ne Debug-Informationen beseitigt werden Man spricht von einem optimierenden Linker.

Die Größe der Ergebnisdatei wird dadurch weiter reduziert und beschleunigt dadurch den Ladeprozeß in den Debugger. Für eine typische Applikation führten die Optimierungen in Compiler und Linker zur Verkürzung der Zyklus-Zeit für Editieren, Compilieren und Debuggen von 40%.

Weniger Platzbedarf

Mehrfach vorhandene Code- und Datenbereiche müssen sowohl durch den Compiler als auch durch den Linker vermieden werden. Der Linker ist das richtige Werkzeug um über Dateigrenzen hinaus zu optimieren.

Neue Optimierungsmethoden generieren Tabellen für virtuelle Funktionen nur in jener Datei, wo diese Funktion definiert wurde. Das Layout von Virtual Function Tables wurde gegenüber der Cfront-Implementierung ebenfalls verbessert.

Das neue Layout vermeidet das oben erwähnte Problem jener Objekte, in denen mehrfach die gleichen Funktionen virtueller Basisklassen vorkommen. Bild 5 zeigt diese Verbesserung. Dies kann zu erheblichen Einsparungen beim RAM-Verbrauch führen, wenn sehr viele Objekte über mehrfache Vererbung und virtuellen Basisklassen erzeugt wurden.

Der Linker optimiert vielfache Kopien von Template- und In-Line-Funktionen. Außerdem werden mehrfach vorhandene Tabellen für virtuelle Funktionen, erzeugt durch In-Line-, Template- oder virtuelle Funktionen, beseitigt. Viele Optimierungen, die früher im Compiler vorgenommen wurden und zu erheblichen Laufzeiten geführt haben, wurden in den Linker verlegt. Als Beispiel: Cfront benötigte sechs Stufen, nur um redundante Template-Funktionen zu vermeiden:

1- Übersetzen von C++ nach C (ausgenommen Templates)
2- Kompilieren von C
3- Prelinker stellt fest wo Templates verwendet werden
4- Übersetzen der Templates von C++ nach C
5- Kompilieren der Templates
6- Linken der Applikation

Cfront erzeugt beim Übersetzten der Quelldatei noch keine Instanzen für die Template-Funktionen. Statt dessen wird ein spezieller Prelinker eingesetzt der feststellt, welche Dateien ein bestimmtes Template verwenden. Cfront ruft anschließend den Compiler auf, um eine einzige Kopie zu erzeugen, die dann von allen Dateien verwendet wird. Wenn alle Template-Funktionen erzeugt wurden, erfolgt der endgültige Linker-Durchlauf.

Die Optimierungsalgorithmen im neuen Linker sorgen dafür, daß redundante Code- und Datenbereiche beseitigt und zusammengelegt werden. Diese Optimierung betrifft auch die Beseitigung überflüssiger Debug-Informationen. Der Compiler kennzeichnet die gewollten Mehrfach-Kopien, wie z.B. statische Funktionen, die der Linker dann im Anschluß nicht wegoptimiert.

Der Optimierungsansatz im Linker verlängert die Durchlaufzeiten kaum. Ein Linker greift intensiv auf Dateien zu und in diesem Fall werden überflüssige Code- und Datenbereiche beseitigt und nicht neuerlich in Dateien abgelegt. Ein Compiler-basierender Ansatz würde erheblichen Aufwand bedeuten, die Abhängigkeiten zwischen den Dateien zu erkennen und auf dieser Basis zu optimieren. Die Laufzeiten wären erheblich länger.

Zusammenfassung

Der steigende Einsatz von C++, speziell auch in Embedded Applikationen, erfordert neue Compilertechnologien um kompakte Code- und Datengrößen zu erreichen, den Speicherbedarf zu verringern und den Entwicklungszyklus zu beschleunigen. Die Optimierungsmethoden herkömmlicher C-Compiler versagen beim Einsatz von C++. Die Optimierung muß speziell an die Anforderungen der neuen objektorientierten Eigenschaften angepaßt sein und den unnötigen Overhead, der sehr oft mit der Sprache C++ in Zusammenhang gebracht wird, verhindern. Nachdem viele C++ Applikationen auch besonders umfangreich sind, kann durch eine neue Compiler-Technologie die Zykluszeit beim Entwickeln beschleunigt werden. Die neue C++ Technologie von Microtec Research reduziert die erforderlichen Zeiten beim Linken und Laden durch vermeiden von redundanten Debug-Informationen und beim Compilieren durch Weglassen des Zwischenschritts der Übersetzung von C++ Programmen nach C. □

Der HTML-Ratgeber

Heinrich.E.G.Bonin, Hanser-Verlag
1996, Preis: ATS 277,-

Dieter Reiermann

Liebenswürdigerweise hat mich der Hanser-Verlag eingeladen, den HTML-Ratgeber zu prüfen. Zur Gestaltung meiner Homepage kam mir dieses Buch gerade recht. Mein erster Eindruck - als ein der Materie noch weitestgehend Unkundiger - „Ich werde mir mehr Zeit nehmen müssen". Abends vor dem Schlafengehen wollte ich im HTML-Ratgeber nicht schmökern. Also kein Lesebuch.

Bei meiner späteren Arbeit habe ich das Buch allerdings noch sehr gut brauchen können. Die HTML-Befehle sind kapitelweise funktionell gegliedert und umfassend beschrieben. Es werden einige interessante Beispiele gezeigt.

Zum Inhalt

Einleitung

Beschreibt hauptsächlich die Intentionen des Autors (Datum: Winter 95/96). Das Buch behandelt HTML 2.0, wieweit HTML 3.2 berücksichtigt wurde, konnte ich nicht herausfinden.

Software

Nur über URL, Diskette wird auf Wunsch zugesendet.

1. Konstrukte

WWW (Hallo World, Syntax der Konstrukte, HTTP), Semantik der Konstrukte, Uniform Resource Locator

2. Konstruktionen

Dynamisches Dokument, Dialog mit Formular, Präsentationen mit Viewer Applikationen, WWW-Server, Realisierung von Sicherheit

3. Konstruktionsempfehlungen

Organisation des Datenmaterials, Gestaltung des einzelnen Dokumentes, Systematisches Testen, Wartung und Betrieb

Anhang

Quellen, Abkürzungen, Index, Abbildungsverzeichnis, Tabellenverzeichnis

Autor

Prof.H.Bonin, Fachhochschule Nordostniedersachsen, Fachbereich Wirtschaft, Lüneburg, `http://cl3.fbw.fh-lueneburg.de:6667/`

Umschlaggestaltung

Das wohlbekannte Layout des Hanser-Verlag, aber was hat HTML mit Heidelbeeren zu tun?

Druck

etwa 10 Punkt, starke Gliederung durch Zeichensatz - vielleicht wirkt der Text dadurch etwas unruhig. Zahlreiche Bildschirmshots von HTML-Dokumenten.

Zusammenfassend

Nicht für „bloody beginners", als Arbeitsbehelf sehr gut geeignet. Aufbau: Ähnlich einer Sprachreferenz.