Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

Kap. 11: Input / Output

Inhalt

- Eine der Hauptaufgaben eines Betriebssystems ist die Überwachung und Steuerung aller E/A-Geräte (I/O-Geräte, I/O Devices):
  - Kommandos an Geräte geben
  - Unterbrechungen (Interrupts) bedienen
  - Fehlerbehandlung durchführen
- Das I/O-System eines Betriebssystems hat i.d.R. einen beträchtlichen Umfang (z.B. ca. 50% in 4.3BSD)
- Ziele eines I/O-Systems:
  - Einfachheit der Schnittstelle zwischen Geräten und dem Rest des Systems
  - Hohes Maß an Geräteunabhängigkeit der Schnittstelle

Gliederung

11.1 I/O-Hardware

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

I/O-Hardware wird hier nicht im einzelnen besprochen. Bzgl. Details vgl. auch Lehrbuch Tanenbaum.

In Stichworten:

- I/O-Geräte
  - Blockorientierte Geräte:
    - Speichern Informationen in Blöcken gleicher Größe mit jeweils eigener (Block-)Adresse
    - Beispiel: Festplatten
  - Zeichenorientierte Geräte:
    - Erzeugen und akzeptieren Zeichenströme
    - Keine Blockgrenzen, Adressen oder Suchoperationen
    - Beispiele: Terminals, Drucker, Mäuse, Netzwerkschnittstellen.
  - Sonstige Geräte:
    - z.B. Uhren

I/O-Hardware (2)

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Steuerwerke (Device Controller, Adapter):
  - I.d.R. spezialisierte kleine Rechner als Bindeglieder zwischen Systembus und Geräten
  - Beispiele für standardisierte Schnittstellen zwischen Controller und Geräten: USB, SCSI/SAS, IEEE 488
  - Bei Großrechnern: Intelligente I/O-Kanäle anstatt Controller, arbeiten eigenständig Kanalprogramme im Hauptspeicher ab
  - Controller sind über adressierbare Register zugreifbar
    - im allg. phys. Adressraum des Prozessors (z.B. MC 680x0)
    - oder in spez. I/O-Space (z.B. Intel 386)
    - Hierdurch erfolgt Initiierung von I/O-Operationen
- Direct Memory Access (DMA):
  - Eigenständiges Übertragen von Daten durch den Controller ohne Zuhilfenahme des Prozessors
  - Indirekte Performance-Beeinflussung durch Cycle Stealing

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

Gliederung

1. Ziele und Grobarchitektur
2. Unterbrechungsbehandlung
3. Gerätetreiber
4. Geräteunabhängige Software des BS-Kerns
5. I/O-Software im Benutzer-Modus

11.2.1 Ziele

- Geräteunabhängigkeit:
  - Schlüsselkonzept beim Entwurf
  - Ein Programm soll nicht wissen müssen, ob eine zugegriffene Datei z.B. auf einer Festplatte oder einem Diskettenlaufwerk gespeichert ist
- Einheitliches Benennungsschema für Dateien und Geräte
- Verwaltung gemeinsam und exklusiv benutzbarer Geräte:
  - Gemeinsam benutzbare Geräte erlauben die gleichzeitige Benutzung durch mehrere Prozesse (z.B. Magnetplatten)
  - Exklusiv benutzbare Geräte erlauben dies nicht (z.B. Drucker)

Ziele (2)

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Synchrone / Asynchrone I/O-Operationen:
  - synchron: Blockierung bis zum Abschluss der Operation
  - asynchron: Ende der Blockierung nach Einleitung der Operation
  - Für Anwendungsprogramme sichtbare E/A-Operationen können durch das Betriebssystem blockierend erscheinen (einfachere Programmierung), auch wenn physikalisches I/O asynchron und unterbrechungsgesteuert abläuft
- Fehlerbehandlung:
  - so nahe an der Hardware wie möglich
  - z.B. durch Wiederholung (Retry) bei transienten Fehlern

Grobarchitektur

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Erreichung der Ziele durch geschichteten Entwurf der I/O-System-Software in vier Schichten (Layers):

Schicht	
4	I/O-Software im Benutzer-Modus
3	Geräteunabhängige SW des BS-Kerns
2	Gerätetreiber
1	Unterbrechungsbehandlung

- **Diese Schichten werden im weiteren von unten nach oben behandelt.**

11.2.2 Unterbrechungsbehandlung

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Technisch: vgl. Vorlesungsteil Rechnerarchitektur
- Unterbrechungsbehandler (Interrupt Handler) sollten in einer möglichst niedrigen Schicht des Betriebssystems verborgen werden
- Gute Möglichkeit der Einkapselung:
  - Jeder Prozess, der I/O-Operation ausführt, blockiert bis zu deren Abschluss, z.B. durch:
    - P (Semaphor), oder
    - Wait (Condition), oder
    - Receive (Message).
  - Tritt eine Unterbrechung auf, die einer Fertigmeldung der I/O-Operation entspricht, entblockiert der Interrupt Handler den blockierten Prozess entsprechend:
    - V (Semaphor), oder
    - Signal (Condition), oder
    - Send (Message).

