Vorlesung Betriebssysteme I

Thema 4: Grundlegende Begriffe, Teil 2

Robert Baumgartl

16. November 2015

Begriffe: Schnittstelle

- beschreibt den *statischen* Aspekt einer Kommunikationsbeziehung
- Kommunikation über Schnittstelle kann synchron und asynchron erfolgen
- kann in Hardware oder in Software vorliegen

Hardwareschnittstellen – Beispiele

- Peripheral Component Interconnect (PCI)
- Controller Area Network (CAN)
- InfiniBand

Softwareschnittstellen = Gesamtheit aller nutzbaren Funktionen einer Bibliothek, eines Betriebssystems, einer Middleware (aka API – **A**pplication **P**rogrammer's **I**nterface)
Beispiele: POSIX, Win32, Qt-API

Begriffe: Protokoll

- beschreibt den *dynamischen* Aspekt einer Kommunikation (also den Ablauf)
- Beispiele
  - Timingdiagramme für das Signalspiel
  - Semantikbeschreibung von Systemrufen
  - Präzedenzen für den Aufruf von Funktionen

Protokoll und Schnittstelle bedingen einander!
Es gibt *proprietäre* und *offene* Schnittstellen und Protokolle.

Beispiel für (Teil einer) Protokollbeschreibung

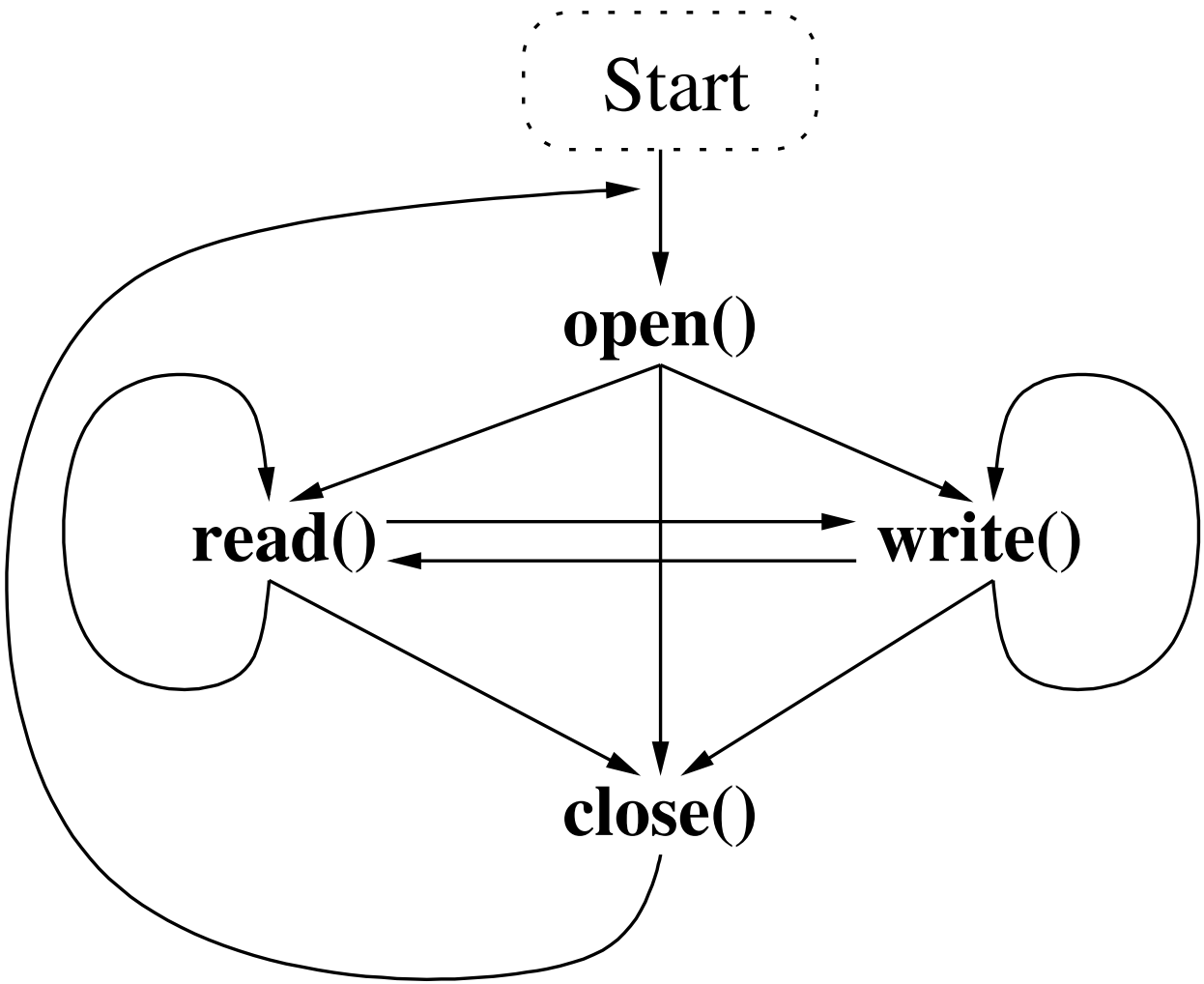
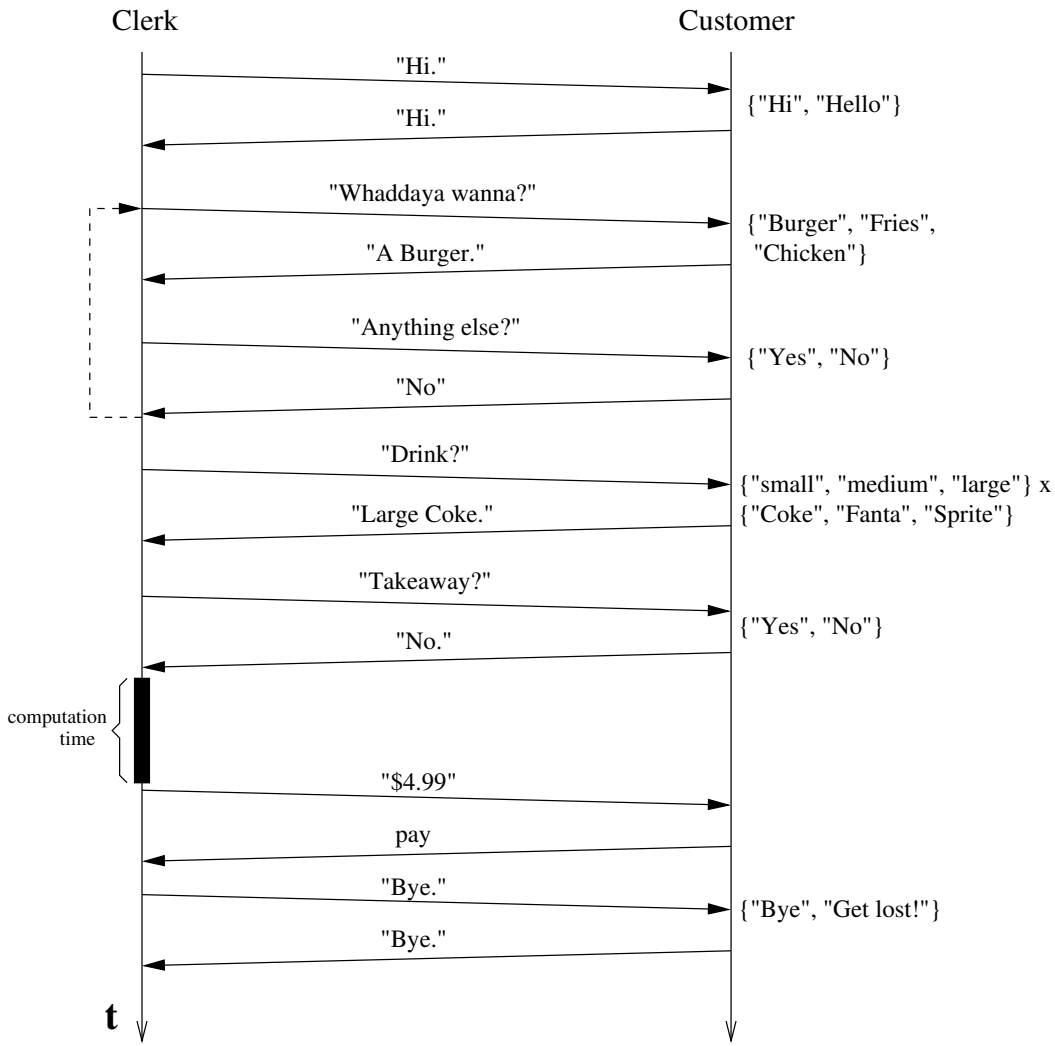


