Betriebssysteme

Tutorium 2

Philipp Kirchhofer
philipp.kirchhofer@student.kit.edu
http://www.stud.uni-karlsruhe.de/~uxbtt/

Lehrstuhl Systemarchitektur
Universität Karlsruhe (TH)

4. November 2009

Was machen wir heute?

1. Organisatorisches

2. Tutorien Übungsblatt
   - OS Terms
   - System Structures
   - System Call Basics

Organisatorisches

Theorieblätter Abgabe

1. Theorieblatt Heute um 12 Uhr

2. Theorieblatt Montag, 16. November, 18 Uhr

Probleme mit OS/161?

Im Forum auf der Vorlesungsseite stehen diverse Patches zur Verfügung.

Tutorien Übungsblatt - OS Terms

Frage 2.1.a

Erkläre die Begriffe Thread, Adressraum und Prozess.
In welchem Zusammenhang stehen diese?

Antwort

Thread (Faden, Ausführungsstrang)
Ein Thread ist die sequentielle Abarbeitung eines Programmflusses.

Adressraum
Ein Adressraum ist eine Menge von Adressen, die eindeutig angesprochen werden können. Alle Threads eines Prozesses liegen in einem Adressraum.
Ein Thread kann im Regelfall nicht auf Daten in einem anderen Adressraum zugreifen.

Prozess
Ein Thread und der zugehörige Adressraum ergeben einen (single-threaded) Prozess.

Tutorien Übungsblatt - OS Terms

Frage 2.1.b

Erkläre die Unterschiede zwischen Batchverarbeitung (single-programming) und Multiprogrammierung (multi-programming).
Welche Vorteile bietet Multiprogrammierung?

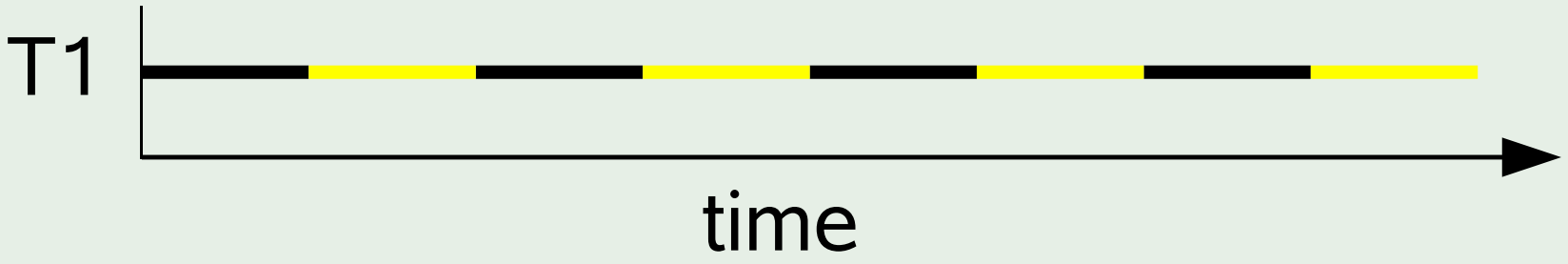
Antwort

Bei Systemen mit Batchverarbeitung wird immer nur ein Programm ausgeführt. Erst nach Ende des Programms kann ein weiteres Programm gestartet werden.

