

5 Eingabe und Ausgabe

- 1 Einführung
- 2 Prozesse und Threads
- 3 Speicherverwaltung
- 4 Dateisysteme
- 5 *Eingabe und Ausgabe***
- 6 Deadlocks
- 7 Virtualisierung und die Cloud
- 8 Multiprozessorsysteme
- 9 IT-Sicherheit
- 10 Fallstudie 1: Linux
- 11 Fallstudie 2: Windows
- 12 Entwurf von Betriebssystemen

5 Eingabe und Ausgabe

- 5.1 Grundlagen der Ein-/Ausgabehardware
- 5.2 Grundlagen der Ein-/Ausgabesoftware
- 5.3 Schichten der Ein-/Ausgabesoftware
- 5.4 Plattenspeicher
- 5.5 Uhren
- 5.6 Benutzerschnittstellen: Tastatur, Maus, Bildschirm
- 5.7 Thin Clients
- 5.8 Energieverwaltung

5.1 Grundlagen der Ein-/Ausgabehardware

5.1.1 Ein-/Ausgabegeräte

5.1.2 Controller

5.1.3 Memory-Mapped-Ein-/Ausgabe

5.1.4 Interruptgesteuerte Ein-/Ausgabe

5.1.5 Ein-/Ausgabe mit DMA

Ein-/Ausgabegeräte (1)

Blockorientierte Geräte:

- Speichern Informationen in Blöcken fester Größe
- Transfers erfolgen in Einheiten von ganzen Blöcken

Zeichenorientierte Geräte:

- Liefern oder akzeptieren einen Zeichenstrom, ohne Berücksichtigung der Blockstruktur
- Nicht adressierbar, keine Suchoperation verfügbar

Ein-/Ausgabegeräte (2)

Tanenbaum, A. S.; Bos, H.: Moderne Betriebssysteme. Pearson Studium 2016

Gerät	Datenrate
Tastatur	10 Byte/s
Maus	100 Byte/s
56-KB-Modem	7 KB/s
Scanner mit 300 dpi	1 MB/s
Digitaler Camcorder	3,5 MB/s
4x Blu-ray-Disk	18 MB/s
WLAN nach 802.11n	37,5 MB/s
USB 2.0	60 MB/s
FireWire 800	100 MB/s
Gigabit Ethernet	125 MB/s
SATA-3-Plattenlaufwerk	600 MB/s
USB 3.0	625 MB/s
Ultra-5-SCSI-Bus	640 MB/s
Single-Lane PCIe-3.0-Bus	985 MB/s
Thunderbolt-2-Bus	2,5 GB/s
SONET OC-768	5 GB/s

Abbildung 5.1: Einige typische Datenraten von E/A-Geräten, Netzwerken und Bussystemen.

Controller

Controller (engl. Steuergerät oder Steuereinheit):

- elektronische Einheiten der Computer-Hardware steuern bestimmte Vorgänge
- Mikrocontroller bestehen nur aus einem integrierten Schaltkreis

Typische Aufgaben:

- Verarbeitung von Unterbrechungsanforderungen (IRQs)
- Verarbeitung der Tastatureingaben
- Steuerung von Zugriffen auf die Festplatte
- u.a.

Memory-Mapped- Ein-/Ausgabe (1)

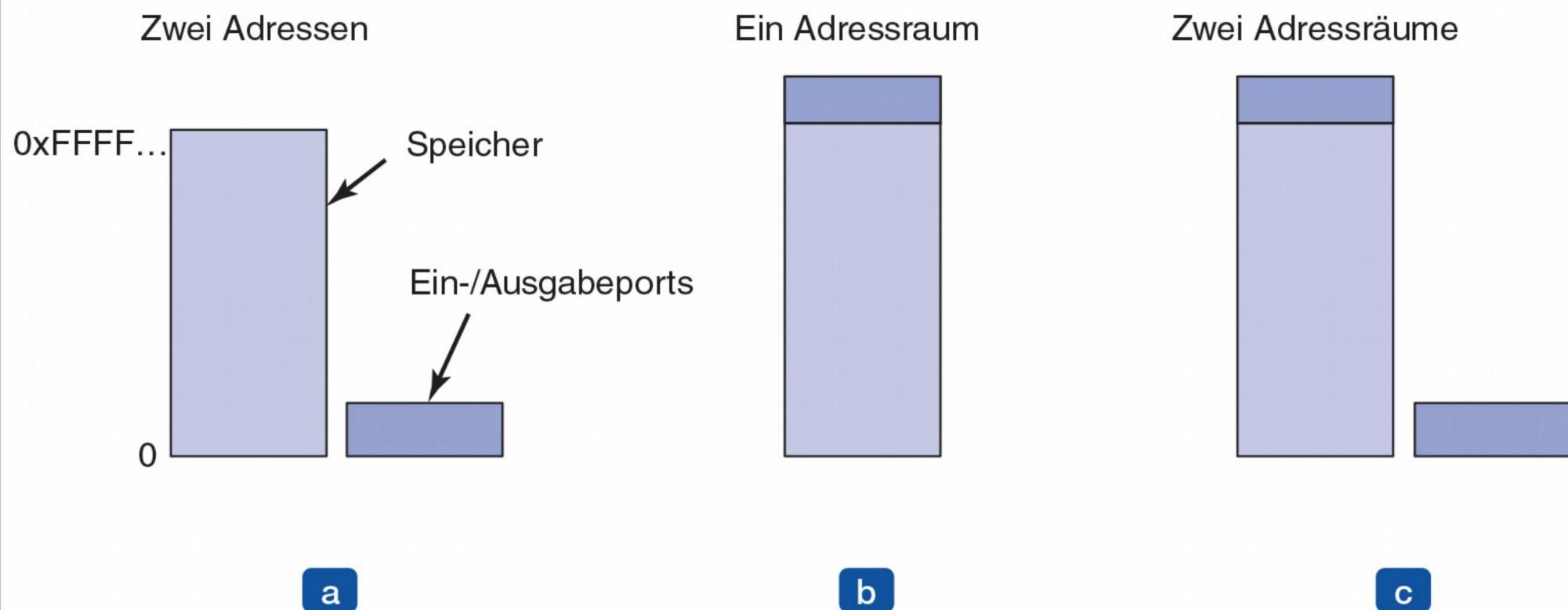


Abbildung 5.2: (a) Getrennter Ein-/Ausgabe- und Arbeitsspeicherbereich. (b) E/A-Adressraum liegt auch im Arbeitsspeicher (Memory-Mapped-E/A). (c) Hybride Lösung.

Memory-Mapped- Ein-/Ausgabe (2)

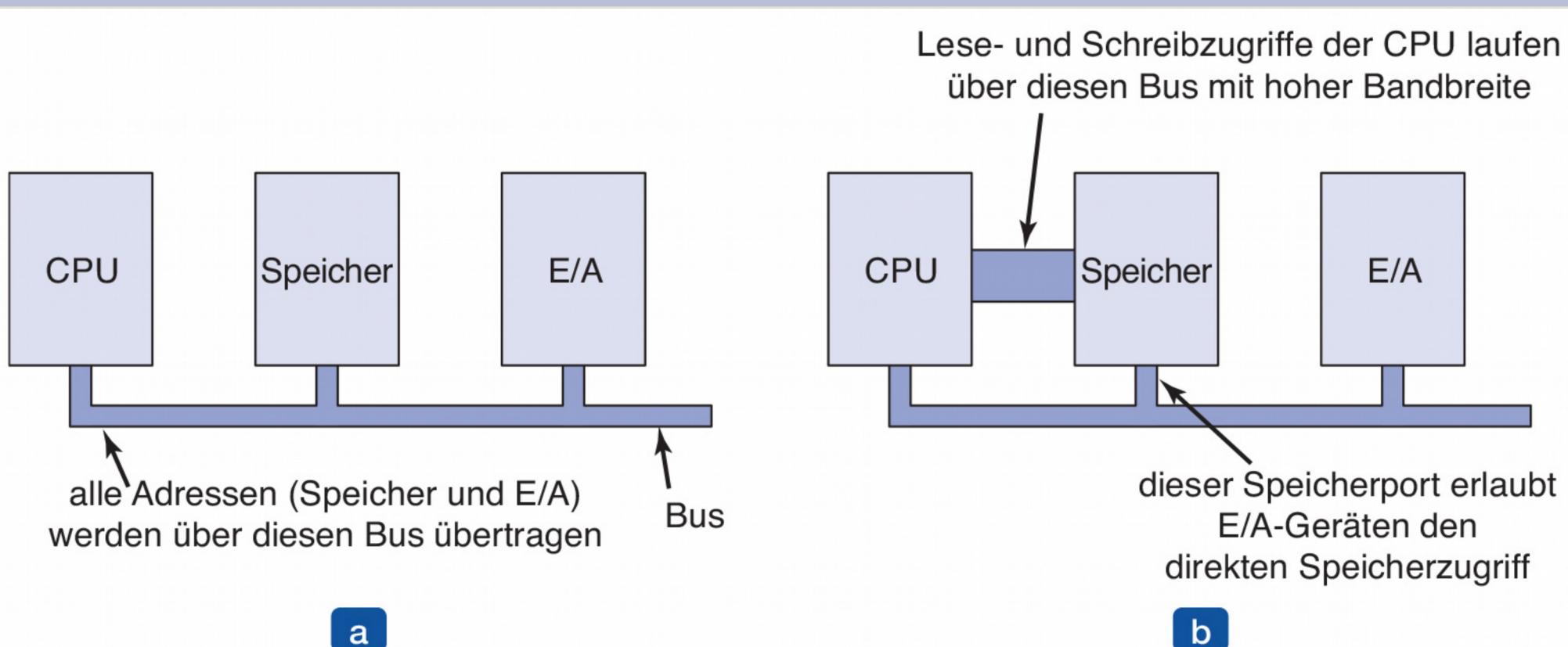


Abbildung 5.3: (a) Architektur mit einem Bus. (b) Architektur mit zwei Bussen.

Interruptgesteuerte Ein-/Ausgabe

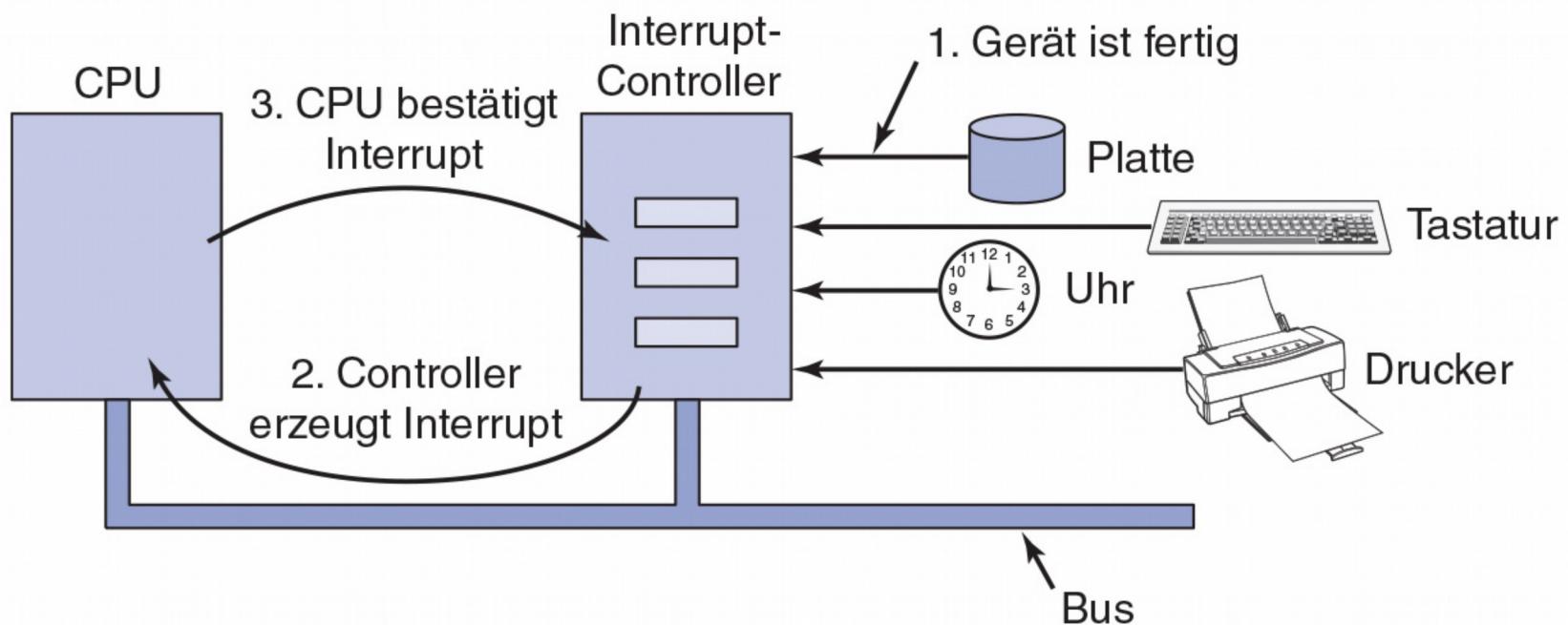


Abbildung 5.5: Ablauf eines Interrupts. Die Verbindung zwischen dem Gerät und dem Controller benutzt statt eigener Leitungen die Interrupt-Leitungen auf dem Bus.

Präzise Interrupts

Vier Eigenschaften eines präzisen Interrupts:

- Der PC wurde an einem bekannten Ort gespeichert.
- Alle Anweisungen, auf die der PC hingewiesen hat, wurden vollständig ausgeführt.
- Es wurde keine weitere Anweisung ausgeführt, auf die der PC verwiesen hat.
- Der Ausführungszustand der Anweisung, auf den der PC zeigt, ist bekannt.

