Fachhochschule Vorarlberg

Betriebssysteme

Informatik

WS13/14

Prozesse:

Prozessverwaltung

Erzeugung/ Hierarchie/ PCB/ Zustände

Armin Simma

Vorlesungs-„Nachlese“

- Folien S.51-62: Lesen Sie Stallings Kapitel 3.2. (4.Auflage): „Prozessbeschreibung“
- Folien S.63-76: Stallings Kapitel 3.1. „Prozesszustände“
- Folie S.77 Stallings Kapitel 9.1 „Arten der Prozessorzuteilung“
- Folien S.78-Ende Stallings Kapitel 9.2: „Scheduling-Algorithmen“
- Folie S.77: Stallings Kapitel 9.1 „Arten der Prozessorzuteilung

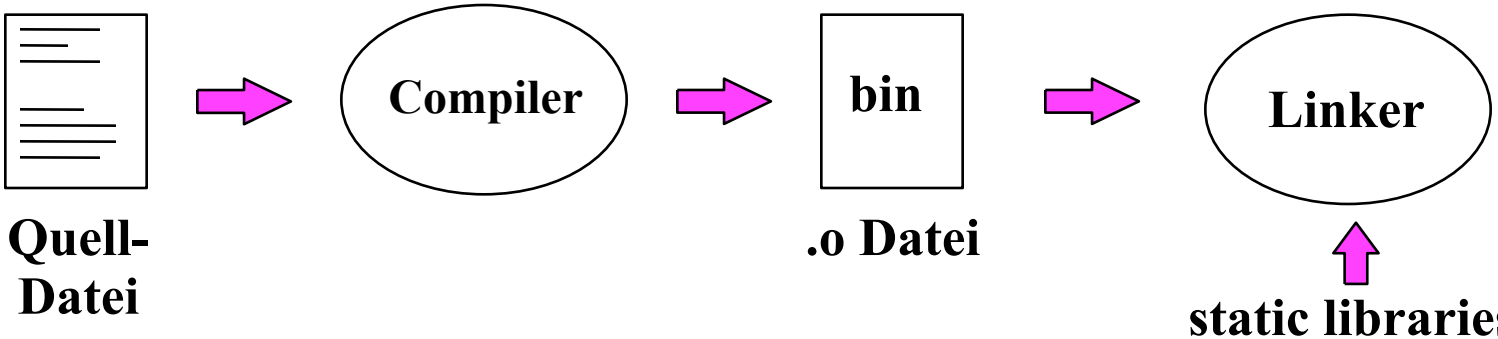
Steuerung ablaufender Programme

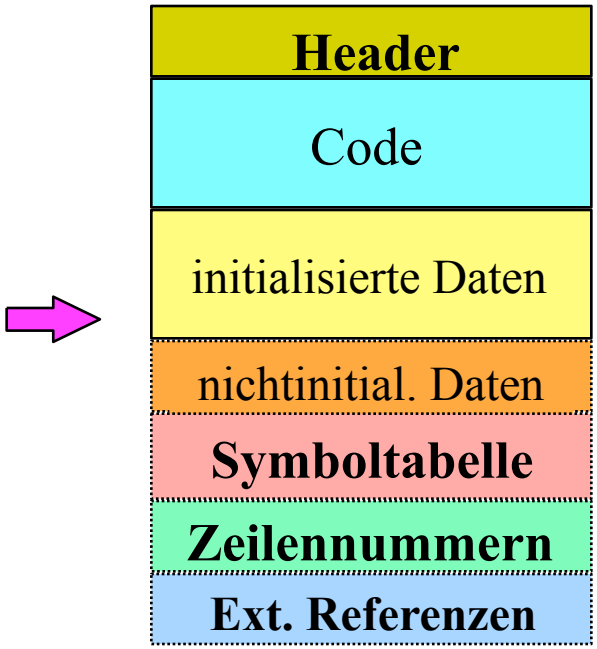
- Ein Betriebssystem startet und verwaltet unterschiedliche Arten von Programmen, die alle als Prozesse ablaufen:
  - Benutzerprogramme (interaktiv)
  - Batch-Jobs
  - Skriptprogramme, z.B. Shell-Scripts
  - Systemprogramme, z.B. Print Spooler, Name Server, File Server.
- Das Betriebssystem steuert den Ablauf sämtlicher Prozesse

Programme und Prozesse

- **Programm**: Formulierung eines Algorithmus; endliche Beschreibung auszuführender Operationen; besteht aus:
  - **Code**: Folge von Maschineninstruktionen
  - **Daten**: Variable, die im Speicher abgelegt und verändert werden
- **Prozess**: eine sich **in Ausführung** befindliche Instanz eines (Teil-)Programms; erfordert:
  - **Speicher**, um Programmcode und Daten aufzunehmen
  - **CPU-Register** für die Ausführung
  - Den **Prozessor** zur Ausführung
  - Eventuell weitere **Resourcen** (Platte/ Drucker/ NW…)

Erzeugen eines (C-) Programms





Ausführbare Datei:
muss standardisiertem Aufbau folgen, z.B. ELF (Linux) PE (Windows), XCOFF (AIX), EXEC (BSD)

Der Programmstart

- Betriebssystem erzeugt einen Prozess und reserviert dafür Speicherplatz
- Der Lader:
  - liest die ausführbare Datei ein
  - richtet den Speicherplatz für den Prozess ein, in dem Code und Daten der ausführbaren Datei liegen.
  - legt ggf. vorhandene Übergabeparameter auf den Stack
    - z.b. Werte für Argumente (argc, argv)
  - richtet die CPU-Register ein
- Programmstart erfolgt durch Aufruf von main()
- Nach Rückkehr von main() wird
  - return-Wert an BS (shell) zurückgegeben
  - der Prozess beendet und die Betriebsmittel freigegeben.