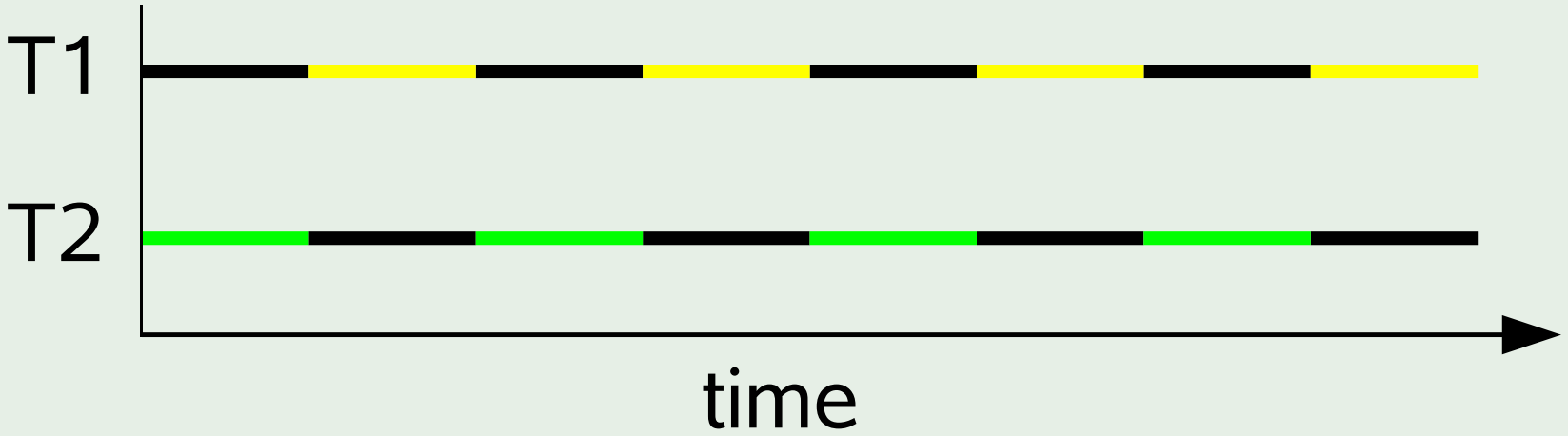
Mit Hilfe von Multiprogrammierung können mehrere Anwendungen gleichzeitig ausführbar sein. Bei Anwendungen mit hohen E/A Latenzen können sich die Programme gegenseitig abwechseln, damit kann ein Leerlauf des Prozessors vermieden werden.

Tutorien Übungsblatt - OS Terms

Beispiel: Batchverarbeitung



Beispiel: Multiprogrammierung



Tutorien Übungsblatt - OS Terms

Frage 2.1.c

Was ist eine Virtuelle Maschine?

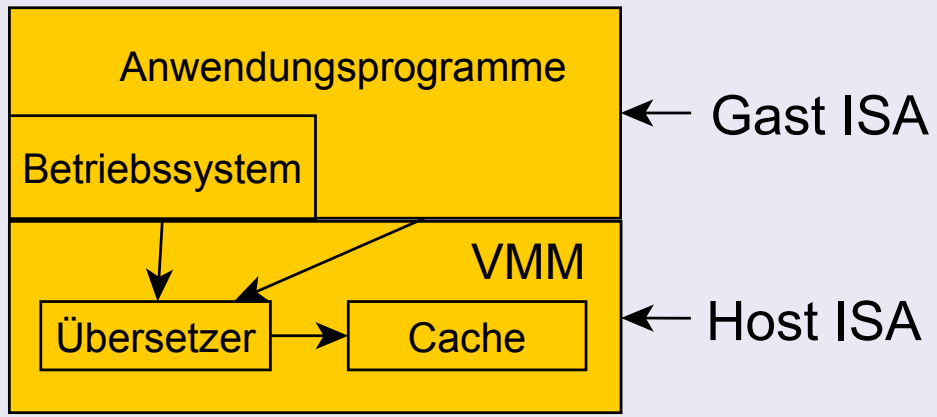
Antwort

Eine virtuelle Maschine ist ein virtueller Computer. Eine solche Maschine besteht nicht aus Hardware, sondern aus Software. Auf einem physischen Computer können gleichzeitig mehrere virtuelle Maschinen betrieben werden.

Beispiele für VMs

- Java Virtual Machine (JVM)
- Transmeta Crusoe / Efficeon
- VMware / VirtualBox

Transmeta Architektur



Tutorien Übungsblatt - System Structures

Frage 2.2.a

Wir nehmen ein typisches PC Betriebssystem (z.B. Windows XP oder Linux) an.
Weshalb sollte ein schlauer Systementwickler manche Betriebssystemfunktionen außerhalb des Kernels implementieren?
Welche Funktionen kommen dafür überhaupt in Frage?

Antwort

Aufgrund der Auslagerung von Funktionen kann der Betriebssystemkern schlank gehalten werden.
Funktionen außerhalb des Kerns laufen im Benutzer-Modus des Prozessors.
Dadurch sind sowohl der Kern als auch andere im Benutzer-Modus laufende Funktionen gegen unzulässige Zugriffe geschützt.

