Implementing static rounding attributes using a global rounding mode

Jim Thomas – 10 Sep 2011

Here is a scheme an implementation with a global rounding mode can use to support static rounding attributes. This scheme can be used whether or not the user code is under FENV_ACCESS ON.

The scheme makes use of these implementation intrinsics:

```
int __swapround(int rnddir);    // equivalent to
                                // { int oldrnddir; oldrnddir = fegetround();
                                //   fesetround(rnddir); return oldrnddir; }

void __setround(int rnddir); // equivalent to fesetround(rnddir)
```

In the pseudo code below, an "affected operation" is one that honors the static rounding attribute. The operations 754 specifies to be affected by the rounding direction are affected operations. Affected operations include some standard C functions, e.g., sqrt, fma, printf, etc.

An "unaffected operation" is one that is not to be affected by the static rounding attribute. It has been proposed that user functions be unaffected operations. The functions in <fenv.h> are to be treated as unaffected operations. Note particularly how the scheme works with fegetround, fesetround, fegetenv, and other functions that access the global rounding mode.

The red code is to be inserted by the implementation.

```
{
#      pragma STDC FENV_ROUND static_rnddir
#      pragma STDC FENV_ACCESS ON
       int __rnddir;
       __rnddir = __swapround(static_rnddir);
       ...
       Affected operation
       ...
       __rnddir = __swapround(__rnddir);
       Unaffcted operation
       __rnddir = __swapround(__rnddir);
       ...
       __setround(__rnddir);
}
```

Any (non returning) exit from the block is to be preceded by
`__setround(__rnddir);`