Generierung von Prozessen

Wann wird ein neuer Prozess erzeugt?

- **Neuer Stapelauftrag**: Betriebssystem wird über Stapeleingabe mit neuem Auftrag versorgt.
- **Interaktives Einloggen**: Neuer Nutzer meldet sich am System an.
- **Dienstleistungsprozesse**: im Auftrag eines Anwendungsprozesses wird ein neuer Prozess erzeugt; z.B. Druckauftrag.
- **Kindprozesse**: Ein Prozess initiiert selbst neue Prozesse; ermöglicht Modularität und Parallelität; z.B. Serverprozess generiert für jede Anfrage, die er erhält, einen neuen Prozess.
  - Benutzer startet neues ausführbares Programm (z.b. ./a.out)
  - ist dann Kind-Prozess der shell (bash)

Generierung von Prozessen
Prozesshierarchie

- Ein Betriebssystem muss eine Möglichkeit bereitstellen, Prozesse zu erzeugen.
- Daraus resultiert das Konzept der **Prozesshierarchie**:

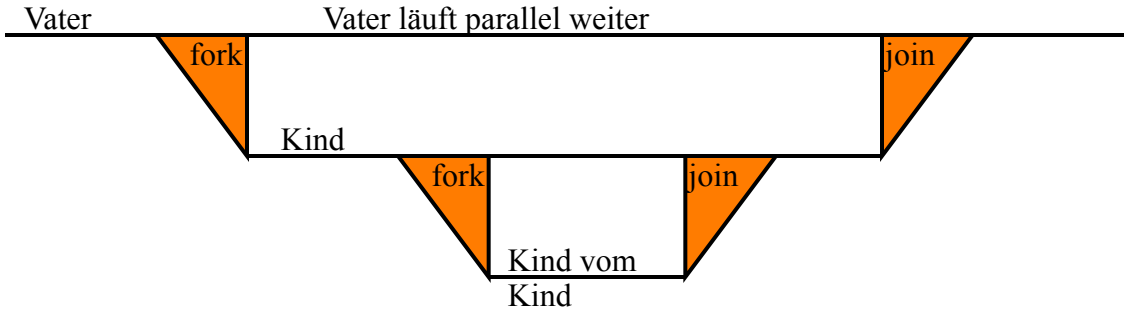
  Ausgehend von einem laufenden Prozess können Prozesse in der Regel neue Prozesse erzeugen: Kindprozesse des Vaterprozesses

➔ Prozesshierarchie:

Generierung von Prozessen
Prozesshierarchie

- UNIX: „*fork*()“ erzeugt exakte Kopie vom aufrufenden Prozess
  - Adressraum v. Prozess mit Inhalt v. neuem Programm ersetzt.
  - Vater- und Kindprozess setzen ihre Ausführungen „parallel“ fort.
  - Bei mehrfachen *fork()*-Aufrufen lassen sich mehrere Kindprozesse parallel zum Vaterprozess starten.
  - Kindprozesse können selbst wieder Kindprozesse starten.

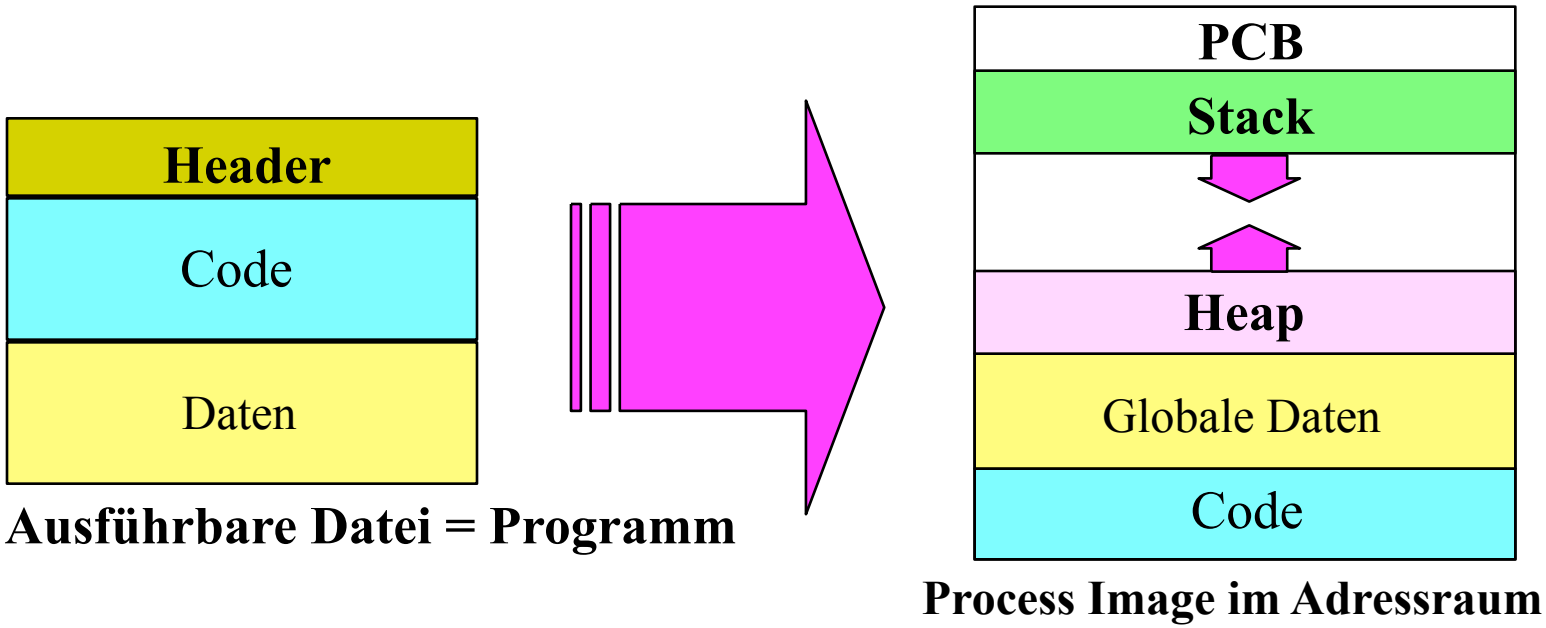
Prozesshierarchie:



Terminierung von Prozessen:

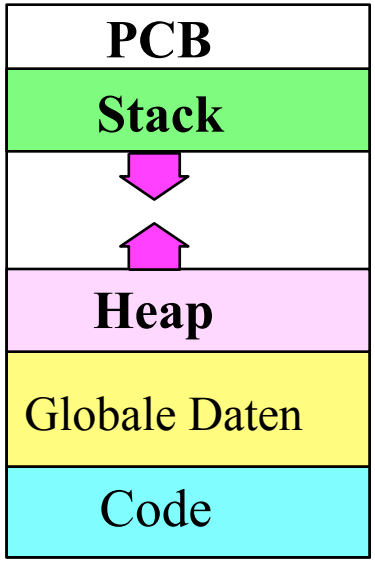
- jedes Betriebssystem stellt Instruktionen zum Anhalten eines Prozesses zur Verfügung
- ein Kindprozess kann durch den Vaterprozess beendet werden, bzw. wird durch das Ende des Vaterprozesses automatisch beendet
- Fehler können zur Terminierung von Prozessen führen.

Adressraum eines Prozesses
(Vereinfacht)



- **Heap**: Dynamisch allokierte Daten (siehe Strukturierte Progr: Pointer, Referenz, malloc(), new etc. folgt noch)
- **Stack**: alle (funktions)lokalen Daten, d.h Variablen innerhalb Funktion. Diese werden erst bei Funktionsaufruf angelegt, darum dynamisch.
- Andere **Daten** sind globale Variablen (für alle Funktionen sichtbar und bei Prozessstart schon fix)

Adressraum eines Prozesses
(Vereinfacht)



Process Image im Adressraum

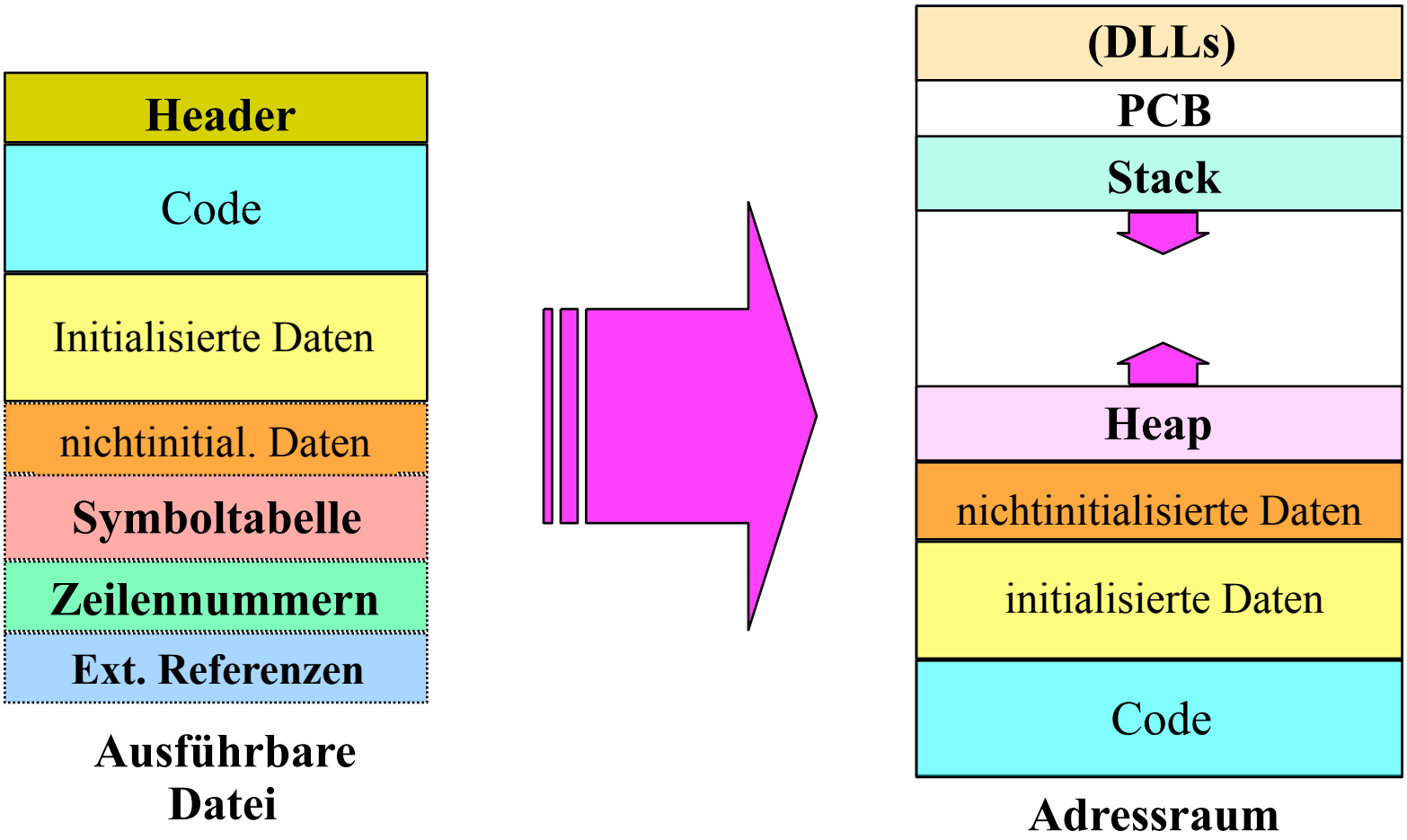
Process Image: Gesamtheit der physischen Bestandteile eines Prozesses; es enthält:

- **Code**
- **Daten**
- **Prozesskontrollblock** (PCB): Daten zur Kontrolle des Prozesses durch das Betriebssystem (s.u.).

- Das Process Image wird im (Haupt-) Speicher abgelegt.

- Damit verfügt jeder Prozess über einen **eigenen** Adressraum, der vor anderen Prozessen geschützt ist.

Adressraum eines Prozesses



Ablauf von mehreren Prozessen

- 3 Prozesse A,B,C und Betriebssystem (Dispatcher)

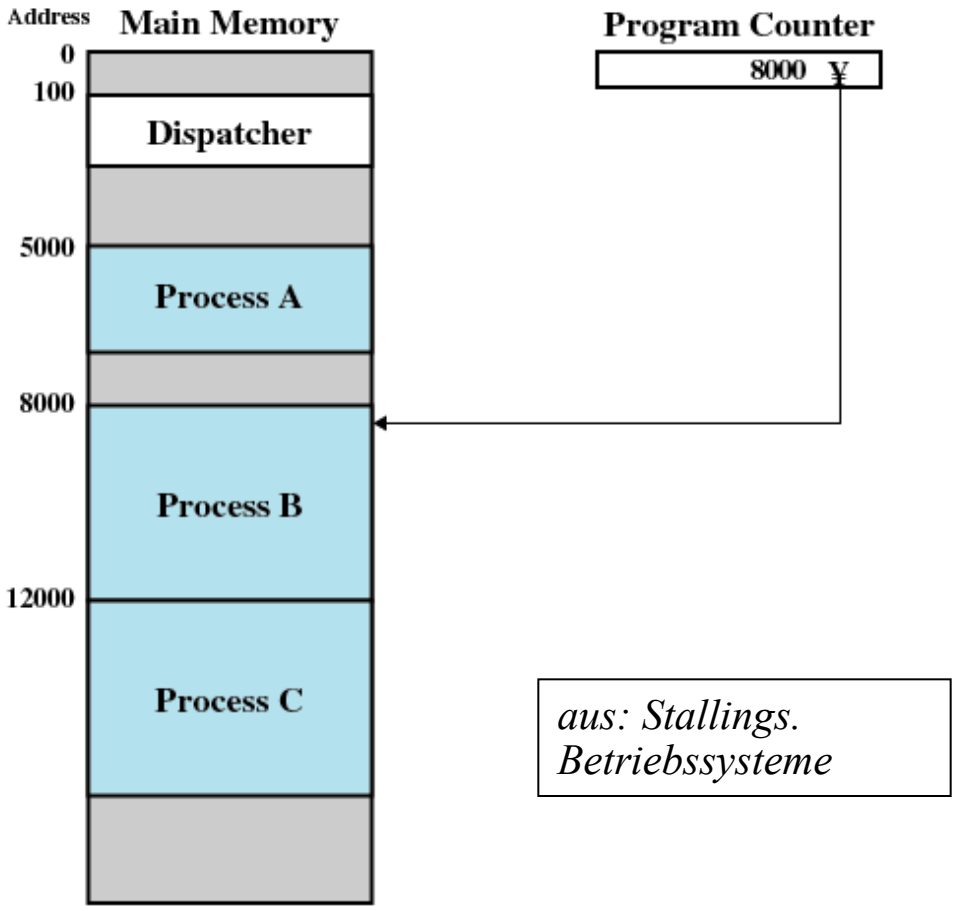


Figure 3.2 Snapshot of Example Execution (Figure 3.4) at Instruction Cycle 13

Ablauf von parallelen Prozessen

(a) Trace of Process A	(b) Trace of Process B	(c) Trace of Process C
5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004		12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011