Alle Funktionen, deren Implementierung außerhalb des Kernels nicht die Systemsicherheit (Schutz, Stabilität) beeinträchtigt

Tutorien Übungsblatt - System Structures

Frage 2.2.a

Welche Funktionen eignen sich besonders gut für die Auslagerung?

Antwort

Für die Auslagerung eignen sich besonders Treiber, die langsame Geräte ansprechen. Bei Funktionen, die viel Daten austauschen, eignet sich die Auslagerung nicht, da ein Overhead durch den nötigen Wechsel zwischen Benutzer- und Kern-Modus entsteht.

Beispiel: Linux FUSE (Filesystem in Userspace)

FUSE erlaubt es Dateisysteme als normale Anwendungen zu entwickeln.

- Generischer Teil im Kern (VFS Anbindung)
- Dateisystemspezifischer Bereich in Anwendungsprogramm

Beispiele:

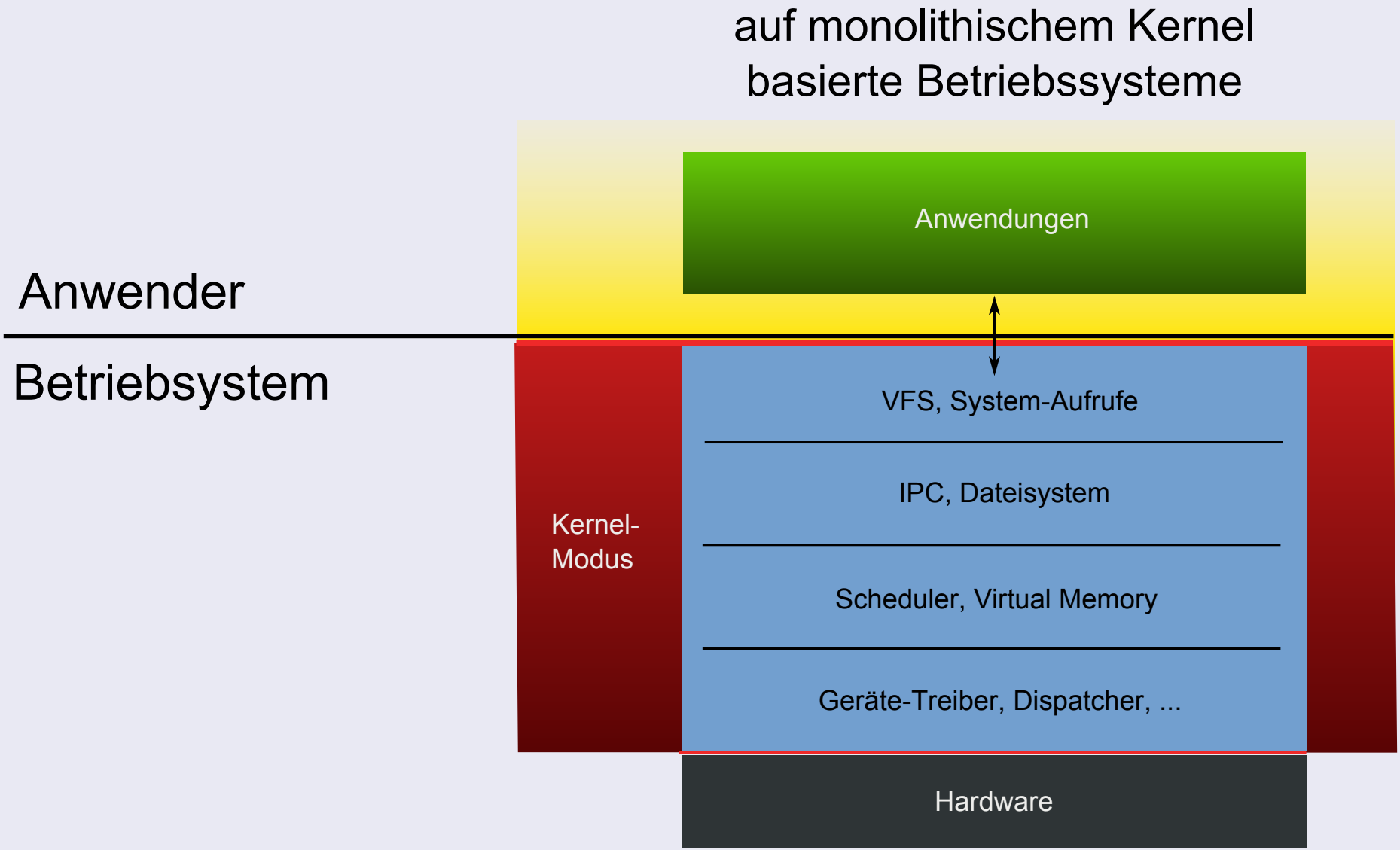
- NTFS
- GmailFS
- SSHFS

Tutorien Übungsblatt - System Structures

Frage 2.2.b

Vergleiche monolithische Kernel mit Mikrokernel-basierten Multi-Server Betriebssystemen.
Was sind die jeweiligen Vor- und Nachteile?