11.2.3 Gerätetreiber (Device Drivers)

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Gerätetreiber kapselt den gesamten geräteabhängigen Code für einen bestimmten Gerätetyp, d.h.:
  - Nur Gerätetreiber kennen Struktur der Controller-Register, setzen Kommandos in ihnen ab und überprüfen die korrekte Ausführung
  - z.B. nur der Plattentreiber muss sich um Sektoren, Spuren, Zylinder, Köpfe, Armbewegungen, Motore, Interleaving usw. kümmern
- Aufgabe eines Treibers ist das Bearbeiten von abstrakten Aufträgen der geräteunabhängigen Software-Schicht des Betriebssystem-Kerns (z.B. Lies Block n)
- Gerätetreiber bearbeitet i.d.R. mehrere Geräte eines Typs
- Gegebener abstrakter Auftrag muss durch den Treiber in eine konkrete Folge von Kommandos an einen Controller umgesetzt werden

Gerätetreiber (2)

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Typische Komponenten eines Gerätetreibers (z.B. UNIX):
  - Autokonfigurations- & Initialisierungsroutinen:
    Einmalig benutzt während des Boot-Vorgangs zur Überprüfung der Existenz von Geräten eines bestimmten Typs sowie deren Initialisierung
  - I/O-Auftragsbearbeitungsroutinen
    (obere Hälfte des Treibers):

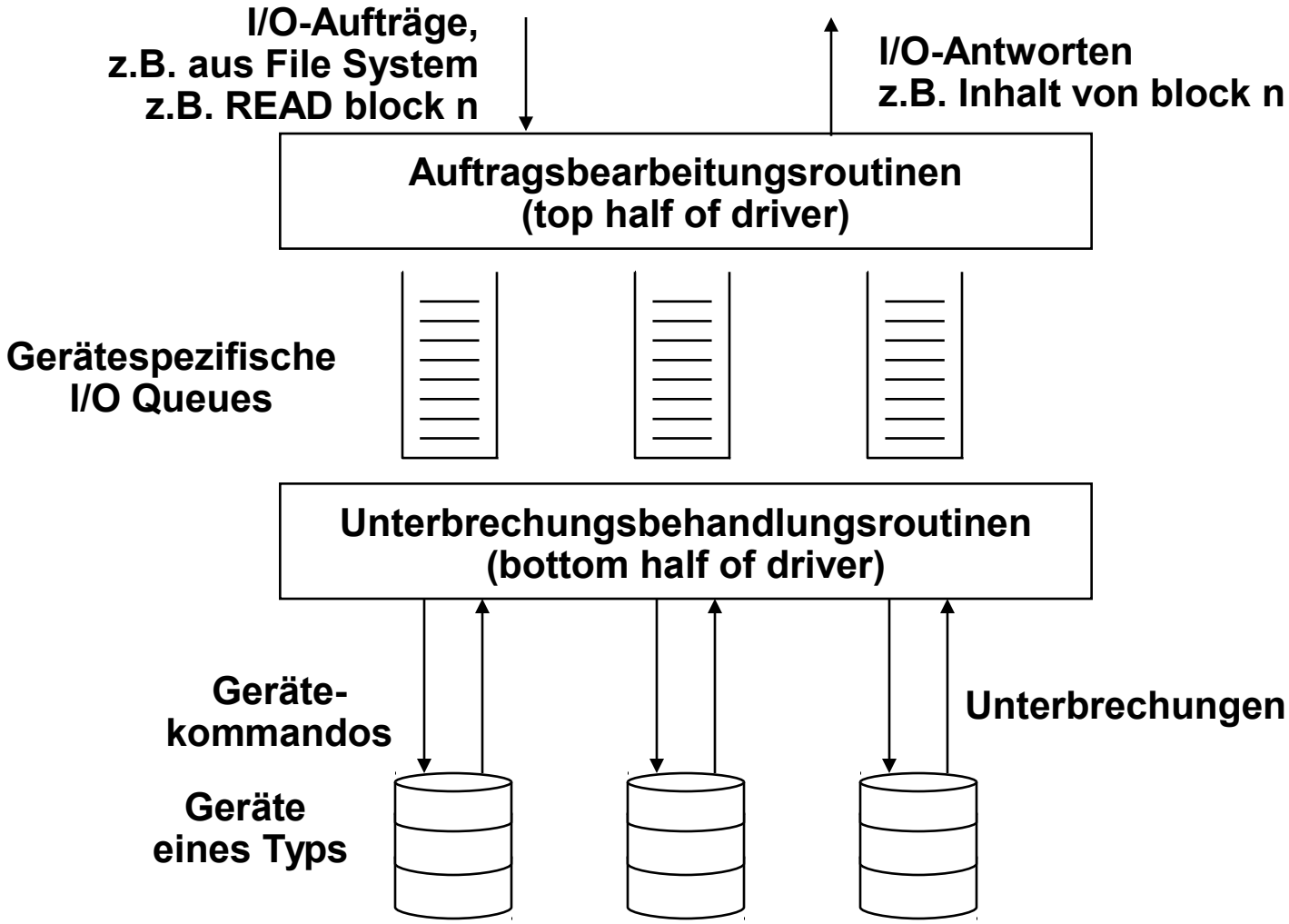
    I/O-Auftrag kann durch Ausführung eines System Calls eines Applikationsprogramms oder durch das virtuelle Speichersystem innerhalb des Betriebssystems entstehen

