Windows Native API Programmierung

Johannes Rudolph

Dirk von Suchodoletz
Prof. Schneider
Lehrstuhl für Kommunikationssysteme
Institut für Informatik
Universität Freiburg

Gliederung

Gliederung

Windows parametrisieren

Situation:

- Poolraumumgebung
- Clients mit VMWare auf Linux
- Windowsimage direkt vom Netzwerk gebootet

Vorteil:

- alle Clients genau gleich
- Neustart: Ausgangszustand wiederhergestellt

Problem?

Nachteile

Windowsinstanzen sind zu gleich:

- Computernamen gleich: Probleme im Netzwerk
- Benutzernamen gleich

Idee: Konfiguration parametrisieren

- Windowskonfiguration = Registrierung
- Möglichkeiten:
  - Registrierung 'Offline' bearbeiten, aber: NTFS-Treiber, proprietäres Registry-Format
  - Anmeldeskript, zu spät im Bootvorgang
  - ?

Die Lösung?

Bootprogramm

- “Bluescreen”
- siehe “autocheck”
- wird früh im Bootvorgang ausgeführt
- hat viele Rechte
- Native API

Gliederung

APIs

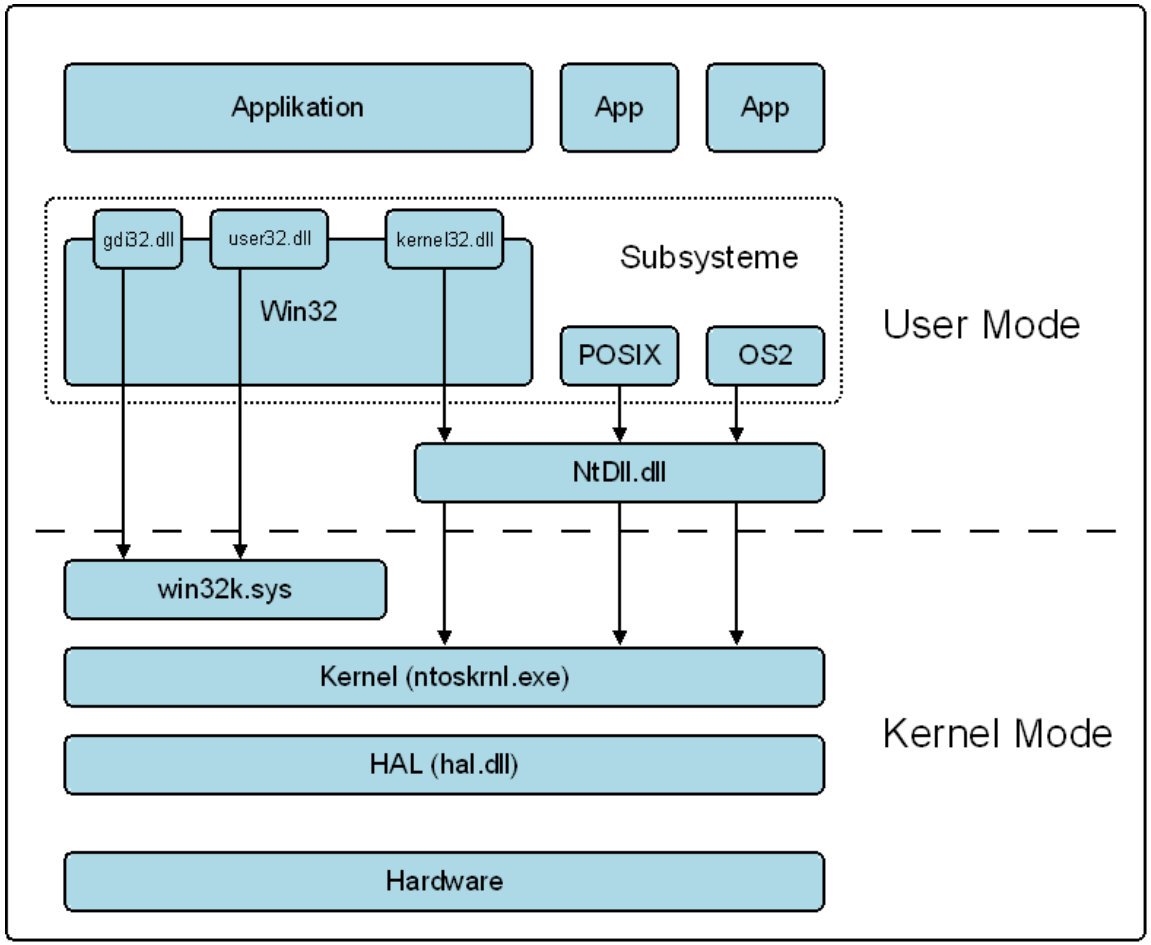

Abbildung: Windows Schichten

Hardware & HAL

Hardwarezugriff durch:

- IO-Ports
- Interrupts
- DMA

Hardware Abstraction Layer:

- kapselt genau diese Zugriffe
- je nach Ausstattung verschiedene Version: hal.dll, halacpi.dll usw.

Kernel & Treiber

Kernel ...

- enthält alle Systemfunktionen
- ist statisch kompiliert in ntoskrnl.exe
- verschiedene Versionen für Multiprozessorsysteme
- durch Treiber (dynamisch) erweiterbar