Monolithischer Kernel

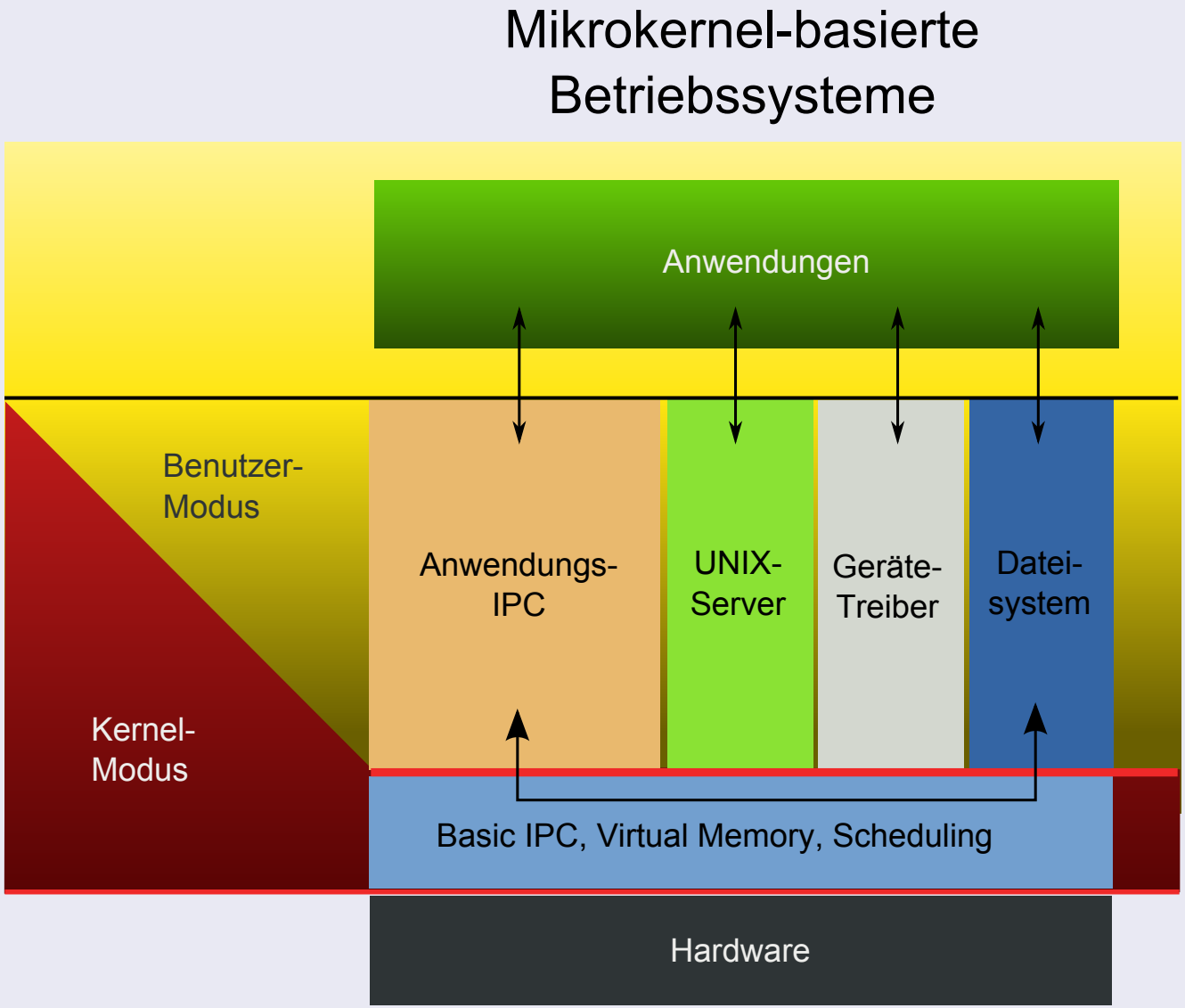


Tutorien Übungsblatt - System Structures

Vor- und Nachteile: Monolithischer Kernel

- ⊕ Schnelle und einfache Dienst-Aufrufe im Kern
- ⊖ Großer Kern mit komplexen Abhängigkeiten
- ⊖ Schlechte Erweiterbarkeit
- ⊖ Keine Isolation zwischen Kern-Komponenten

