Fachhochschule Nordwestschweiz
Hochschule für Technik

Modul Betriebssysteme (bsys-iC)

Feedback aus der Hausaufgabe

- Was ist Ihnen aufgefallen?
- Gab es grundlegende neue Erkenntnisse?
- Was hat gefehlt?
- Wieviel Zeit haben Sie aufgewendet?

Lektion 8: Speicher-Management und Ressourcenverwaltung/-zuteilung

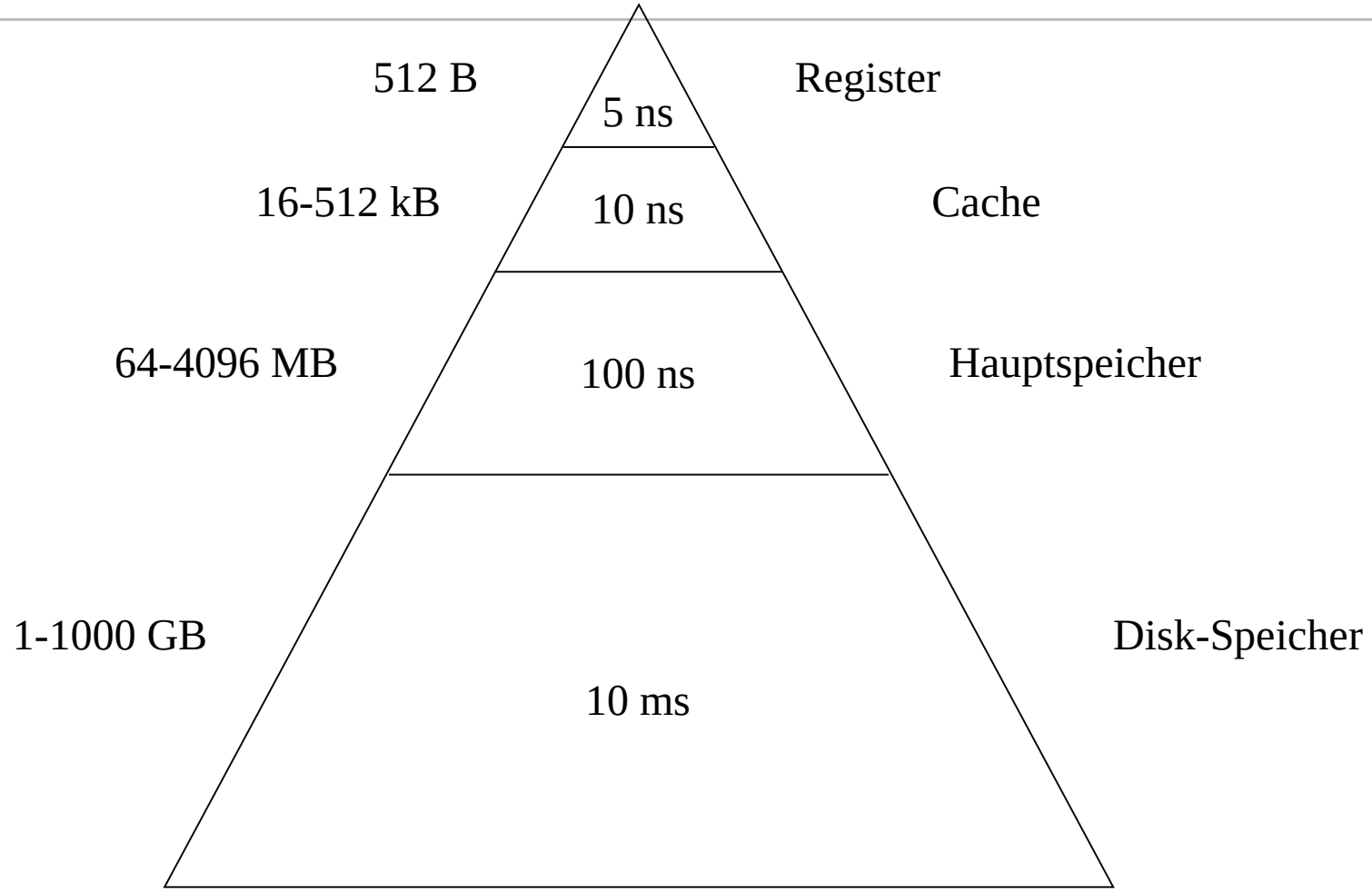
Inhalt

- Hauptspeicherverwaltung und -zuteilung (Paging, Swapping)
- Sekundärspeicherverwaltung und -zuteilung (Quota-System am Beispiel UFS)
- Scheduling-Strategien - Funktionsweise, Vor- und Nachteile / Einsatzgebiete

