Work in progress

Geräteverwaltung (2) / Systemstart (Booten)

Ute Bormann, TI2

2023-10-13

Inhalt

1. Ansteuerung von Terminals
2. Systemaufrufe und Gerätetreiber
3. Booten eines Unix-Systems

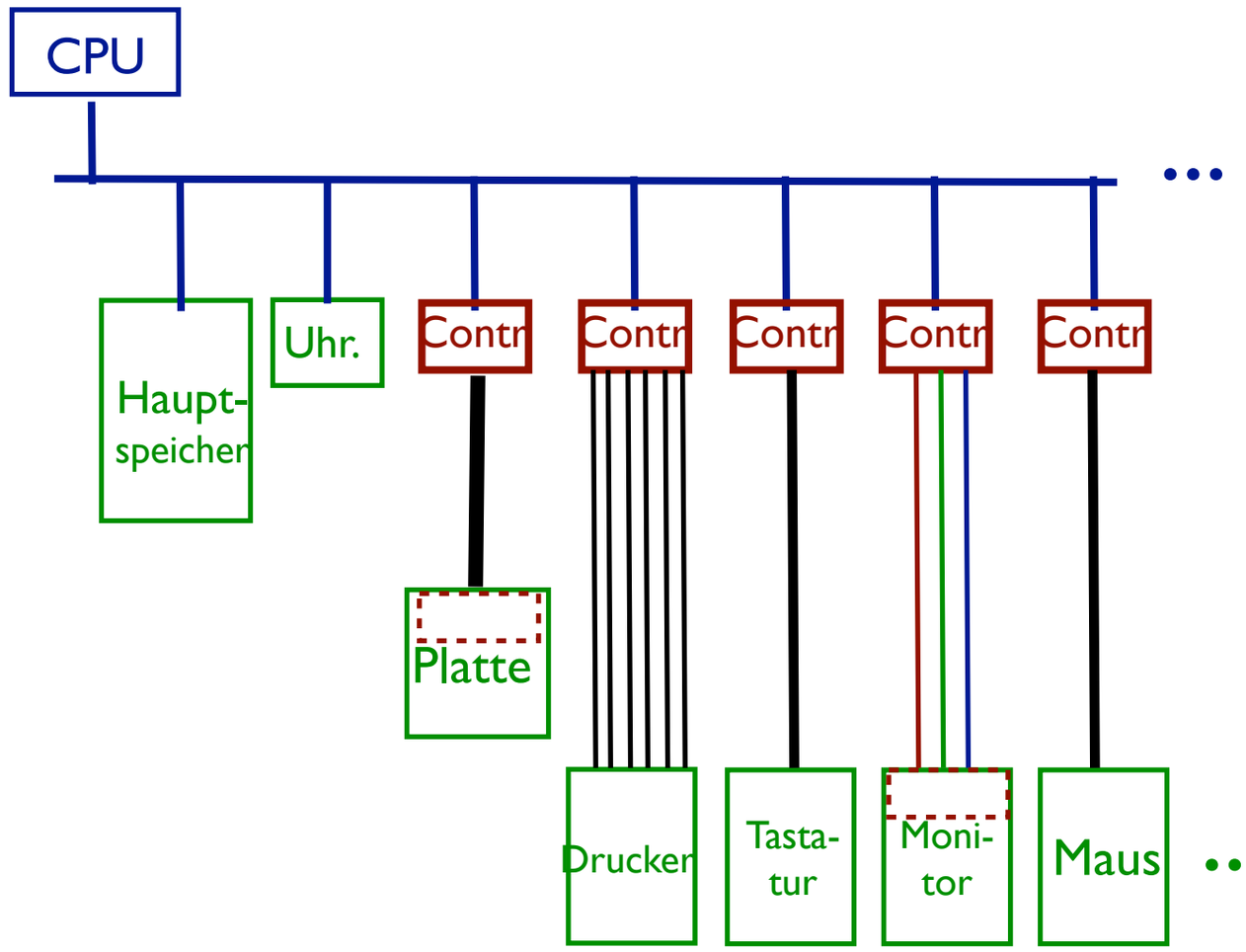
Anschluss von Geräten (vereinfacht)

- Klassischer Rechneraufbau:
  CPU über Bus mit Vielzahl von „Geräten“ verbunden

  ⇒ (in Grenzen) eigenständige Arbeit der Geräte

- Mehrere Geräte desselben Typs möglich
- U.U. mehrere Busabschnitte unterschiedlicher Länge
- Ansteuerung über Geräteadressen
- Anschlussleitungen
  - unterschiedlich „lang“
  - geräteabhängige Schnittstellen

  ⇒ Geräte-Controller:
  - Adapter zum Busanschluss
  - Ansteuerungshardware (zunehmend in Geräte integriert)



In Unix: Grobe Unterscheidung:

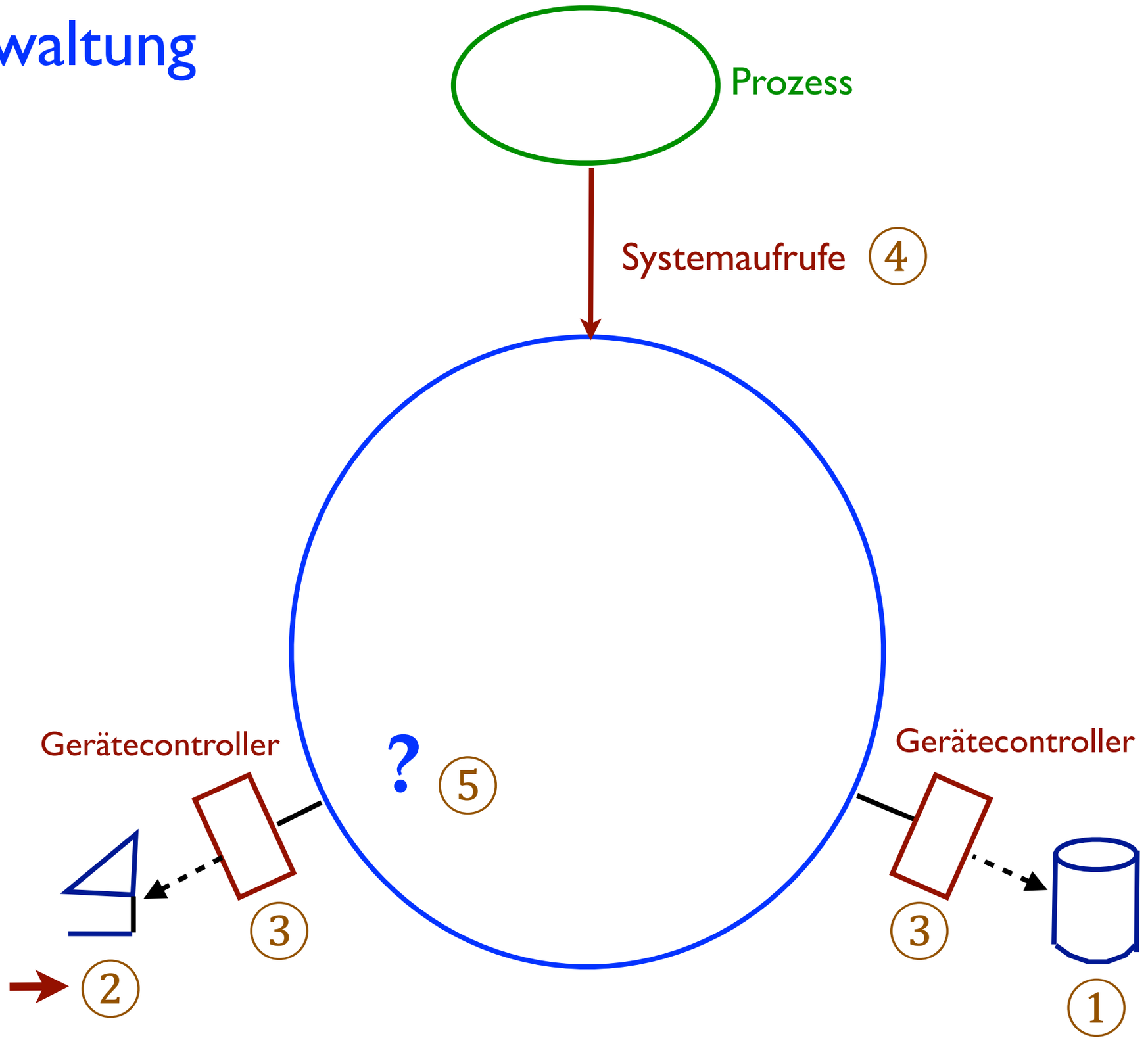
a) Blockgeräte:

- Zugriffseinheit sind Blöcke (z.B. 512 B ⇒ 4 KiB)
- I.d.R. wahlfrei adressierbar
- Beispiele: Platten, (früher auch Disketten)...