Mikro-Kernel

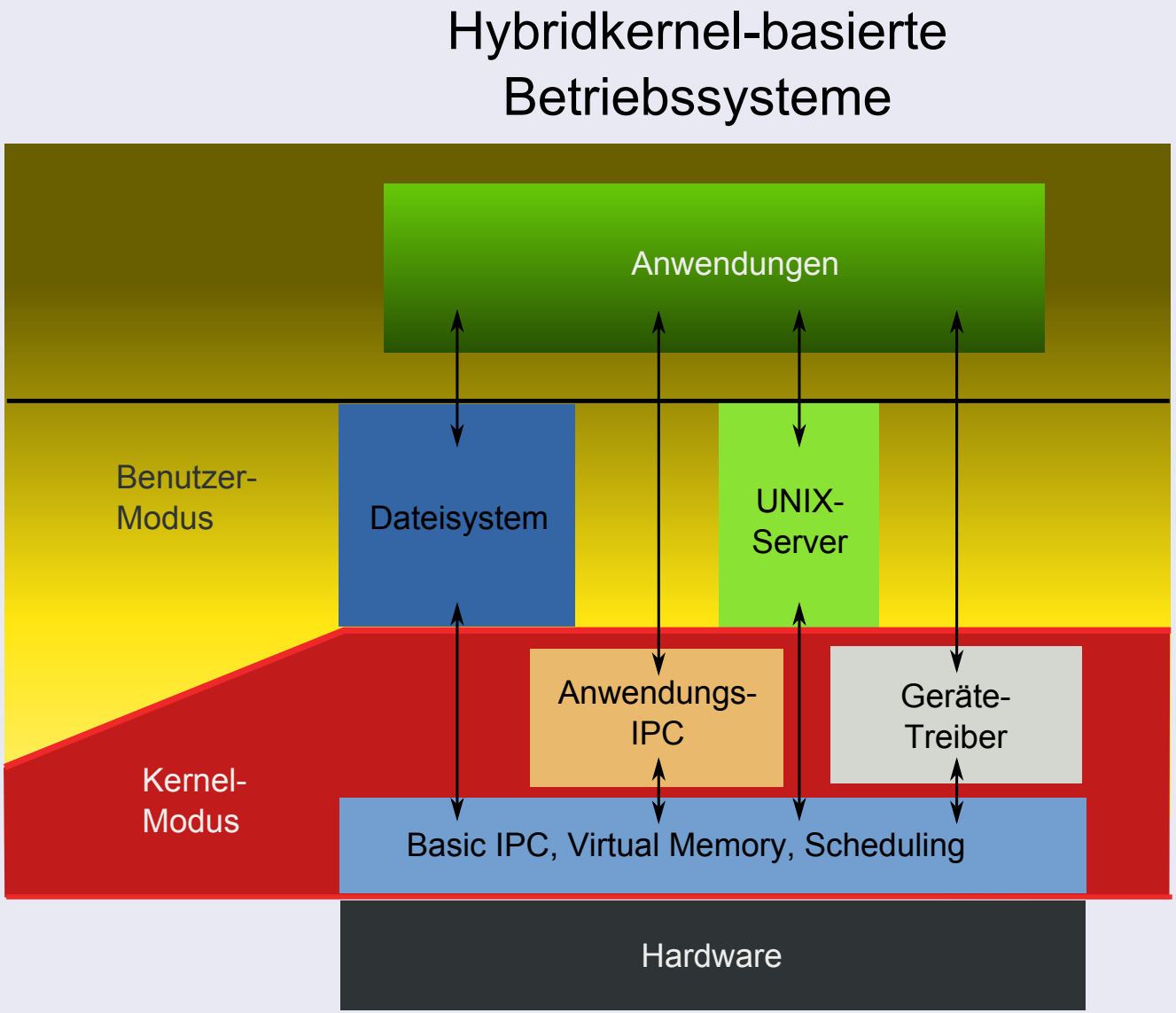


Vor- und Nachteile: Mikro-Kernel

Strikte Isolierung von Kern-Komponenten

- ⊕ Schlanker Mikro-Kern
- ⊕ Sehr gute Isolation zwischen Kern-Komponenten
- ⊕ Klar definierte Server-Schnittstellen erleichtern Abhängigkeitsverwaltung
- ⊖ Kommunikationsoverhead

Hybrid-Kernel



Vor- und Nachteile: Hybrid-Kernel

Kombination aus den Vor- und Nachteilen von monolithischen und Mikro-Kernel Systemen

Tutorien Übungsblatt - System Structures

Frage 2.2.c

Eine Anwendung möchte eine Datei schreiben. Sie führt dazu einen „write“ Systemaufruf (Syscall) durch.