5000 = Starting address of program of Process A
8000 = Starting address of program of Process B
12000 = Starting address of program of Process C

aus: Stallings. Betriebssysteme

Figure 3.3 Traces of Processes of Figure 3.2

Ablauf von parallelen Prozessen

1	5000
2	5001
3	5002
4	5003
5	5004
6	5005
----------	Time out
7	100
8	101
9	102
10	103
11	104
12	105
13	8000
14	8001
15	8002
16	8003
----------	I/O request
17	100
18	101
19	102
20	103
21	104
22	105
23	12000
24	12001
25	12002
26	12003

Instruktionszyklus (Prozessor)

27	12004
28	12005
----------	Time out
29	100
30	101
31	102
32	103
33	104
34	105
35	5006
36	5007
37	5008
38	5009
39	5010
40	5011
----------	Time out
41	100
42	101
43	102
44	103
45	104
46	105
47	12006
48	12007
49	12008
50	12009
51	12010
52	12011
----------	Time out

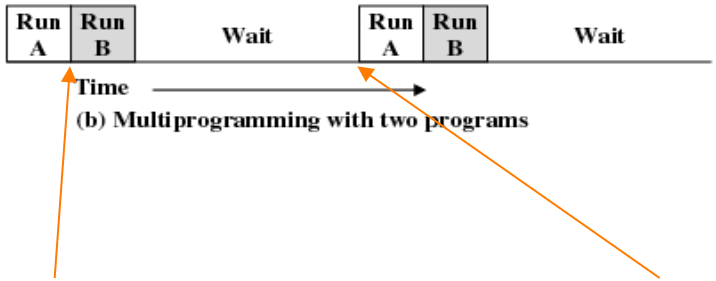
Adresse im Speicher

100 ist Startadresse vom Dispatcher

schattierter Bereich: Dispatcher

Der Prozesskontrollblock (PCB)

- Wh: Multiprogramming:
  - BS schaltet Prozesse um wenn notwendig
    - wegen E/A
    - oder Timesharing-System (Zeitscheibe zu Ende)



- Unterbrochener Proz. muß so weitermachen wie unterbrochen
- -> **Program counter** merken
- -> **Register** merken
- …

Der Prozesskontrollblock (PCB)

- Wozu PCB?
- Motivation:
  - Wählen Sie 2 Programme aus Programmieren aus
  1. Machen Sie für Programm1 einen Schreibtischtest
     - **im Kopf!**
       - keine Zeilennummern; keine Variableninhalte aufgeschrieben
  2. Unterbrechung nach dem n-ten (z.b. 5.) Befehl
  3. Programm2 Schreibtischtest (im Kopf)
  4. Unterbrechung nach dem n-ten (z.b. 5.) Befehl
  5. Zurück zum Programm1 .
  6. Werte der Variablen; Zeilennummer noch im Kopf?

  - so geht's auch dem Betriebssystem ;-)

  ➔ Zeilennummern und Variablen aufschreiben!

Der Prozesskontrollblock (PCB)

- Wozu PCB?
- Motivation:
  - Strukturiertes Programmieren:

```
integer a;
integer b;
...
```

  - Wie kann man die Inhalte von a und b vertauschen?

```
integer temp;
temp := a;
a:=b;
b:=temp;
```

  - Hilfsvariable **temp** entspricht PCB

Der Prozesskontrollblock (PCB)

- Wozu PCB?
  - Statt einer Hilfsvariable **temp** brauchen wir **pro Prozess** einen eigenen PCB
  - Grundproblem:
    - Es gibt nur einen Registersatz
      - Ein ProgramCounter PC
      - Ein Statusregister
      - ...