APIs

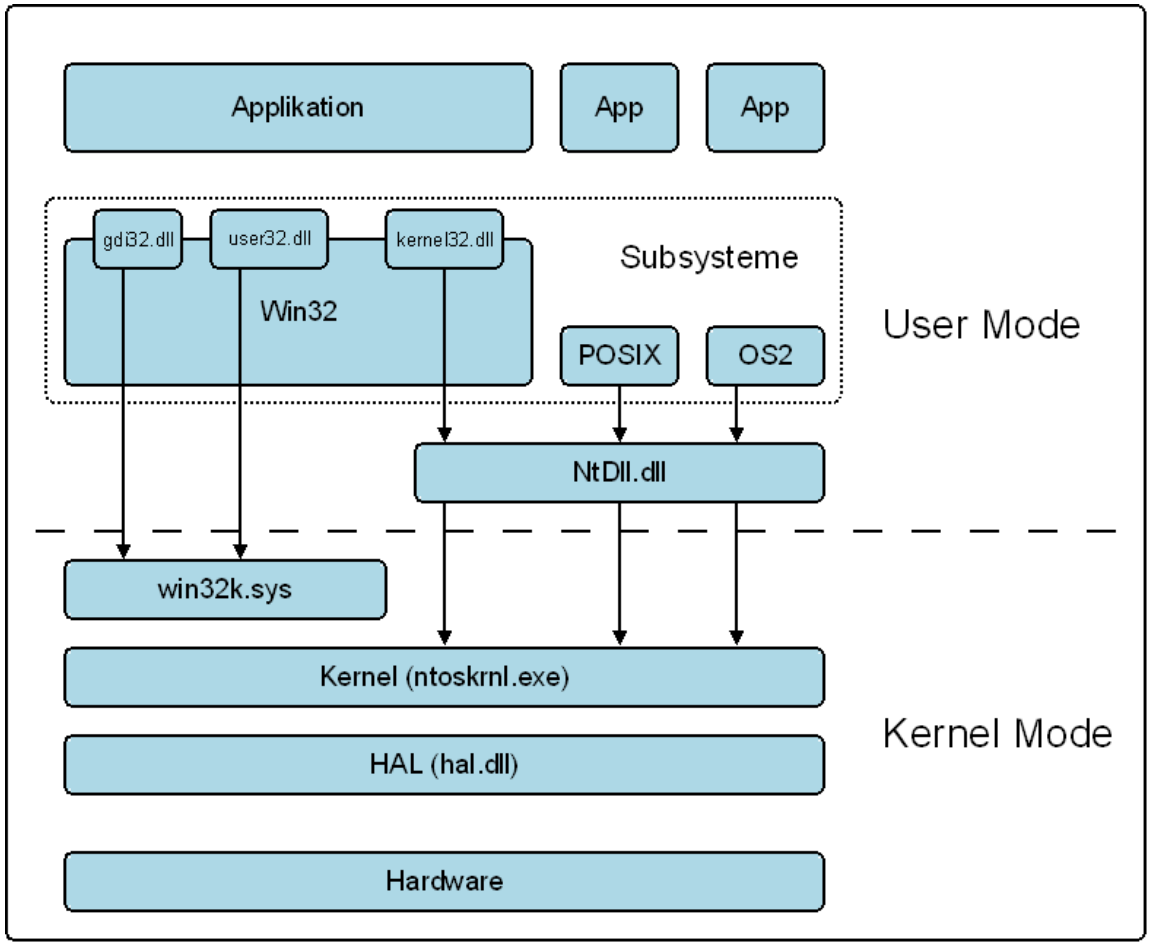

Abbildung: Windows Schichten

Kernelmode, Usermode

Prozessor hat mehrere Ausführungsebenen (auch “Ringe”):

- Usermode
  - Speicherzugriff nur auf bestimmte Bereiche
  - Hardwarezugriff nicht möglich
- Kernelmode: keine Einschränkungen

Interfacing

Sprung von UM nach KM durch

- Interrupt
- sysenter/syscall
- Dispatchtabelle enthält Zeiger zu Funktionen (system services)

Native API

- Bibliothek von Systemfunktionen
- ntdll.dll enthält “Stubs” mit Sprungbefehlen in den Kernel
- Vista: 360 Funktionen

Stub

```
ntdll!NtReadFile
mov eax,0xb7
mov edx,0x7ffe0300
call edx
ret 0x24

0x7ffe0300 SharedUserData!SystemCallStub
mov edx,esp
sysenter
ret
```