    Treibercode ist i.d.R. synchron zur aufrufenden Seite (Auftraggeberseite) hin, d.h. Aufrufer können im Treiber blockieren
  - Unterbrechungsbehandlungsroutinen
    (untere Hälfte des Treibers)

    Treibercode ist i.d.R. asynchron zur Geräteseite hin
  - Dazwischen liegen gerätespezifische Warteschlangen
    für I/O-Kommandos an die vom Treiber verwalteten Geräte

Gerätetreiber (3)

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim



11.2.4 Geräteunabh. SW des BS-Kerns

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Aufgaben:
  - Bereitstellung von Funktionen, die unabhängig von bestimmten I/O-Gerätetypen sind
  - Bereitstellung einer einheitlichen Schnittstelle zur I/O-Software in den Benutzerprozessen
- Typische Funktionen:
  - Bereitstellung einer einheitlichen Schnittstelle zur Einbindung von Gerätetreibern
  - Benennung von Geräten
  - Zugriffsschutz für Geräte
  - Bereitstellung von Blöcken mit einer festen geräteunabhängigen Blockgröße
  - Pufferung
  - Speicherverwaltung auf blockorientierten Geräten
  - Zuteilung und Freigabe von Geräten
  - Fehlermeldung

11.2.5 I/O-Software im Benutzermodus

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- I/O-Bibliotheken
  - Funktionen, die zu Anwendungsprogrammen hinzugebunden werden
  - Beispiele: `write` und `printf` im C-Programm, die mit wenig bzw. mehr Aufwand zu einem WRITE system call führen
- Spooling-System
  - Verwaltung von Druckaufträgen für eine Menge von Druckern
  - realisiert durch einen speziellen Dämon-Prozess und ein spezielles (Spooling)-Verzeichnis
- E-mail-System
  - analog
- ...

11.3 Plattentreiber

- Plattentreiber
  - sind Treiber für Magnetplatten
  - besitzen alle unter 11.2.3 besprochenen Eigenschaften
- Im folgenden Plattenarm-Scheduling-Verfahren
  - in Plattentreibern bei der Auswahl des nächsten durchzuführenden Auftrags eines Plattenlaufwerks eingesetzt
- Ausgangspunkt: Zugriffszeit zu einem Plattenblock ist bestimmt durch:
  - Positionierzeit des Plattenarms (Seek Time)
  - Latenzzeit (bis Block unter Schreib-/Lesekopf, im Mittel 1/2 Rotationszeit)
  - Transferzeit (Übertragung des Blocks)

Positionierzeit hat i.d.R. einen hohen Anteil (vgl. 9.3.3.2)

⇒ Leistungssteigerung kann also durch Reduktion der mittleren Positionierzeit erreicht werden.

Plattentreiber (2)

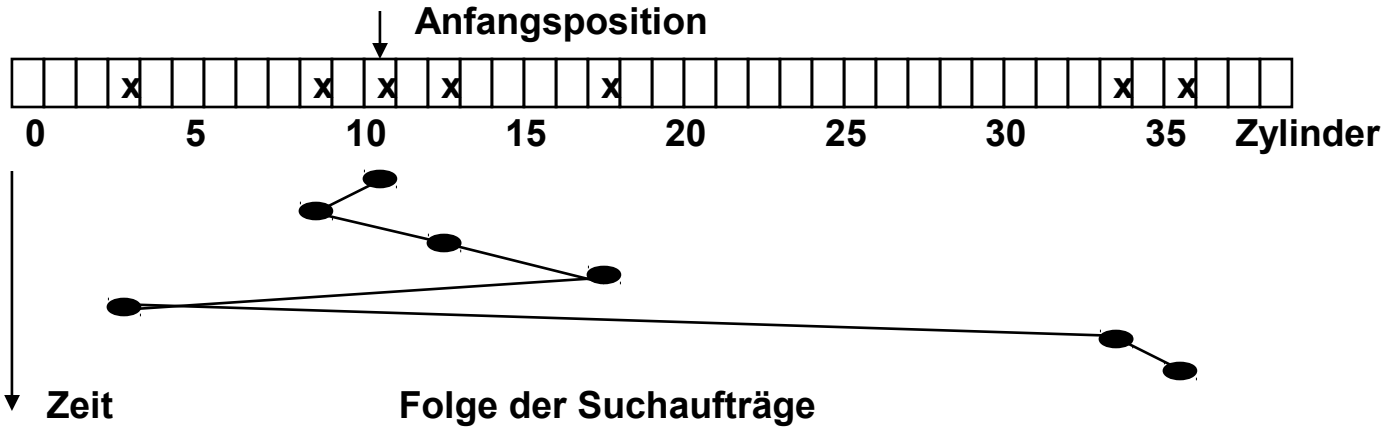
Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Algorithmen, die für eine Menge von Platten-I/O-Aufträgen die mittlere Positionierzeit gegenüber FCFS verringern:
  - Shortest-Seek-First (SSF)
  - Fahrstuhl-Algorithmus