Vergleiche den durch Kernel Ein- und Austritte sowie Adressraumwechsel verursachten Overhead dieses Systemaufrufs bei einem monolithischen und einem Mikro-Kernel basierten Betriebssystem.

Antwort

Bei einem Mikro-Kern basierten Betriebssystem ist im Vergleich zu einem monolithischen System eine höhere Anzahl an Kernel Ein- und Austritten sowie Adressraumwechseln zu beobachten.

Tutorien Übungsblatt - System Structures

Kernel-Benutzer-Wechsel

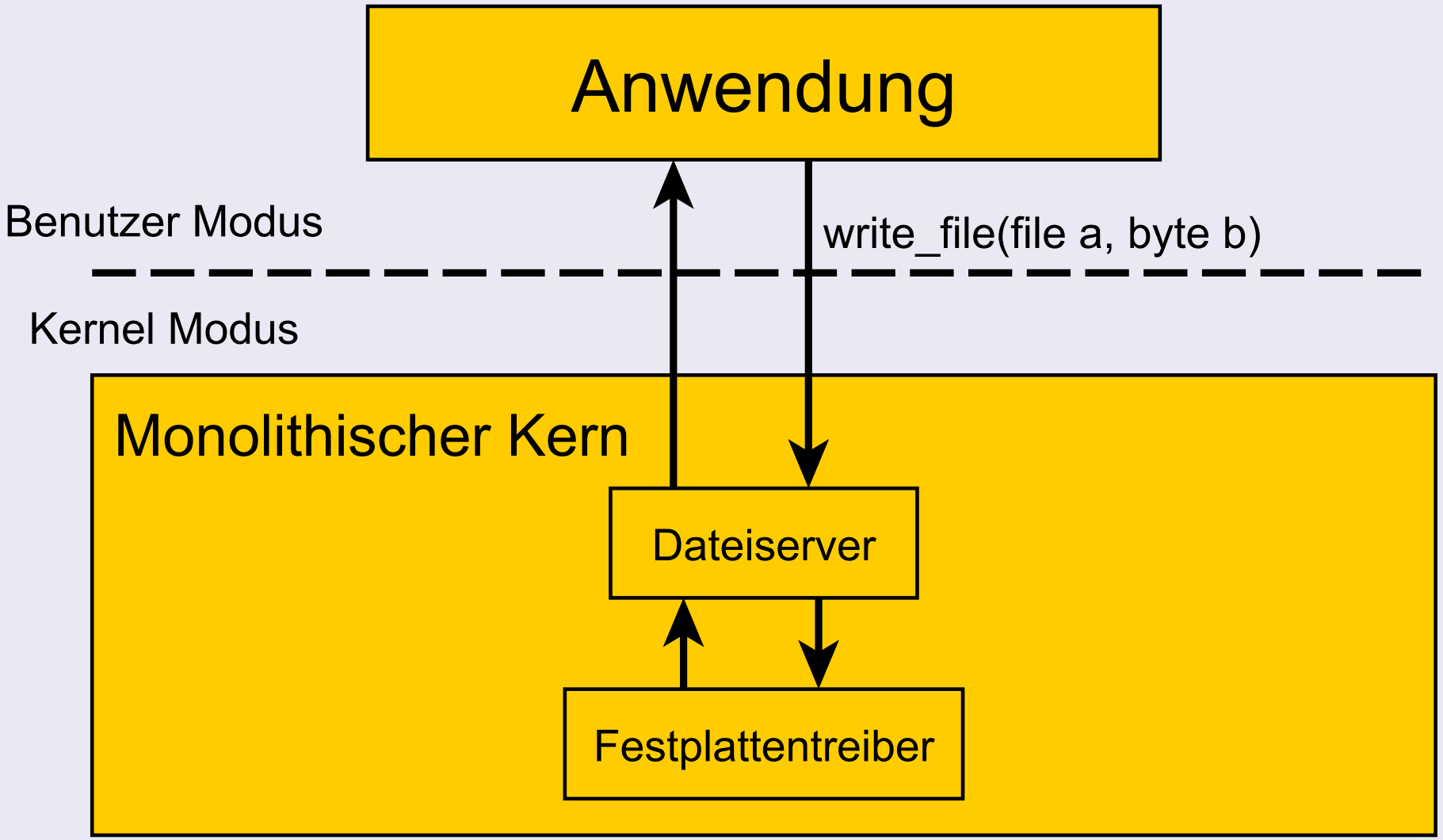
- Prozessormodus umschalten
- Stack auf Kernel-Stack umschalten