→b) Charactergeräte:

- Zugriffseinheit sind Bytes
- I.d.R. nur sequentieller Zugriff (Bytestrom)
- Beispiele: Monitor, Drucker, Tastatur,…

Überblick Geräteverwaltung



Teil 1:
Ansteuerung von Terminals

Eigenschaften von Terminals/Monitoren

Historische Entwicklung;

a) Schreibende Terminals (Teletypes, tty)

- Eingegebene Zeichen auf Papier gedruckt
- Zeilenweises Arbeiten

Eigenschaften von Terminals/Monitoren

Historische Entwicklung;

a) Schreibende Terminals (Teletypes, tty)

- Eingegebene Zeichen auf Papier gedruckt
- Zeilenweises Arbeiten

b) Glass-ttys

- Gleiches Modell wie bei a), aber Bildschirm
- i.d.R. 24+1 Zeilen à 80 Zeichen
- „Papierrolle“ durch Scrollen simuliert

⇒ Zugriff über V.24-Schnittstelle (seriell),
⇒ Fester Zeichenvorrat (ASCII)

- Schriftzeichen:
  z.B. a, b,c,…
- Steuerzeichen,
  z.B. CR, LF, BS,...

Eigenschaften von Terminals/Monitoren

Historische Entwicklung;

a) Schreibende Terminals (Teletypes, tty)

- Eingegebene Zeichen auf Papier gedruckt
- Zeilenweises Arbeiten

b) Glass-ttys

- Gleiches Modell wie bei a), aber Bildschirm
- i.d.R. 24+1 Zeilen à 80 Zeichen
- „Papierrolle“ durch Scrollen simuliert

⇒ Zugriff über V.24-Schnittstelle (seriell),
⇒ Fester Zeichenvorrat (ASCII)

c) Monitore (Memory-mapped)

- von Tastatur entkoppelt
- Punktmatrix („Picture Element“ = Pixel)
- unterschiedliche Auflösung

- Schriftzeichen:
  z.B. a, b,c,…
- Steuerzeichen,
  z.B. CR, LF, BS,...

Auflösung, z.B.:

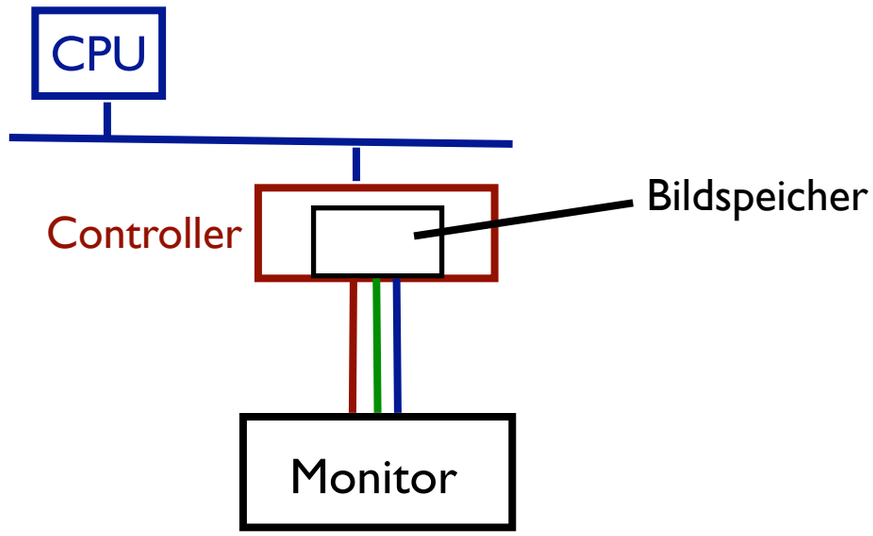
- 1024 x 768
- 1920 x 1080
- 3840 x 2160

Zugriff über Videosignal-Schnittstelle (analog ⇒ digital)

⇒ RGB-Schnittstelle ⇒ VGA ⇒ DVI / HDMI ⇒ Display Port ⇒ ...

USB-C

- Ansteuerung des „Elektronenstrahls“
  - (Positionierung)/Austastlücke...
  - Intensität
  - i.a. drei Signale für rot, grün, blau
- Regelmäßige Auffrischung nötig (Bildwiederholsequenz z.B. 60 Hz)
- Bildspeicher enthält Abbild
  ⇒ zur Auffrischung auslesen

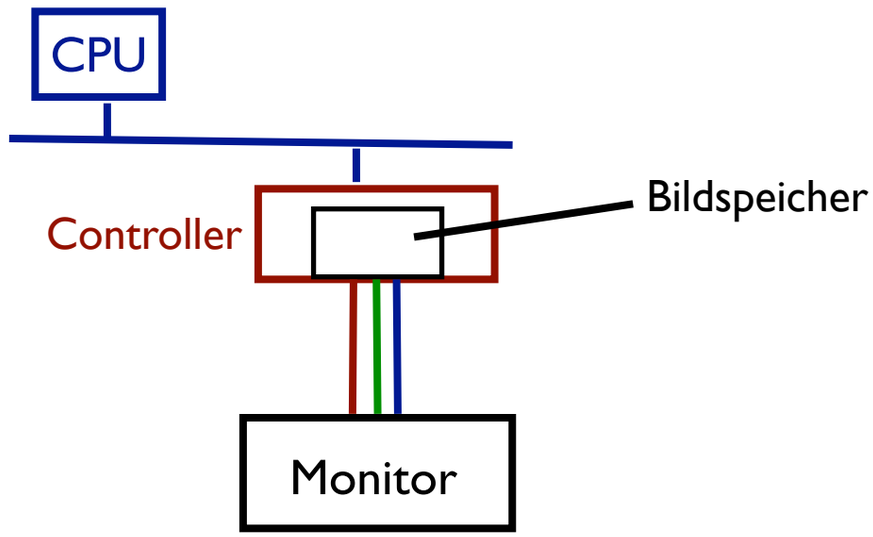


Zugriff über Videosignal-Schnittstelle (analog ⇒ digital)

⇒ RGB-Schnittstelle ⇒ VGA ⇒ DVI / HDMI ⇒ Display Port ⇒ ...