Abbildung: Typische Präzedenzen bei Funktionen eines Dateisystems

Protokollbeispiel

Kommunikation eines Kunden mit dem *Clerk* bei McDonald's



Aktivitäten und Ressourcen

In einem Rechensystem gibt es zwei Kategorien von grundsätzlichen Objekten

1. Aktivitäten: das, was abgearbeitet wird
   - Task
   - Prozess
   - Thread
   - Routine
   - ...
   - (siehe später)
2. Ressourcen: das, was Aktivitäten „zum Leben“ benötigen

Ressourcen

- „alles das, was keine Aktivität ist“
- Aktivitäten konkurrieren um Ressourcen
- existieren in allen Schichten eines Systems
- Beispiele: Datei, Festplatte, Programmcode, Hauptspeicherblock
- = Hardware und alle passiven Abstraktionen eines Rechensystems (d. h. auch CPU und Geräte)
- besitzen zu jedem Zeitpunkt einen inneren Zustand (z. B. CPU: Gesamtheit der Inhalte aller Register)
- Ressourcen werden durch Aktivitäten *angefordert*,durch eine zentrale Instanz *zugeteilt* und nach Nutzung durch die Aktivität *zurückgegeben*($\leftarrow$ Protokoll!)

Entziehbare Ressourcen

Def. Eine *entziehbare Ressource* kann nach ihrer Zuteilung der Aktivität jederzeit entzogen werden. Der Vorgang ist für die Aktivität transparent.
Ablauf:

1. Aktivität anhalten
2. Zustand der Ressource sichern (z.B. auf Datenträger schreiben)
3. [Ressource anderweitig verwenden]
4. Zustand der Ressource restaurieren
5. Aktivität fortsetzen

Voraussetzung für Entziehbarkeit:

- Zustand der Ressource ist vollständig auslesbar
- Zustand der Ressource kann beliebig manipuliert werden.

Entziehbare Ressourcen - Beispiele

- CPU (Zustand kann in den Hauptspeicher ausgelagert werden)
- Hauptspeicherblock (Zustand kann auf Massenspeicher ausgelagert werden)
- Datei

Die meisten Ressourcen sind nicht entziehbar:

- CPU-Cache
- Drucker
- Netzwerkkarte

Exklusiv nutzbare Ressourcen

Def. Eine *exklusiv nutzbare* Ressource darf zu jedem Zeitpunkt maximal von *einer* Aktivität genutzt werden.

- Beispiele: Hardware, (beschreibbarer) Speicher, zum Schreiben eröffnete Datei
- BS muss Exklusivität durchsetzen ($\rightarrow$ Synchronisationsmechanismen)
- Zuteilung kann mittels verschiedener Strategien erfolgen:
  - Fairness
  - Minimierung der Wartezeit
  - Garantie einer maximalen Wartezeit

Klassifikation und Beispiele für Ressourcen

entziehbar	*nicht entziehbar*
Prozessor, Speicher	Datei, alle verbrauchbaren BM
gleichzeitig nutzbar	*exklusiv nutzbar*
Programmcode, Datei, Speicher	Prozessor, Drucker, Signal
wiederverwendbar	*verbrauchbar*
Prozessor, Datei, Speicher	Signal, Nachricht, Interrupt
physisch	*logisch oder virtuell*
Prozessor, Speicher, Geräte	Datei, Signal, Prozessor (!)

