TransSkel Programmer's Notes

9: Type Changes in TransSkel 3.06

Who to blame: Paul DuBois, dubois@primate.wisc.edu
Note creation date: 01/05/94
Note revision: 1.00
Last revision date: 01/05/94
TransSkel release: 3.06

This Note discusses changes to function and variable types made to TransSkel 3.06.

Earlier versions of TransSkel defined certain derived types which were used for portability purposes or to compensate for certain behaviors of the THINK C compiler. These types have been eliminated from TransSkel. This Note explains the rationale for the types and why they're no longer used. If your first experience with TransSkel is with release 3.06, you may still find it useful to read this Note, since older TPN's still contain references to types that are no longer used.

`Integer` and `LongInt` Are No More. Ditto *Compiler.h*

The very earliest releases of TransSkel used `int` for 2-byte integer variables. This is also how the header files for THINK C (then LightspeedC) were written. However, `int` is ambiguous, since the C standard treats `int` as merely "whatever is the natural word type". To eliminate this ambiguity, all 2-type integers in TransSkel were typed as `Integer` (the same name as the Pascal 2-byte integer type used in Inside Macintosh), and `Integer` was `typedef`'d to an integer type that the compiler treated as a 2-byte quantity. At the time, this was either `short` or `int` for LightspeedC. To avoid possible problems for 4-byte integers, all long variables were typed as `LongInt`, which also was `typedef`'d appropriately.

Eventually, the `typedef`'s for `Integer` and `LongInt` were migrated into a separate file *Compiler.h* that was included by all TransSkel source files. The advantage of this was that `Integer` and `LongInt` provided a consistent pair of type names which could be easily mapped to the proper underlying types for a given C compiler. To port TransSkel to a different compiler, you could just make sure the `typedef`'s were correct and all

integer variables would be the right size. The chief disadvantage was the necessity of `#include`'ing *Compiler.h* in anything TransSkel-related.

The preceding rationale for `Integer` and `LongInt` now longer seems to apply. It appears to be universally true that Macintosh C compilers use `short` and `long` for 2- and 4-byte integer types. That being so, TransSkel no longer uses `Integer` or `LongInt`, and the source files no longer include *Compiler.h*.

`SkelDlogFilter()` Type Change

Earlier releases of TransSkel declared `SkelDlogFilter()` like this:

```
SkelDlogFilterProcPtr
SkelDlogFilter (SkelDlogFilterProcPtr filter, Boolean doReturn);
```

where `SkelDlogFilterProcPtr` was `typedef`'d as a pointer to functions that looked like this:

```
pascal Boolean
Filter (DialogPtr dlog, EventRecord *evt, Integer *item)
{
}
```

This is the same kind of function to which the `ModalFilterProcPtr` type refers. However, early versions of the THINK C compiler either did not define `ModalFilterProcPtr` or else by default precompiled MacHeaders to treat function pointers as `void *`, losing prototype information. Thus, although `SkelDlogFilterProcPtr` and `ModalFilterProcPtr` were logically equivalent, for type-checking purposes they were not. Explicitly defining the `SkelDlogFilterProcPtr` type allowed better type-checking. The THINK C 6 default precompiled MacHeaders retains full prototype information, so `SkelDlogFilterProcPtr` is fully equivalent to `ModalFilterProcPtr` and has become vestigial. `SkelDlogFilter()` is now declared like this:

```
ModalFilterProcPtr
SkelDlogFilter (ModalFilterProcPtr filter, Boolean doReturn);
```