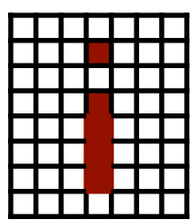
USB-C

- Ansteuerung des „Elektronenstrahls“
  - (Positionierung)/Austastlücke...
  - Intensität
  - i.a. drei Signale für rot, grün, blau



- Regelmäßige Auffrischung nötig (Bildwiederholsequenz z.B. 60 Hz)
- Bildspeicher enthält Abbild
  ⇒ zur Auffrischung auslesen
- Klassisch: Zwei Anwendungsmodi des Bildspeichers:

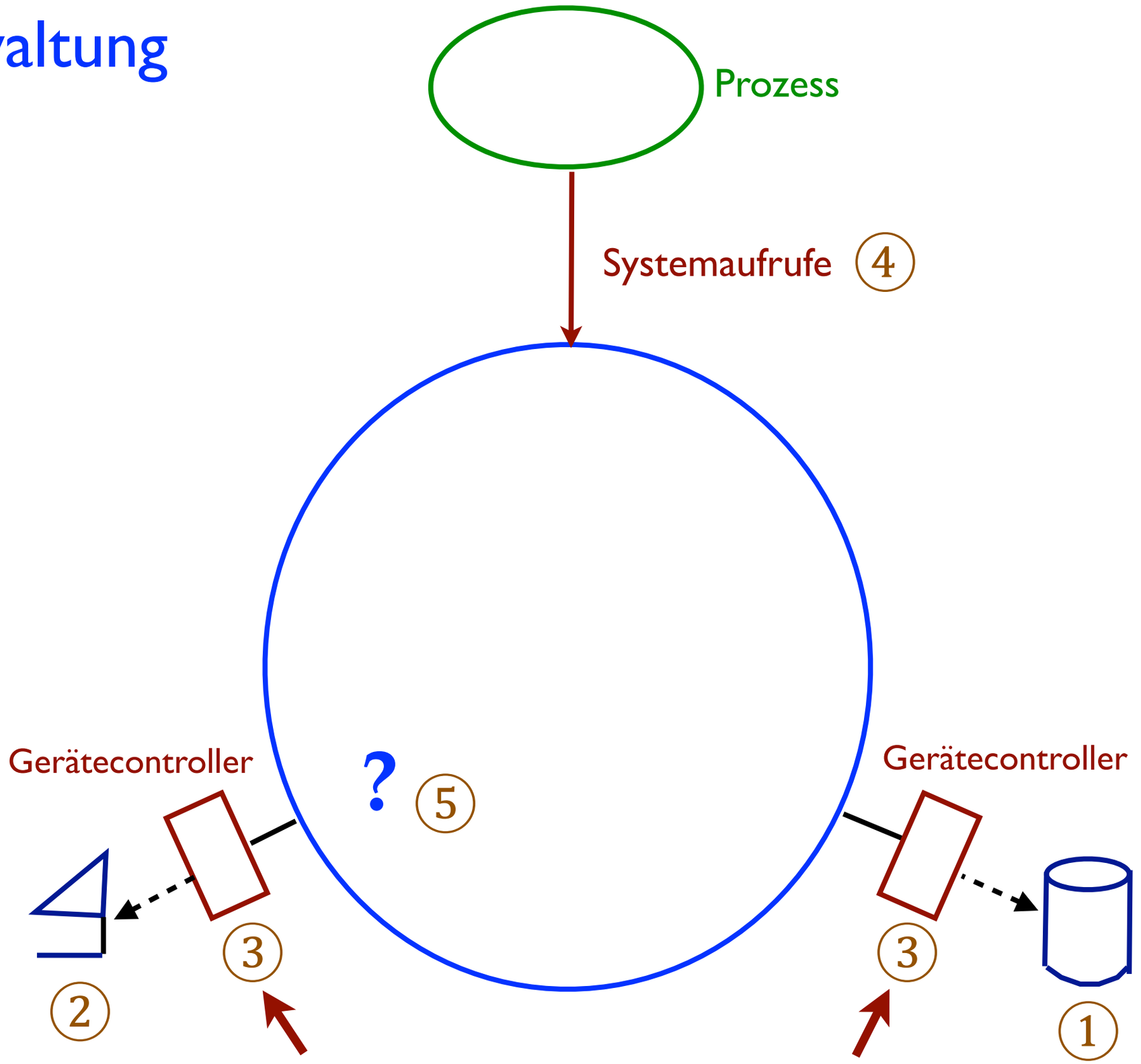
a) Früher auch: Textmode
  - Bildspeicher enthält 25x80 2-Byte-Folgen (Zeichencode + Attribute)
    ⇒ Umsetzen in Videosignal



Zeichen-Muster

b) Heute fast immer: Grafikmode
  - Bildspeicher enthält Farbe/Intensität je Pixel

Überblick Geräteverwaltung



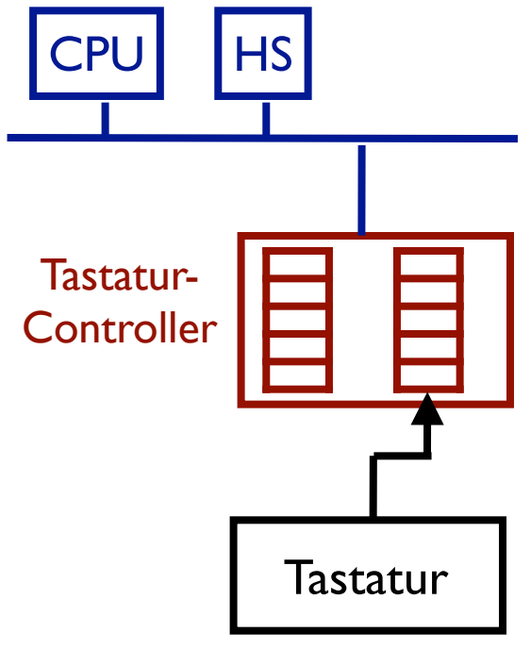
Geräte-Controller

- Ansteuerungslogik für Gerät; erzeugt z.B.
  - Impulse für Armbewegung der Platte
  - Impulse für „Elektronenstrahl“ für Bildschirm
- Ggf. Pufferbereiche für Informationen, z.B.
  - Puffer für gelesenen Plattenblock (oft mehrere)
  - Bildspeicher für regelmäßige Auffrischung

Geräte-Controller

- Ansteuerungslogik für Gerät; erzeugt z.B.
  - Impulse für Armbewegung der Platte
  - Impulse für „Elektronenstrahl“ für Bildschirm
- Ggf. Pufferbereiche für Informationen, z.B.
  - Puffer für gelesenen Plattenblock (oft mehrere)
  - Bildspeicher für regelmäßige Auffrischung
  - ggf. UARTs (Universal Asynchronous Receiver/Transmitter)

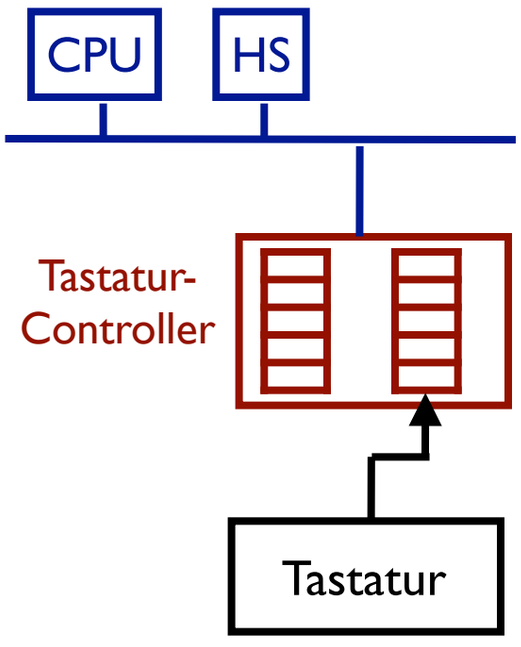
    ⇒ Parallel-/Seriellwandler für serielle Schnittstellen (Schieberegister)



Geräte-Controller

- Ansteuerungslogik für Gerät; erzeugt z.B.
  - Impulse für Armbewegung der Platte
  - Impulse für „Elektronenstrahl“ für Bildschirm
- Ggf. Pufferbereiche für Informationen, z.B.
  - Puffer für gelesenen Plattenblock (oft mehrere)
  - Bildspeicher für regelmäßige Auffrischung
  - ggf. UARTs (Universal Asynchronous Receiver/Transmitter)

    ⇒ Parallel-/Seriellwandler für serielle Schnittstellen (Schieberegister)



- Enthält eigenen „Prozessor“, kann unabhängig von CPU Aufträge abarbeiten
- Enthält Geräteregister, in dem Aufträge abgelegt werden ⇒ Aktivierung
- Enthält Register für „Ergebnis“/Fehlermeldungen
- Oft Fertigmeldung durch Interrupt