Shortest-Seek-First (SSF)

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Bei der Auswahl des nächsten I/O-Auftrags wird derjenige gewählt, der "am nächsten" ist, d.h. die geringsten Kopfbewegungen erfordert
- Beispiel:

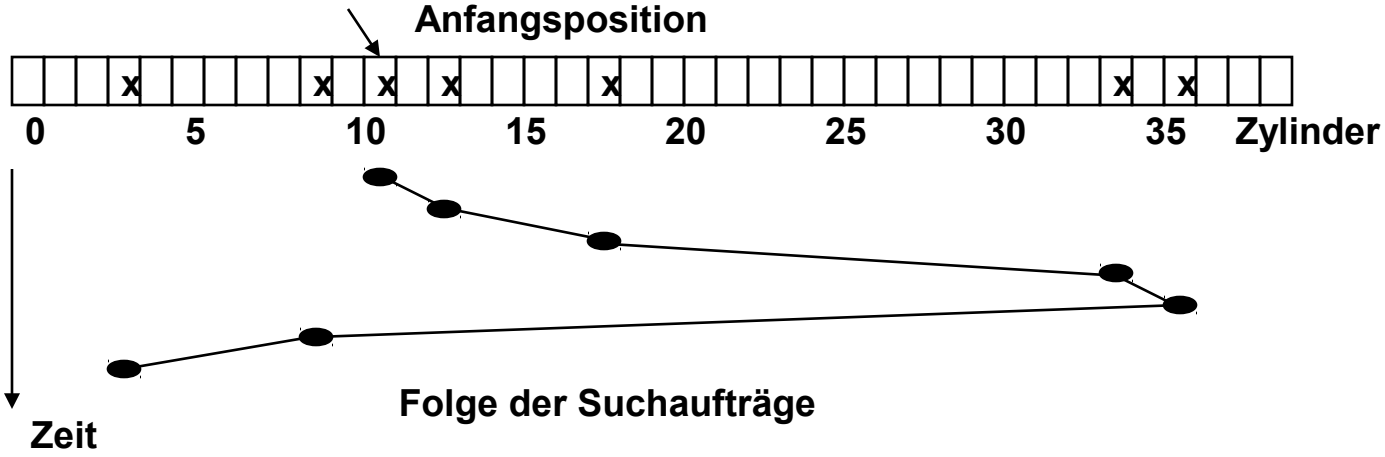


- SSF reduziert die Positionierzeiten gegenüber FCFS deutlich
- Aber: Shortest-Seek-First ist unfair:
  - Bei stark belasteter Platte tendiert der Arm dazu, sich in der Mitte der Platte aufzuhalten. Anforderungen bzgl. äußerer und innerer Zylinder werden nachteilig bedient.

Fahrstuhl-Algorithmus (Elevator Seek)

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Wie bei einem Aufzug bewegt sich der Plattenarm solange in eine Richtung, bis es in dieser Richtung keine Plattenaufträge mehr gibt, und wechselt dann die Richtung
- Beispiel:

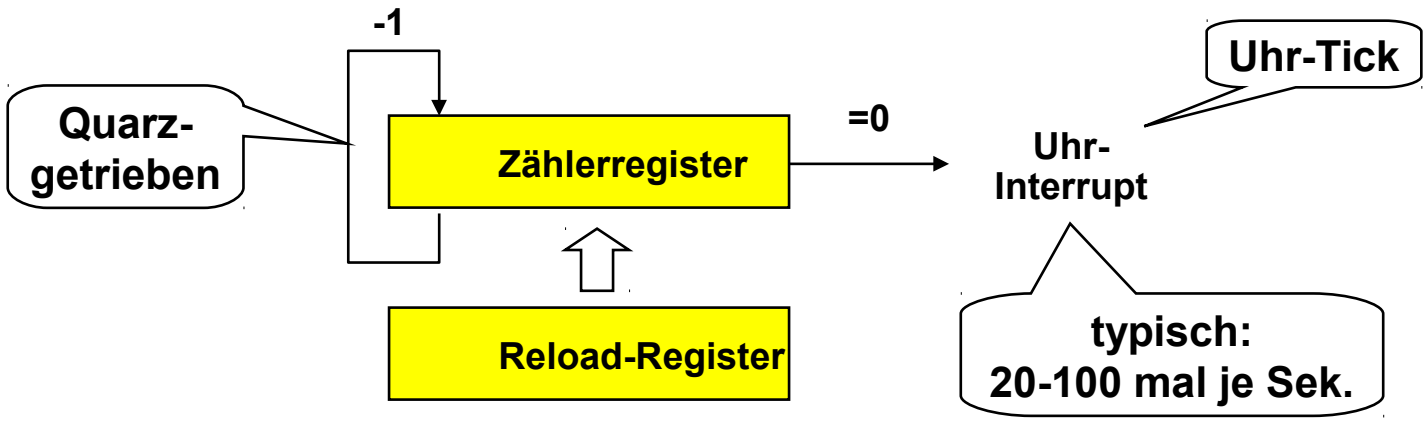


- Für jede gegebene Menge von Aufträgen ist die obere Grenze der Summe der Armbewegungen bekannt, nämlich zweimal die Anzahl der vorhandenen Zylinder

