Aufbau eines modernen Betriebssystems (Windows NT 5.0)

Moritz Mühlenthaler

14.6.2004

Proseminar KVBK

Gliederung

1. Das Designproblem
   a) Überblick
   b) Design Goals
   c) Möglichkeiten der Strukturierung

2. Umsetzung der Design Goals in NT 5.0
   a) Architektur-Überblick
   b) Aufbau der User Mode Komponenten
   c) Aufbau des Kernels

3. Zusammenfassung

Gliederung

1. Das Designproblem
   a) Überblick
   b) Design Goals
   c) Möglichkeiten der Strukturierung

2. Umsetzung der Design Goals in NT 5.0
   a) Architektur-Überblick
   b) Aufbau der User Mode Komponenten
   c) Aufbau des Kernels

3. Zusammenfassung

Probleme (1)

- Sehr hohe Lebensdauer
  - UNIX ist etwa 25 Jahre alt
  - Windows ist etwa 10 Jahre alt
- Platformunabhängigkeit trotz
  - Unterschiedlicher Hardware-Umgebungen
  - Unabhängig voneinander designter HW-Komponenten
- Komplexes Resourcen Management
  - Benutzer wollen ihre Daten vor Zugriffen anderer schützen
  - I/O Geräte müssen sich HW Resourcen teilen (Interrupts, DMA Kanäle, ...)
  - Komponenten hängen stark zusammen

Probleme (2)

- Abwärtskompatibilität
  - ➔ Einschränkungen / Eigenarten der früheren Version(en) müssen berücksichtigt werden
- Betriebssysteme sind sehr grosse Programme
  - ➔ UNIX i.d.R > 1 Mio Zeilen Code
  - ➔ Windows NT ~ 29 Mio Zeilen Code
  - ➔ Unmöglich zu überblicken!

➢ Betriebssysteme sollten trotzdem einfach und einheitlich programmierbar sein...

Design Goals

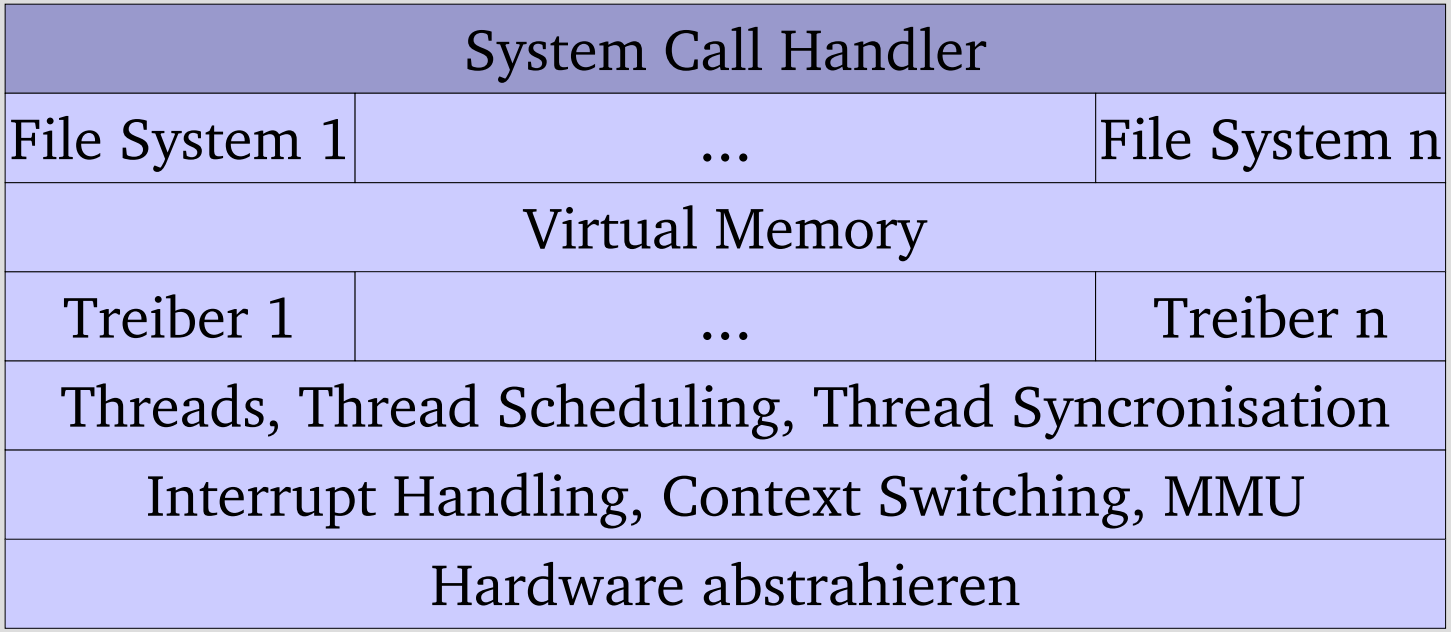
Welche Ziele sollte man also beim Entwurf im Auge behalten?