APIs

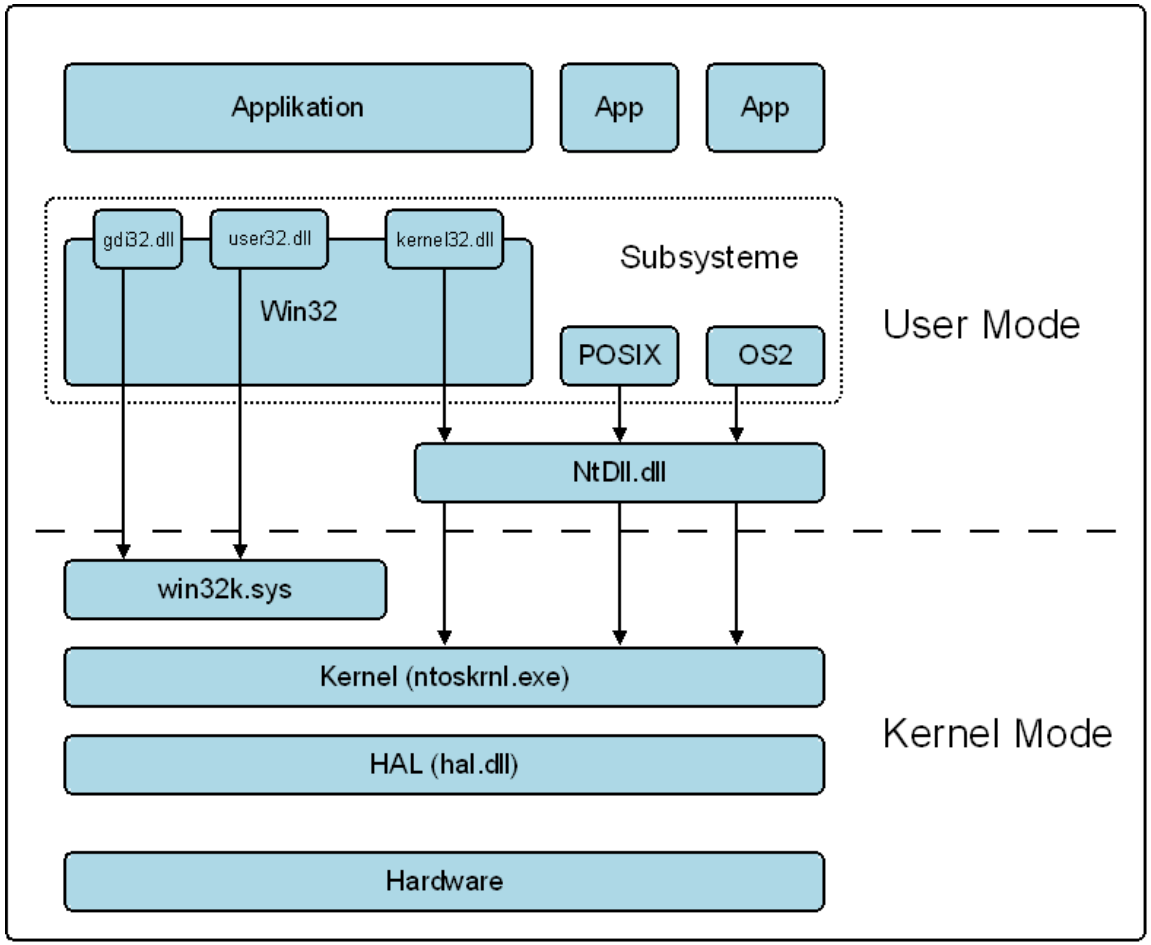

Abbildung: Windows Schichten

Subsysteme

- Idee: ein Kernel, mehrere Usermode-APIs
- Subsysteme: POSIX, OS/2, Win32
- OS/2 Support inzwischen nicht mehr enthalten
- Win32 essentiell, da Code für GUI darin enthalten
- Subsysteme: Usermodeprozesse, die Native API verwenden

Win32

- aufgeteilt in Usermode- und Kernelmodeteil: csrss.exe, win32k.sys
- Funktionen für Grafik (GDI), GUI, Shell, usw.
- Wrapper für NtXXX-Funktionen: dokumentiert bei MSDN
- Kernelmodefunktionen verfügbar über Dispatching wie ntdll.dll