Adressraumwechsel

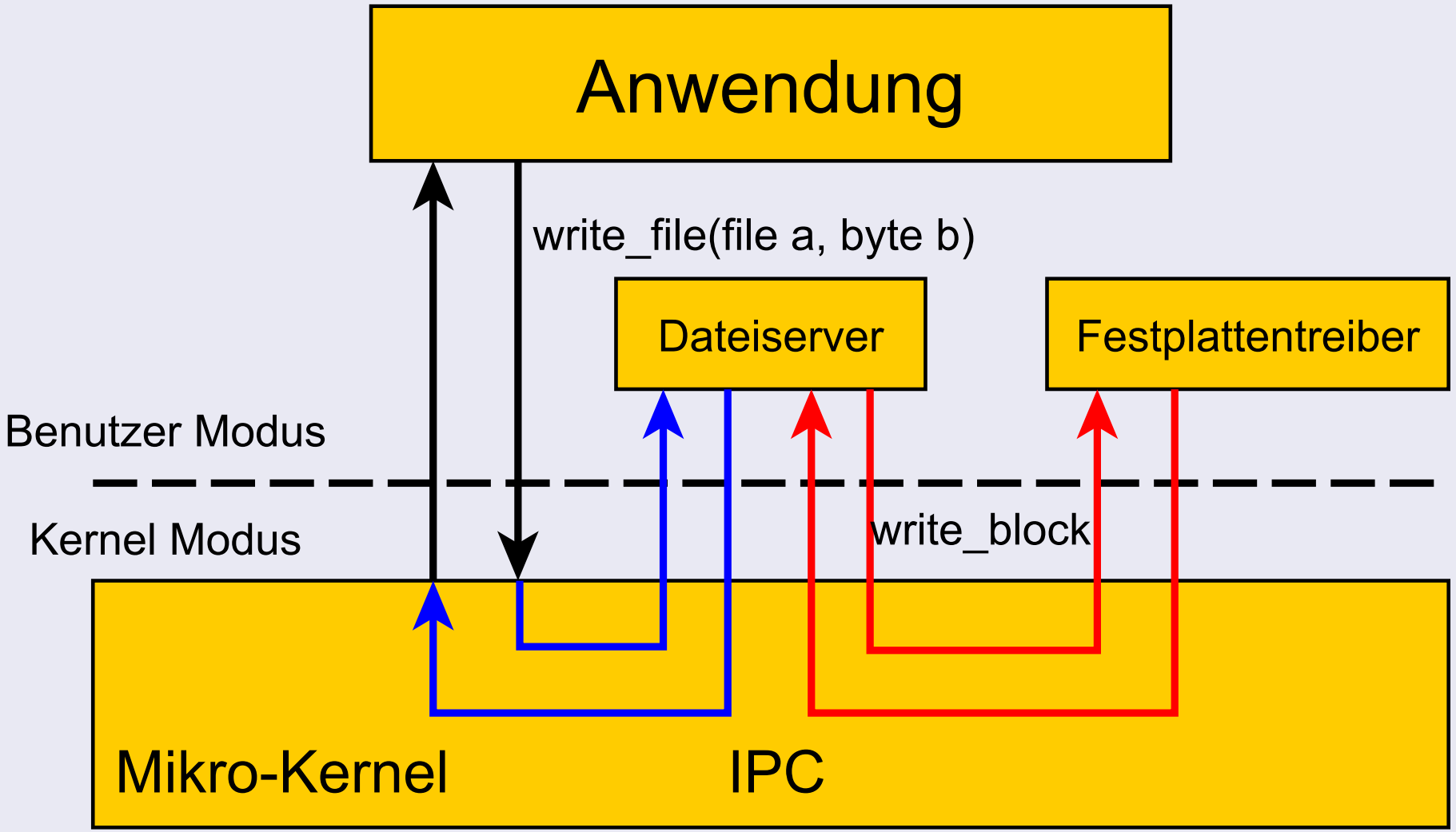
- Wechsel der Tabellen für Verwaltung des virtuellen Speichers
- TLB Cache leeren