11.4 Uhrtreiber

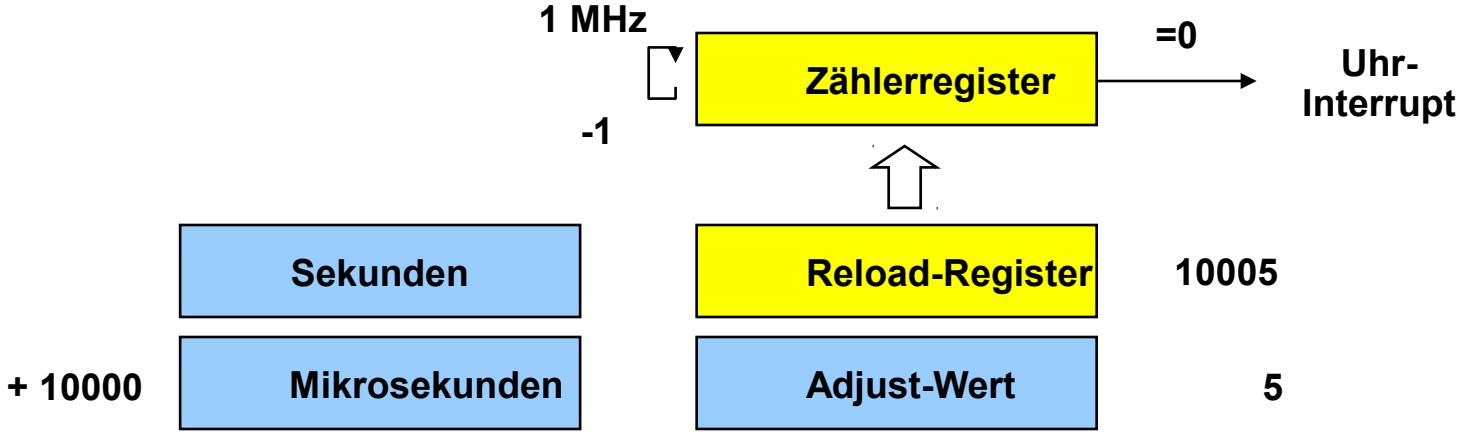
Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Hardware eines Timers besteht i.d.R. aus:
  - einem Oszillator fester Frequenz
  - einem Zähler
    - bei jeder Schwingung des Oszillators dekrementiert
    - bei Erreichen des Werts Null eine Unterbrechung erzeugt
  - einem Reload-Register
    - ladbarer Wert initialisiert automatisch den Zähler neu, wenn dieser den Wert Null erreicht
- Uhrtreiber erbringt im Falle eines Uhrticks durchzuführende Funktionen



Aufgaben des Uhrtreibers

- Verwalten der Uhrzeit (Time-of-Day, Wall Time):
  - Variable des Betriebssystemkerns
  - Inkrementieren bei jedem Uhrtick
  - Beispiel UNIX
    - zwei 32-Bit (oder 64-Bit) Integer-Variablen
      - Anzahl Sekunden seit 1.1.1970
      - Anzahl µs (oder ns) in der aktuellen Sekunde
    - typ. 100 Interrupts/s
    - bei Interrupt werden Variablen um nominelle Anzahl µs erhöht
    - Korrekturwert (Adjust-Wert) für Ausgleich der Drift des Quartzes
    - Systemdienste `settimeofday`, `adjtime`



Aufgaben des Uhrtreibers

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Quantum-Überwachung:
  - Bei Prozesswechsel wird ein Zähler mit der Dauer des dem Prozess zugewiesenen Quantums, gemessen in Uhrticks, initialisiert
  - Bei jedem Uhrtick wird der Zähler dekrementiert
  - Wird der Wert Null erreicht, ruft der Uhrtreiber den Scheduler zur Neuvergabe des Prozessors auf

Aufgaben des Uhrtreibers (2)

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Alarm-Systemdienst (Interval Timer):
  - Ziel: Beauftragung des Betriebssystems, nach einer gewählten Zeitspanne einen Alarm (ein Signal, eine Nachricht o.ä.) zuzustellen.
  - Beispiel UNIX: `setitimer` system call
    - erlaubt Einplanen von SIG(VT)ALRM-Signalen nach Ablauf einer bestimmten Zeitspanne der realen Zeit oder der virtuellen Zeit des Prozesses, die nur fortschreitet, wenn der Prozess rechnend ist
    - anwendungsspezifische Reaktion kann mittels `sigaction` zugeordnet werden
  - Interval Timer verwaltet i.d.R. eine nach den Zeitpunkten geordnete Liste der Zeit-Alarme.