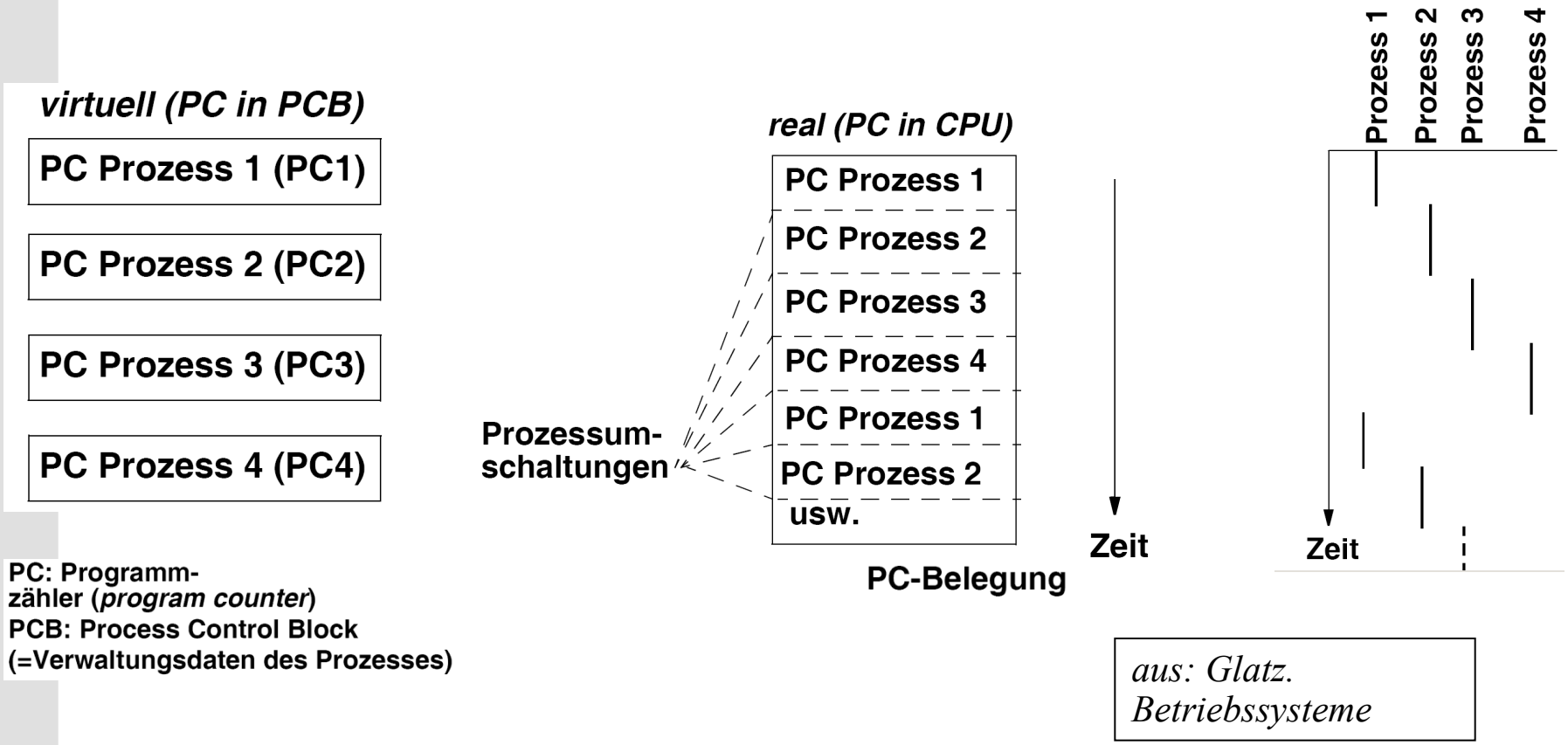
Der Prozesskontrollblock (PCB)

Der Prozesskontrollblock enthält die folgenden Informationen zum jeweiligen Prozess:

- **Prozessidentifikation (PID): Prozessnummer**
- **Prozessorzustand (PC, Registerinhalte, Interrupt-Masken)**
- **aktueller Prozesszustand**
- **Priorität des Prozesses für das Scheduling**
- **Informationen zum Speichermanagement**
  - **Wo ist der Prozess (Process Image = Code + Daten…) gespeichert?**
  - Seitentabellen, Segmenttabellen…
- E/A-Status: eingesetzte Ressourcen (Datenzeiger, offene Dateien)
- Referenz auf Vaterprozess und ggf. Kindprozesse
- Abrechnungsdaten: Rechenzeit, Ressourcenverbrauch
- Informationen bzgl. Privilegien des Prozesses, Eigentümerverhältnissen von Ressourcen.

Der Prozesskontrollblock (PCB)