- Erweiterbarkeit
- Portierbarkeit
- Zuverlässigkeit
- Kompatibilität
- Sicherheit
- Effizienz

Möglichkeiten der Strukturierung (1)

1. Schichten-Modell

- Verringerung der Komplexität durch Einteilung in Schichten
- Weit verbreitet (siehe UNIX, Windows)



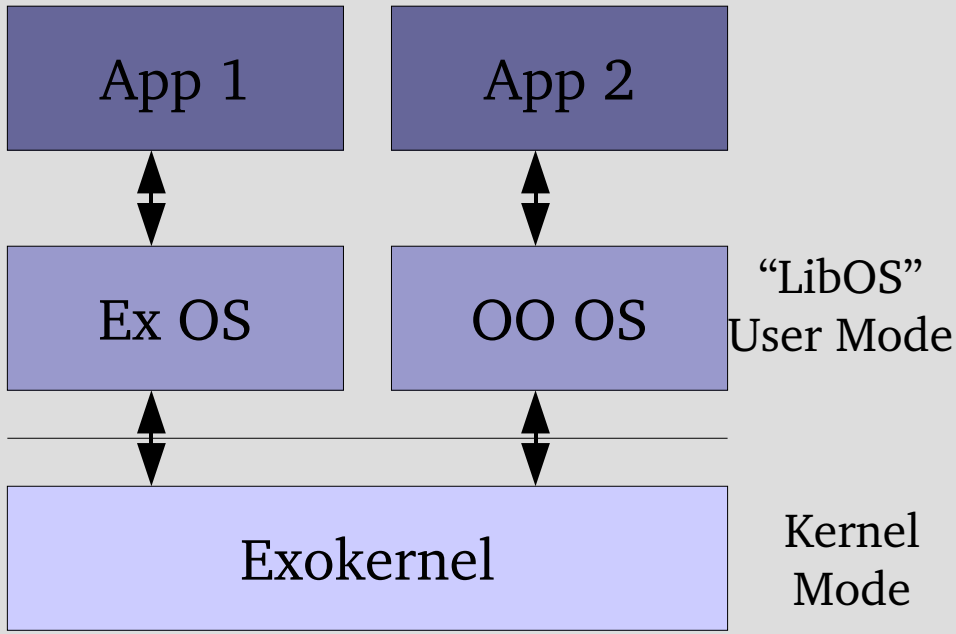
Nach Tanenbaum

Möglichkeiten der Strukturierung (2)

2. Exokernel

Prinzip:
Alles, was der Benutzer programmieren kann, soll er auch selber machen

- Es wird z.B. kein Dateisystem vorgegeben
- Einzige Aufgabe des Kernels ist das sichere Belegen und Freigeben von Resourcen
- Alle anderen Dienste sind User-Libraries (“LibOS”)

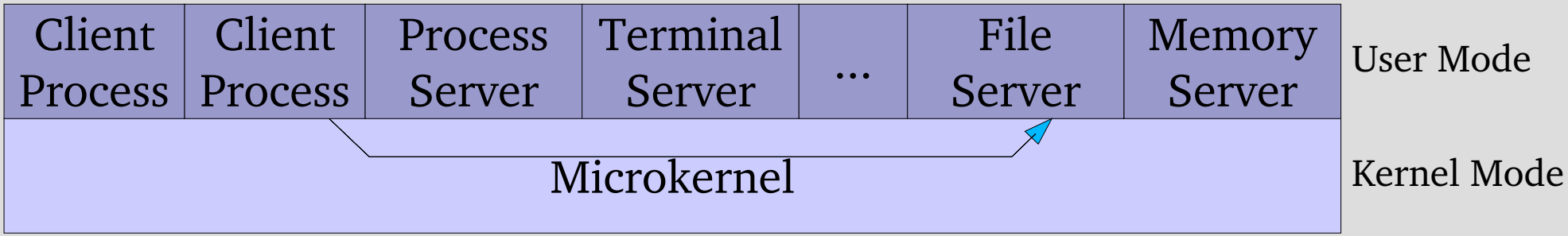


Nach Engeler '95

Möglichkeiten der Strukturierung (3)