Tabelle: Klassifikation von Ressourcen

Ressourcentransformation

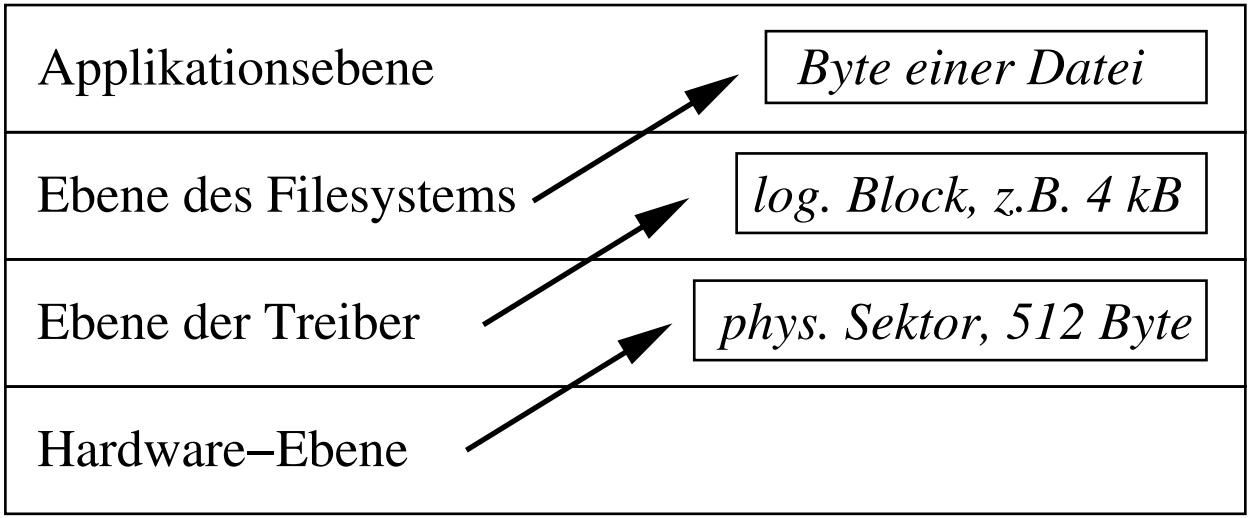


Abbildung: Transformation der Ressource *physischer Sektor* in *Datei*

Es kann dabei sogar eine neue Qualität entstehen:

Speicher + Identifikator + Programmcode = neuer Prozess

User Mode und Kernel Mode

- Idee: nur in einem privilegierten Modus (*Kernel Mode*) dürfen alle Operationen ausgeführt werden (z.B. Zugriff auf die Hardware, Manipulation von systemrelevanten Datenstrukturen wie der Prozesstabelle)
- dieser ist dem Betriebssystem vorbehalten
- Applikationen werden in einem restriktiven Modus (*User Mode*) ausgeführt (z.B. erfolgt automatische Prüfung der Gültigkeit jeder Speicherreferenz)
- bei Verletzung der Restriktionen wird die Applikation abgebrochen
- Unterscheidung Kernel Mode vs. User Mode analog zur Einteilung Administratoren vs. gewöhnliche Nutzer
- Ziel: Etablierung eines grundlegenden Schutzkonzeptes

User Mode und Kernel Mode

Was darf man nur im Kernel Mode?

- neuen Prozess erzeugen
- Treiber ins System laden oder daraus entfernen
- generell: Diensterbringung des Betriebssystems
- **nicht** jedoch: typische Adminaufgaben