TLB beschleunigt Zuordnung von logischen (virtuellen) zu physikalischen Speicheradressen.

Systemaufruf Monolithischer Kernel



Systemaufruf Mikro-Kernel



Tutorien Übungsblatt - System Call Basics

Frage 2.3.a

Welche Ereignissen können zu einem Kern Aufruf führen?

Antwort

- Ausnahmen (Exceptions)
  - Division durch Null
  - Zugriff auf ungültigen Speicher
- Unterbrechungen (Interrupts)
  - Gerät ist fertig mit Arbeiten
  - Interner Zeitgeber (Timer Interrupt)
- System Aufrufe (Syscalls)
  - Prozess starten
  - Datei schreiben, lesen

Um in den Kern zu wechseln verwenden Syscalls meistens spezielle Ausnahmen.

Tutorien Übungsblatt - System Call Basics

Frage 2.3.b

Wie hängt eine „trap“ Instruktion mit Systemaufrufen zusammen?

Antwort

Die „trap“ Instruktion schaltet den Prozessor in den privilegierten Modus (Kernel-Modus) und springt zu einer vorher vom Betriebssystem-Kern vorgegebenen Adresse im Speicher.
Die „trap“ Instruktion erlaubt es also einen Mechanismus auf höherer Ebene (Systemaufrufe) umzusetzen.
Andere Methoden zur synchronen Ausführung des Kerns sind auch denkbar, so könnte z.B. eine bestimmte ungültige Instruktion einen Wechsel zum Kern auslösen und dort besonders interpretiert werden.

Tutorien Übungsblatt - System Call Basics

Frage 2.3.c

Ein Systemaufruf soll auf einen Speicherbereich in einem Anwendungsprogramm schreiben. Welche Probleme können dabei auftreten?

Antwort

Der Kern muss sicherstellen, dass der Speicherbereich in einem Anwendungsprogramm liegt. Ansonsten könnte ein bösartiges Programm Daten im Kern überschreiben.

Frage 2.3.c

Kann es auch zu Problemen kommen wenn der Kern nur lesend auf den Speicherbereich zugreift?

Antwort

Bei einem Lesezugriff können keine Kern-Daten verändert werden, allerdings können „geheime“ Informationen wie z.B. Passwörter ausgelesen werden.

Fragen & Kommentare?

Ende

xkcd: Compiling