Aufgaben des Uhrtreibers (3)

- Watchdog-Funktion (Watchdog Timer):
  - Ziel: Zeitüberwachung Betriebssystem-interner Vorgänge (Timeouts)
  - Vorgehensweise analog

- Accounting:
  - Ziel: "Faire" Abrechung der CPU-Nutzung durch einen Prozess
  - Bearbeitung von Unterbrechungen wird i.d.R. dem gerade betroffenen Prozess zugeordnet (nicht dem Verursacher, Einfachheit)
  - Bei jedem Uhrtick wird ein CPU-Nutzungszähler im Prozesskontrollblock des sich gerade in Ausführung befindlichen Prozesses inkrementiert

Aufgaben des Uhrtreibers (4)

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Profiling:
  - Ziel: Ermittlung statistischer Aussagen über den Zeitverbrauch bestimmter Programmteile (z.B. Erwartungs-werte für die Ausführungsdauern von Prozeduren).
  - Dazu: Erstellen einer Häufigkeitsverteilung über das Antreffen des Programmzählers (Program Counters) in bestimmten Adressbereichen.

    Beispiel: UNIX `profil` System Call
  - Mittels Analyse des Stacks (Call Graph) und Symbol Table des Programms sowie Anwendung des "Gesetzes der Großen Zahl" werden die gewünschten Erwartungswerte bestimmt
  - Bei jedem Uhrtick überprüft der Uhrtreiber, ob Profiling für den rechnenden Prozess eingeschaltet ist. Wenn ja, wird bestimmt, in welche Klasse der Häufigkeitsverteilung der aktuelle Programmzählerwert fällt, und der Zähler dieser Klasse wird inkrementiert.

11.5 Terminal-Treiber

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Typen von Terminals:
  - Serielle Terminals bestehen aus Tastatur und Bildschirm sowie einer seriellen Schnittstelle (z.B. RS 232) zur zeichen-orientierten Kommunikation mit einem Rechner (Host)
  - Memory-Mapped Terminals
    - im phys. Adressraum sogenanntes Video-RAM vorhanden, das mittels üblicher Maschinenbefehle oder durch Befehle eines HW-Beschleunigers beschrieben werden kann
    - Video-Controller liest zyklisch den Inhalt des Speichers und erzeugt daraus die Video-Signale für den Bildschirm
    - Tastatur und Maus werden als separate Geräte mit eigenem Treiber eingesetzt

    Unterscheidung:
    - Zeichenorientierte Geräte
      (z.B. der ursprüngliche IBM-PC, heute unüblich)
    - Bitmap-Terminals erlauben Pixel-Verarbeitung (heute Standard)
  - X-Terminals sind dagegen keine Terminals, sondern eigen-ständige Rechner, die mittels des X-Protokolls mit einem Host kommunizieren, um entfernte graph. Ausgabe zu bewirken

Tastatur-Treiber

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

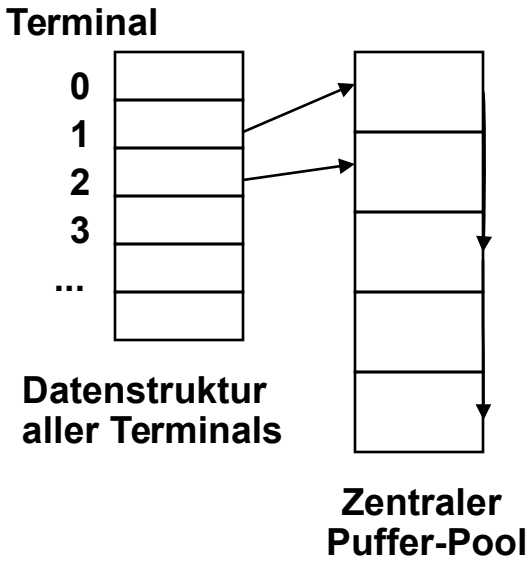
- Eigenschaften:
  - Aufgabe: Sammeln von Eingaben der Tastatur und Übergabe an Benutzerprogramme, wenn diese vom Terminal lesen
  - I.d.R. bewirkt jeder Tastendruck eine Unterbrechung
  - Falls Tastennummern an den Treiber übergeben werden, erfolgt zunächst eine Umwandlung anhand einer Code-Tabelle z.B. in ASCII Code
  - Raw Mode-Betrieb:
    - Zeichenorientiert: Die Zeichen werden ihrer exakten Folge unverarbeitet weitergegeben (z.B. für Editoren sinnvoll)
  - Cooked Mode-Betrieb:
    - Zeilenorientiert: Treiber übernimmt das Editieren innerhalb einer Zeile (z.B. Backspace-Bearbeitung)
    - übergibt nur bearbeitete Zeilen an Anwendung (z.B. Shell)
    - Zeichen müssen gespeichert werden, bis Zeile vollständig
  - Modus kann i.d.R. mittels Systemaufruf gewählt werden

    Beispiel: UNIX `ioctl` System Call

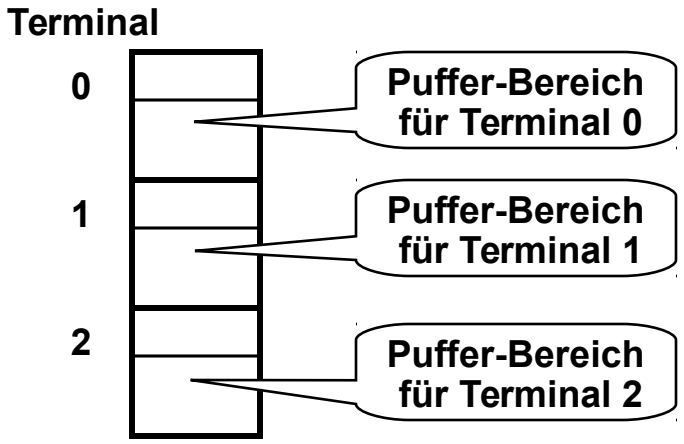
Tastatur-Treiber (2)