APIs

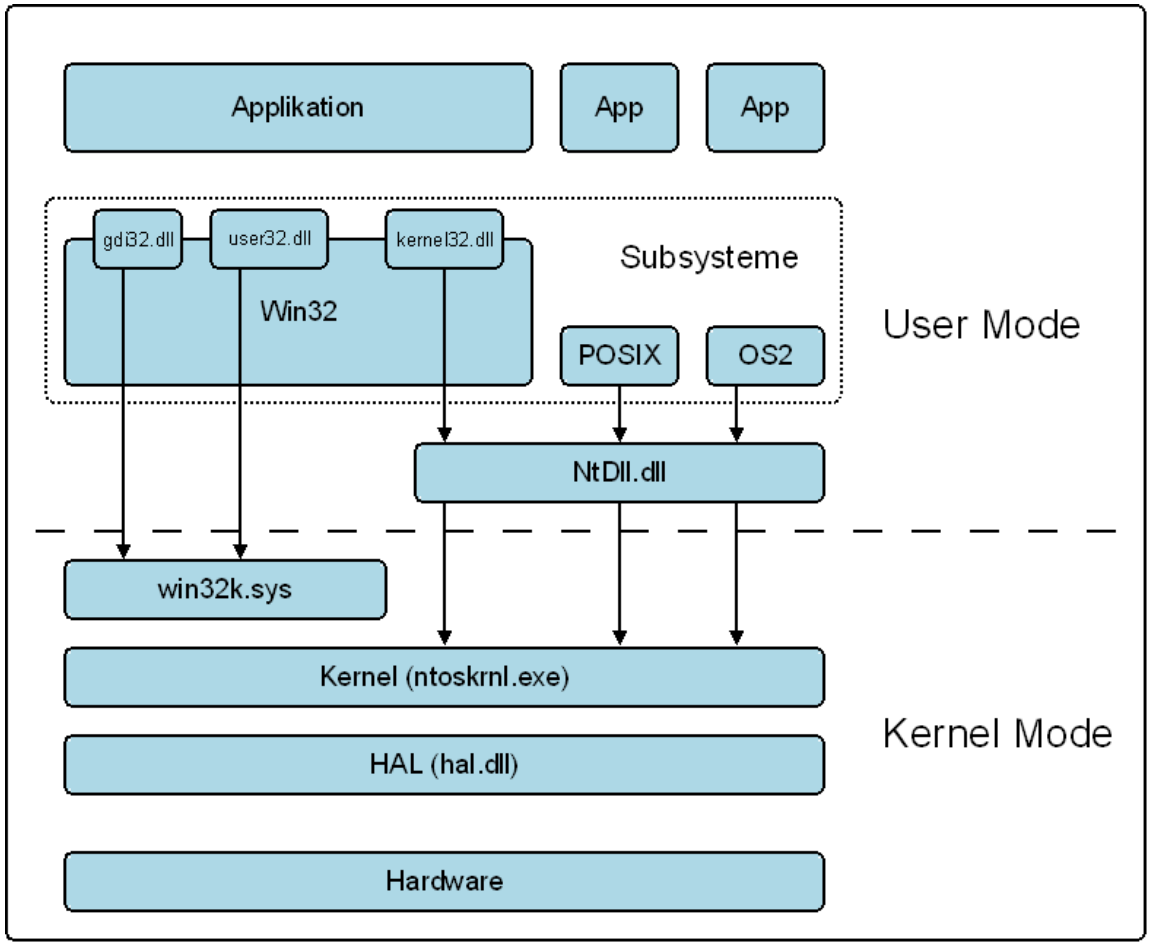

Abbildung: Windows Schichten

Objekte als Verwaltungseinheiten

- Sicherheit mit ACLs
- allgemeiner Namensraum
- einheitliche Zugriffsstruktur
- Referenzzählung

Vorteile:

- Sicherheit nur einmal implementiert
- Buchhaltung zentralisiert und Resourcenverbrauch kontrollierbar
- Lebenszyklus standardisiert

Objekt API

Ablauf:

- NtCreateXXX (&handle,...)
- operiere mit *handle*
- NtClose(handle);

Handle:

- Zeiger auf ein Objekt
- pro Prozess
- enthält gewährte Rechte

Dateien & Registrykeys

Dateien:

- NtCreateFile
- abgespeckt: NtOpenFile
- + NtReadFile
- + NtWriteFile
- ca. 30 Funktionen

Keys:

- NtCreateKey, NtOpenKey
- + NtQueryValueKey, ZwSetValueKey usw.
- ca. 20 Funktionen

Andere Objektfunktionen

- ZwClose
- ZwDuplicateHandle
- ZwQueryObject
- ZwCreateSymbolicLinkObject
- & andere

Namensraum

- enthält benannte Objekte
- sysinternals: WinObject

Registrierung

- zentraler Konfigurationsspeicher
- modular aufgebaut
- Teile werden pro Benutzer eingeblendet
- regedit.exe

Logischer Aufbau

- Baumstruktur
- Key: Knoten, haben Namen, ACL
- Value: Blätter, haben Namen, Typ und Daten
- Wurzelknoten:
  - Machine (HKEY_LOCAL_MACHINE)
  - Users (HKEY_USERS)

Binärer Aufbau

- Hives: Datei, die einen Ast der Registry enthält
- /Windows/system32/config/ ...
  - SAM = Machine/SAM: Security Access Manager
  - software = Machine/Software: Softwarekonfiguration
  - system = Machine/System: Systemkonfiguration
- %USERPROFILE%/NTUSER.dat = Users/%SID% : Benutzerdaten

Bootvorgang

- Bootsektor
- NtLdr
- Kernel
- smss.exe: Session Manager Subsystem
  - BootExecute
  - Registrierung vollständig laden
  - Win32 laden
  - Winlogon
    - Service Control Manager

Gliederung

Bootprogramm

Benutzernamen ändern

Gliederung

Native API

- Windows objektorientiert
- Native API: low-level(?) Usermode-API
- Funktionen zur Arbeit mit Objekten
- leider ziemlich undokumentiert

Fragen?

Danke fürs Zuhören! :)