Präzise und unpräzise Interrupts

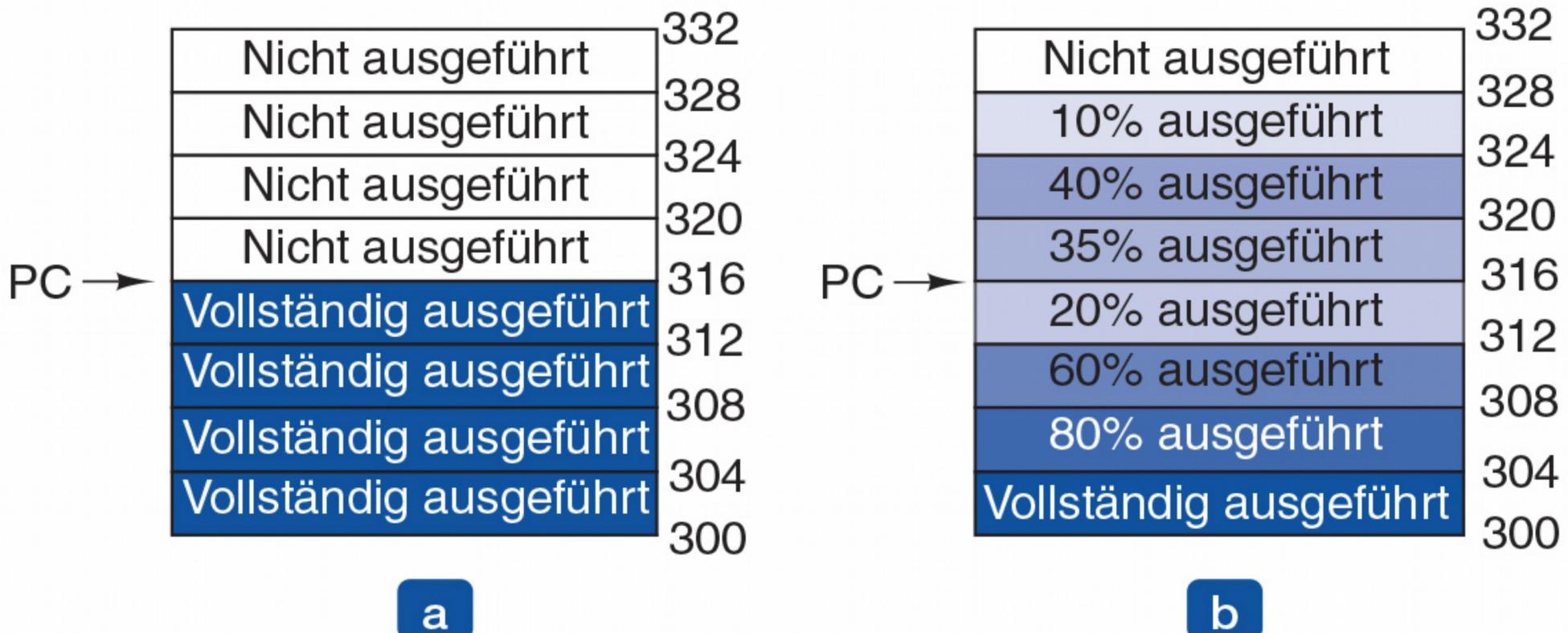


Abbildung 5.6: (a) Präziser Interrupt. (b) Unpräziser Interrupt.

Ein-/Ausgabe mit DMA

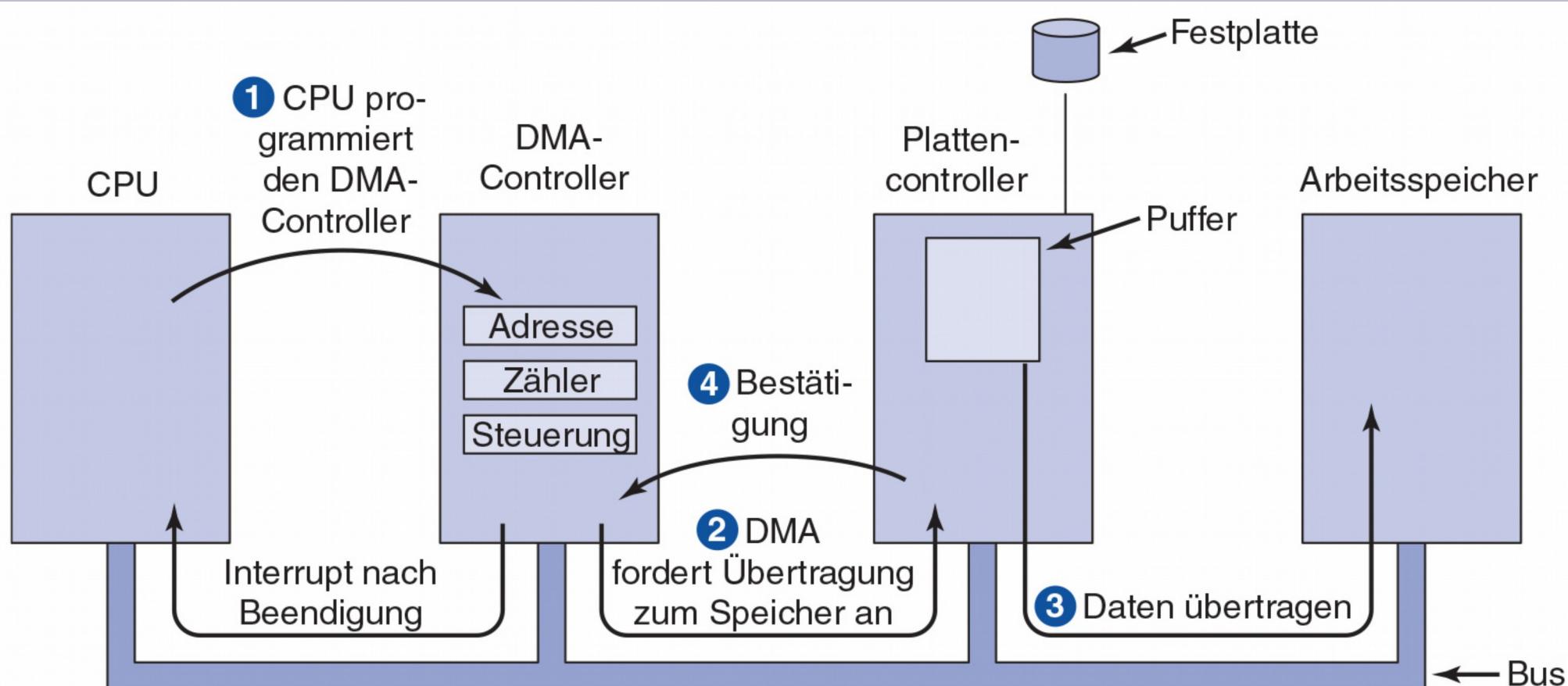


Abbildung 5.4: Ablauf eines DMA-Transfers.

5.2 Grundlagen der Ein-/Ausgabesoftware

5.2.1 Ziele von Ein-/Ausgabesoftware

5.2.2 Programmierte Ein-/Ausgabe

5.2.3 Interruptgesteuerte Ein-/Ausgabe

5.2.4 Ein-/Ausgabe mit DMA

Grundlagen der Ein-/Ausgabesoftware

Problemstellungen:

- Geräteunabhängigkeit
- Einheitliche Benennung
- Fehlerbehandlung
- Synchron versus Asynchron
- Pufferung

Programmierte Ein-/Ausgabe (2)

```
copy_from_user(buffer, p, count);          /* p ist der Kern-Puffer */
for (i = 0; i < count; i++) {               /* Schleife über alle Zeichen */
    while (*printer_status_reg != READY);   /* Wiederhole bis zum Ende */
    *printer_data_register = p[i];           /* Gebe ein Zeichen aus*/
}
return_to_user( );
```

Abbildung 5.8: Das Schreiben einer Zeichenkette auf den Drucker mit programmierter Ein-/Ausgabe.

Interruptgesteuerte Ein-/Ausgabe

```
copy_from_user(buffer, p, count);
enable_interrupts( );
while (*printer_status_reg != READY);
*printer_data_register = p[0];
scheduler( );
```

a

```
if (count == 0) {
    unblock_user( );
} else {
    *printer_data_register = p[i];
    count = count - 1;
    i = i + 1;
}
acknowledge_interrupt( );
return_from_interrupt( );
```

b

Abbildung 5.9: Das Ausgeben einer Zeichenkette auf den Drucker mittels interruptgesteuerter Ein-/Ausgabe: (a) Code, der zum Zeitpunkt des Systemaufrufs zum Drucken ausgeführt wird; (b) Unterbrechungsroutine für den Drucker.

Ein-/Ausgabe mit DMA

```
copy_from_user(buffer, p, count);  
set_up_DMA_controller( );  
scheduler( );
```

a

```
acknowledge_interrupt( );  
unblock_user( );  
return_from_interrupt( );
```

b

Abbildung 5.10: Das Drucken einer Zeichenkette mit DMA: (a) der ausgeführte Code, wenn der Systemaufruf zum Drucken benutzt wurde; (b) Unterbrechungsroutine.

5.3 Schichten der Ein-/Ausgabesoftware

5.3.1 Unterbrechungsroutinen

5.3.2 Gerätetreiber

5.3.3 Geräteunabhängige Ein-/Ausgabesoftware

5.3.4 Ein-/Ausgabesoftware im Benutzeradressraum

Schichten der Ein-/Ausgabesoftware



Abbildung 5.11: Schichten des Ein-/Ausgabesoftwaresystems.

Unterbrechungsroutinen (1)

Typische Schritte nach Auslösung eines Hardware-Interrupts:

1. Alle Register speichern einschließlich des PSW (Statusregister, engl. Program Status Word)
2. Kontext für die Interrupt-Service-Prozedur einrichten
3. Einen Stack für die Interrupt-Service-Prozedur einrichten
4. Bestätigung an den Interrupt-Controller (Falls kein zentraler Interrupt-Controller vorhanden: Interrupts wieder aktivieren)
5. Register kopieren wie sie in die Prozesstabellen gespeichert wurden

Unterbrechungsroutinen (2)

Typische Schritte nach Auslösung eines Hardware-Interrupts (2):

6. Interrupt-Service-Prozedur ausführen: Extrahieren der Informationen aus den Registern des unterbrechenden Controllers.
7. Prozess auswählen, der als nächster ausgeführt werden soll.
8. Einrichten des MMU-Kontext für den nächsten Prozess.
9. Laden der neuen Prozessorregister einschließlich des PSW.
10. Starten des neuen Prozesses.

Gerätetreiber

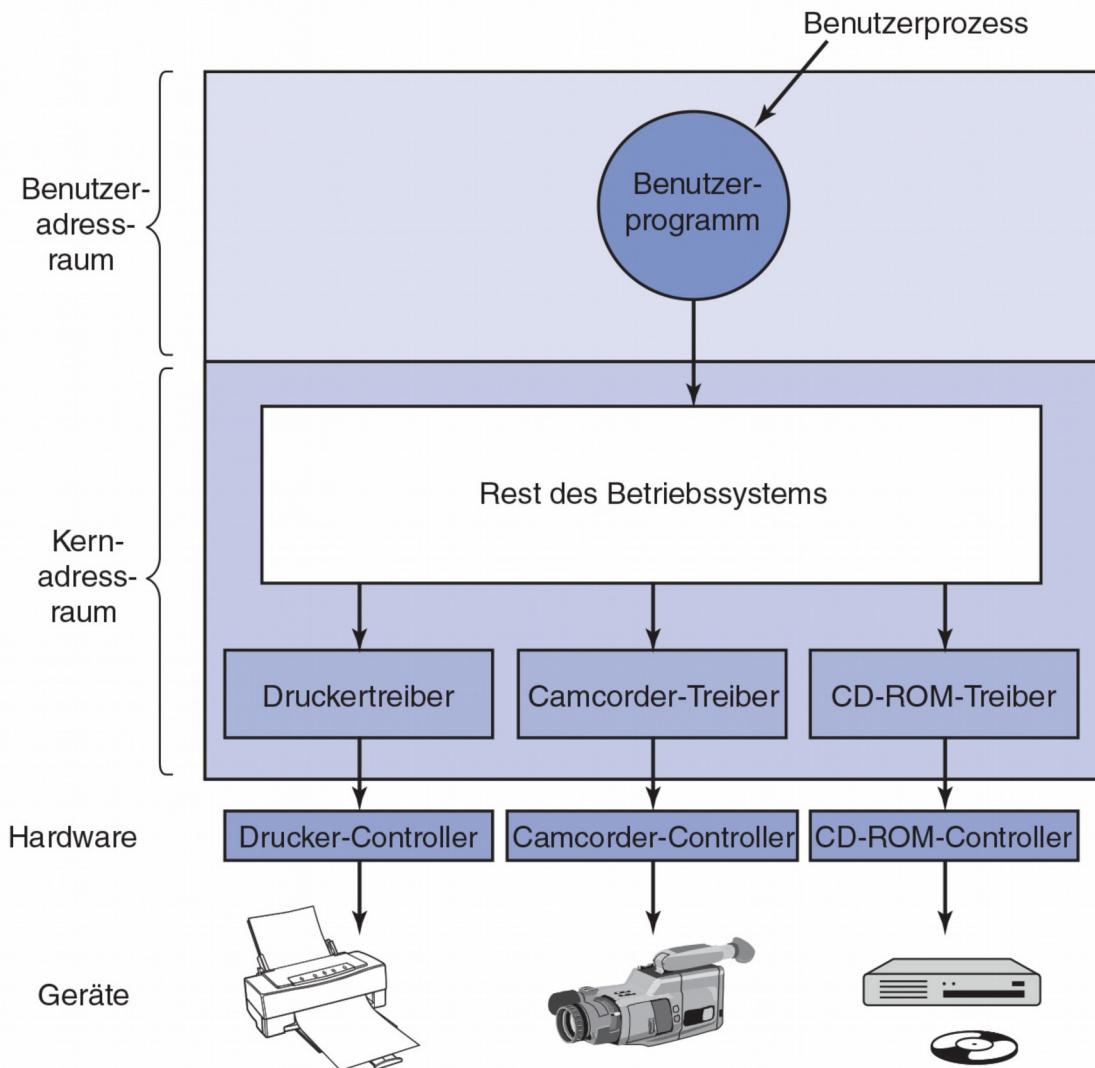


Abbildung 5.12: Logische Positionierung der Gerätetreiber. In Wirklichkeit erfolgt jede Kommunikation zwischen den Treibern und den Controllern über den Bus.

Geräteunabhängige Ein-/Ausgabesoftware

Einheitliche Schnittstelle für Gerätetreiber

Pufferung

Fehlerbericht

Anforderung und Freigabe von exklusiv zugewiesenen Geräten

Geräteunabhängige Blockgröße zur Verfügung stellen

Abbildung 5.13: Funktionen der geräteunabhängigen Ein-/Ausgabesoftware.

