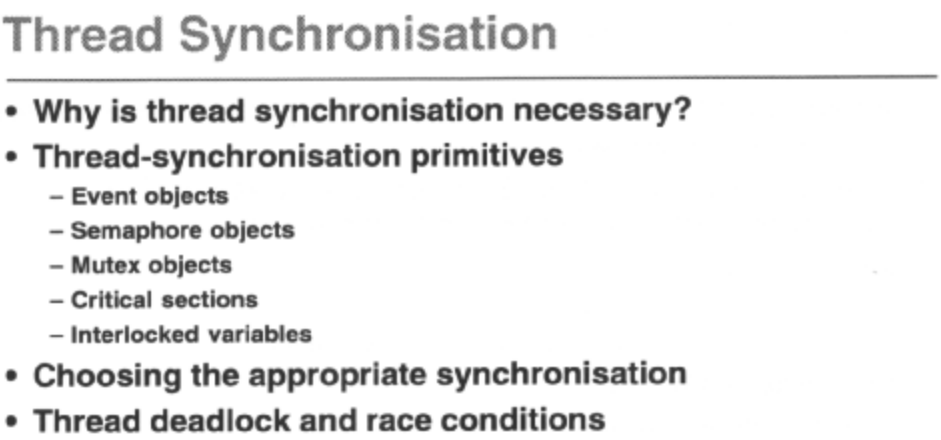

'Static linking' creates an executable file that is 100% complete; the linker copies the relevant code from all objects and libraries referenced by the program. Although this code can be physically shared at run time by multiple invocations of the same application, it cannot be shared by different applications, which use the same library routines. All Windows applications use a large number of Windows routines; this would be a waste of memory if the Windows libraries were statically linked.

The diagram above illustrates the creation of a simple but typical application, *PROGA.* Two source files, *PROGA.C* and *USEFUL.C* are compiled, and the two object code files *PROGA.OBJ* and *USEFUL. OR]* thus produced are statically linked.

PROGA.C refers to the function `Sqrt()`, which is defined in *USEFUL.C.* Provided *PROGA.C* declares this function to be an external function, the compiler will accept and compile the source code without any problems.

At link time, the external references are resolved, and the linker ensures that the reference to the `Sqrt()` function is resolved against the `Sqrt()`code, which is physically present in the executable *file.*