OpenRTM-aist (C++) - バグ #2507

ARM<->i386間で通信するとdouble型のデータが化ける

2012/08/10 11:57 - n-ando

ステータス:	終了	**開始日:**	2012/08/10
優先度:	通常	**期日:**	
担当者:		**進捗率:**	100%
カテゴリ:		**予定工数:**	0.00時間
対象バージョン:			

説明

ARM<->i386間で通信するとdouble型のデータが化ける

ARM側 SeqOutComp

```
-:  131.345  131.12  131  131  131[  ]

0:  1301.35  1301.12  1301  1301  31[  ]
1:  1302.35  1302.12  1302  1302  32[  ]
2:  1303.35  1303.12  1303  1303  33[!]
3:  1304.35  1304.12  1304  1304  34["]
4:  1305.35  1305.12  1305  1305  35[#]
5:  1306.35  1306.12  1306  1306  36[$]
6:  1307.35  1307.12  1307  1307  37[%]
7:  1308.35  1308.12  1308  1308  38[&]
8:  1309.35  1309.12  1309  1309  39[']
9:  1310.35  1310.12  1310  1310  40[(]
```

i386側 SeqInComp

```
-------------------------------------------------------------------
                Double          Float          Long          Short          Octet
-------------------------------------------------------------------
                                   Basic  type
-------------------------------------------------------------------
        9.45875e-013        131.12          131          131          131[  ]
-------------------------------------------------------------------
                                   Sequence  type
-------------------------------------------------------------------
   0:     1.99916e+037      1301.12      1301      1301      31[  ]
   1:     1.99916e+037      1302.12      1302      1302      32[  ]
   2:     1.99916e+037      1303.12      1303      1303      33[!]
   3:     1.99916e+037      1304.12      1304      1304      34["]
   4:     1.99916e+037      1305.12      1305      1305      35[#]
   5:     1.99916e+037      1306.12      1306      1306      36[$]
   6:     1.99916e+037      1307.12      1307      1307      37[%]
   7:     1.99916e+037      1308.12      1308      1308      38[&]
   8:     1.99916e+037      1309.12      1309      1309      39[']
   9:     1.99916e+037      1310.12      1310      1310      40[(]
```

逆向きについても調査。やはりdoubleがおかしい

i386側 SeqOutComp

```
-:  24.345  24.12  24  24  24[  ]

0:  231.345  231.12  231  231  231[  ]
1:  232.345  232.12  232  232  232[  ]
2:  233.345  233.12  233  233  233[  ]
3:  234.345  234.12  234  234  234[  ]
4:  235.345  235.12  235  235  235[  ]
5:  236.345  236.12  236  236  236[  ]
```

```
6:  237.345  237.12  237  237  237[  ]
7:  238.345  238.12  238  238  238[  ]
8:  239.345  239.12  239  239  239[  ]
9:  240.345  240.12  240  240  240[  ]
```

ARM側 SeqInComp

```
---------------------------------------------------------------------
              Double             Float         Long         Short        Octet
---------------------------------------------------------------------
                                  Basic  type
---------------------------------------------------------------------
      -8.67922e+209          24.12          24            24            24[  ]
---------------------------------------------------------------------
                                Sequence  type
---------------------------------------------------------------------
  0:      9.45875e-13          231.12          231          231          231[  ]
  1:      9.45875e-13          232.12          232          232          232[  ]
  2:      9.45875e-13          233.12          233          233          233[  ]
  3:      9.45875e-13          234.12          234          234          234[  ]
  4:      9.45875e-13          235.12          235          235          235[  ]
  5:      9.45875e-13          236.12          236          236          236[  ]
  6:      9.45875e-13          237.12          237          237          237[  ]
  7:      9.45875e-13          238.12          238          238          238[  ]
  8:      9.45875e-13          239.12          239          239          239[  ]
  9:      9.45875e-13          240.12          240          240          240[  ]
```

- なお、ARM同士の通信ではデータ化けは発生しない。
- ARM (C++) -> i386 (Python) の通信でもdoubleが化けているため、ARM版のomniORBが原因の可能性大。

履歴

#1 - 2012/08/10 12:08 - n-ando

- omniORB-4.1.2を新たにコンパイルしなおして、既存のライブラリと入れ替え->×

omniORB-4.1.2のコンパイル

```
./configure  --prefix=$PREFIX  \
             CXX=/usr/bin/arm-linux-gnueabi-g++  \
             CC=/usr/bin/arm-linux-gnueabi-gcc  \
             CXXFLAGS=-s  \
             CFLAGS=-s  \
             --build=i386-linux-gnu  \
             --host=arm-linux-gnueabi  \
             --target=arm-linux-gnu  \
             --with-omniorb=/usr/arm-linux-gnueabi  \
             --with-includes=-I/usr/arm-linux-gnueabi/include
```

さらに、

```
make  clean
make  CC=gcc  -C  src/tool/omniidl/cxx/cccp
make  CXX=g++  -C  src/tool/omniidl/cxx
make  CC=gcc-3.4  -C  src/tool/omkdepend
make
```