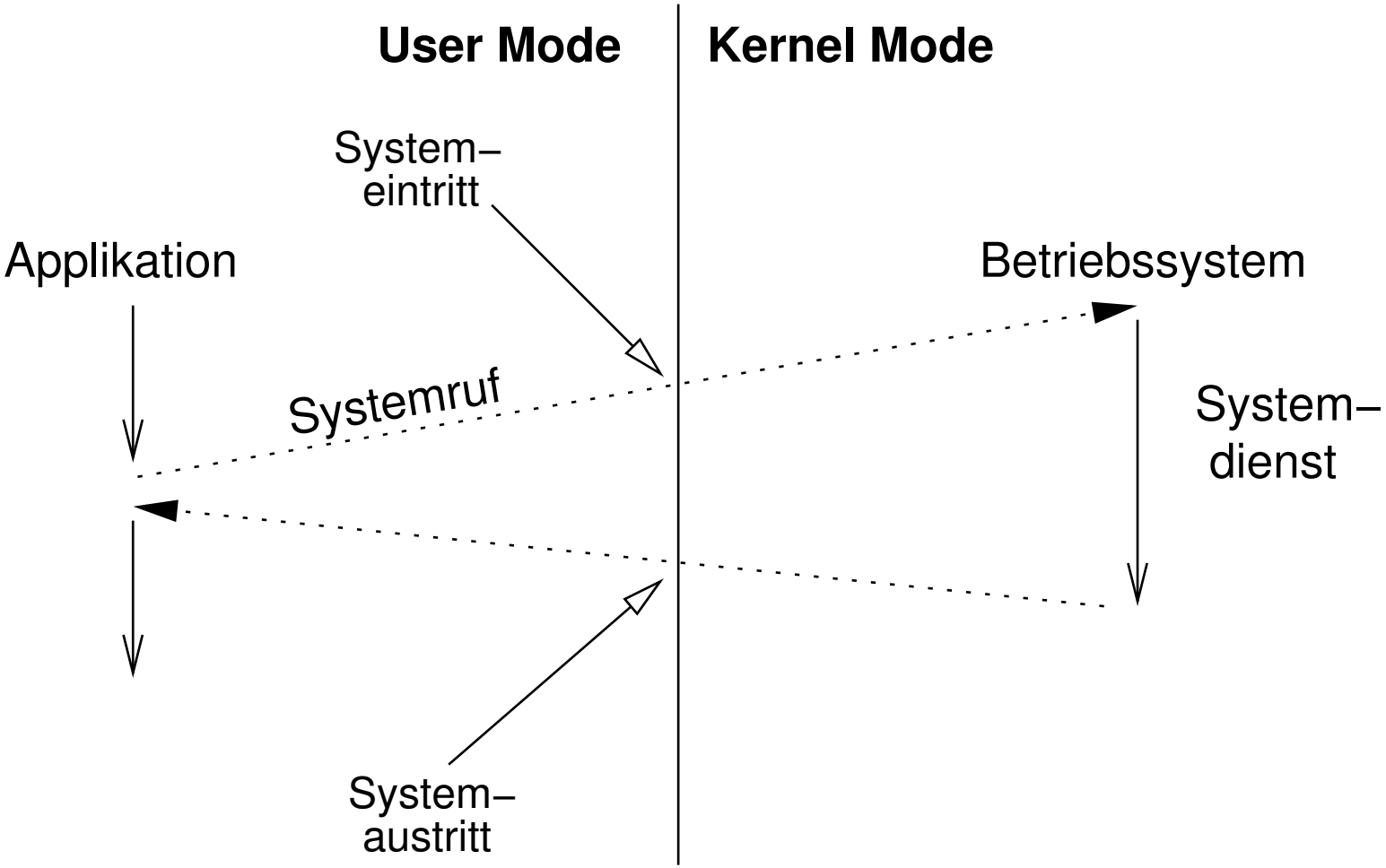
Die CPU muss User Mode/Kernel Mode unterstützen, d.h., verschiedene Privilegierungsmodi unterscheiden.

Systemruf

Damit der „gewöhnliche“ Nutzer die Funktionen des Kernels überhaupt anwenden darf, gibt es den Mechanismus des Systemrufs.

- BS bietet dem Programmierer Funktionen, diese werden über Systemrufe zur Verfügung gestellt
- Gesamtheit aller Systemrufe eines BS ist dessen Application Programmer’s Interface (API)
- Nutzung analog den Funktionen einer Bibliothek mit einem Unterschied: Diensterbringung erfolgt im Kernel Mode
- $\rightarrow$ gewöhnlicher Funktionsaufruf als Mechanismus unbrauchbar!
- Systemrufe können blockieren!