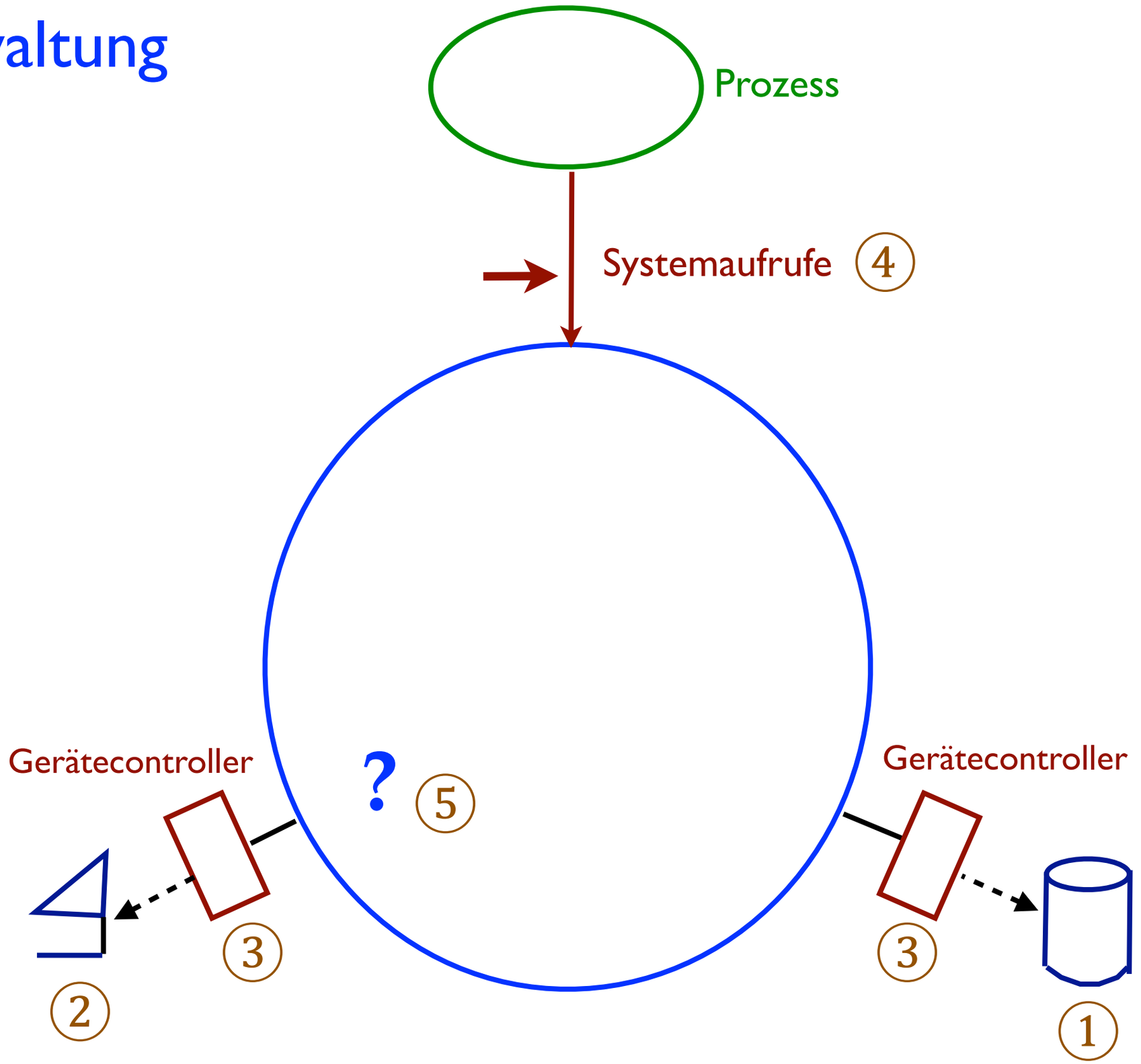
Fragen – Teil 1

- Was ein *Geräte-Controller*? Welche Aufgaben hat er?

Teil 2:
Systemaufrufe und Gerätetreiber

13 Geräteverwaltung (2), Booten, Ute Bormann

Überblick Geräteverwaltung

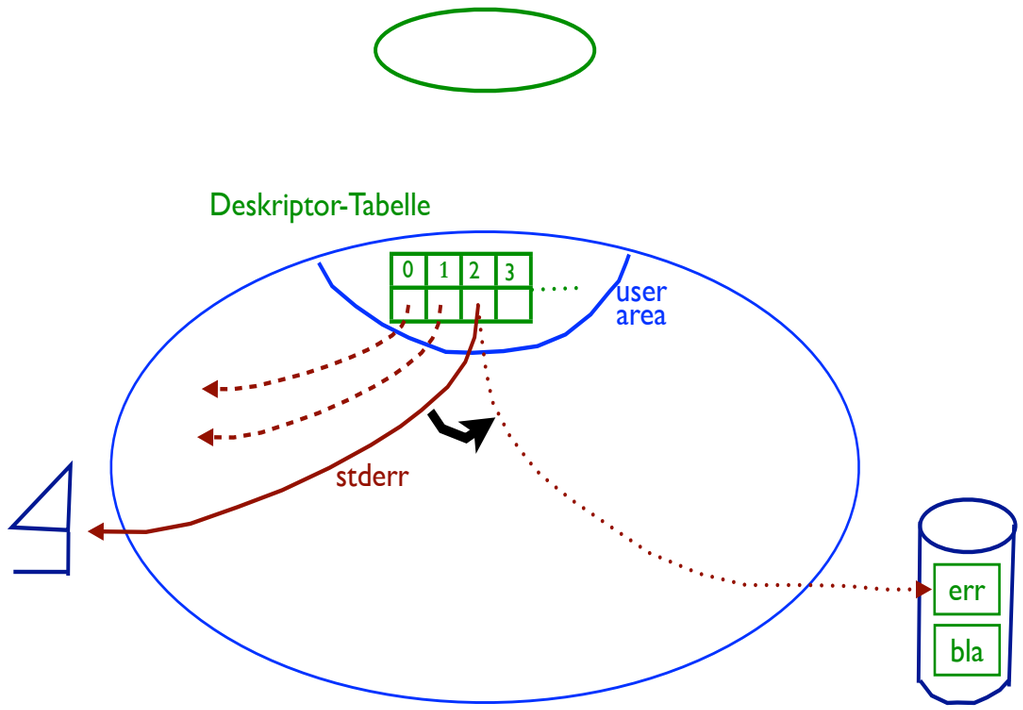


Systemaufrufschnittstelle zur Geräteansteuerung in Unix

- Unterschiede der Geräte vor Benutzer weitgehend verbergen
  ⇒ Geräteunabhängigkeit
- Vereinheitlichte Schnittstelle mit Dateizugriff

  ⇒ Geräte über spezielle Dateien (special files) ansprechen
  - mknod() (⇒ Erzeugen der Gerätedatei)
  - open() / close() (⇒ Anmelden/Reservieren bzw. Abmelden)
  - read() / write()

⇒ Ermöglicht Namen, vereinfacht Ein-/Ausgabeumlenkung

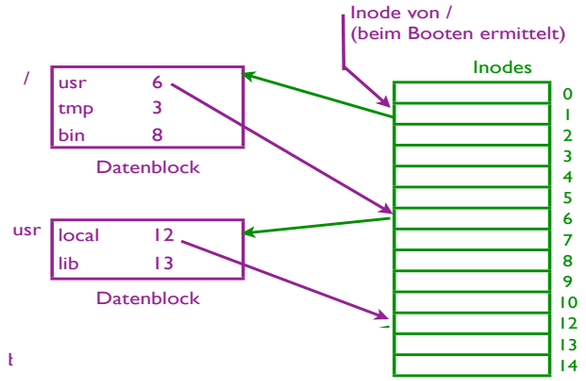


Systemaufrufschnittstelle zur Geräteansteuerung in Unix

- Unterschiede der Geräte vor Benutzer weitgehend verbergen
  ⇒ Geräteunabhängigkeit
- Vereinheitlichte Schnittstelle mit Dateizugriff

  ⇒ Geräte über spezielle Dateien (special files) ansprechen
  - mknod() (⇒ Erzeugen der Gerätedatei)
  - open() / close() (⇒ Anmelden/Reservieren bzw. Abmelden)
  - read() / write()