3. Client-Server Systeme

- Grösster Teil des OS läuft im User Mode
- Kommunikation durch den Microkernel
- Eignet sich gut für verteilte Systeme
- Viele Context Switches
  - ➔ nicht sehr effizient



Nach Tanenbaum

Gliederung

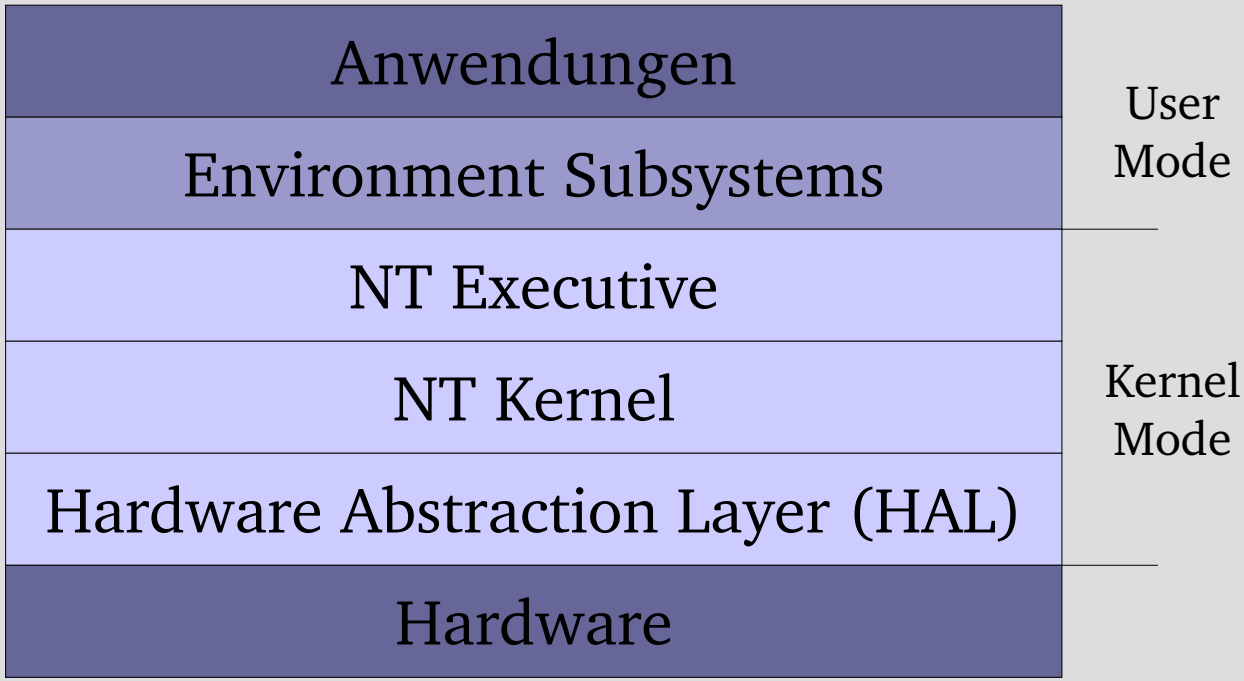
1. Das Designproblem
   a) Überblick
   b) Design Goals
   c) Möglichkeiten der Strukturierung

2. Umsetzung der Design Goals in NT 5.0
   a) Architektur-Überblick
   b) Aufbau der User Mode Komponenten
   c) Aufbau des Kernels

3.Zusammenfassung

Architektur-Überblick

- Schichten-Modell
  - NT Executive im Kernel Mode -› Effizienz
  - Hardwarezugriff in HAL gekapselt -› Portierbarkeit