Einheitliches Interface für Gerätetreiber

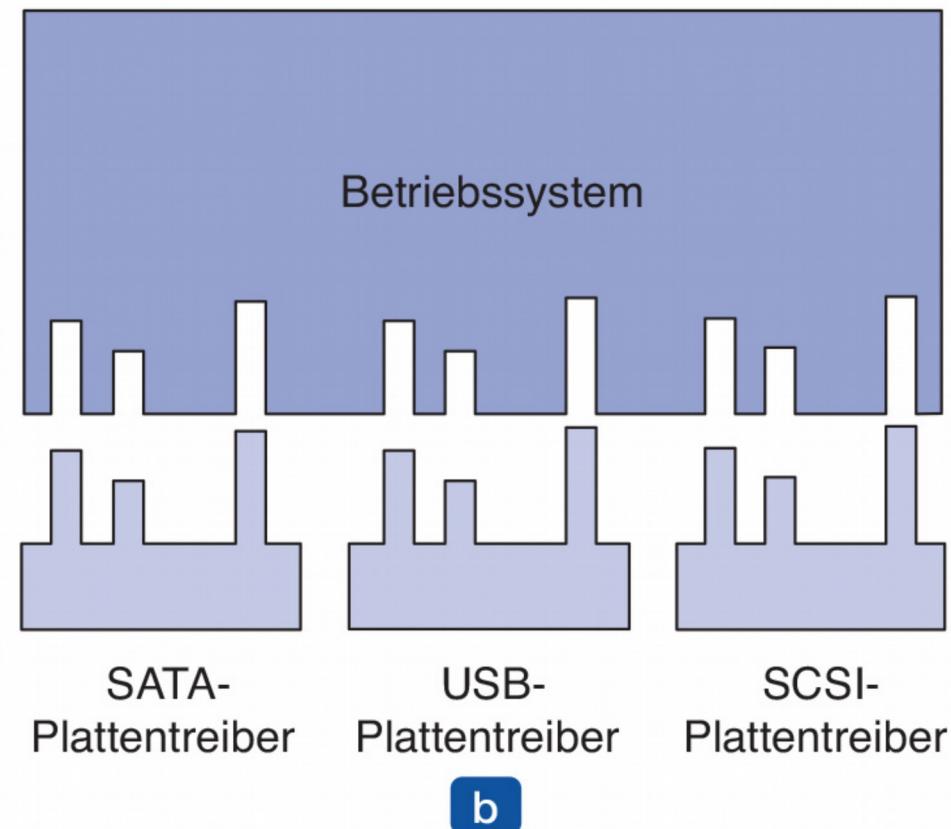
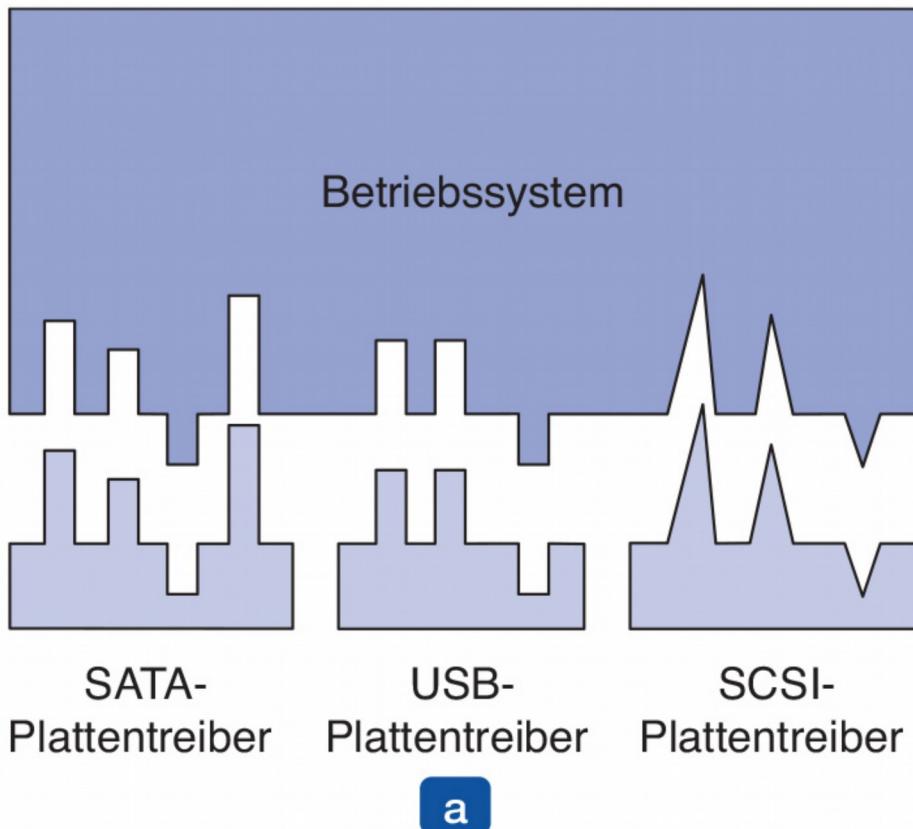


Abbildung 5.14: (a) Ohne eine Standardschnittstelle für Treiber. (b) Mit einer Standardschnittstelle für Treiber.

Puffern (1)

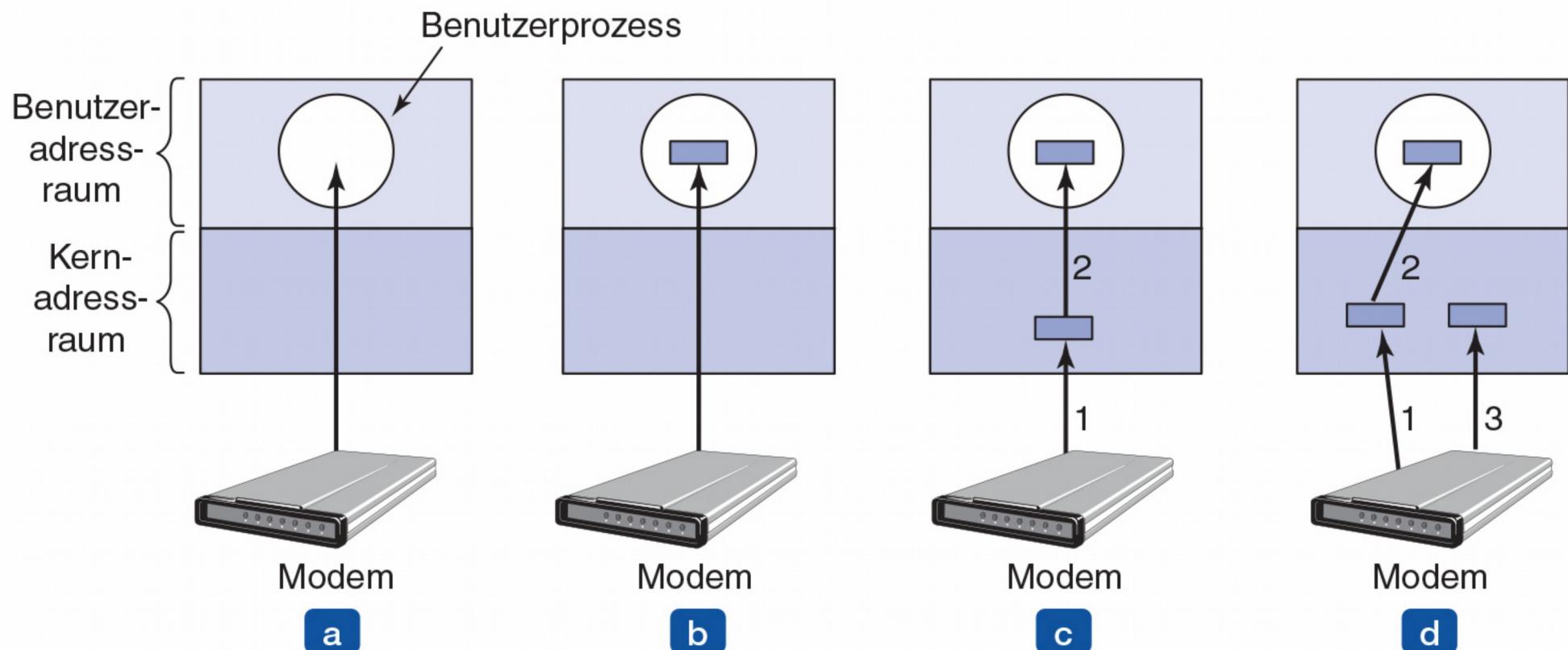


Abbildung 5.15: (a) Ungepufferte Eingabe. (b) Pufferung im Benutzeradressraum. (c) Pufferung im Kern, gefolgt vom Kopieren in den Benutzeradressraum. (d) Doppelte Pufferung im Kern.

Puffern (2)

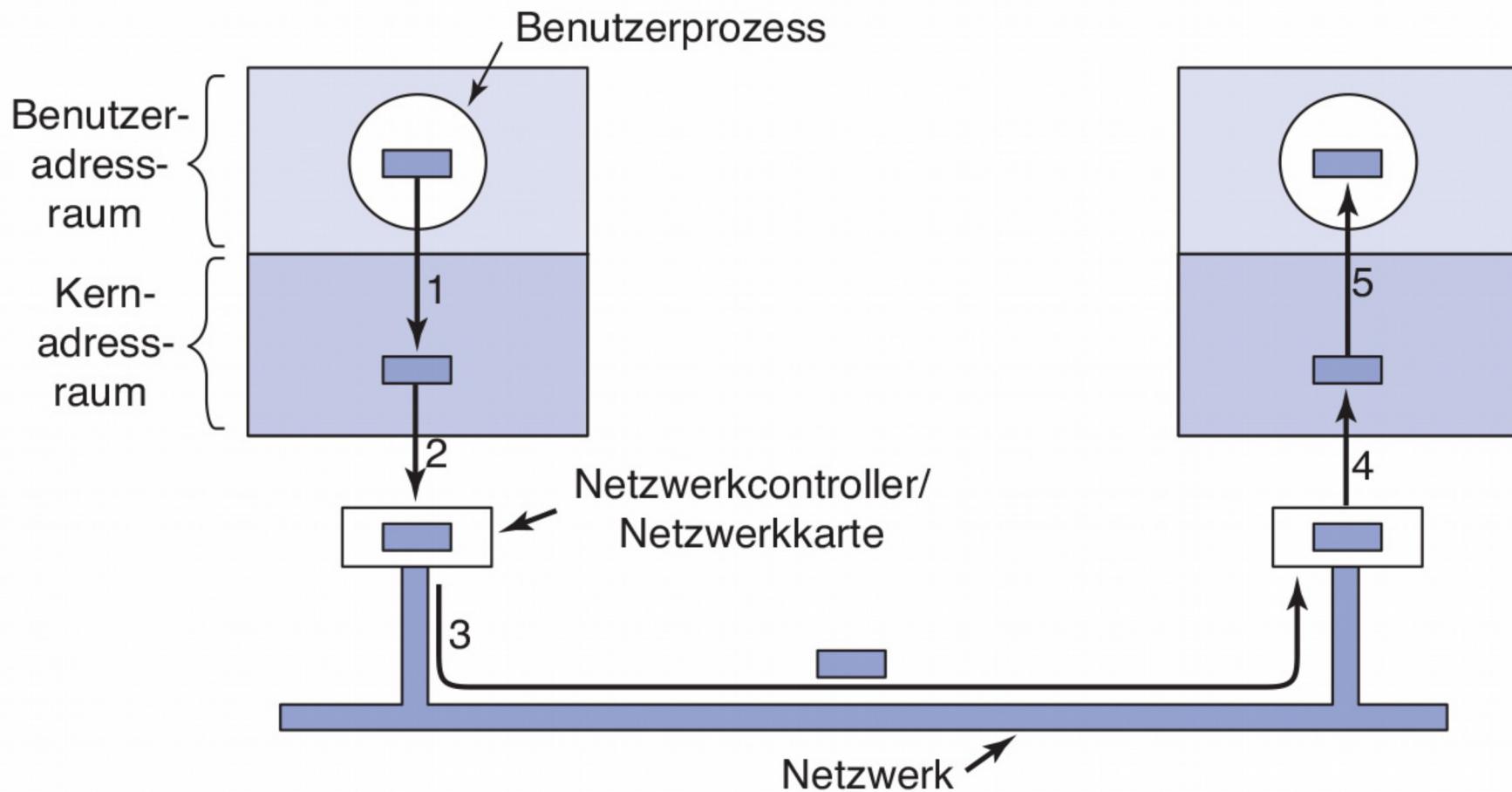


Abbildung 5.16: Netzwerkverwaltung kann das Kopieren von vielen Paketen bedeuten.

Ein-/Ausgabesoftware im Benutzeradressraum

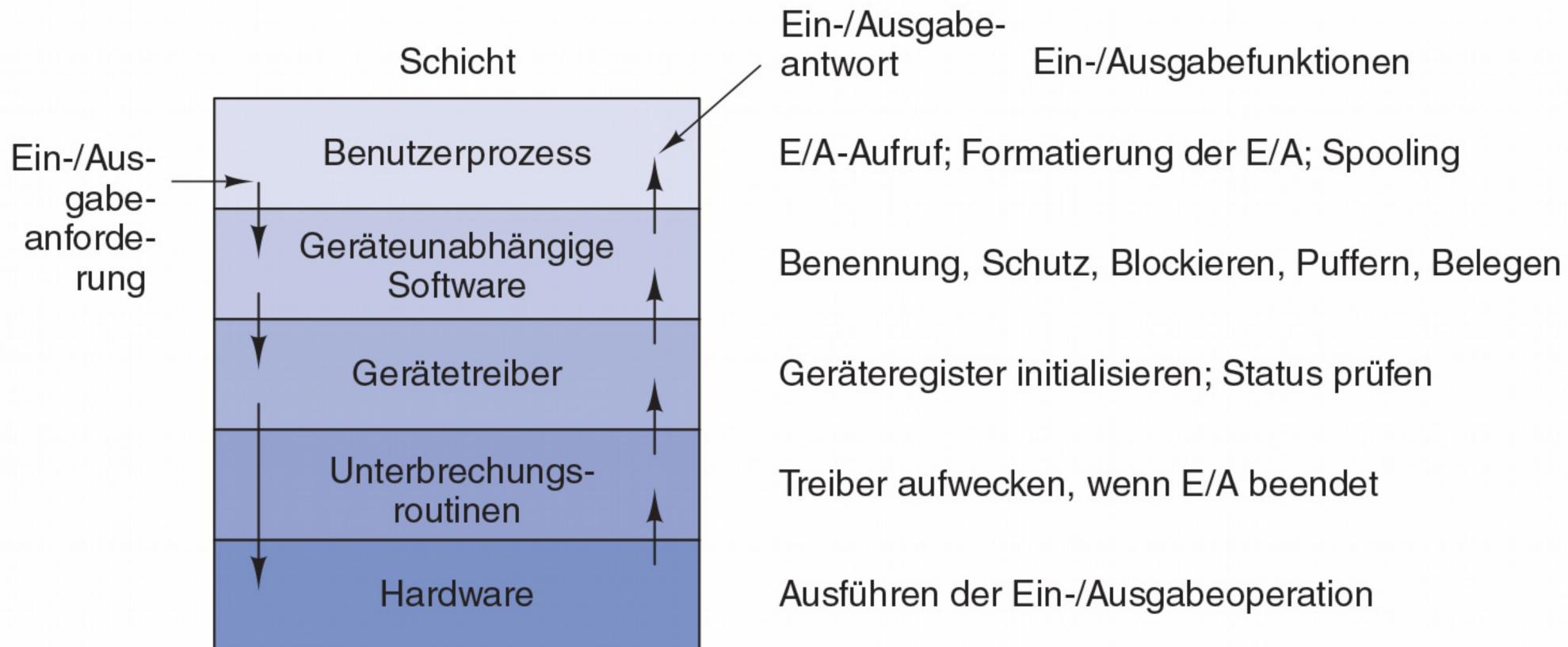


Abbildung 5.17: Schichten des Ein-/Ausgabesystems und die Hauptfunktion jeder Ebene.

