Betriebssysteme SS2024

Geräteverwaltung

Tutorium 11

Inhaltsverzeichnis

Treiber

- Gerätetreiber implementieren die Schnittstelle zu dedizierten Geräten
- Ermöglichen einheitliche Operationen für den Gerätezugriff
- Übernehmen (unter anderem) folgende Aufgaben:
  - Verwaltung von Zugriffsrechten
  - Optimierung der Gerätenutzung
  - Zuteilung von Ressourcen
  - Auflösen von Zugriffskonflikten
  - Datentransport vom und zum Gerät

Treiberimplementierung

Problem: Ein-/Ausgabe Geräte haben hohe Latenzzeiten

Mögliche Betriebsmodi für Gerätetreiber:

- Polling-Betrieb
  - Der Prozessor erkundigt sich ob Teile der angeforderten Daten vorliegen
  - *Aktives Warten*, falls noch keine Daten vorliegen
    $\rightarrow$ Ineffizient bei größeren Datenmengen
- Unterbrechungsbetrieb
  - Ein angefragtes Gerät löst eine Unterbrechung aus, sobald ein Teilauftrag vollständig abgearbeitet wurde
  - Der Prozess der die Daten angefordert hat kann nun blockiert werden

Disk Scheduling Strategien

IO Optimierung, wegen vieler Einschränkungen

- FCFS: First Come First Serve
- SSTF: Shortest Seek-Time First
- SCAN: Fahrstuhl Algorithmus
- C-SCAN: Circular SCAN

Die Anzahl der Spurwechsel ist interessant, weil dies ein Anhaltspunkt für Gesamtdauer des Speicherzugriffs ist

FCFS

- Neue Anfragen werden am Ende der Liste eingereiht
- Gehe zum nächsten Auftrag in der Liste
- Quasi keine Optimierung

Beispiel:
Position: 4, Richtung: Unten, Aufträge: $\{2, 9, 5, 1, 8\}$

FCFS

- Neue Anfragen werden am Ende der Liste eingereiht
- Gehe zum nächsten Auftrag in der Liste
- Quasi keine Optimierung

Beispiel:
Position: 4, Richtung: Unten, Aufträge: $\{2, 9, 5, 1, 8\}$

Lösung:
Reihenfolge: 2-9-5-1-8, Anzahl Spurwechsel: 24

SSTF

- Der am nähesten gelegene Auftrag wird ausgewählt, Heuristik bei Gleichstand vorhanden
- Kann “schwingen” und zur Aushungerung führen
- Besonders problematisch wenn regelmäßig neue Aufträge dazukommen

Beispiel:
Position: 4, Heuristik: Unten, Aufträge: $\{2, 9, 5, 1, 8\}$

SSTF

- Der am nähesten gelegene Auftrag wird ausgewählt, Heuristik bei Gleichstand vorhanden
- Kann “schwingen” und zur Aushungerung führen
- Besonders problematisch wenn regelmäßig neue Aufträge dazukommen

Beispiel:
Position: 4, Heuristik: Unten, Aufträge: $\{2, 9, 5, 1, 8\}$

Lösung:
Reihenfolge: 5-2-1-8-9, Anzahl Spurwechsel: 13

SCAN

Aufzug Algorithmus

- Folge Aufträgen in eine Richtung
- Bei unterstem/obersten Auftrag, erfolgt ein Richtungswechsel

Beispiel:
Position: 4, Richtung: Unten, Aufträge: $\{2, 9, 5, 1, 8\}$

SCAN

Aufzug Algorithmus

- Folge Aufträgen in eine Richtung
- Bei unterstem/obersten Auftrag, erfolgt ein Richtungswechsel

Beispiel:
Position: 4, Richtung: Unten, Aufträge: $\{2, 9, 5, 1, 8\}$

Lösung:
Reihenfolge: 2-1-5-8-9, Anzahl Spurwechsel: 11

C-SCAN

Circulärer SCAN

- Immer nur eine Richtung (nach unten)
- Wenn der unterste Auftrag erreicht wurde, springe zum höchsten Auftrag
- Vorteilhaft falls ein Ende-zu-Ende Sprung günstig ist

Beispiel:
Position: 4, Richtung: Unten, Aufträge: $\{2, 9, 5, 1, 8\}$

C-SCAN

Circulärer SCAN

- Immer nur eine Richtung (nach unten)
- Wenn der unterste Auftrag erreicht wurde, springe zum höchsten Auftrag
- Vorteilhaft falls ein Ende-zu-Ende Sprung günstig ist

Beispiel:
Position: 4, Richtung: Unten, Aufträge: $\{2, 9, 5, 1, 8\}$

Lösung:
Reihenfolge: 2-1-9-8-5, Anzahl Spurwechsel: 15

Beispielaufgaben

- Position: 32
- Richtung: Unten
- Aufträge: { 72, 28, 70, 9, 91, 5, 34, 56, 11, 94 }

	1	2	3	4	5	6	7	8	9	10	$\sum$
FCFS											
SSTF											
SCAN											
C-SCAN											

Lösung

- Position: 32
- Richtung: Unten
- Aufträge: { 72, 28, 70, 9, 91, 5, 34, 56, 11, 94 }