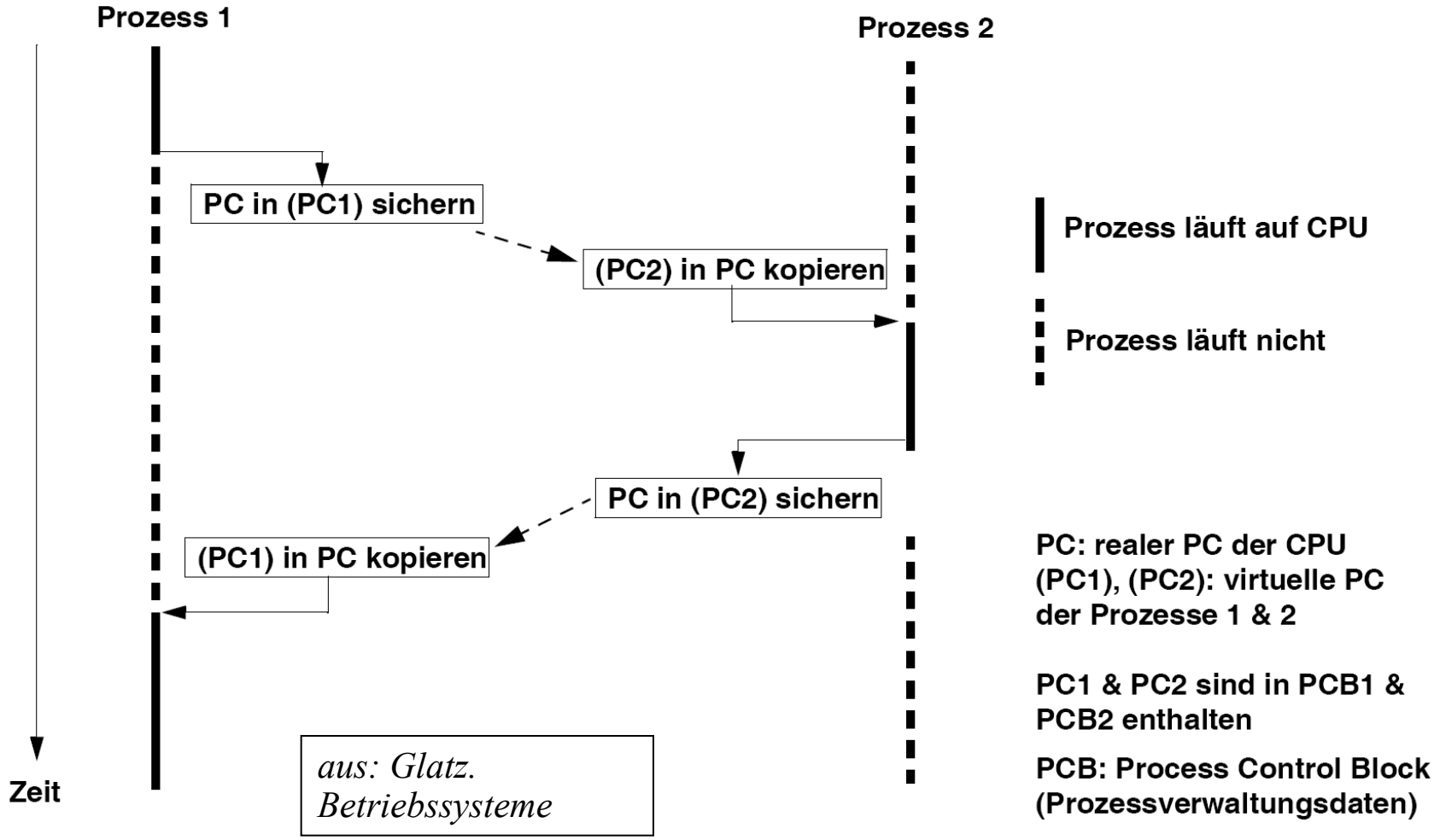
- Durch PCB hat jeder Prozess scheinbar (virtuell) seinen eigenen Program counter



aus: Glatz. Betriebssysteme

Der Prozesskontrollblock (PCB)

Prozessumschaltungen und PCB



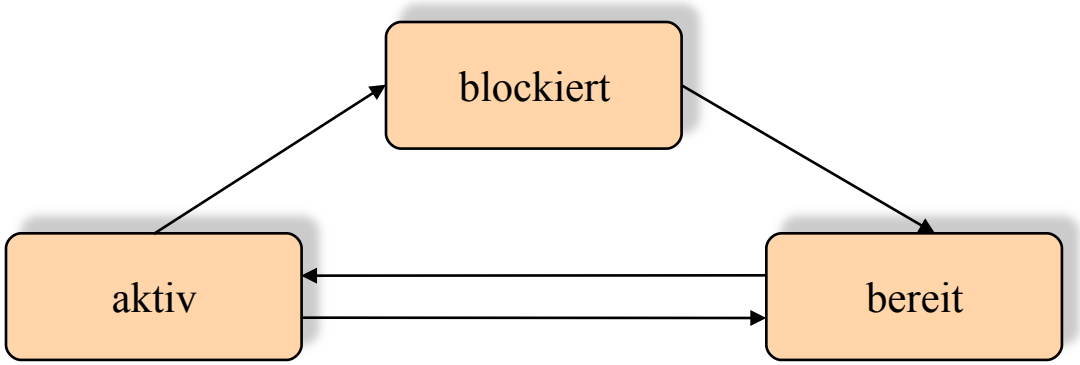
Ablaufplanung/-steuerung (scheduling):

- **Ziel**: möglichst effiziente und scheinbar parallele Bearbeitung mehrerer Aufgaben.
- **Voraussetzung:** effiziente Auftragsverwaltung - Verwaltung der Zustände und Übergänge von Prozessen (aufgrund äußerer oder innerer Ereignisse).
- Dazu muss das Betriebssystem Informationen über den aktuellen Status jedes Prozesses haben.
- Info ist abgelegt im Process Control Block (PCB):
  - Der PCB beschreibt den Status aller Betriebsmittel, die vom Prozess verwendet werden.
  - Die Gesamtheit aller PCBs definiert für das Betriebssystem alle in Arbeit befindlichen Aktivitäten.

Zustände eines Prozesses
einfaches Modell 1(3 states)

Ein Prozess kann im Laufe seiner Bearbeitung im wesentlichen folgende unterscheidbaren Zustände annehmen:

- **aktiv** (*running)*: Prozess belegt aktuell die CPU; bei Einprozessormaschinen kann sich zu jedem Zeitpunkt nur ein Prozess in diesem Zustand befinden.
- **bereit** (*ready*): Prozess ist zur Ausführung bereit, aber der Prozessor ist momentan belegt.
- **blockiert** (*blocked*): Prozess wartet auf ein Ereignis (I/O, Signal,…) und kann erst dann zur Ausführung kommen.



Zustände eines Prozesses

P1

läuft auf

CPU

P3

P5

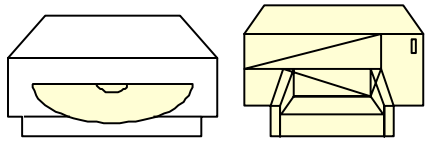
P6

könnten abgearbeitet werden aber CPU ist durch P1 belegt!