5.4 Plattenspeicher

5.4.1 Hardware von Plattenspeichern

5.4.2 Formatierung von Plattenspeichern

5.4.3 Strategien zur Steuerung des Plattenarms

5.4.4 Fehlerbehandlung

5.4.5 Zuverlässiger Speicher

Hardware von Plattspeichern (1)

Parameter	360-KB-Diskette von IBM	WD-3000-HLFS-Festplatte
Anzahl der Zylinder	40	36481
Spuren pro Zylinder	2	255
Sektoren pro Spur	9	63 (im Durchschnitt)
Sektoren pro Platte	720	586072368
Bytes pro Sektor	512	512
Plattenkapazität	360 KB	300 GB
Zugriffszeit (benachbarter Zylinder)	6 ms	0,7 ms
Zugriffszeit (Durchschnitt)	77 ms	4,2 ms
Rotationszeit	200 ms	6 ms
Übertragungszeit eines Sektors	22 ms	1,4 µs

Abbildung 5.18: Plattenparameter der ursprünglichen 360-KB-Diskette des IBM-PC und einer WD-3000-HLFS-Festplatte („Velociraptor“) von Western Digital.

Hardware von Plattspeichern (2)

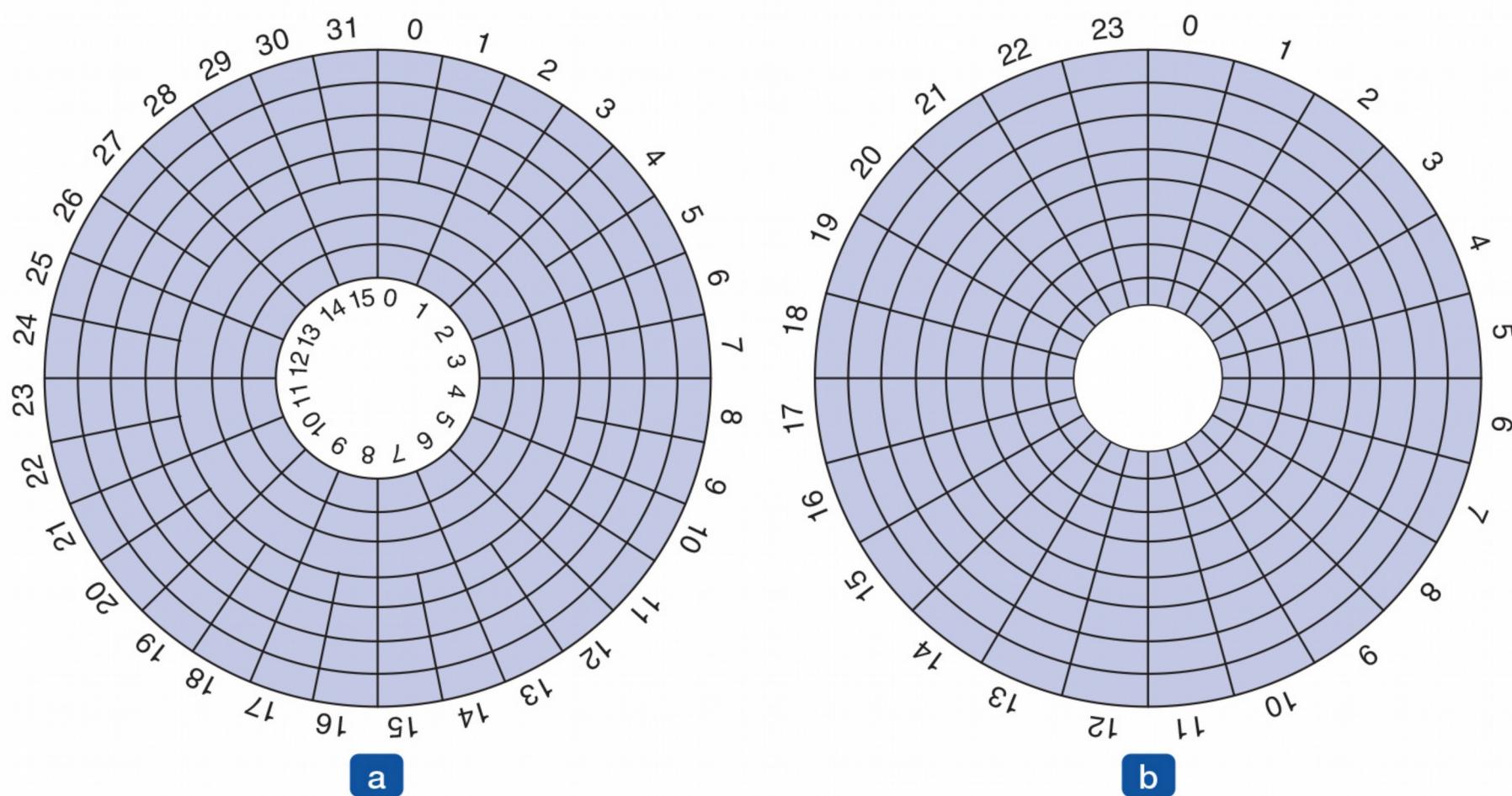
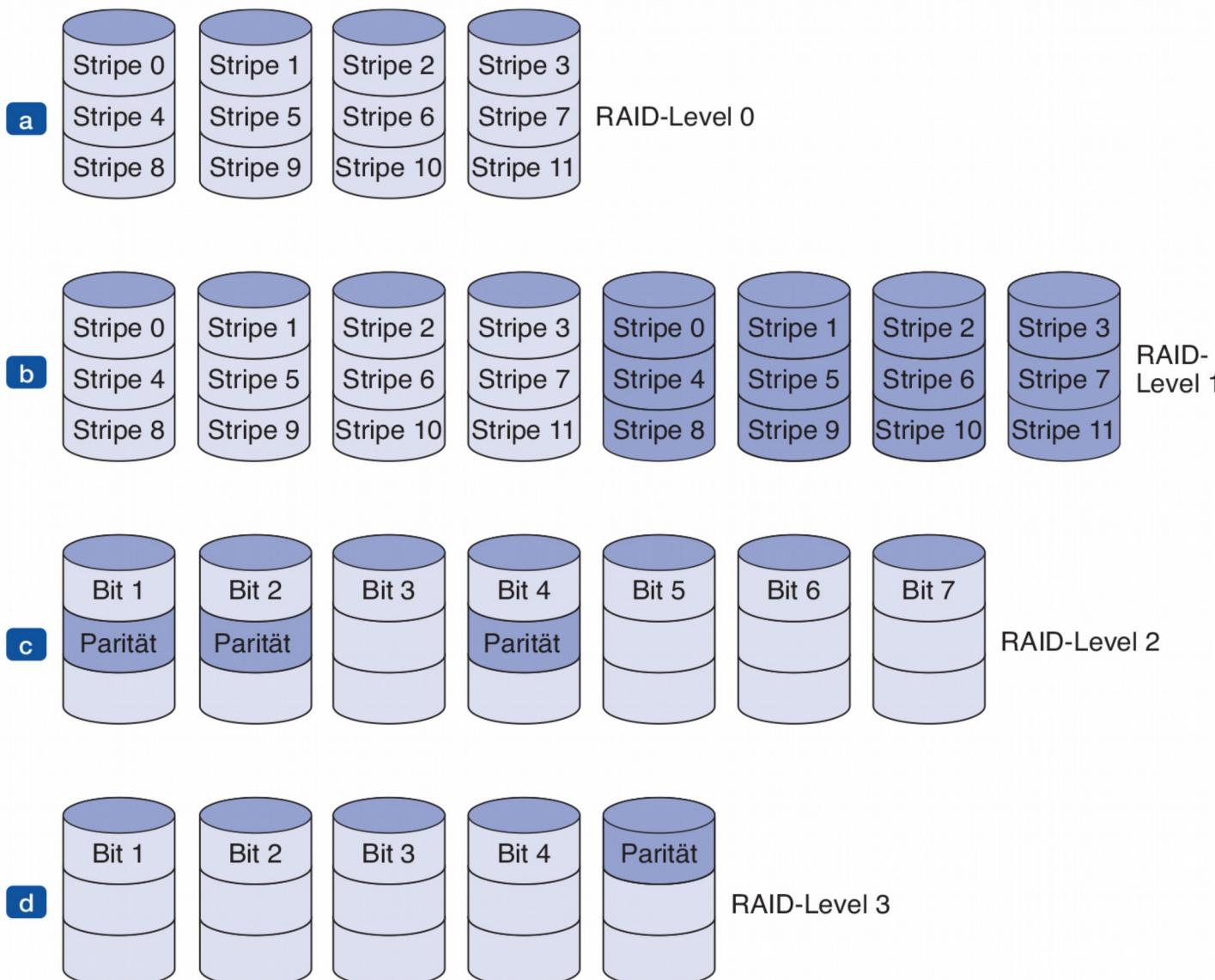


Abbildung 5.19: (a) Physische Geometrie einer Platte mit zwei Zonen. (b) Eine mögliche virtuelle Geometrie für diese Platte.

RAID (1)



RAID (2)

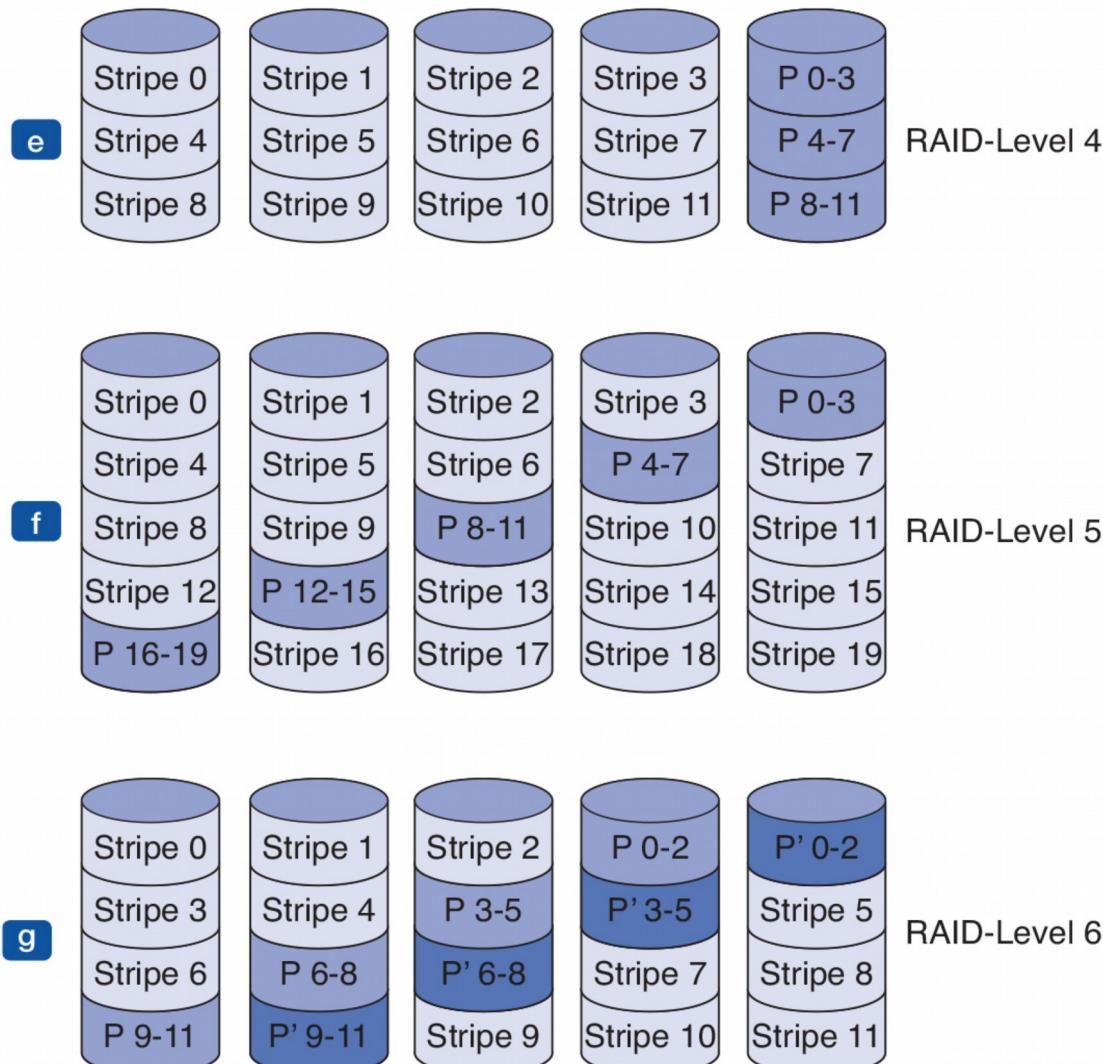


Abbildung 5.20: RAID-Level 0 bis 6. Die Sicherungs- und Paritätlaufwerke sind dunkler dargestellt.

Formatierung von Plattsenspeichern (1)



Abbildung 5.21: Festplattensektor.