とした。

#2 - 2012/08/10 15:33 - n-ando

CORBA::Double型のマーシャリング・アンマーシャリングは omniORB4/cdrStream.h にてinlineで定義されている。

```
  friend inline void operator>>= (_CORBA_Double a, cdrStream& s) {
    struct LongArray2 { _CORBA_ULong l[2]; };
    convertFromFloat(_CORBA_Double, LongArray2);
    if (s.pd_marshal_byte_swap) {
      LongArray2 m;
      m.l[0] = Swap32(l.l[1]);
      m.l[1] = Swap32(l.l[0]);
```

```
                l   =   m;
        }
#ifdef   OMNI_MIXED_ENDIAN_DOUBLE
        {
                _CORBA_ULong   v   =   l.l[0];
                l.l[0]   =   l.l[1];
                l.l[1]   =   v;
        }
#endif
        CdrMarshal(s,LongArray2,omni::ALIGN_8,l);
    }
```

ここで、OMNI_MIXED_ENDIAN_DOUBLEが定義されていると挙動が変わる。
OMNI_MIXED_ENDIAN_DOUBLE はCORBA_sysdep.hにて定義されている。

```
#if   defined(__arm__)
#      if   defined(__armv5teb__)
#           define   NO_OMNI_MIXED_ENDIAN_DOUBLE
#      else
#           define   OMNI_MIXED_ENDIAN_DOUBLE
#      endif
#endif
```

ここで、
http://marc.info/?l=omniorb-list&m=130929162028726&w=4
のような話があり、__VFP_FP__ が宣言されている場合 NO_OMNI_MIXED_ENDIAN_DOUBLE が定義されるべきであるらしい。

そこで、

```
#if   defined(__arm__)
#      if   defined(__armv5teb__)   ||   defined(__VFP_FP__)
#           define   NO_OMNI_MIXED_ENDIAN_DOUBLE
#      else
#           define   OMNI_MIXED_ENDIAN_DOUBLE
#      endif
#endif
```

のようにする。

Doubleのoperator>>=などはインライン関数なため、OpenRTMのコンパイル時、RTCのコンパイル時に効いてくる。
したがって、omniORBをコンパイルしなおしてもあまり意味がなかったようだ。

#3 - 2012/08/10 16:50 - n-ando

修正手順

修正手順は以下の通り

1. CORBA_sysdep.h の修正
2. OpenRTMの再コンパイル
3. コンポーネントの再コンパイル

CORBA_sysdep.hの修正

/usr/arm-linux-*/include/omniORB4/CORBA_sysdep.h 内の #if defined(*arm*) 付近を以下のように修正する。

```
#if   defined(__arm__)
#      if   defined(__armv5teb__)   ||   defined(__VFP_FP__)
#           define   NO_OMNI_MIXED_ENDIAN_DOUBLE
#      else
#           define   OMNI_MIXED_ENDIAN_DOUBLE
#      endif
#endif
```

#if defined(*armv5teb*) のみになっている部分を # if defined(*armv5teb*) ¦¦ defined(*VFP_FP*) のように修正。

OpenRTMの再コンパイル

OpenRTMを再度コンパイルする。すでに同一環境でconfigure済みなら、configureは必要ないが、コンパイルした環境がすでにない場合は、

```
        ./configure   --prefix=/usr/arm-linux-gnueabi   \
                      CXX=/usr/bin/arm-linux-gnueabi-g++   \
```

```
CC=/usr/bin/arm-linux-gnueabi-gcc  \
CXXFLAGS=-s  \
CFLAGS=-s  \
--build=i386-linux-gnu  \
--host=arm-linux-gnueabi  \
--target=arm-linux-gnu  \
--with-omniorb=/usr/arm-linux-gnueabi  \
--with-includes=-I/usr/arm-linux-gnueabi/include
```

arm-linux-gnueabi の部分は arm-linux-gnu 等の場合もあるので、環境に応じて読みかえること。

その後

```
$ make install
```

として、環境にOpenRTMをインストールしなおす。

コンポーネントの再コンパイル

コンポーネントがOpenRTMの標準データ型のみ使用している場合はコンポーネントの再コンパイルはおそらく不要。
独自IDLや独自データ型を使用している場合は、再度コンパイルする必要がある。

#4 - 2012/08/10 17:09 - n-ando

参考

ちなみに、この変更はomniORB-4.1.6以降には適用されている。4.1.6を使用してOpenRTMをコンパイルすれば問題ない。
したがって、これの変更が必要なのはomniORB-4.1.5以前である。

omniORBをコンパイルするためのスクリプトを添付する。

#5 - 2012/08/10 17:14 - n-ando

- ファイル omniorb_crosbuild.sh を追加

#6 - 2012/08/10 17:14 - n-ando

- ステータス を 新規 から 解決 に変更

- 進捗率 を 0 から 100 に変更

以上をもって解決とする。

#7 - 2013/04/15 17:45 - n-ando

- ステータス を 解決 から 終了 に変更

ファイル

omniorb_crosbuild.sh	2.11 KB	2012/08/10	n-ando