  ⇒ Ermöglicht Namen, vereinfacht Ein-/Ausgabeumlenkung



Kern-intern:

- open(): im Prinzip wie bei normaler Datei
  - Erzeugt Eintrag in Deskriptor-Tabelle des Prozesses (liefert fd zurück)
  - Erzeugt Eintrag in File-Tabelle
  - Liest Inode der Gerätedatei von Platte (⇒ Inode-Tabelle)
- Inode enthält keine Blocknummern sondern:
  - Major Number: Gerätetyp
  - Minor Number: Konkretes Exemplar von Gerät diesen Typs

- Über Major Number Zugriff auf Gerätetreiber

  ⇒ indiziert Device Switch Table (Block vs. Character Devices)

  ⇒ Verweise auf Zugriffsoperationen

- Grobe Unterscheidung:

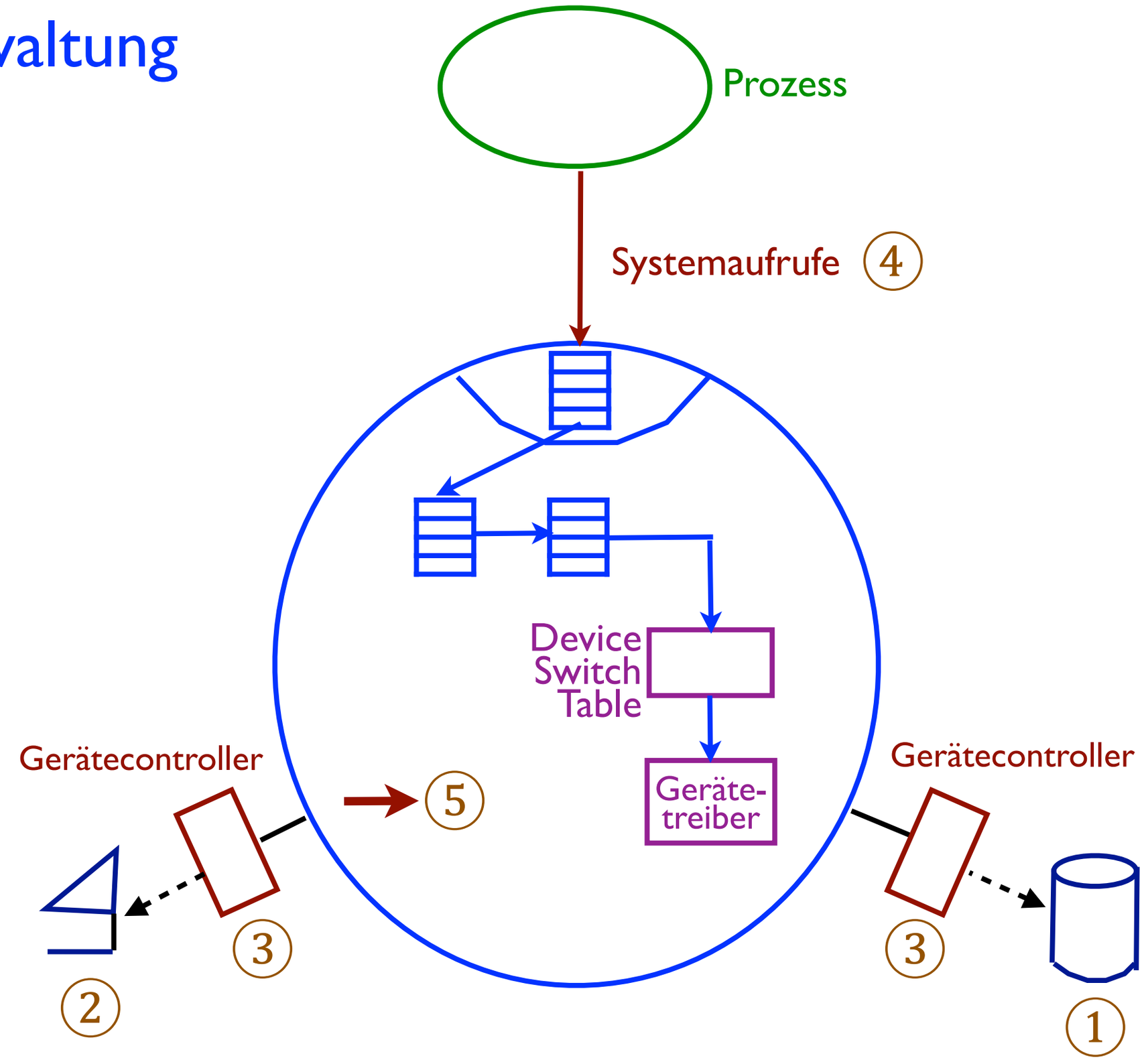
  a) Blockgerätetreiber

     open()
     close()
     strategy() (Eine Routine für read()/write() + Richtungsparameter)
     ...

  b) Charactergerätetreiber

     open()
     close()
     read()
     write()
     ioctl() (Parametrisierung des Gerätetreibers)
     ...

Überblick Geräteverwaltung

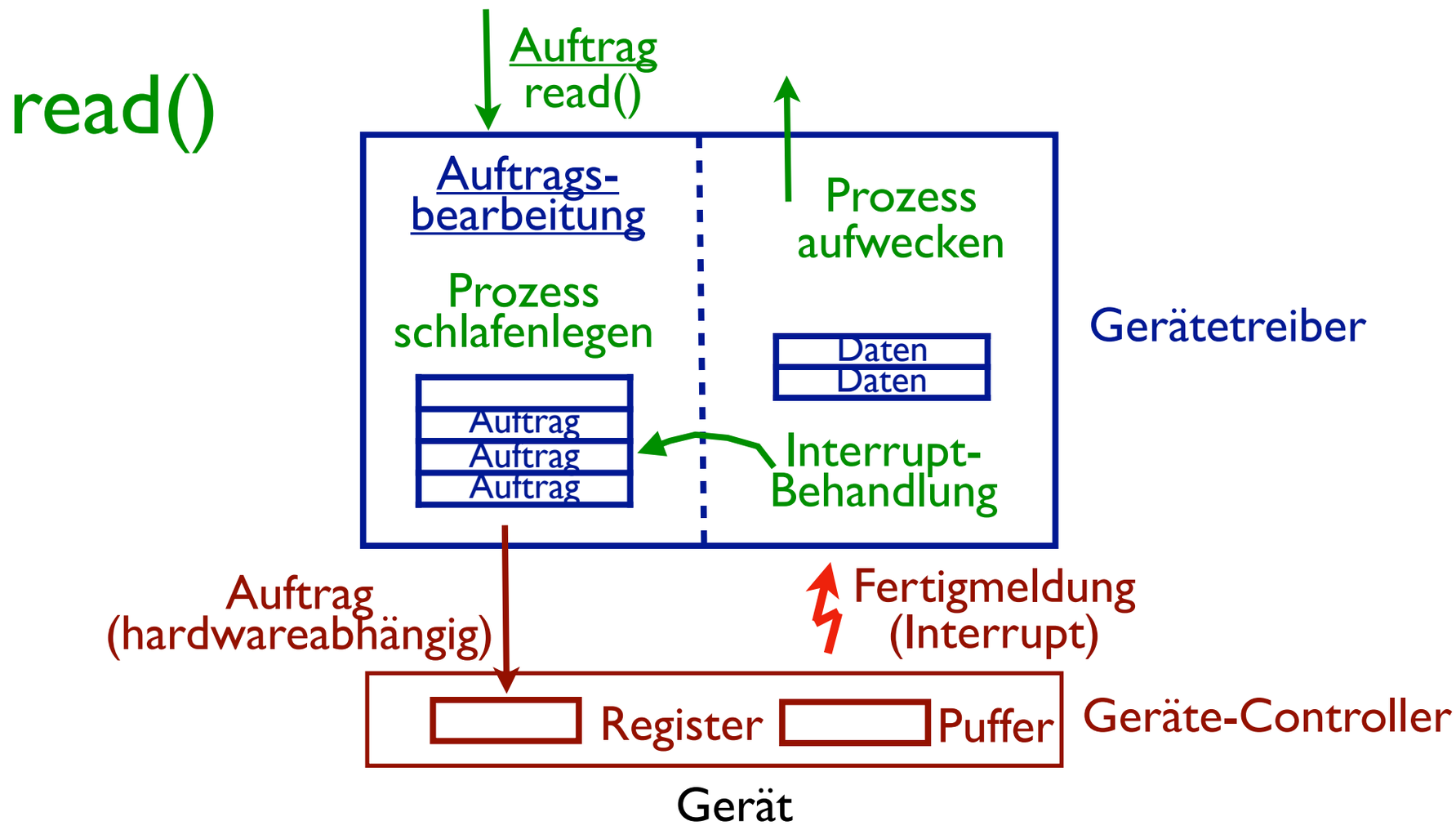


Gerätetreiber (Allgemeines)