Formatierung von Plattspeichern (2)

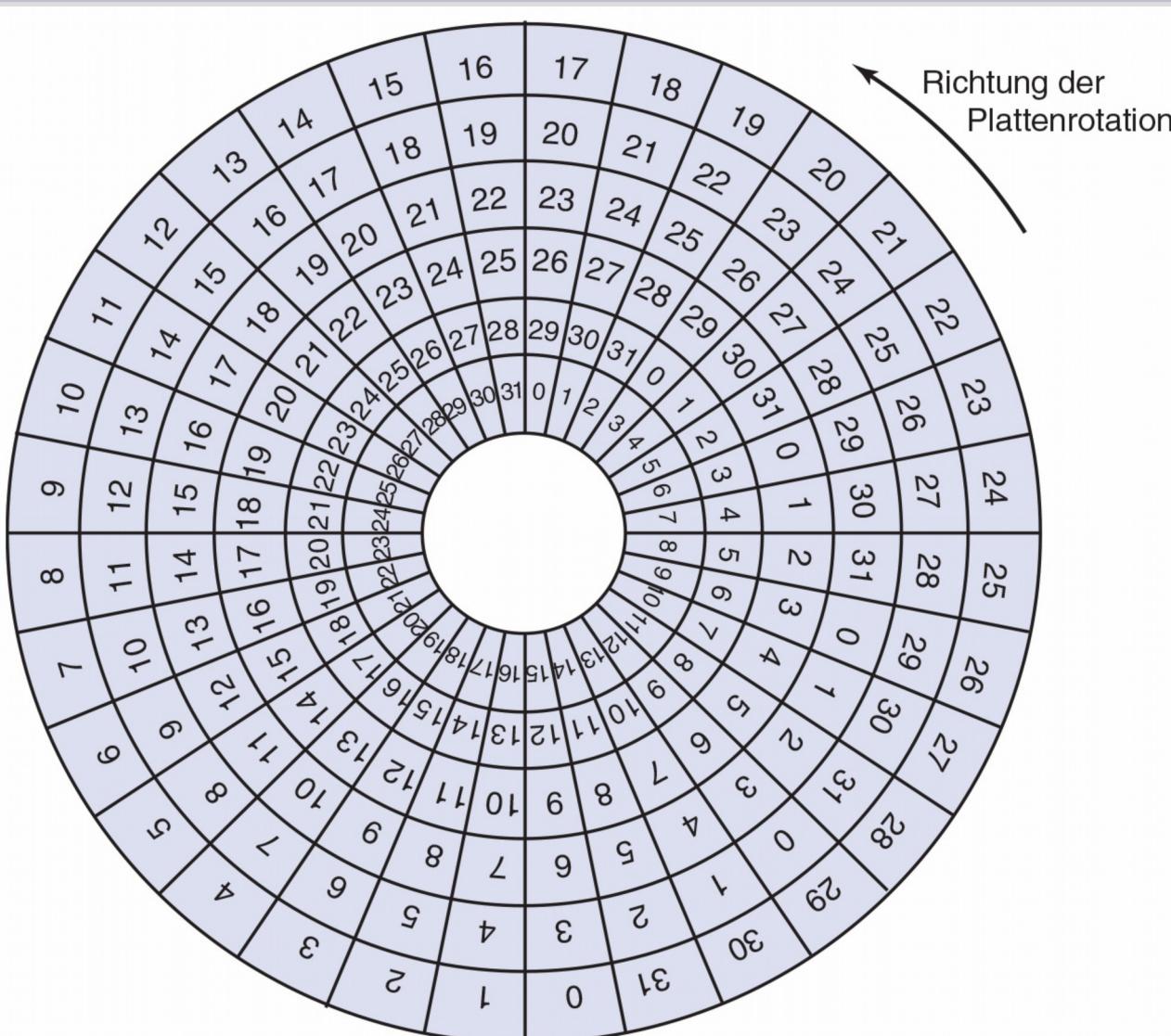
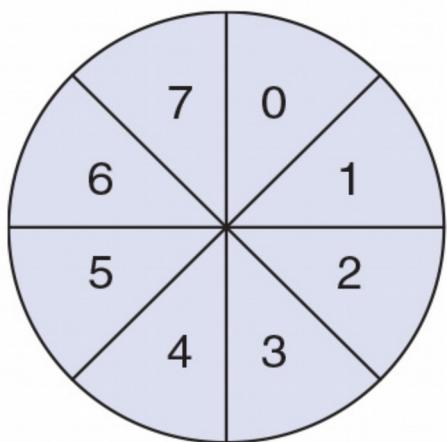
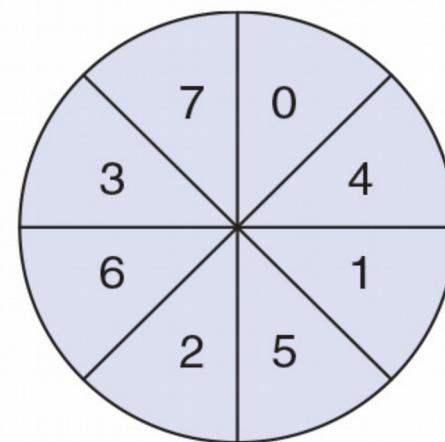


Abbildung 5.22: Darstellung des Zylinderversatzes.

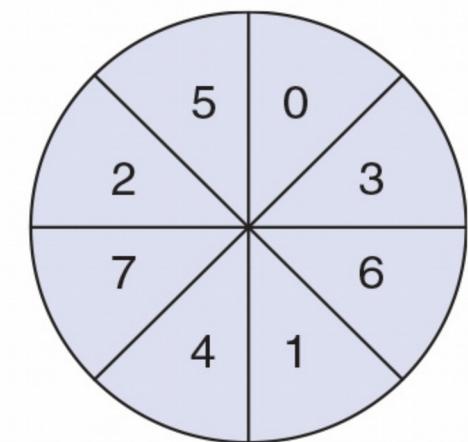
Formatierung von Plattsenspeichern (3)



a



b



c

Abbildung 5.23: (a) Keine Verschachtelung. (b) Einfache Verschachtelung. (c) Doppelte Verschachtelung.

Strategien zur Steuerung des Plattenarms (1)

Faktoren, welche die Zeit zum Lesen/Schreiben eines Plattenblocks beeinflussen:

1. Suchzeit (Seek Time, die Zeit, um den Arm zum richtigen Zylinder zu bewegen).
2. Rotationsverzögerung (Rotational Latency, wie lange dauert es, bis der richtige Sektor unter den Kopf kommt).
3. Tatsächliche Datenübertragungszeit.

Strategien zur Steuerung des Plattenarms (2)

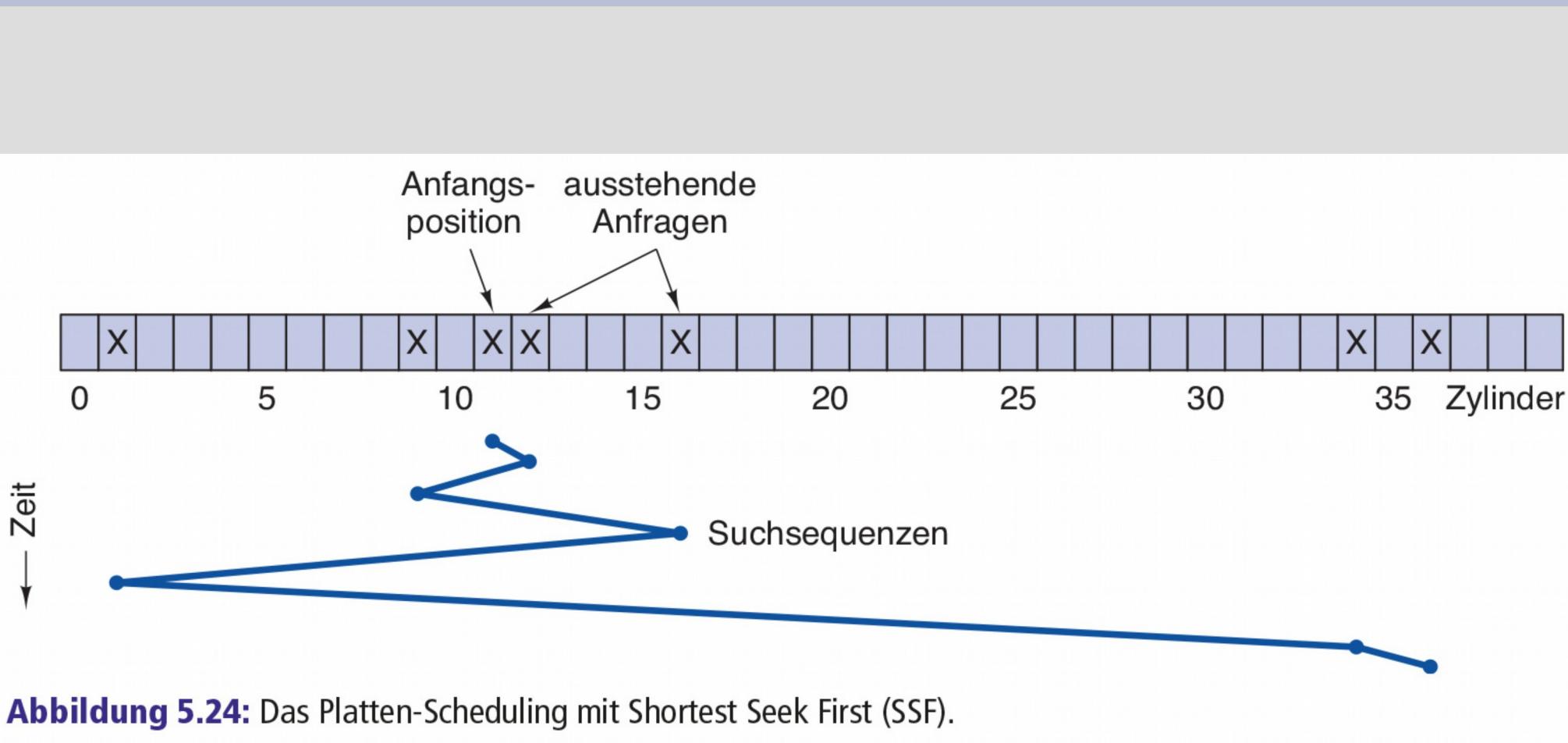


Abbildung 5.24: Das Platten-Scheduling mit Shortest Seek First (SSF).

Strategien zur Steuerung des Plattenarms (3)

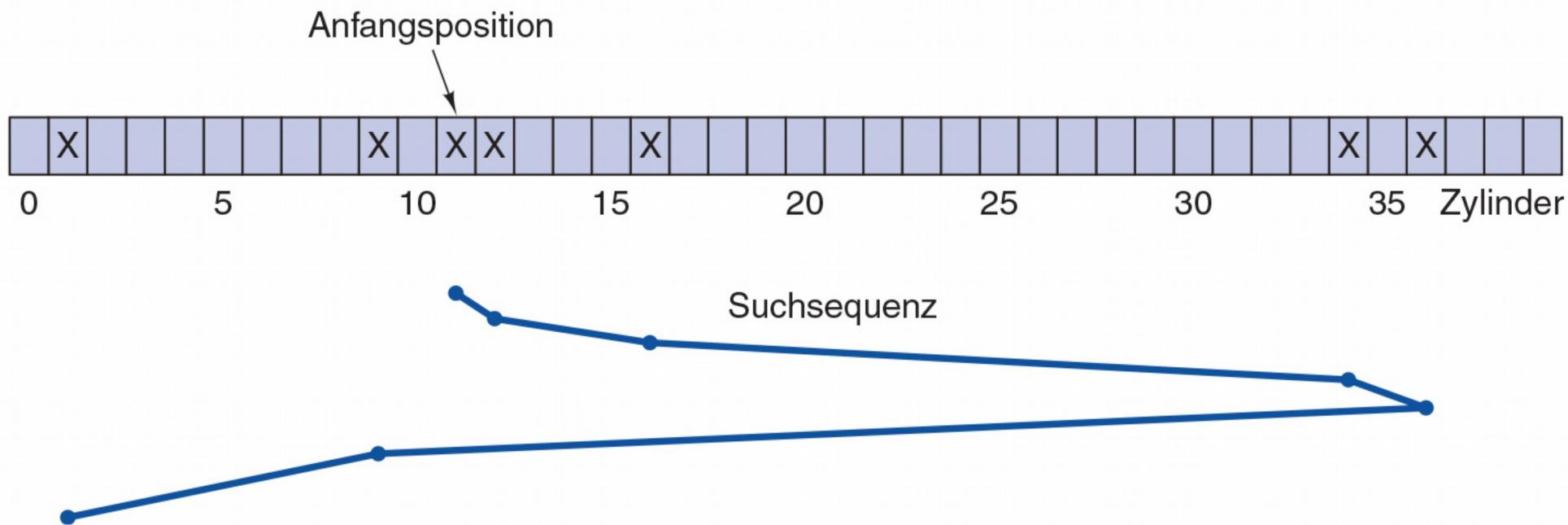


Abbildung 5.25: Der Aufzugsalgorithmus zur Planung von Plattenzugriffen.

Fehlerbehandlung

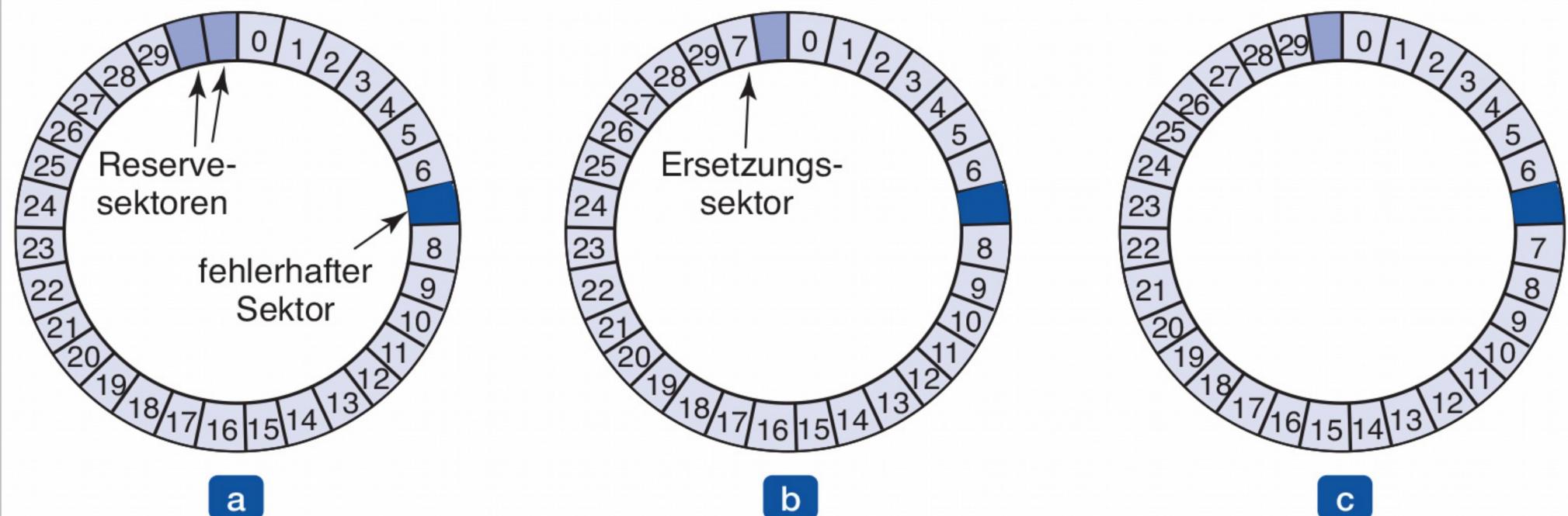


Abbildung 5.26: (a) Plattenspur mit einem fehlerhaften Sektor. (b) Ersetzung des defekten Sektors durch einen Reservesektor. (c) Verschiebung des Sektors, um den fehlerhaften Sektor zu umgehen.