P2

P4

warten auf



Geräte

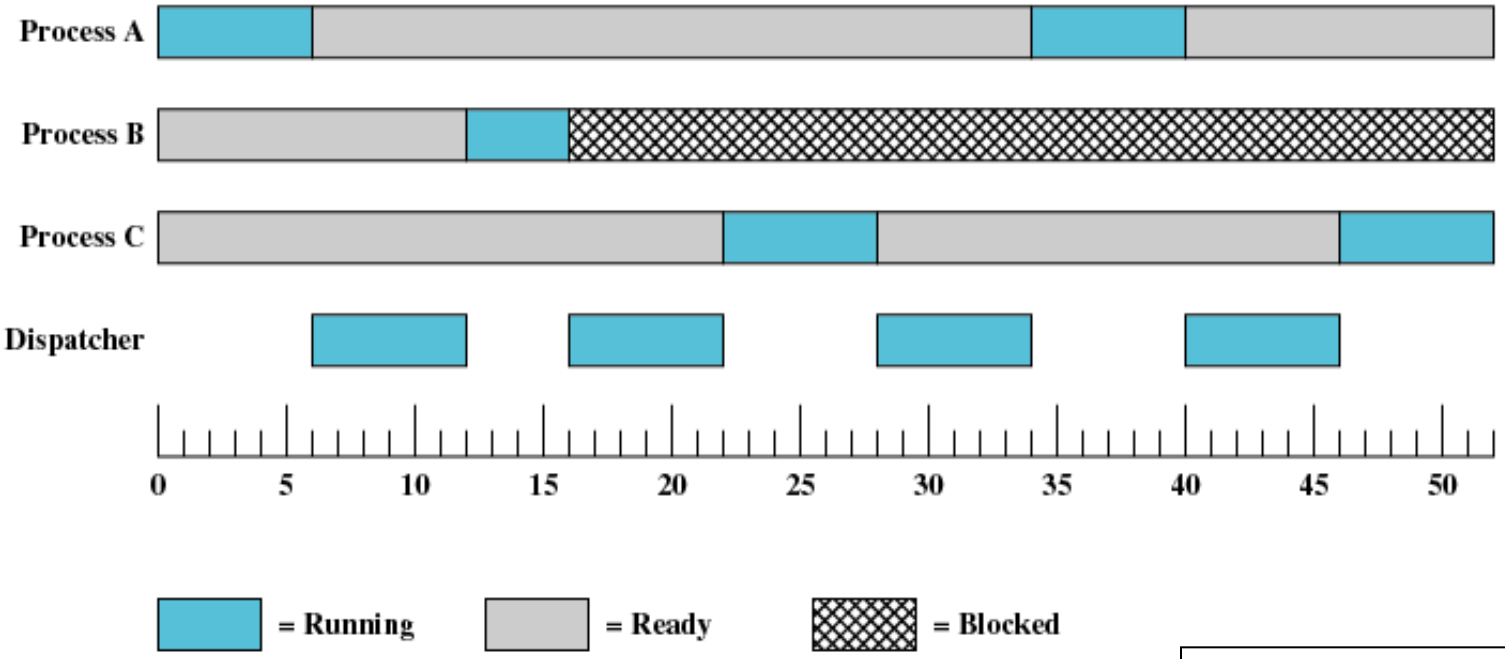
Zustände eines Prozesses
Zustandsübergänge

- Aktiv ➔ Bereit: Zeitscheibe zu Ende. Dispatcher (=Scheduler) entzieht dem Prozess die CPU
- Bereit ➔ Aktiv : Anderer Prozess (oder dessen Zeitscheibe) ist zu Ende. Prozess bekommt CPU
- Aktiv ➔ Blockiert: Prozess ruft (langandauernde) E/A-Operation auf, gibt „freiwillig“ CPU ab.
- Blockiert ➔ Bereit : Interrupt von E/A-Gerät erhalten: E/A-Operation für diesen Prozess ist beendet.

- Entscheidung **wann welcher** Prozess aktiv wird:
  - Wechsel Bereit ➔➔ Aktiv
  - Als **short-term** scheduling (oder CPU scheduling) bezeichnet

Zustände für Trace-Beispiel

- Trace siehe BS7_prozesse_verwaltung.pdf Seite 14-16
  - Kopie: Folgeslide



aus: Stallings. Betriebssysteme

Figure 3.7 Process States for Trace of Figure 3.4

Zustände für Trace-Beispiel

1	5000
2	5001
3	5002
4	5003
5	5004
6	5005
----------	Time out
7	100
8	101
9	102
10	103
11	104
12	105
13	8000
14	8001
15	8002
16	8003
----------	I/O request
17	100
18	101
19	102
20	103
21	104
22	105
23	12000
24	12001
25	12002
26	12003
27	12004
28	12005
----------	Time out
29	100
30	101
31	102
32	103
33	104
34	105
35	5006
36	5007
37	5008
38	5009
39	5010
40	5011
----------	Time out
41	100
42	101
43	102
44	103
45	104
46	105
47	12006
48	12007
49	12008
50	12009
51	12010
52	12011
----------	Time out