Motivation

**"I always give 110% to my job.
40% on Monday, 30% on Tuesday, 20% on
Wednesday, 15% on Thursday, and 5% on Friday."**

Die “Speicher-Pyramide”

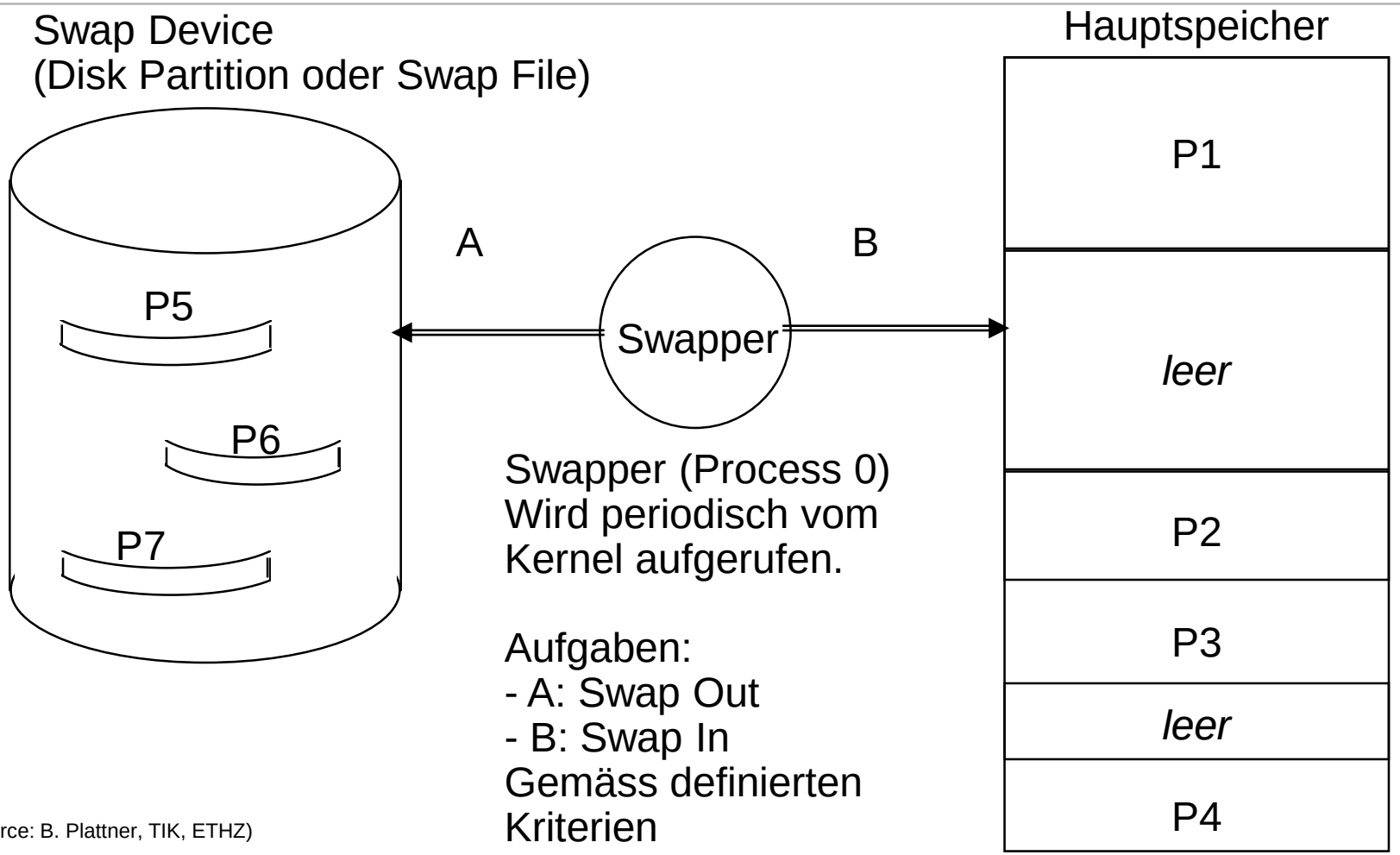


Nach: J. O’Gorman, Operating Systems with Linux, Figure 8.2

Virtual Memory

- Erweiterung des Hauptspeichers pro System (mehr Prozesse im System als Speicher verfügbar), oder pro Prozess (einzelner Prozess grösser als verfügbarer Hauptspeicher).
- Systematische Abstraktion für system-spezifische Overlay-Techniken.
- Organisation des Hauptspeichers in gleich grosse, einheitlich adressierbare Einheiten (Seiten, pages).
- Benötigt hardware-unterstütze Abbildung zwischen physischen und virtuellen Adressen:
  - Statische Tabelle
  - Assoziationstabelle / Cache
  - Dynamische Auflösung wenn eine Adresse referenziert wird (Basis Register)
  - …

Swapping



Swapper (Process 0)
Wird periodisch vom Kernel aufgerufen.