Zuverlässiger Speicher (1)

- Verwendet ein Paar identischer Festplatten
- Jede Festplatte liefert beim Lesen die gleichen Ergebnisse
- Zu diesem Zweck definierte Operationen:
 1. Stabiles Schreiben (Write)
 2. Stabiles Lesen (Read)
 3. Crash-Wiederherstellung

Zuverlässiger Speicher (2)

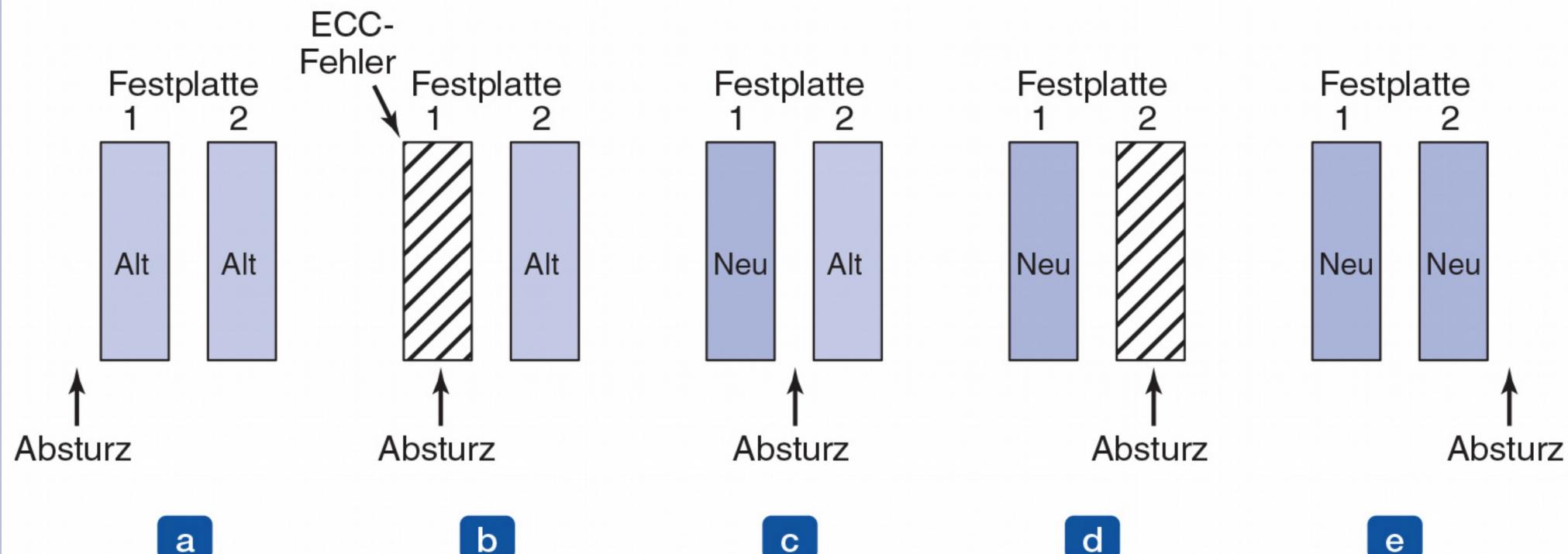


Abbildung 5.27: Analyse des Einflusses eines Absturzes bei zuverlässigerem Schreiben.

5.5 Uhren

5.5.1 Hardwareuhren

5.5.2 Softwareuhren

5.5.3 Soft-Timer

Hardwareuhren

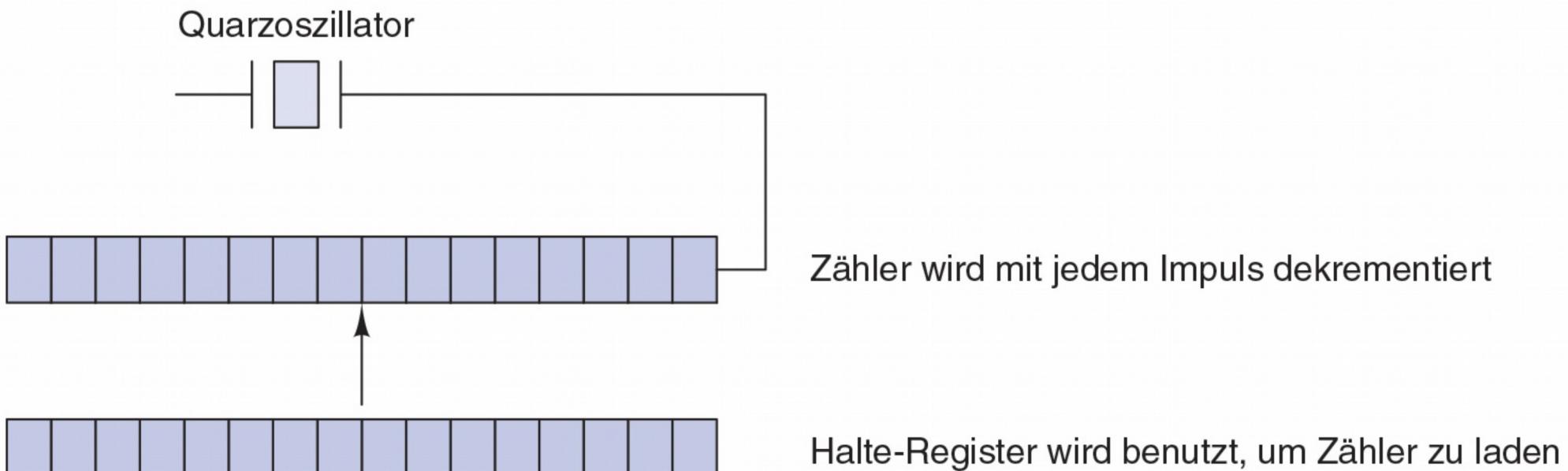


Abbildung 5.28: Programmierbare Uhr.

Software für Uhren (1)

Typische Aufgaben eines Uhrentreibers:

1. Pflege der Tageszeit.
2. Verhindern, dass Prozesse länger als zulässig ausgeführt werden.
3. Berücksichtigung der CPU-Auslastung.
4. Alarmsystemaufruf von Benutzerprozessen verarbeiten.
5. Bereitstellung von Watchdog-Timern für Teile des Systems.
6. Profilerstellung, Überwachung, Statistikerfassung.

Software für Uhren (2)

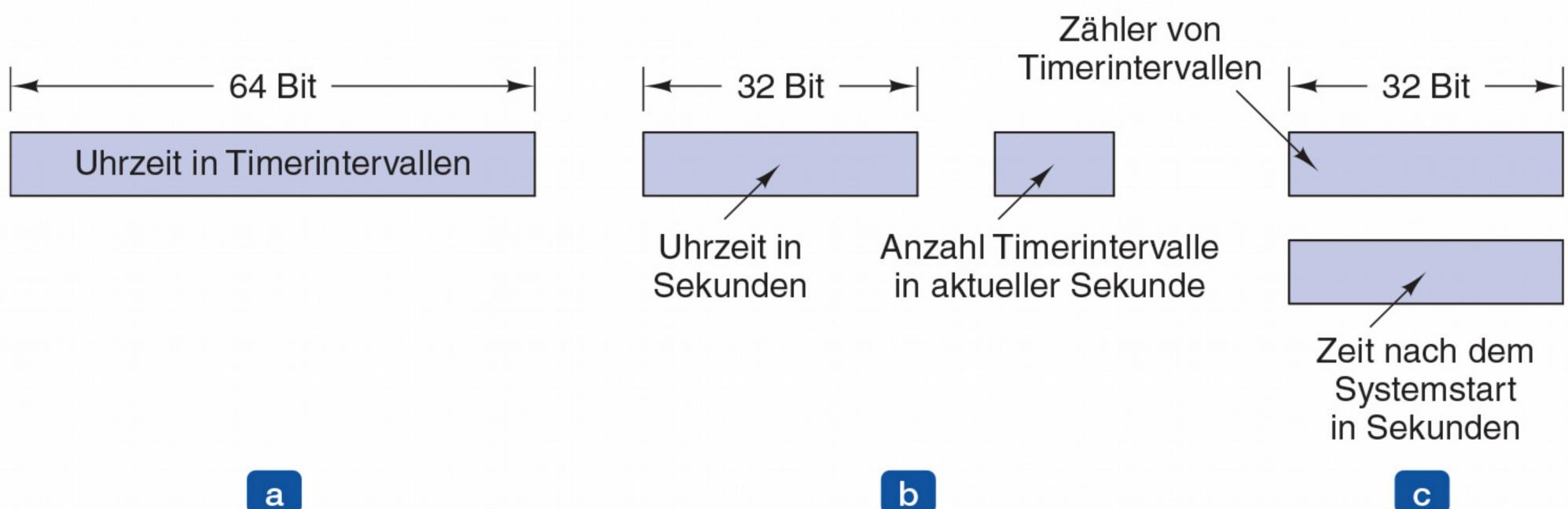


Abbildung 5.29: Drei Arten, die aktuelle Zeit zu verwalten.

Software für Uhren (3)

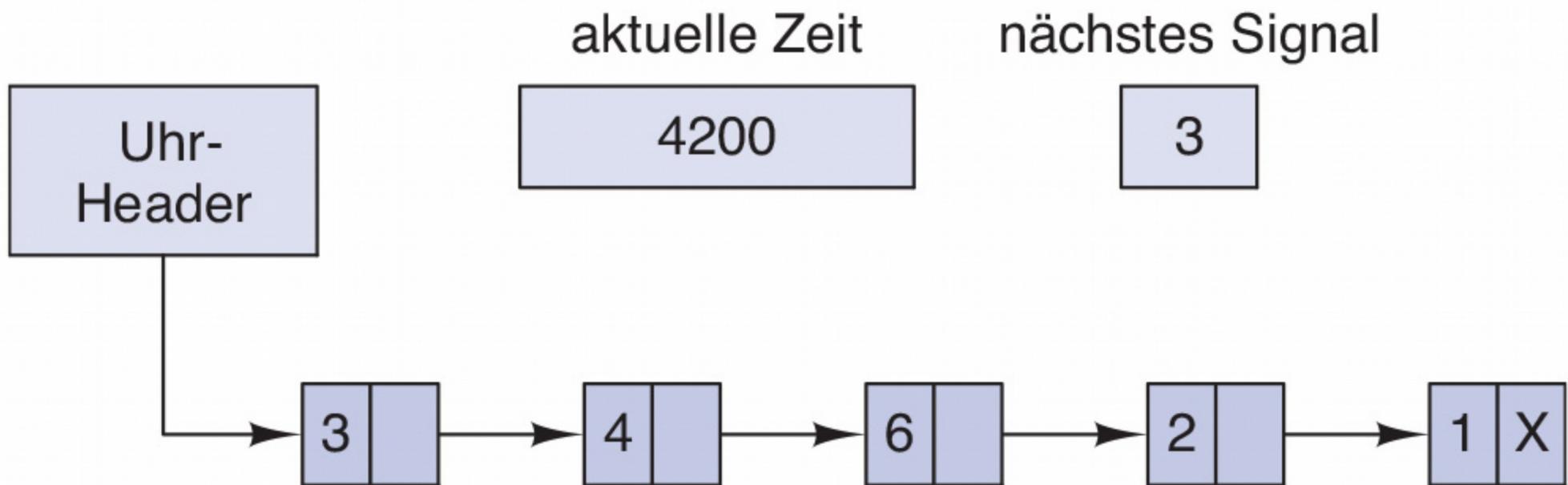


Abbildung 5.30: Simulation mehrerer Timer mit einer einzigen Uhr.

Soft-Timer

Soft-Timer stehen oder fallen mit der Rate, mit der Kernel-Einträge aus anderen Gründen vorgenommen werden. Diese Gründe beinhalten:

1. Systemaufrufe.
2. TLB fehlt.
3. Seitenfehler.
4. E / A-Interrupts.
5. Die CPU geht in den Leerlauf.

5.6 Benutzerschnittstellen: Tastatur, Maus, Bildschirm

5.6.1 Eingabesoftware

5.6.2 Ausgabesoftware

Tastatur-Software

Zeichen	POSIX-Name	Kommentar
STRG-H	ERASE	Zeichen löschen und eine Spalte zurück
STRG-U	KILL	Gesamte Zeile löschen
STRG-V	LNEXT	Nächstes Zeichen interpretieren
STRG-S	STOP	Ausgabe anhalten
STRG-Q	START	Ausgabe starten
ENTF	INTR	Prozess unterbrechen (SIGINT)
STRG-\	QUIT	Core Dump erzwingen (SIGQUIT)
STRG-D	EOF	Ende der Datei
STRG-M	CR	Wagenrücklauf (unveränderlich)
STRG-J	NL	Zeilenvorschub (unveränderlich)

Abbildung 5.31: Zeichen, die im kanonischen Modus speziell behandelt werden.

Ausgabesoftware - Textfenster

Escape-Sequenz	Bedeutung
ESC [<i>n</i> A	<i>n</i> Zeilen nach oben
ESC [<i>n</i> B	<i>n</i> Zeilen nach unten
ESC [<i>n</i> C	<i>n</i> Zeichen nach rechts
ESC [<i>n</i> D	<i>n</i> Zeichen nach links
ESC [<i>m; n</i> H	Cursor an Position (<i>m, n</i>) bewegen
ESC [<i>s</i> J	Lösche Bild ab Cursor (0 bis Ende, 1 vom Anfang, 2 alles)
ESC [<i>s</i> K	Lösche Zeile ab Cursor (0 bis Ende, 1 vom Anfang, 2 alles)
ESC [<i>n</i> L	Füge <i>n</i> Zeilen ab Cursor ein
ESC [<i>n</i> M	Lösche <i>n</i> Zeilen ab Cursor
ESC [<i>n</i> P	Lösche <i>n</i> Zeichen ab Cursor
ESC [<i>n</i> @	Füge <i>n</i> Zeichen ab Cursor ein
ESC [<i>n m</i>	Zeichendarstellung (0=normal, 4=fett, 5=blinkend, 7=invers)
ESC M	Bild zurückrollen, wenn Cursor in der obersten Zeile steht

Abbildung 5.32: Die ANSI-Escape-Sequenzen, die vom Terminaltreiber bei der Ausgabe akzeptiert werden. ESC beschreibt die ASCII-Sequenz (0x1B) und *n*, *m* und *s* sind die optionalen numerischen Parameter.