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Verbreitete Ansätze zur Pufferung von Zeichen:
  - Zentraler Puffer-Pool für alle Terminals mit Verkettung von Puffern fester Länge (z.B. 64 Zeichen) für jedes Terminal (a)
  - Privater Pufferbereich für jedes Terminal (b)



(a)



(b)

Tastatur-Treiber (3)

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Problembereiche:
  - Tastatureingaben
    - erzielen keine automatische Ausgabe (z.B. bei Passwort-Eingabe wichtig!).
    - Ausgabe von eingebenen Zeichen erfolgt durch Software und wird Echoing genannt
  - Behandlung von Tabulatoren
  - Zeilenfortschaltung mittels Wagenrücklauf und/oder Zeilenvorschub
  - Behandlung von Sonderzeichen im Cooked Modus (z.B. Löschzeichen, Escape-Zeichen, Dateiende)

Bildschirm-Treiber

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Eigenschaften:
  - Terminal-Ausgabe i.d.R. einfacher als Behandlung von Eingabe.
  - Treiber für serielle Terminals und Treiber für Memory-Mapped Terminals sind dabei sehr unterschiedlich.
  - Serielles Terminal
    - Zuordnung eines Puffers (u.U. aus demselben Puffer Pool wie die für Tastatur-Treiber benutzten).
    - Ausgabe: Daten werden in einen oder mehrere Puffer kopiert
    - Treiber führt I/O zeichenweise über die serielle Schnittstelle mit zwischenzeitlichem Warten auf Unterbrechung durch
  - Memory-Mapped zeichenorientiertes Terminal
    - Ausgabe unmittelbar in Video-RAM
    - Notwendig: Verwaltung einer "aktuellen Position"
    - Behandlung von Sonderzeichen (wie z.B. Backspace)
    - Besondere Beachtung auch für Zeilenvorschub am Seitenende sowie die Cursor-Positionierung

11.6 Beispiel: Das UNIX I/O-System

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- BSD UNIX I/O-System besitzt alle in 11.2 beschriebenen Komponenten und prinzipiellen Eigenschaften.
- In jüngerer Zeit dynamisch ladbare Gerätetreiber
- Geräte haben Repräsentierung als Inode im Dateisystem mit Typ "special file".
- UNIX unterscheidet
  - blockorientierte Spezialdateien (Block Devices, z.B. Platten)
  - zeichenorientierte Spezialdateien
    (Character Devices, z.B. Terminals (ttys))
  - jeweils festgelegte Schnittstelle für Gerätetreiber dieser Klassen
- Inode einer Spezialdatei enthält
  - Major und Minor Device Number des Gerätes
  - Major Device Number selektiert den Gerätetyp und damit den zuständigen Treiber
  - Minor Device Number selektiert ein logisches Gerät
    (z.B. Platten-Partition oder bestimmtes Terminal)

UNIX I/O-System (2)

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Für alle blockorientierten Geräte gilt:
  - Jeder Typ besitzt einen Eintrag in einer Tabelle (4.3BSD `bdevsw` block device switch).
  - Eintrag entspricht Treiber und enthält Adressen (Einsprungstellen) der Operationen des Treibers
  - Major Device Number eines Gerätes selektiert Eintrag
  - Jeder blockorientierte Treiber muss folgende Operationen bereitstellen:
    - open
    - strategy (Initiieren von read oder write)
    - close
    - dump (Hauptspeicherabzug bei Crash)
    - psize (Größe einer Partition in Blöcken)
  - Gemeinsame Pufferverwaltung für alle blockorientierten Geräte, den Block Buffer Cache (vgl. 9.3.6). Er reduziert die Anzahl der notwendigen Plattenzugriffe. Die Pufferverwaltung erfolgt gemäß LRU.

UNIX I/O-System (3)

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Für alle zeichenorientierten Geräte gilt:
  - Jeder Typ besitzt einen Eintrag in einer Tabelle (4.3BSD `cdevsw` character device switch)
  - Character Device Typen werden auch verwendet, um "Raw Access" zu blockorientierten Geräten zu erhalten, d.h. unter Umgehung des Block Buffer Cache (z.B. für fsck, dump)
  - Jeder zeichenorientierte Treiber muss folgende Operationen bereitstellen:
    - open
    - close
    - ioctl (I/O control operation)
    - mmap (map device contents into memory)
    - read
    - reset
    - select (poll device for I/O readiness)
    - stop
    - write