Aufgaben:
- A: Swap Out
- B: Swap In

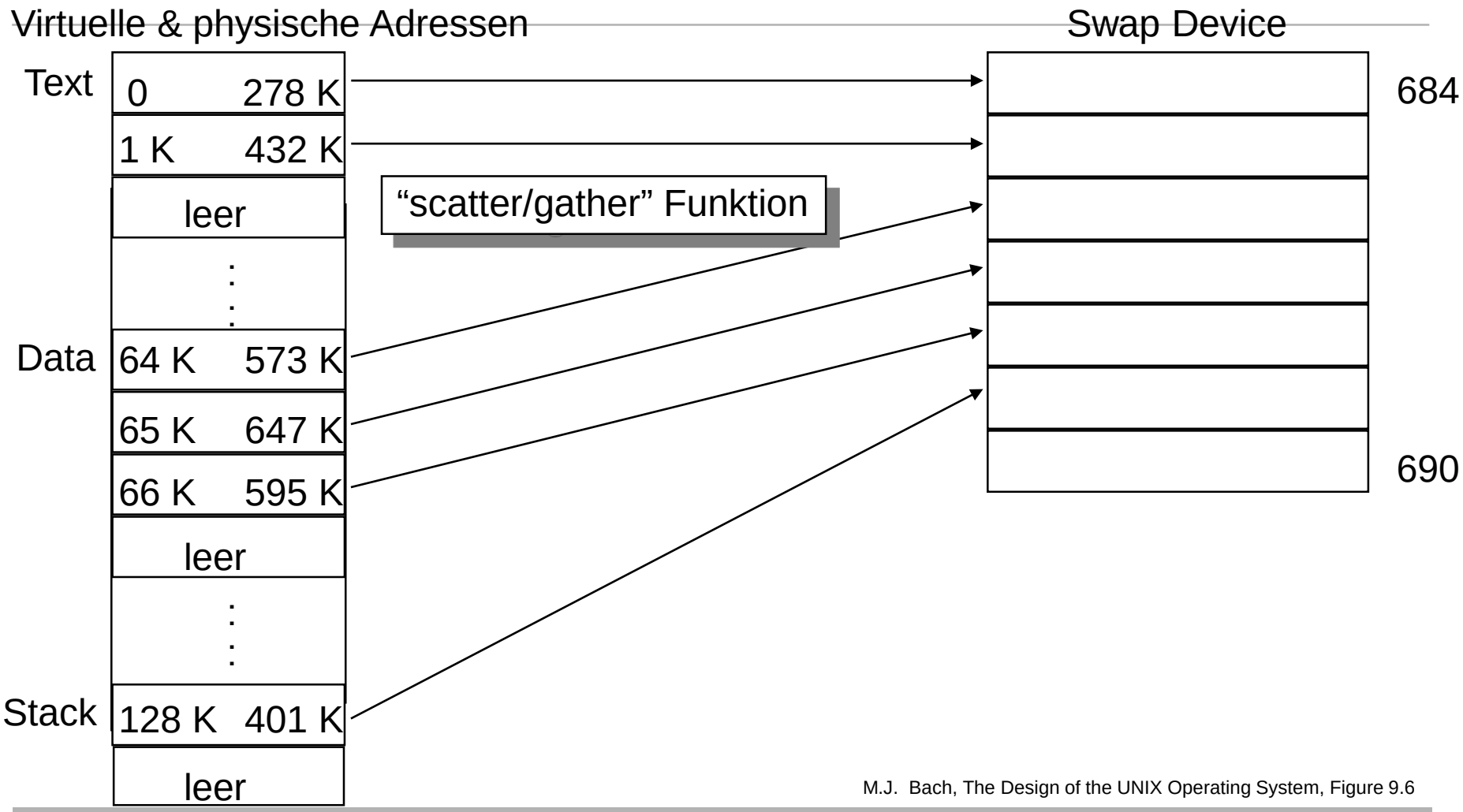
Gemäss definierten Kriterien

(Source: B. Plattner, TIK, ETHZ)

Kriterien für das Swapping

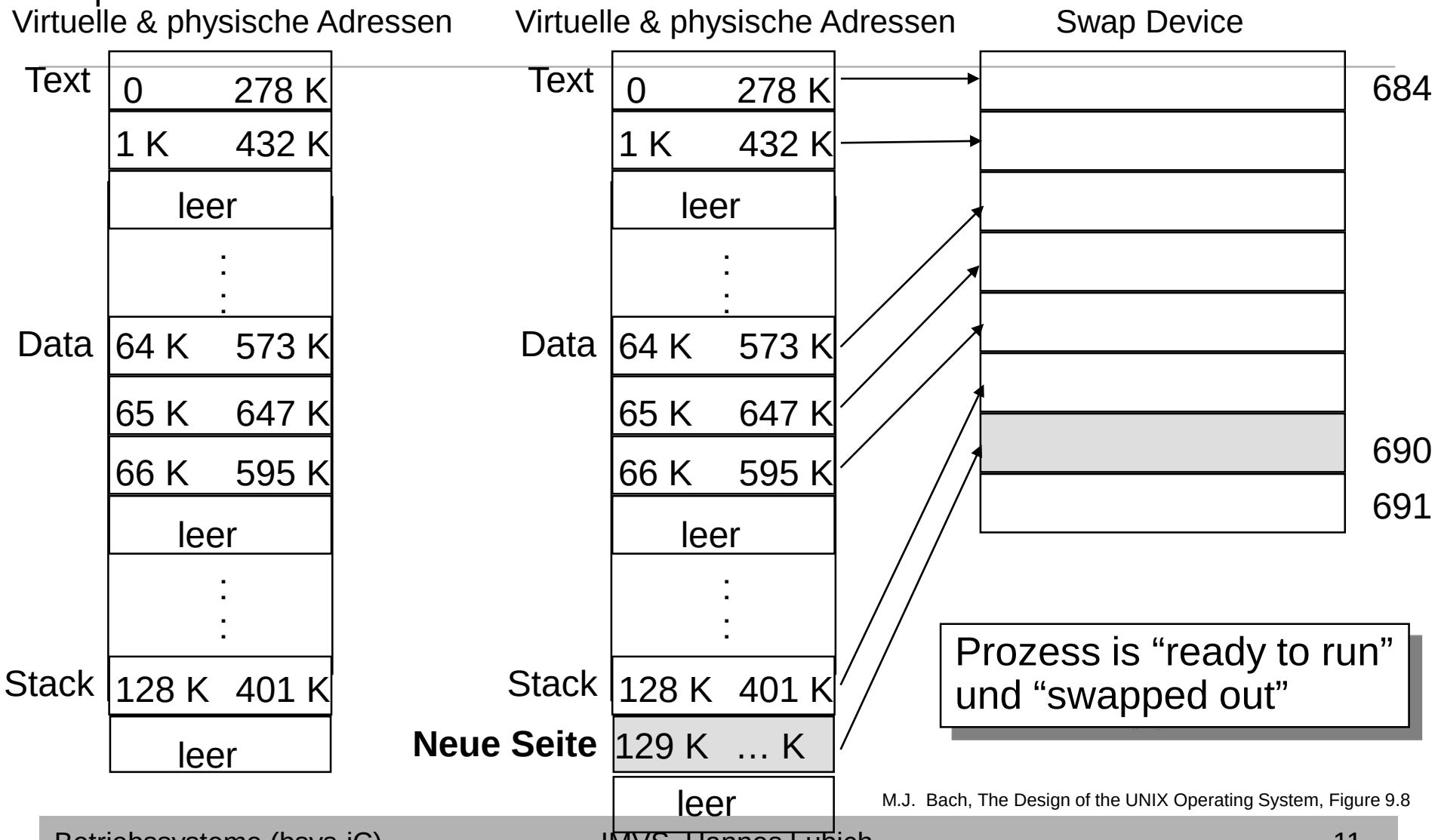
- A: Verlagere Prozesse vom Hauptspeicher auf das Swap Device: “swap out”
  1. Kein Platz im HS für weitere Prozesse, aber Prozess ruft “fork” auf → fork swap
  2. Kein Platz im HS, aber Prozess wächst, z.B. weil der Stack wächst → expansion swap
  3. Auf Swap Device ausgelagerter wartender Prozess wird “ready to run” und wird vom scheduler ausgewählt → exchange swap
     - Zombie Prozesse und “locked in memory” Prozesse sind nicht wählbar
     - Schlafende Prozesse werden “ready to run” prozessen vorgezogen
     - Präferenz auf Prozessen mit niedriger Priorität
     - Präferenz auf Prozessen die > 2 sec. im HS waren
- B: Verlagere Prozesse vom Swap Device in den Hauptspeicher: “swap in”
  - Nur “ready to run” Prozesse sind wählbar
  - Präferenz auf Prozessen die > 2 sec. ausgelagert waren