- Code innerhalb des Betriebssystemkerns zur Geräteverwaltung
- Ein Treiber pro Gerätetyp (Major Number)
- Parametrisiert mit Minor Number zur Auswahl des konkreten Geräts

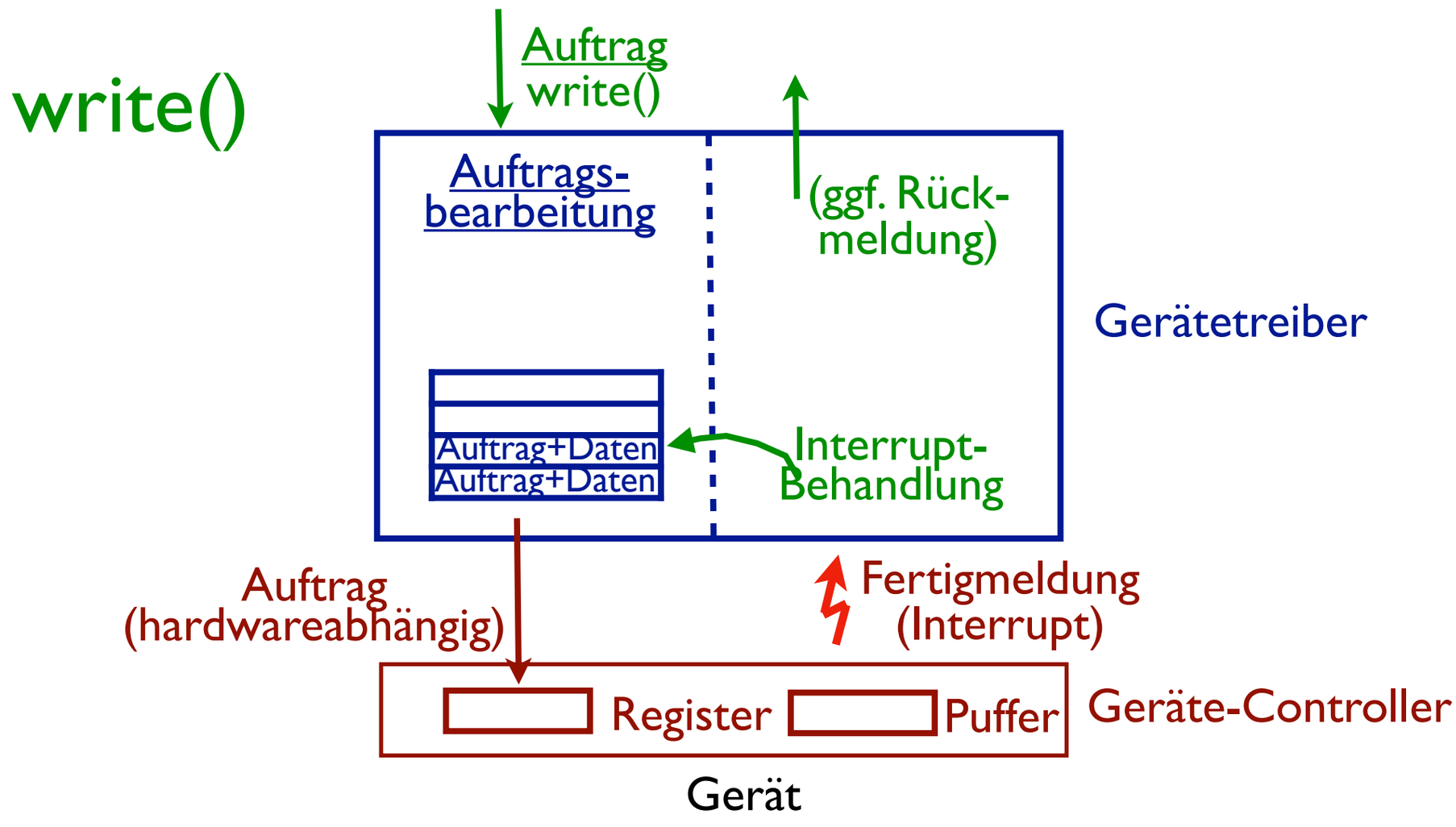
Gerätetreiber (Allgemeines)

- Code innerhalb des Betriebssystemkerns zur Geräteverwaltung
- Ein Treiber pro Gerätetyp (Major Number)
- Parametrisiert mit Minor Number zur Auswahl des konkreten Geräts
- Grundsätzliche Arbeitsweise (Beispiele: read()/write(), vereinfacht)



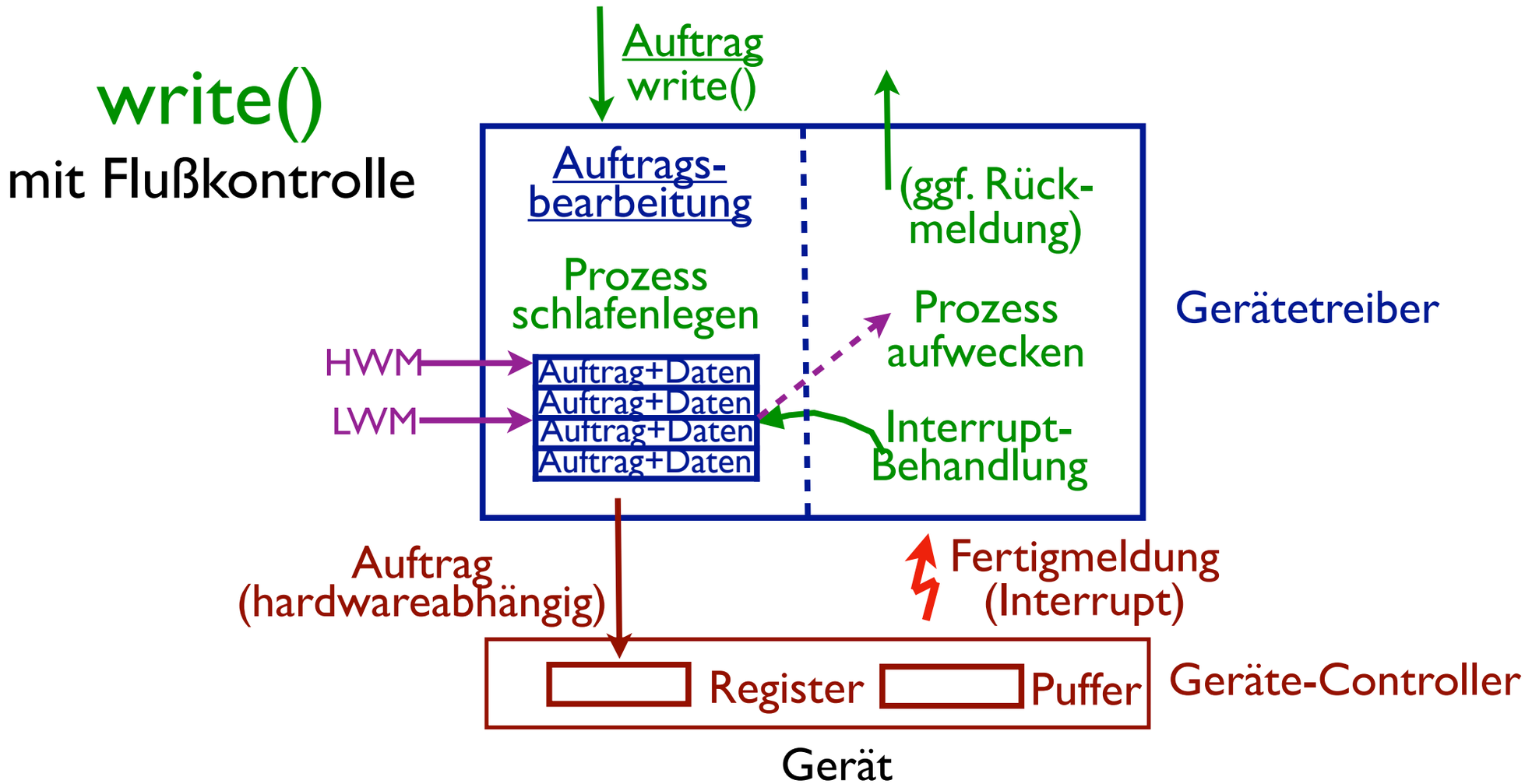
Gerätetreiber (Allgemeines)

