—- Allgemeine Informationen—-

Projektziele

Michael Felke Martina Mostert

Revision 2009-03-11

In Zusammenarbeit mit

Marc Horst, Christian Janßen

Copyright © by owl's Team. *Alle Rechte vorbehalten.*

Weiterhin beigetragen haben:

Oliver Höfken, Nicholas Vollmer

Copyright © by owl's Team. *Alle Rechte vorbehalten.*

Inhaltsverzeichnis

1 Zielbeschreibung

Ziel des Projekts ist die Schaffung eines Systems aus Modellen, Spezifikationen, Prüfverfahren und Werkzeugen, welches es ermöglicht, Programme ohne erneutes Compilieren über Hardwaregrenzen hinweg auf jedem Rechner zu benutzen, sofern:

- dieser funktional dazu in der Lage ist
- genügend Ressourcen zur Verfügung stehen
- die Ablaufumgebung[1] über die benötigten Programmierschnittstellen verfügt.

Es soll insbesondere ermöglichen:

- Treiber auszuliefern, welche auf jedem Rechner verwendet werden können
- hierfür entwickelte Betriebssysteme auf unterschiedlicher Hardware zu booten.

Dies soll mittels eines generalisierten Maschinencodes in Kombination mit einer Normdarstellung für die binäre Speicherung persistenter Daten erreicht werden. Ein abstrahiertes, dynamisches Computing Modell ermöglicht den so erstellten Maschinenprogrammen sich in ihre jeweilige Ablaufumgebung einzupassen. Auch der Aufruf von Programmierschnittstellen soll vereinheitlicht werden, so dass ein Programm mit verschiedenen Betriebssystemen verwendet werden kann, wenn diese die gleichen, vom Programm benötigten, Schnittstellen zur Verfügung stellen. Ein einheitliches Paketformat für Maschinenprogramme ergänzt dies.

Dabei soll das zu schaffende System und insbesondere der generalisierte Maschinencode so wenig statische Grenzen wie möglich enthalten.

2 Liste der Ziele / Pflichtenheft

1. Das System soll die Erstellung von Programmen unabhängig von der Zielhardware ermöglichen.
2. Programme sollen unter optimaler Ausnutzung des jeweiligen Prozessorbefehlssatzes ausgeführt werden können.
3. Trotzdem soll es möglich sein Programme zu schreiben, welche die vorhandene Hardware voll ausnutzen.
4. Der zu schaffende generalisierte Maschinencode soll
   - interpretierbar sein
   - leicht auf die Zielhardware übersetzbar sein
   - performant von JIT-Engines verwendet werden können

[1] im weiteren Ausführungssphäre genannt

Copyright © by owl's Team. Alle Rechte vorbehalten.

- es ermöglichen, nur die gerade benötigten Teile zu übersetzen (compiling on demand)
- mittels jeder Compilersprache erstellt werden können.

5. Ausliefern von kompilierten bzw. assemblierten Programmen in einem einheitlichen Format für alle Betriebssysteme, welches es erlaubt Programme
   - aus dem Paket mittels eines Systemprogramms in angepasster Form zu installieren
   - direkt mittels eines Loaders aus dem Paket heraus zu verwenden
   - stückweise von einem Server zu holen
   - in unterschiedlichen Varianten (z.B. Sprachvarianten deutsch/englisch) auszuliefern, von denen dann die passende verwendet wird
   - an ihren Schnittstellen immer mit der passenden Version zu verbinden
   - für verschiedene Betriebssysteme und Hardware auf einem Server bereitzustellen
   - automatisch in der aktuellsten Reversion zu verwenden
   - mit weiteren Informationen wie Quelltext, Hilfetexte etc. abzulegen.
6. Der entstehende Code soll so speichersparend wie möglich ist.
7. Ein Programm soll nur soviel Ressourcen belegen wie es wirklich benötigt
8. Die erzeugten Programme müssen die Möglichkeit der Überprüfung der von ihnen statisch zu belegenden Ressourcen vor der Ausführung bieten.
9. Im Rahmen eines Sicherungssystems sollen
   - auf einem Rechner unterschiedliche Ausführungssphären zur Verfügung gestellt werden
   - die einem Programm zur Verfügung stehenden Maschinenoperationen beschränkt werden
   - die Veränderung von anderen Programmen verhindert werden
   - Speicherschutz berücksichtigt werden.
10. Der Programmcode soll in Bezug auf die Speicherform (z.B. ROM, Binär-Datei, Textdatei, Datenbank, ISAM-Datei so neutral wie möglich sein, um in fast allen heutigen und zukünftigen Formen speicherbar zu sein.
11. Der Lese-Zugriff auf den Programmcode bzw. Teile des Programmcodes soll über Prozeduren geleitet werden können, die im Programm selber stehen, um Dekodierungen und Verifizierungen durchführen zu können.
12. Bei der Verarbeitung von Daten soll das System ermöglichen Daten
    - in einem rechnerunabhängigen Format (auch binär) zu speichern

Alle Rechte vorbehalten. Copyright © by owl's Team.

- blockorientiert und strukturiert zu speichern
- sowohl High- als auch Low-Byte-First zu lesen und zu schreiben.

13. Alle atomaren Datenarten sollen verarbeitet werden können.
14. Unterstützung von Echtzeit-Verarbeitung.
15. Unterstützung von Multitasking/-thread Programmierung.

3 Minimalrechnersystem

Als kleinster gemeinsamer Nenner wird ein Minimalrechnersystem definiert, ab dem die o.g. Ziele und Anforderungen realisiert werden können.

Spezifikation:

- 8-Bit, 6502 Befehlssatz
- 512 Byte RAM
- optional Ports (seriell, parallel, etc.)
- ROM 32 KiB [2]

[2] Kibibyte=1024 Byte, gemäss IEC 60027-2, IEEE 1541

Copyright © by owl's Team.

Alle Rechte vorbehalten.

Index