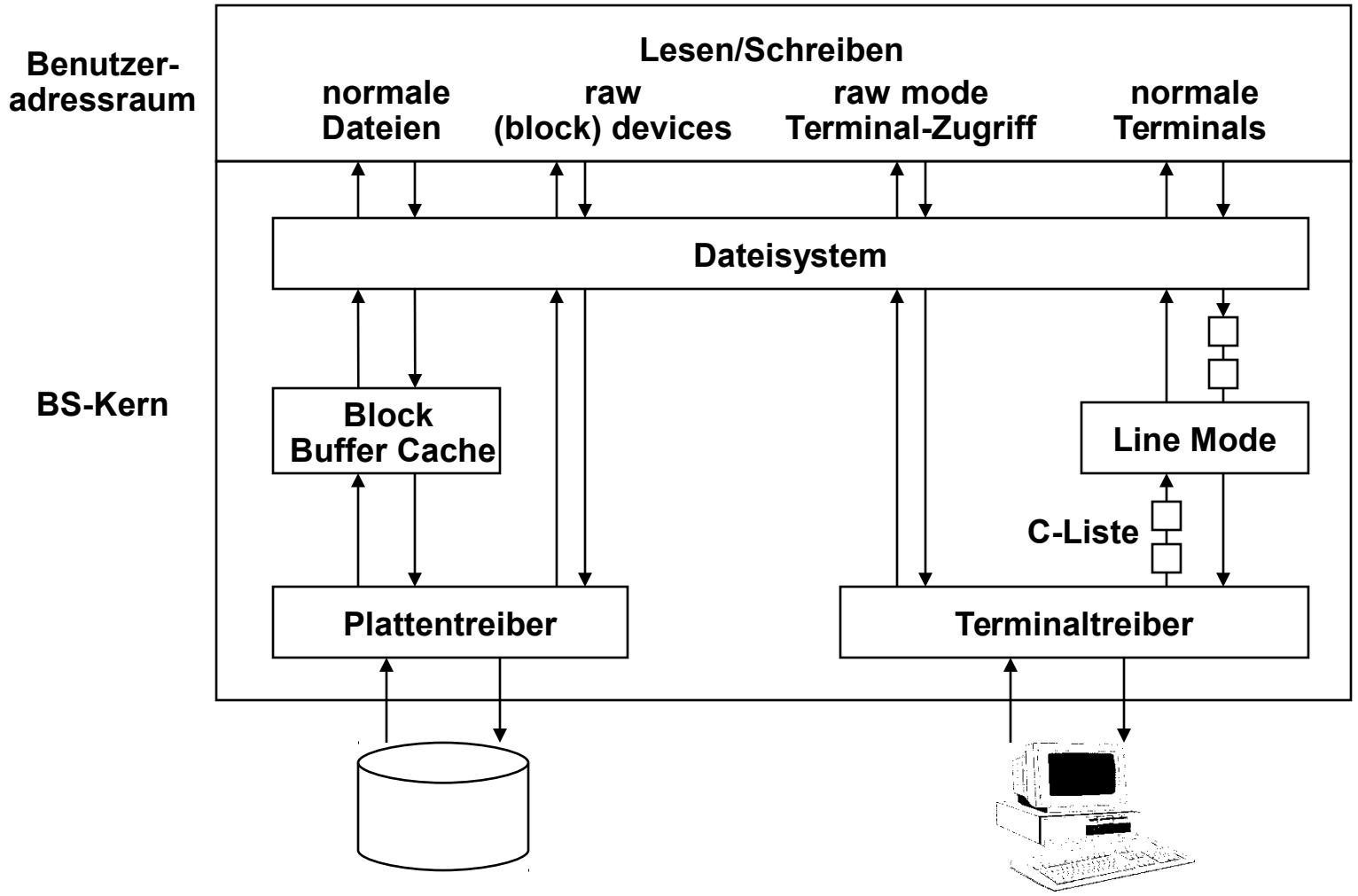
UNIX I/O-System (4)

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- Gemeinsame Pufferverwaltung ohne Caching (unnötig) für alle wirklichen zeichenorientierten Geräte, die sogenannten C-Listen, je Zeichenstrom bestehend aus verketteten kleinen Blocken mit bis zu 64 Zeichen.
- Ein Zeichenstrom kann im Treiber durch eine sogenannte Line Discipline vorverarbeitet werden (Cooked Character Stream) oder diese umgehen (Raw Character Mode).

UNIX I/O-System (Zusammenfassung)

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim



11.7 Zusammenfassung

Hochschule RheinMain
University of Applied Sciences
Wiesbaden Rüsselsheim

- I/O ist ein wichtiges, aber aufgrund der vielen spezifischen Eigenarten oft vernachlässigtes Thema
- Anteil des Betriebssystem-Codes für Ein-/Ausgabe ist beträchtlich
- In 11.1 wurden einige Eigenschaften von I/O-Hardware stichwortartig zusammengestellt
- In 11.2 wurden die vier Software-Schichten zur Ein-/Ausgabe allgemein besprochen
  - Unterbrechungsbehandlung
  - Gerätetreiber
  - geräteunabhängige Software im BS-Kern
  - I/O-Software auf Benutzerebene, wie z.B. I/O-Bibliotheken und Spooling-Dämonen
- In 11.3 bis 11.5 wurden einige Eigenschaften von Treibern für Platten (speziell Plattenarm-Scheduling-Verfahren) sowie für Uhren und Terminals besprochen
- In 11.6 wurde schließlich die Struktur der BSD UNIX I/O-Software als Beispiel behandelt