General Porting Guidelines

Porting 32-bit applications to 64-bit Microsoft Windows will be easier than it was porting 16-bit applications to 32-bit Windows. However, the move will go more smoothly with some careful planning. The following are some general guidelines.

Planning

- Determine the magnitude of the effort required for the port. Gauge how much work is involved by identifying the following items:
    - Problem 32-bit code. Compile your 32-bit code with the 64-bit compiler and examine the extent of the errors and warnings.
    - Shared components or dependencies. Determine which components in your application originate from other teams and whether those teams plan to develop 64-bit versions of their code.
    - Legacy or assembly code. 16-bit Windows-based applications do not run on 64-bit Windows and must be rewritten. While x86 assembly code runs in WOW64, you may want to rewrite this code to take advantage of the speed of the Intel Itanium architecture.
- Port the entire application, not just portions of it.
  Although it is possible to port pieces of an application or to limit code to 2G with /LARGEADDRESSAWARE:NO, this strategy trades short-term gain for long-term pain.

- Find substitutes for technologies that will not be ported.
  Some technologies, including DAO (Data Access Object) and the Jet Red database engine, will not be ported to 64-bit Windows.

- Treat your 64-bit version as a separate product release.
  Even though your 64-bit product may share the same code base as your 32-bit, it needs additional testing and may have other release considerations.

Development

- Start developing compliant code now.
  Developers can start writing compliant code by using the latest Windows header files and the new data types with no adverse effects on 32-bit product development. For more information, see Getting Ready for 64-bit Windows.

- Ensure that your code can be compiled for both 32- and 64-bit Windows.
  The new data model was designed to allow 32- and 64-bit applications to be built from a single code base with few modifications. The SQL Server and Windows development teams are developing 32- and 64-bit versions of their products from the same code base.

- Use the compiler's new optimization features for best performance.
  Code optimization for Intel Itanium processors is more important than it was for the x86. The compiler assumes many of the optimization functions previously handled by the microprocessor. You can maximize the performance of a 64-bit application by using two new optimization features of the compiler: *Profile Guided*

Optimization and *Whole Program Optimization*. Both features result in longer build times and require the early development of good test scenarios.

Profile Guided Optimization involves a two-step compile process. During the first compile, the code is instrumented to capture the execution behavior. This information is used during the second compile to guide all optimization features.

Whole Program Optimization analyzes the code in all application files, not just a single one. This approach increases performance in several ways, including better inlining, as well as improved side-effect analysis and custom calling conventions.

Testing

- Determine whether you'll test 64- or 32-bit code running in WOW64.
  Some applications include both native 64-bit code and 32-bit code running in WOW64. Investigate this closely while developing a test plan, and decide whether your test tools should be 64-bit, 32-bit, or a combination. You will often need to test both the 64- and 32-bit versions of your application on 64-bit Windows.

- Test frequently used 32-bit components.
  First, recompile your code to 64-bit and test. Second, fix problems, recompile in 32-bits, and then test. Third, recompile to 64-bit and test.

- Test COM and RPC components.
  Make sure that both 32- and 64-bit COM and RPC components communicate correctly. You may also have to test communications with 16-bit components over a network.

- Test your 32-bit version on 64-bit Windows.
  Customers can continue to use 32-bit applications on 64-bit Windows where performance and memory issues are not major considerations.

- Test different memory configurations.
  Adding large amounts of memory on the server sometimes exposes previously unnoticed problems in either the application or the operating system.

© 2017 Microsoft