- Code innerhalb des Betriebssystemkerns zur Geräteverwaltung
- Ein Treiber pro Gerätetyp (Major Number)
- Parametrisiert mit Minor Number zur Auswahl des konkreten Geräts
- Grundsätzliche Arbeitsweise (Beispiele: read()/write(), vereinfacht)



Gerätetreiber (Allgemeines)

- Code innerhalb des Betriebssystemkerns zur Geräteverwaltung
- Ein Treiber pro Gerätetyp (Major Number)
- Parametrisiert mit Minor Number zur Auswahl des konkreten Geräts
- Grundsätzliche Arbeitsweise (Beispiele: read()/write(), vereinfacht)



Aufgaben des Gerätetreibers

- Entgegennehmen von Aufträgen und Absetzen im Geräte-Controller (Hardware-abhängig)
- Entgegennehmen der Fertigmeldungen (Interrupts)

  ⇒ ggf. Controller-Puffer auslesen

  ⇒ ggf. Meldung an Prozess
- Ggf. Sammeln von Aufträgen und „Ergebnismeldungen“

  ⇒ Verwaltung von Warteschlangen

  ⇒ weitergehende Entkopplung von Prozessen und Geräten
  - Folge: Nach Fertigmeldung automatisch „nächster“ Auftrag absetzbar
  - (Achtung: nicht zwingend FCFS ⇒ z.B. Fahrstuhlalgorithmus bei Platten)
- Ggf. „Flusskontrolle“ der Warteschlangen
  - High Water Mark (maximaler Füllstand)
  - Low Water Mark („minimaler“ Füllstand)

    ⇒ ggf. Prozess informieren ⇒ Nachfüllen

Abgrenzung der Aufgaben von Gerätetreiber und Geräte-Controller nicht scharf.
Verschiedene Formen:

a) Programmed I/O: (z.B. Tastatur)

- I/O-Ablauf im Grundsatz wie oben beschrieben
- „Kontrolle“ bleibt beim Treiber (CPU)
- Sonderfall Tastatur-Treiber: Keine expliziten Leseaufträge an Controller…

b) DMA (Direct Memory Access) (z.B. Platte)

- Controller greift nach Anstoß selbst auf Hauptspeicher zu und übernimmt notwendige Kopiervorgänge
- Auftrag muss entsprechende Speicherbereiche angeben

  ⇒ müssen verfügbar sein
- CPU weniger belastet

c) (gibt auch andere Formen, Mischformen...)

- …

Terminal-Treiber

- Können in verschiedenen „Modes“ laufen (z.B. Übertragungsrate bit/s)
  ⇒ Parametrisierung des Treibers

Terminal-Treiber

- Können in verschiedenen „Modes“ laufen (z.B. Übertragungsrate bit/s)
  ⇒ Parametrisierung des Treibers

- Grundsätzliche Unterscheidung:

  a) Raw-Modus:
     - Tasteneingabe wird unverändert an Prozess durchgereicht (zeichenweise)

  b) Cooked-Modus: (+ Varianten)
     - Tasteneingabe wird zeilenweise an Prozess gereicht
     - Einige Tasten(kombinationen) im Treiber abgefangen (Sonderbedeutung)
       - Zeileneditierfunktionen: z.B. BS, DEL, ctrl-u...
       - Zeilenende: z.B. CR/LF (+ Varianten)
       - Flusskontrolle: ctrl-s (stoppt Ausgabe), ctrl-q (weiter)
       - Signale: z.B. ctrl-c (SIGINT)

- Verwaltung des Cooked-Modes (weitgehend) geräteunabhängig
  ⇒ Auslagerung in Line Disciplines
  ⇒ von unterschiedlichen Terminal-Treibern benutzbar
  (parametrisiert)

- Parametrisierung erfolgt über Systemaufruf ioctl()
- Im Shell-Prozess über Kommando stty aktivierbar
- stty -a Anzeige der aktuellen Einstellungen

- Vielzahl von Parametern:
  - Signalrate: z.B. 38400 baud
  - Echoverhalten: ...
  - Zeichenumwandlungen: z.B. Zeilenende vs. CR/LF
  - Flusskontrolle: ...
  - Raw mode vs. Cooked mode
  - Tastenkombinationen für Signale: ctrl-c, ctrl-z, ...

Kleine Aufgabe

Gegeben sei der folgende Auszug aus einer Ausgabe des Kommandos stty -a. Was könnte sie bedeuten?

```
speed 38400 baud; rows 26; columns 80;

intr = ^C; kill = ^U; eof = ^D; start = ^Q; stop = ^S; susp = ^Z;

icrnl iutf8

isig echo –echoctl
```

Fragen – Teil 2

- Welche Vorteile bietet eine vereinheitlichte Betriebssystemschnittstelle zum Zugriff auf Geräte? Wie sieht sie in Unix in etwa aus?
- Was ist ein *Gerätetreiber*? Welche Aufgaben hat er?
- Warum erfolgt der Zugriff auf Geräte häufig über Warteschlangen?
- Worin unterscheidet sich *DMA (Direct Memory Access)* von *Programmed I/O*?
- Warum werden Terminal-Treiber in Unix parametrisiert? Nenne typische Parameter.