Das X Window System (1)

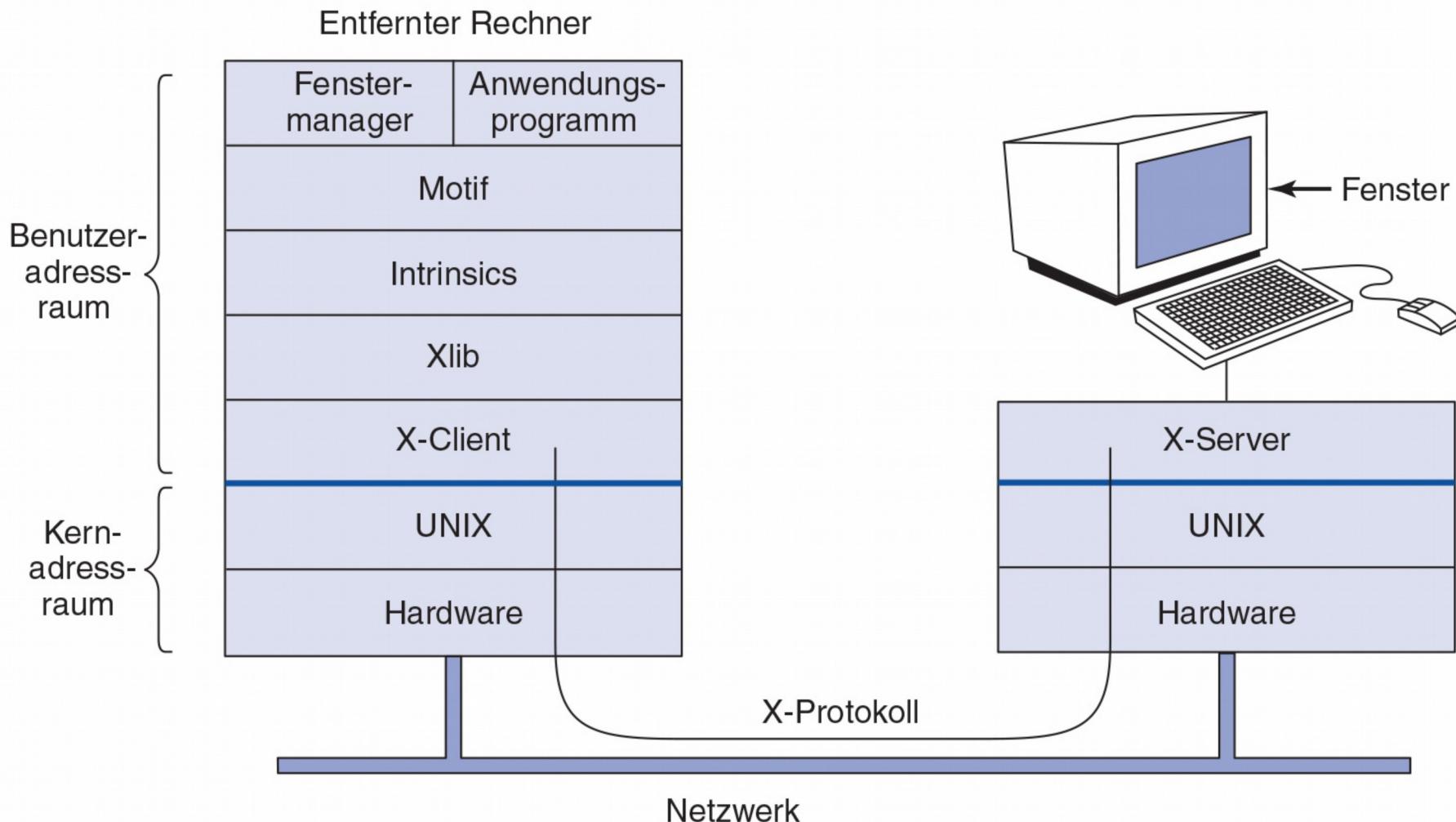


Abbildung 5.33: Clients und Server im X-Window-System des M.I.T.

Das X Window System (2)

Nachrichtentypen zwischen Client und Server:

1. Zeichenbefehle vom Programm zur Arbeitsstation.
2. Antworten von der Workstation auf Programmanfragen.
3. Tastatur-, Maus- und andere Ereignisankündigungen.
4. Fehlermeldungen.

Das X Window System (3)

Grundgerüst eines X-Window Programms

```
#include <X11/Xlib.h>
#include <X11/Xutil.h>

main(int argc, char *argv[])
{
    Display disp;                      /* Server-ID */
    Window win;                        /* Fenster-ID */
    GC gc;                            /* Grafikkontext-ID */
    XEvent event;                     /* Speicher für ein Ereignis */
    int running = 1;

    disp = XOpenDisplay("display_name"); /* Verbindung zum X-Server */
                                         /* herstellen */
    win = XCreateSimpleWindow(disp, ... ); /* Speicher für neues */
                                         /* Fenster belegen */
    XSetStandardProperties(disp, ...);   /* Fenster bei mgr ankündigen */
    gc = XCreateGC(disp, win, 0, 0);    /* Grafikkontext erzeugen */
    XSelectInput(disp, win, ButtonPressMask | KeyPressMask | ExposureMask);
                                         /* Fenster anzeigen; */
                                         /* Expose-Ereignis schicken */
    XMapRaised(disp, win);
```

Das X Window System (3)

Grundgerüst eines X-Window Programms

```
while (running) {
    XNextEvent(disp, &event);           /* nächstes Ereignis holen */
    switch (event.type) {
        case Expose: ...; break;       /* Fenster neu zeichnen */
        case ButtonPress: ...; break;  /* Mausklick verarbeiten */
        case Keypress: ...; break;    /* Tastatureingabe verarbeiten */
    }
}

XFreeGC(disp, gc);                  /* Grafikkontext freigeben */
XDestroyWindow(disp, win);          /* Fensterspeicher freigeben */
XCcloseDisplay(disp);               /* Netzwerkverbindung trennen */
```

Abbildung 5.34: Das Grundgerüst einer X-Window-Anwendung.

Graphical User Interfaces (1)

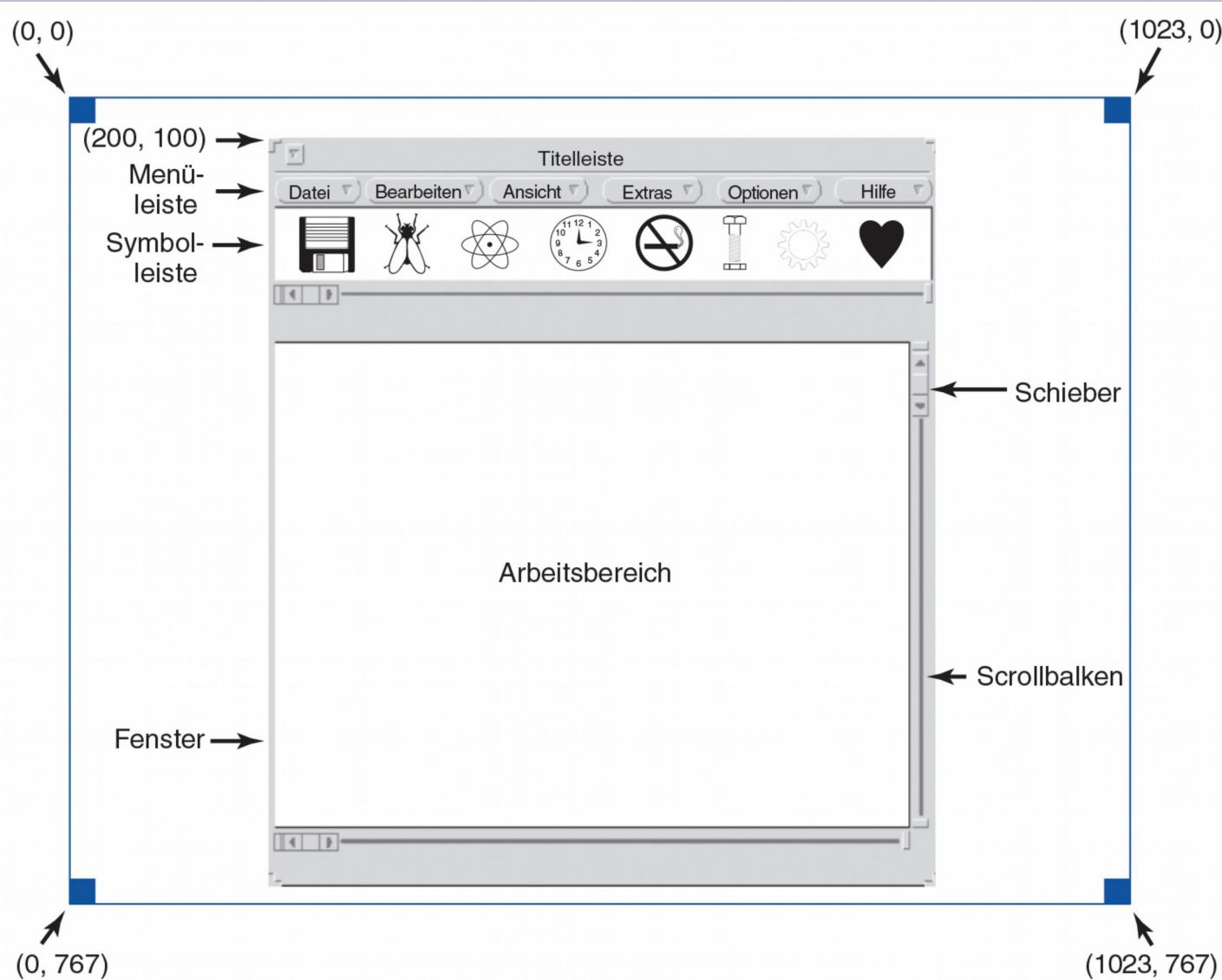


Abbildung 5.35: Beispelfenster an den Koordinaten (200, 100) auf einem Anzeigegerät mit XGA-Auflösung.

Graphical User Interfaces (2)

Grundgerüst eines MS Windows Hauptprogramms

```
#include <windows.h>

int WINAPI WinMain(HINSTANCE h, HINSTANCE, hprev, char *szCmd, int iCmdShow)
{
    WNDCLASS wndclass;           /* Klassenobjekt für dieses Fenster */
    MSG msg;                    /* ankommende Nachrichten werden hier */
                                /* gespeichert */
    HWND hwnd;                  /* Zeiger auf das Fenster-Objekt */

    /* Initialisierung von wndclass */
    wndclass.lpfnWndProc = WndProc; /* Angabe, welche Prozedur */
                                    /* aufgerufen wird */
    wndclass.lpszClassName = "Program name"; /* Text für Titelleiste */
    wndclass.hIcon = LoadIcon(NULL, IDI_APPLICATION); /* Programm-Icon laden */
    wndclass.hCursor = LoadCursor(NULL, IDC_ARROW); /* Mauszeiger laden */

    RegisterClass(&wndclass);      /* wndclass an Windows übergeben */
    hwnd = CreateWindow ( ... )    /* Speicher für das Fenster belegen */
    ShowWindow(hwnd, iCmdShow);   /* Fenster auf dem Bildschirm anzeigen */
    UpdateWindow(hwnd);          /* Anweisung an Fenster, sich zu */
                                /* zeichnen */
```

Graphical User Interfaces (3)

Grundgerüst eines MS Windows Hauptprogramms

```
while (GetMessage(&msg, NULL, 0, 0)) { /* Nachricht aus der */
    /* Warteschlange holen */
    /* Nachricht übersetzen */
    /* msg zur entsprechenden */
    /* Prozedur senden */
    TranslateMessage(&msg);
    DispatchMessage(&msg);
}

return(msg.wParam);
}

long CALLBACK WndProc(HWND hwnd, UINT message, UINT wParam, long lParam)
{
    /* Deklarationen stehen hier. */

    switch (message) {
        case WM_CREATE: ... ; return ... ; /* Fenster erzeugen */
        case WM_PAINT: ... ; return ... ; /* Fensterinhalte neu zeichnen */
        case WM_DESTROY: ... ; return ... ; /* Fenster zerstören */
    }
    return(DefWindowProc(hwnd, message, wParam, lParam)); /* Standard */
}
```

Abbildung 5.36: Grundgerüst eines Windows-Hauptprogramms.

Graphical User Interfaces (4)

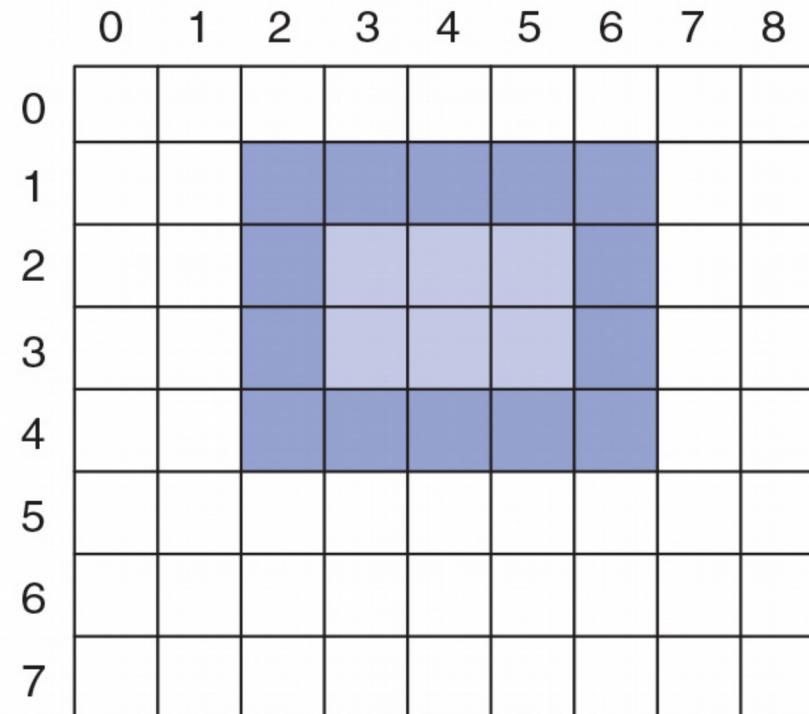


Abbildung 5.37: Ein Beispiel für ein Rechteck, das mit der Funktion *Rectangle* gezeichnet wurde. Jedes Feld stellt ein Pixel dar.