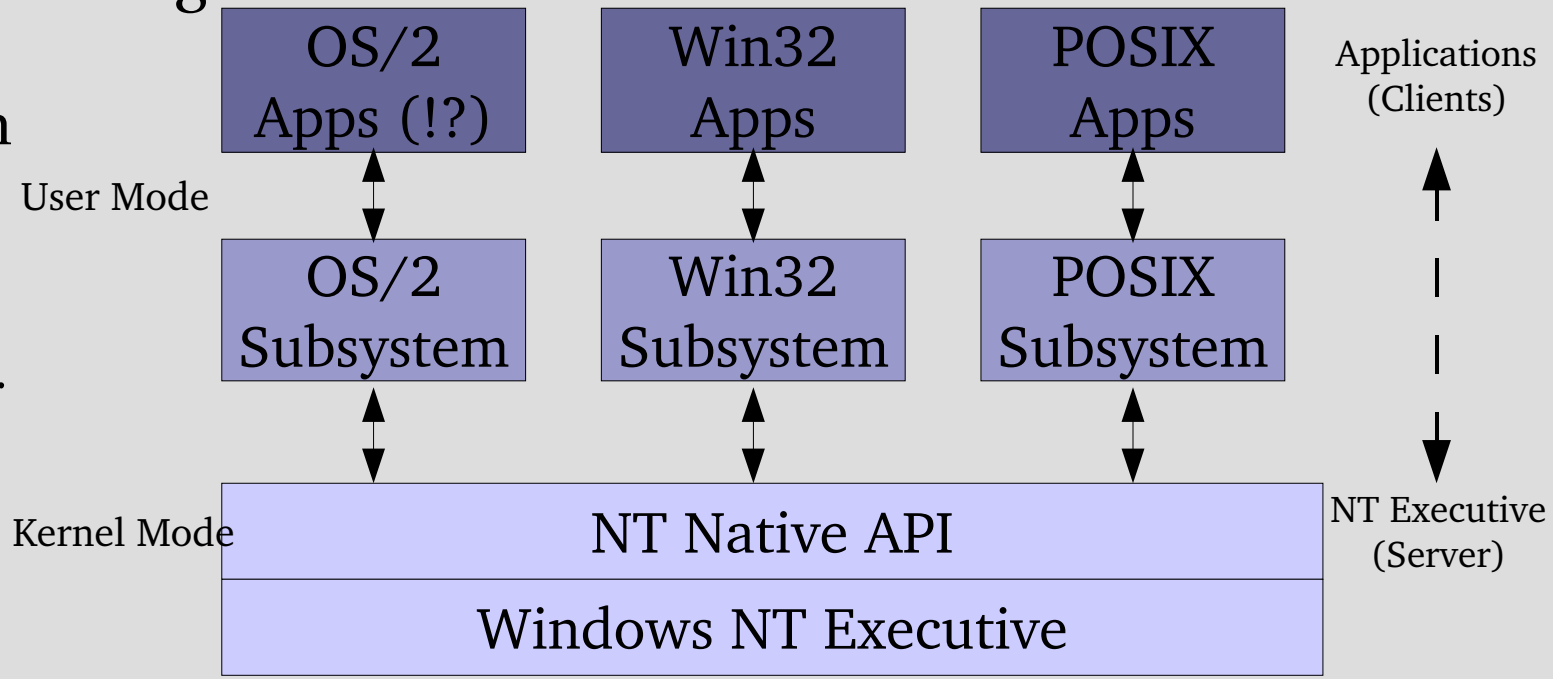


Aufbau der User Mode Komponenten

- Client-Server Modell:

➔Anwendungen kommunizieren mit dem Kernel über die Environment Subsystems -› Kompatibilität