Prinzip eines Systemrufs



Ablauf eines Systemrufs

```
count = read(fd, buffer, nbytes);
```

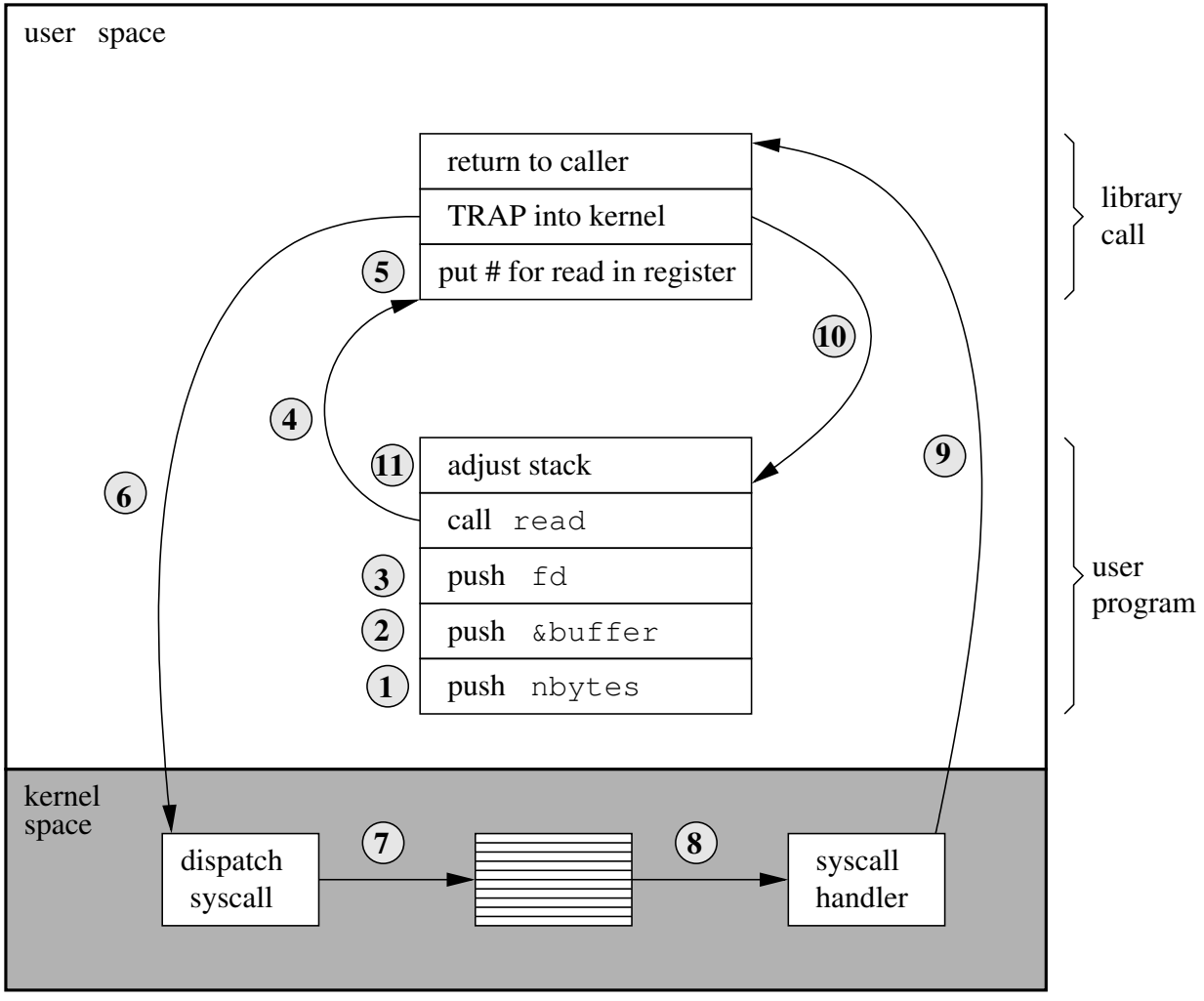
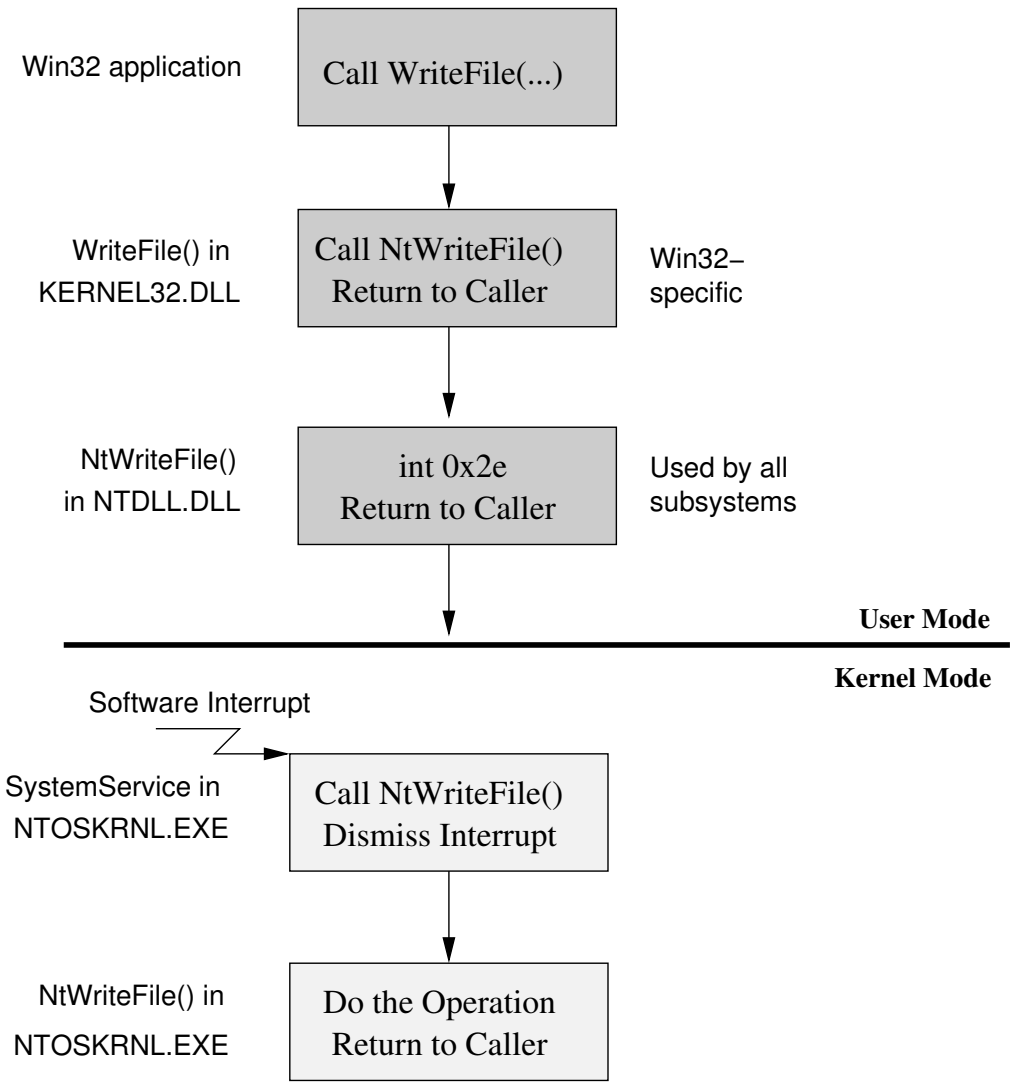


Abbildung: Allgemeiner Ablauf eines Systemrufs `read()`

Ablauf von WriteFile() in Windows 2000/XP/Vista



Quelle: David Solomon, *Inside Windows XP*, Microsoft Press, 2000

Was haben wir gelernt?

1. Protokoll und Schnittstelle
2. Ressourcen
   - entziehbare
   - exklusiv nutzbare
   - Ressourcentransformation
3. Kernel Mode und User Mode
4. Was ist ein Systemruf?