Teil 3:
Booten eines Unix-Systems

Betriebssysteme

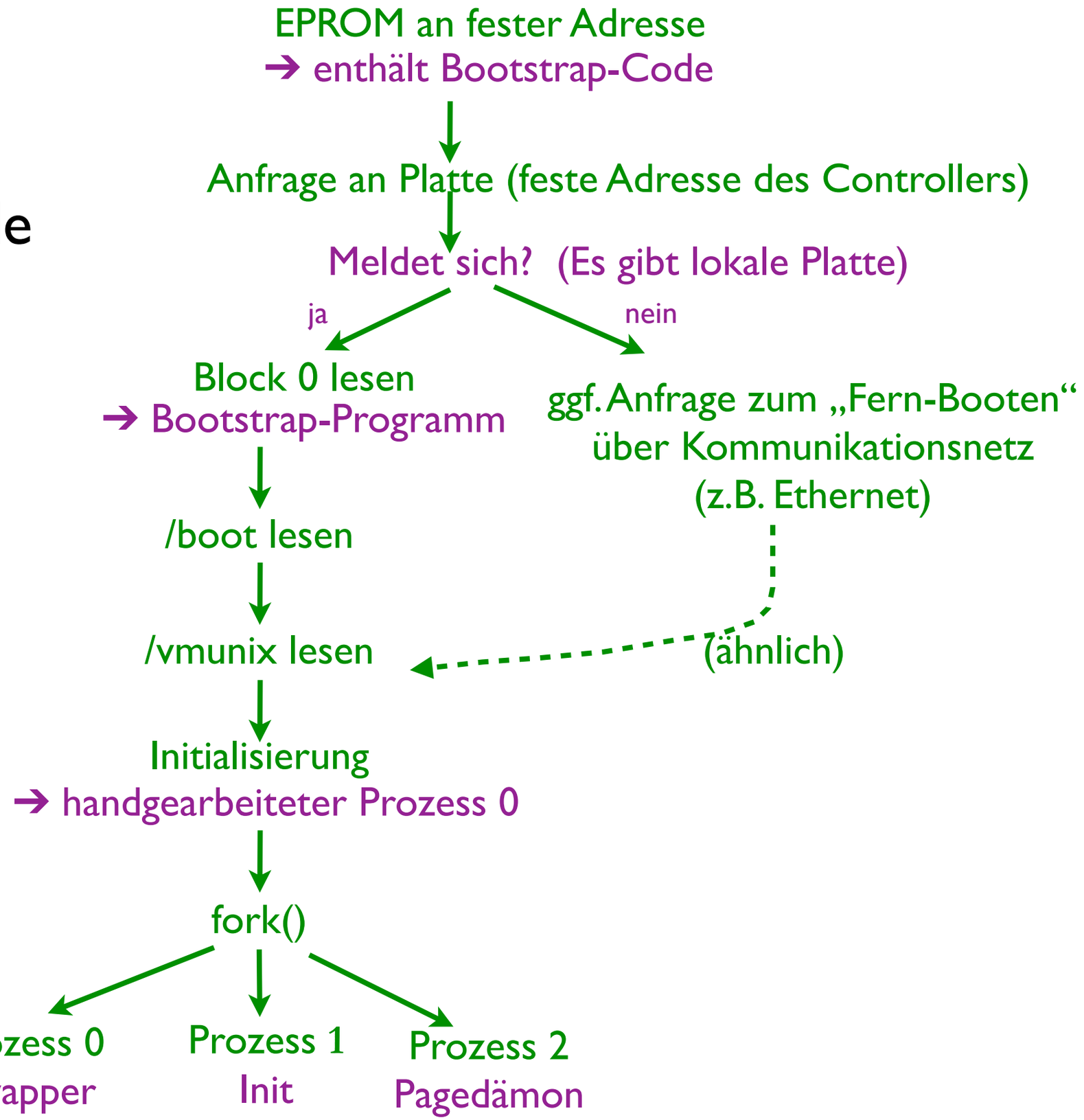
- Prozessverwaltung
- Speicherverwaltung
- Dateiverwaltung
- Geräteverwaltung

⇒ • „Hochfahren“ (Booten) / „Runterfahren“ eines Systems

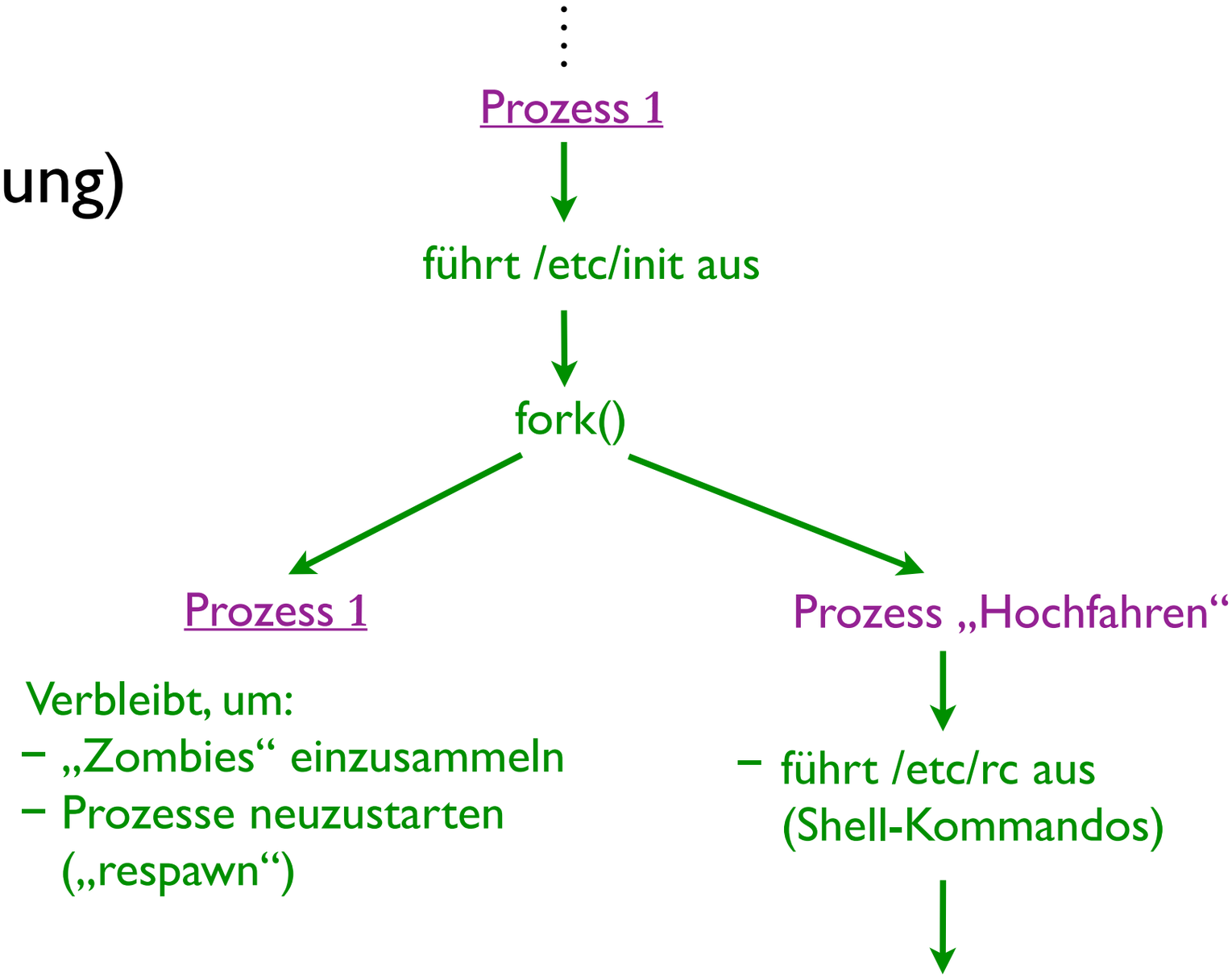
- Prozessverwaltung
  ⇒ Nebenläufigkeit ⇒ Kommunikation

„Booten“ eines Unix-Systems (vereinfacht)

- Feinheiten stark systemabhängig
- Ungefähr folgende Schritte:



(Fortsetzung)



– Prozesse für Terminals starten (getty/login ➔ sh)

„Runterfahren“ des Systems

- Ausgelöst z.B. durch Shell-Kommando `shutdown`
- Signal an Prozess 1 senden
- Daraufhin:
  - Prozess 1 führt kein „Respawn“ mehr durch
  - Sonstige Prozesse „killen“
  - Systemaufruf `sync()`
  - Systemaufruf `halt()`

Fragen – Teil 3

- Nenne einige wesentliche Aufgaben beim Booten eines Unix-Systems.

Zusammenfassung

- Eigenschaften von Terminals/Monitoren
- Gerätecontroller
- Systemaufrufe zur Ansteuerung von Geräten
- Gerätetreiber
- „Hochfahren“ / „Runterfahren“ eines Unix-Systems

Geräteverwaltung 2 / Booten – Fragen

1. Was ein *Geräte-Controller*? Welche Aufgaben hat er?
2. Welche Vorteile bietet eine vereinheitlichte Betriebssystemschnittstelle zum Zugriff auf Geräte? Wie sieht sie in Unix in etwa aus?
3. Was ist ein *Gerätetreiber? Welche Aufgaben hat er?*
4. Warum erfolgt der Zugriff auf Geräte häufig über Warteschlangen?
5. Worin unterscheidet sich *DMA (Direct Memory Access)* von *Programmed I/O*?
6. Warum werden Terminal-Treiber in Unix parametrisiert? Nenne typische Parameter.
7. Nenne einige wesentliche Aufgaben beim Booten eines Unix-Systems.