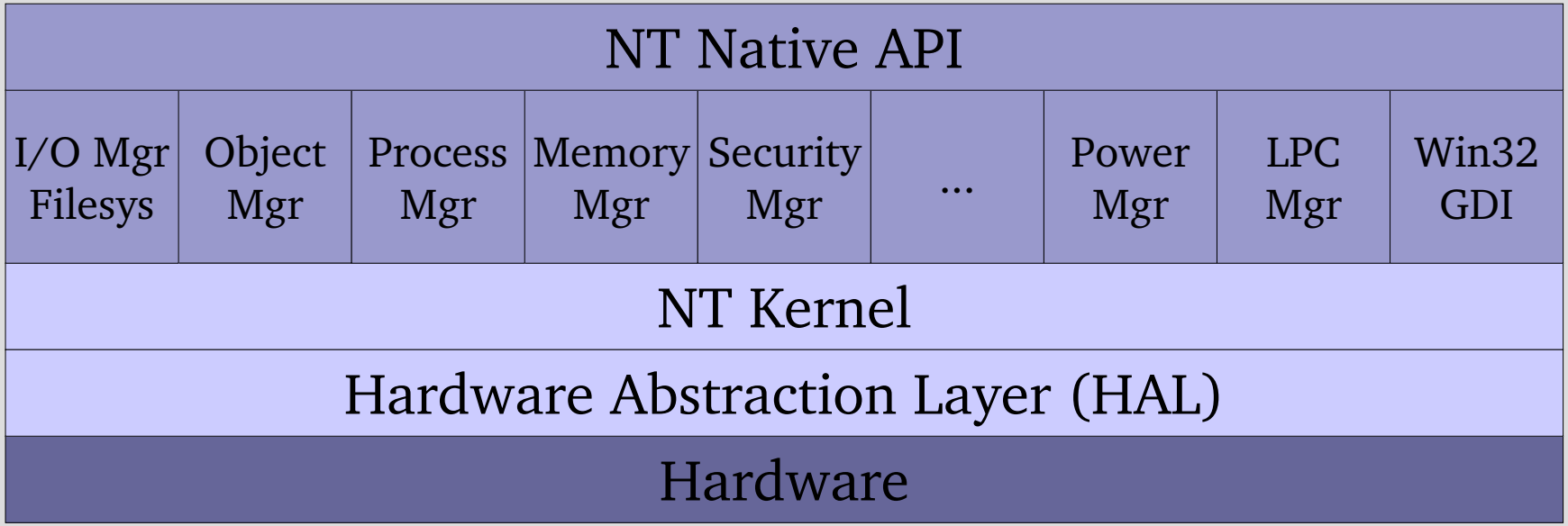
- NT Native API ständig erweitert
  - ➔ Normale Anwendungen sind von der Native API unabhängig, Emulation der jeweiligen OS API im Env Subsystem



Aufbau des Kernels (1)

- Wegen starker Abhängigkeiten nur schwer in Module gliederbar
  -› im Prinzip Monolithisch

- Frühere User Mode Module wie das GDI jetzt im Kernel Mode
  -› Effizienz



Aufbau des Kernels (2)

Einige Kernel Module:

- Object Manager
  - Zentrale Rolle: Alle Resourcen sind Objekte (Threads, Files, ...)
  - Repräsentation durch Handles
  - Reference Counting
- Security Manager
  - Vergibt und überprüft Zugriffsrechte auf Objekte
  - “Access Token” wird an Objekte angehängt
- Process Manager
  - Erstellt, löscht, startet und stoppt Prozesse
  - kein Scheduling (-> Kernel)

Aufbau des Kernels (3)

- LPC Manager
  - ➔ Interprozesskommunikation: Apps ⇔ Env Subsystems
  - ➔ Aus Geschindigkeitsgründen keinen normalen IPC Mechanismen
- NT Kernel
  - ➔ Thread Scheduling (Preemptive Scheduler, 32 Priority Levels)
  - ➔ Software / Hardware Interrupts, Exceptions
  - ➔ Stellt “Kernel Objects” zur Verfügung (Mutex, Semaphor, Event, ...)
- Hardware Abstraction Layer (HAL)
  - Bildet Bus Adressen auf Logische Adressen ab
  - Clock-, Timer-Management
  - Hardware-unabhängiger DMA Datentransfer

Zusammenfassung

- Betriebssysteme sind sehr komplexe Programme
- 3 Gängige Strukturierungsmöglichkeiten
  - Layered
  - Exokernel
  - Client-Server
- NT User-Mode Komponenten:
  - Env Subsystems sind für Kompatibilität verantwortlich
  - Client-Server Modell bei der Kommunikation App <-> Subsystem
- NT Kernel Mode Komponenten:
  - Grob in Schichte eingeteilt: HAL, NT Kernel, NT Executive
  - Vielzahl stark voneinander abhängiger Module in der NT Executive

Literatur / Quellen

- “Modern Operating Systems”
  Andrew S. Tanenbaum, Prentice Hall, 2001

- “Sliding through Operating Systems”
  Daniel Menasce, George Mason University, 1997

- “Exokernel: an operating system architecture for application-level resource management”
  Dawson R. Engler, M. Frans Kaashoek et al, 1995