	1	2	3	4	5	6	7	8	9	10	$\sum$
FCFS	72	28	70	9	91	5	34	56	11	94	534
SSTF	34	28	11	9	5	56	70	72	91	94	120
SCAN	28	11	9	5	34	56	70	72	91	94	116
C-SCAN	28	11	9	5	94	91	72	70	56	34	176

Direct Memory Access (DMA)

- Einführung eines speziellen Eingabe/Ausgabe-Bausteins, welcher auf dem Speicherbus agiert um den Prozessor zu entlasten
- Ablauf einer Anfrage:
  - I/O-Auftrag wird angestoßen
  - Gerät gibt Resultat an den DMA-Baustein, welcher dieses direkt in den Hauptspeicher schreibt
  - Prozessor kann nun auf die Resultate zugreifen

DMA-Betriebsmodi

- Der Prozesser und der DMA-Baustein müssen sich den Speicherbus teilen
- Umsetzung durch verschiedene Betriebsmodi:
  - Im **Burst** Modus übernimmt der DMA-Baustein die Kontrolle über den Speicherbus für die Dauer des vollständigen Transfers
  - Im **Cycle Stealing** Modus teilen sich die CPU und der DMA-Baustein die Buszyklen mit einer festen Aufteilung

Kombination von DMA und Caching

- Moderne Prozessoren haben mehrere Cache-Level
- Problem: DMA läuft an den Caches vorbei
- Das Betriebssystem muss vor einem DMA-Transfer dafür sorgen, dass keine Inkonsistenzen auftreten
  - Zurückschreiben von gecachten Werten
  - Invalidieren der gecachten Werte

I/O-Geräte unter UNIX

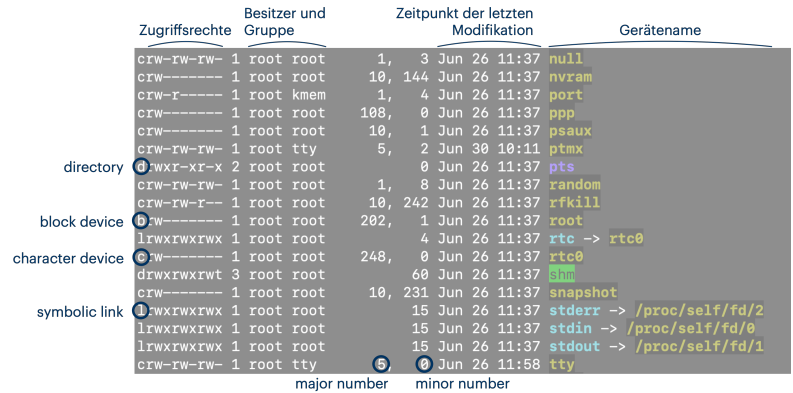
- Geräte werden unter UNIX Betriebssystemen durch spezielle Gerätedateien repräsentiert welche im Verzeichnis `/dev` zu finden sind
- Unterscheidung zwischen zwei Arten von Geräten:
  - *blockorientierte Geräte:*
    z.B. Festplatten oder USB-Sticks
  - *zeichenorientierte Geräte:*
    z.B. Eingabegeräte, Audiokanäle, ...
- Die Gerätedateien können mit Hilfe von Systemaufrufen beschrieben und gelesen werden

Beispiel - Gerätedateien unter Linux

```
crw-rw-rw- 1 root root      1,   3 Jun 26 11:37 null
crw------- 1 root root     10, 144 Jun 26 11:37 nvram
crw-r----- 1 root kmem      1,   4 Jun 26 11:37 port
crw------- 1 root root    108,   0 Jun 26 11:37 ppp
crw------- 1 root root     10,   1 Jun 26 11:37 psaux
crw-rw-rw- 1 root tty       5,   2 Jun 30 10:11 ptmx
drwxr-xr-x 2 root root            0 Jun 26 11:37 pts
crw-rw-rw- 1 root root      1,   8 Jun 26 11:37 random
crw-rw-r-- 1 root root     10, 242 Jun 26 11:37 rfkill
brw------- 1 root root    202,   1 Jun 26 11:37 root
lrwxrwxrwx 1 root root            4 Jun 26 11:37 rtc -> rtc0
crw------- 1 root root    248,   0 Jun 26 11:37 rtc0
drwxrwxrwt 3 root root           60 Jun 26 11:37 shm
crw------- 1 root root     10, 231 Jun 26 11:37 snapshot
lrwxrwxrwx 1 root root           15 Jun 26 11:37 stderr -> /proc/self/fd/2
lrwxrwxrwx 1 root root           15 Jun 26 11:37 stdin -> /proc/self/fd/0
lrwxrwxrwx 1 root root           15 Jun 26 11:37 stdout -> /proc/self/fd/1
crw-rw-rw- 1 root tty       5,   0 Jun 26 11:58 tty
```

- Ausschnitt des Listing des Verzeichnisses `/dev` unter Ubuntu 22.04
- Abrufbar mit Hilfe des Befehls `ls -l /dev`

Beispiel - Gerätedateien unter Linux



- Ausschnitt des Listing des Verzeichnisses `/dev` unter Ubuntu 22.04
- Abrufbar mit Hilfe des Befehls `ls -l /dev`