Bitmaps

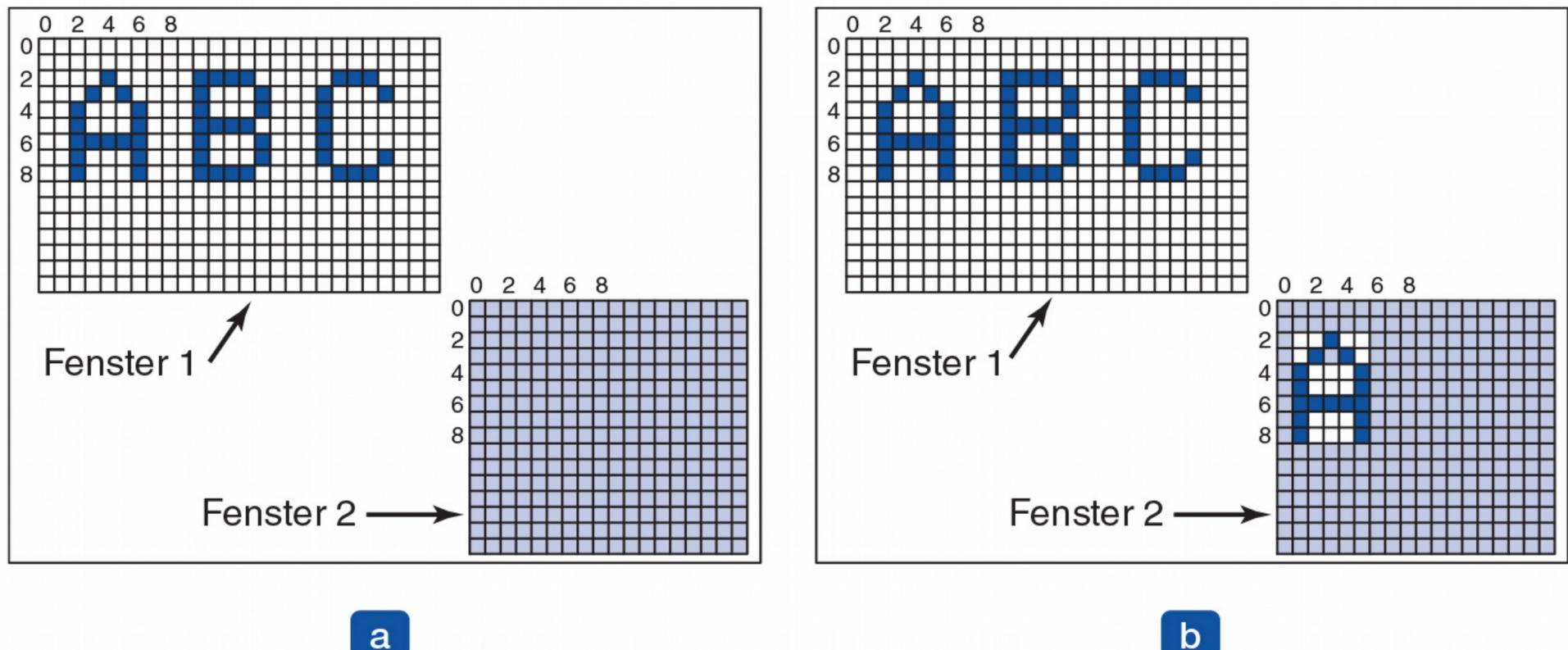


Abbildung 5.38: Kopieren von Bitmaps mithilfe von BitBlt: (a) vorher; (b) nachher.

Fonts

20 pt: abcdefgh

53 pt: abcdefgh

81 pt: abcdefgh

Abbildung 5.39: Einige Beispiele von Zeichenkonturen in verschiedenen Punktgrößen.

Thin Clients (1)

Thin Client (Lean Client oder Slim Client):

- Client (Computer oder Programm), welcher zur Erfüllung seiner Aufgaben auf die Hilfe eines Servers angewiesen ist
- Herstellerabhängige Bezeichnungen:
 - Cloud Client
 - Zero Client
 - Universal Desktop
 - Clever Client

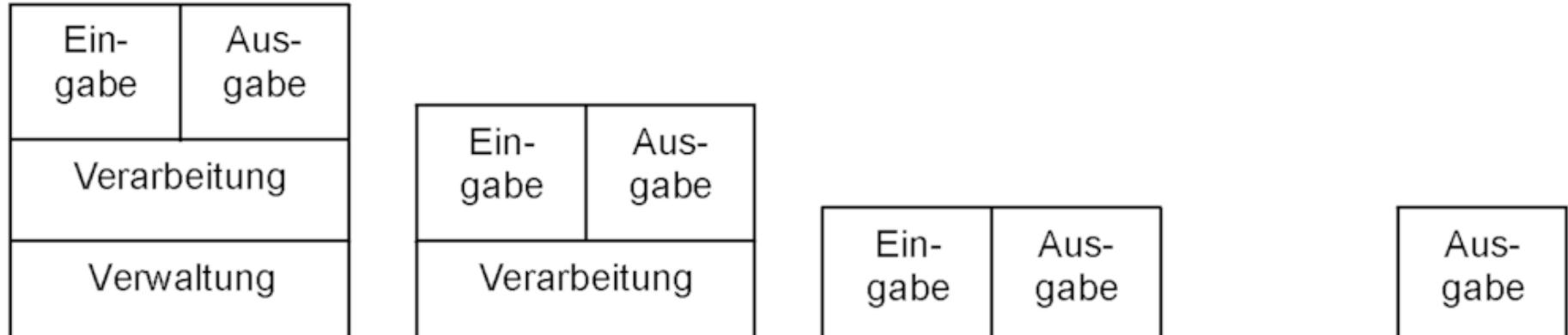
Gegensatz Fat Client (auch Full Client):

- Hard- und Software so gebaut, dass er seine eigenen Aufgaben selbst erledigen kann.

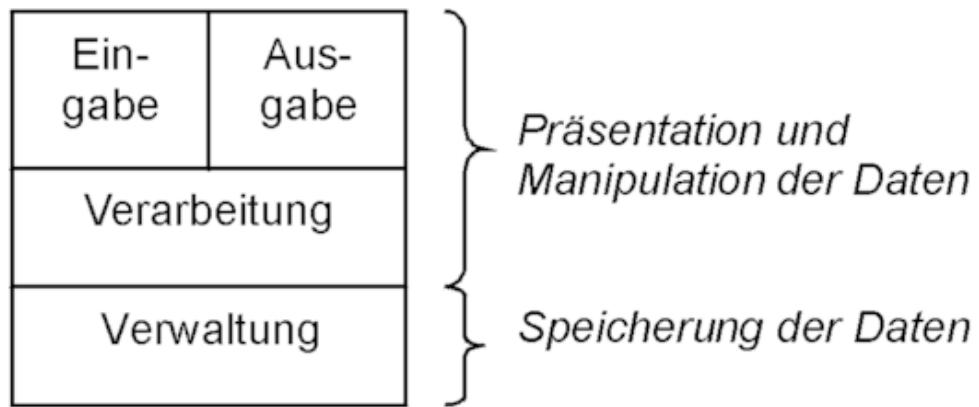
Trend im Web 2.0, Portables und Cloud-Computing: prinzipiell hin zu Thin Clients.

Thin Clients (2)

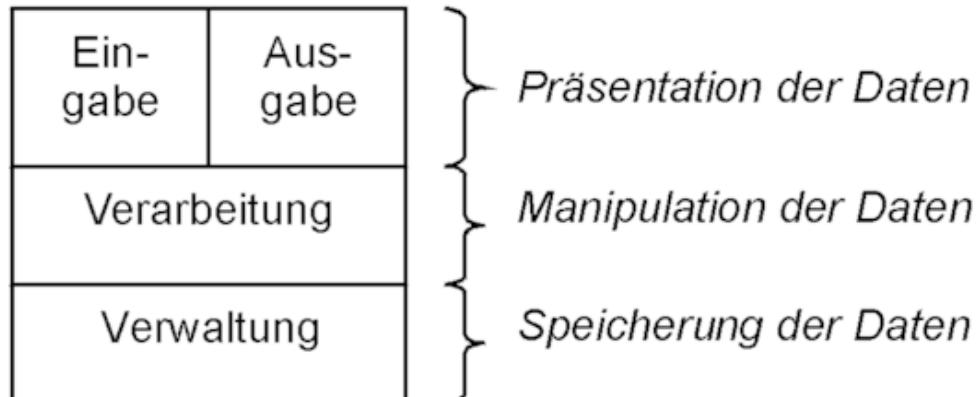
Welche Aufgaben übernimmt das Endgerät?



2 Tier Architektur:



3 Tier Architektur:



Von unbekannt - Erstellt von Florian Hoffmann am 12. März 2005.,
PD-Schöpfungshöhe, <https://de.wikipedia.org/w/index.php?curid=597730>

5.8 Energieverwaltung

5.8.1 Hardwareaspekte

5.8.2 Betriebssystemaspekte

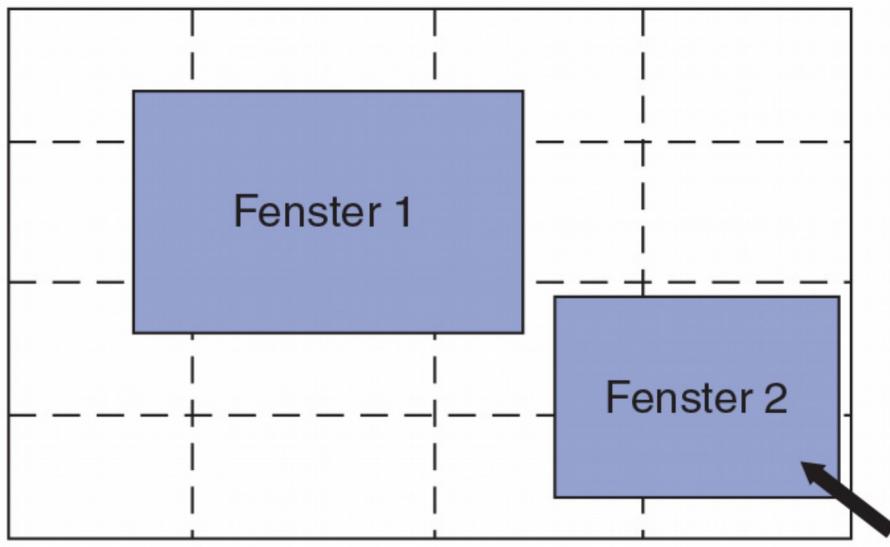
5.8.3 Energieverwaltung und Anwendungsaspekte

Hardwareaspekte

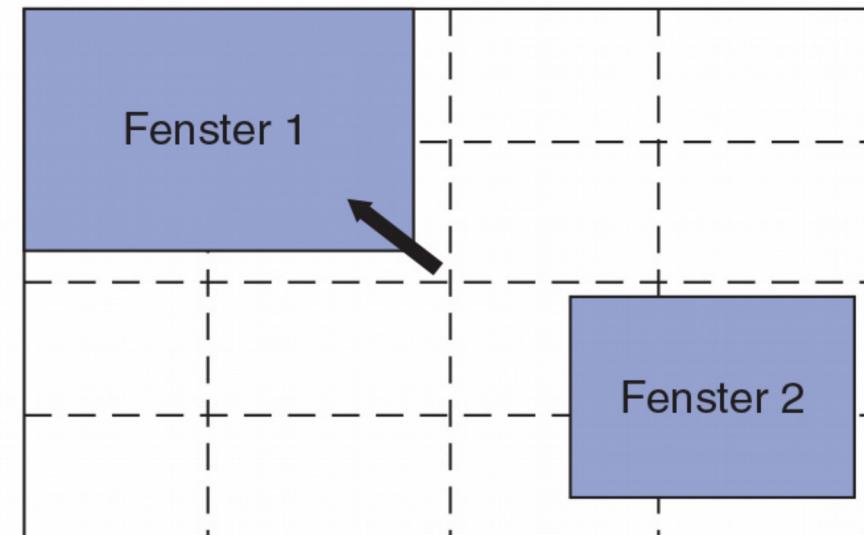
Gerät	Li et al. (1994)	Lorch und Smith (1998)
Bildschirm	68 %	39 %
CPU	12 %	18 %
Festplatte	20 %	12 %
Modem		6 %
Sound		2 %
Speicher	0,5 %	1 %
Andere		22 %

Abbildung 5.40: Energieverbrauch verschiedener Komponenten eines Notebooks.

Betriebssystemaspekte – der Bildschirm



a



b

Abbildung 5.41: Die Benutzung von Zonen für die Hintergrundbeleuchtung des Bildschirms: (a) Wenn Fenster 2 ausgewählt wird, wird es nicht bewegt. (b) Wenn Fenster 1 ausgewählt wird, wird es verschoben, um die Anzahl der beleuchteten Zonen zu reduzieren.

Energieverwaltung und Anwendungaspekte

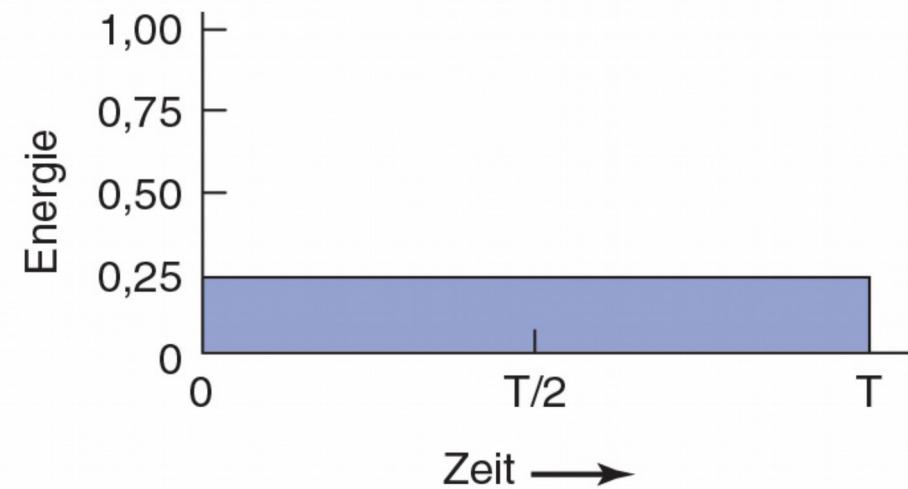
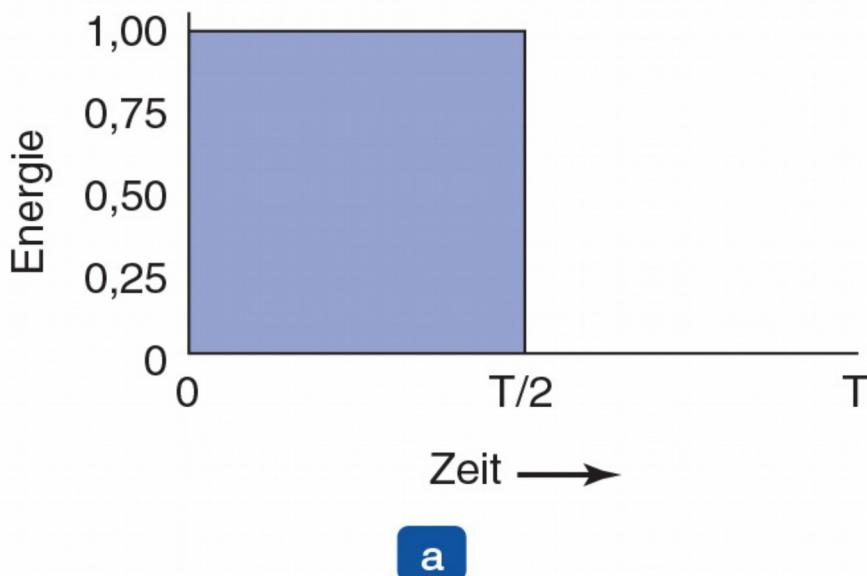


Abbildung 5.42: (a) Der Prozessor läuft bei höchster Taktrate. (b) Die Reduzierung der Spannung um die Hälfte vermindert die Taktrate um die Hälfte und senkt den Energieverbrauch auf ein Viertel.