - 上記の型をメンバーとして含む aggregate 型または union 型
2. 異なる型のポインター (例えば int* と float*) は同じオブジェクトをポイントできません。次に例を示します。

```
int a = 20;
int* pa = &a;
unsigned int* pua = (unsigned int* )&a;  // pua は pa のエイリアス
```

3. 構造体のフィールドへのアクセスは、同じ構造体の別のフィールドへのアクセスとエイリアスできません。
4. 例外があります: char*
   この例では、2 つのメモリーアクセスのみエイリアスできます。

```
typedef struct { short field1; } S1;
typedef struct { double field1; double field2; } S2;
void foo( int *p1, float *p2, S1 *p3, S1 *p4, S2 *p5)
{
  *p1 = 10;           // エイリアス不可
  *p2 = 20.0;         // エイリアス不可
```

```
  p3->field1 = 30;     // エイリアス可
  p4->field1 = 40;     // エイリアス可
  p5->field1 = 50.0;   // エイリアス不可
  p5->field2 = 60.0;   // エイリアス不可
}
```

エイリアシング規則違反とは?

「型パンニング」は、2 つの異なるデータ型が同じ表現を持つという仮定に基づいたエイリアシング規則違反です。例えば、ポインター型と整数型や、数学ライブラリーでよく見られる浮動小数点型と整数型があります。

これは良いプログラミング手法ではありません。以下は、SPEC* CPU2006 ベンチマーク 403.gcc の varasm.c:3723 にある不適切なコード例です。

```
int *strp; struct rtx_const value;
 strp = (int *) &value;
 while (--len >= 0)
  if (*p++ != *strp++)
```

エイリアシング規則違反を解決するには?

本当に異なる型で特定のデータ・オブジェクトにアクセスする必要があるのでしょうか? しないほうが良いですが、どうしても必要な場合は、union または char* を使用します。例えば、前出の 403.gcc の例で union を使用します。

```
union { int strp[]; struct rtx_const value; } foo;
int* pv = foo.strp;
...
while (--len >= 0)
  if (*p++ != *pv++)  // 構造体の 2 つのフィールドと見なされる (上記の規則 3)
```

より包括的な解決策は、構造体の対応するフィールドを比較することです。

コンパイラー・サポートを有効にするには?

-ansi-alias コンパイラー・オプションを使用して、C++ プログラムが C++ 標準に準拠していることをインテル® C++ コンパイラーに知らせます。これは、-O2 でコンパイルする場合のデフォルトです。このオプションを有効にすると、コンパイラーはベクトル化を含むより積極的な最適化を適用します。

この規則に従っていないコードをこのオプションを指定してコンパイルすると、コンパイラーは正しくないコードを生成する可能性があることに注意してください。

インテル® C++ コンパイラーの場合、ansi-alias のデフォルトは OS によって異なります。デフォルトは開発者が簡単に変更できます。

- Windows* では、最適化で ANSI エイリアシング規則が**デフォルトで無効**になります。

  /Qansi-alias は、規則に準拠していることをコンパイラーに知らせます。
  /Qansi-alias- は、規則に準拠していないことをコンパイラーに知らせます。

- Linux* および macOS* では、最適化で ANSI エイリアシング規則が**デフォルトで有効**になります。

  -ansi-alias は、規則に準拠していることをコンパイラーに知らせます。
  -no-ansi-alias は、規則に準拠していないことをコンパイラーに知らせます。

-ansi-alias-check は、ansi-alias チェッカーを有効にする便利なコンパイラー・オプションです。ansi-alias チェッカーは、潜在的な ANSI エイリアシング規則違反についてソースコードを確認し、潜在的な違反と識別された文のコードに関連する安全ではない最適化を無効します。このオプションを使用して、潜在的な違反を識別できます。ただし、ansi-alias checker はすべての違反を検出できるわけではありません。このオプションは、デフォルトで無効になります。

パフォーマンス向上の可能性は?

パフォーマンス向上の可能性は、アプリケーションとコードのコンパイルに使用される特定のコンパイラー・オプションによって異なります。インテル® コンパイラーでは、次の場合、アプリケーションが ANSI エイリアシング規則に準拠しているかどうか示すことが重要です。

- **-ipo** が使用されていない
- プログラム全体が利用できない
- アプリケーションに多くのポインター参照が含まれている

まとめ

- **-ansi-alias** を使用してください!
- 違反が見つかった場合は、char* と union を使用して解決します。
- 本当に必要な場合を除き、異なる型で同じデータ・オブジェクトにアクセスしないようにします。
- パフォーマンスの向上を達成してください!

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。