Adressabbildung während des Swaps



M.J. Bach, The Design of the UNIX Operating System, Figure 9.6

Expansion Swap Beispiel



M.J. Bach, The Design of the UNIX Operating System, Figure 9.8

Swap Device Verwaltung

Adresse Einheit

1	10000

(a)

Adresse Einheit

101	9900

(b)

151	9850

(c)

251	9750

(d)

251	9750

(a)

Adresse Einheit

1	150
251	9750

(c)

101	50
251	9750

(b)

1	150
251	9750

(a)

1	150
451	9550

(b)

M.J. Bach, The Design of the UNIX Operating System, Figures 9.3, 9.4, 9.5

Sequenz der Swap Operationen

Time	Proc A	B	C	D	E
			swap out	swap out	swap out
0	0	0	0	0	0
	runs				
1	1	1	1	1	1
		runs			
2	2	2	2	2	2
	swap out	swap out	swap in	swap in	
	0	0	0	0	
			runs		
3	1	1	1	1	3
				runs	
4	2	2	2	2	4
	swap in		swap out	swap out	swap in
	0		0	0	0
					runs
5	1	3	1	1	1
	runs				
6	2	4	2	2	2
	swap out	swap in	swap in		swap out
	0	0	0	3	0
		runs			

M.J. Bach, The Design of the UNIX Operating System, Figure 9.10

“Thrashing” beim Swapping

Fachhochschule Nordwestschweiz
Hochschule für Technik

Time	Proc A	B	C	D	E
				nice 25	
			swap out	swap out	swap out
0	0 runs	0	0	0	0
1	1	1 runs	1	1	1
2	2 swap out 0	2 swap out 0	2 swap in 0 runs	2 swap in 0	2
3	1	1	1	1 swap out 0	3 swap in 0 runs
4	2 swap in 0 runs	2	2 swap out 0	1	1
5	1	3 swap in 0 runs	1	2	2 swap out 0
6	2 swap out 0	1	2	3 swap in 0 runs	1

M.J. Bach, The Design of the UNIX Operating System, Figure 9.11

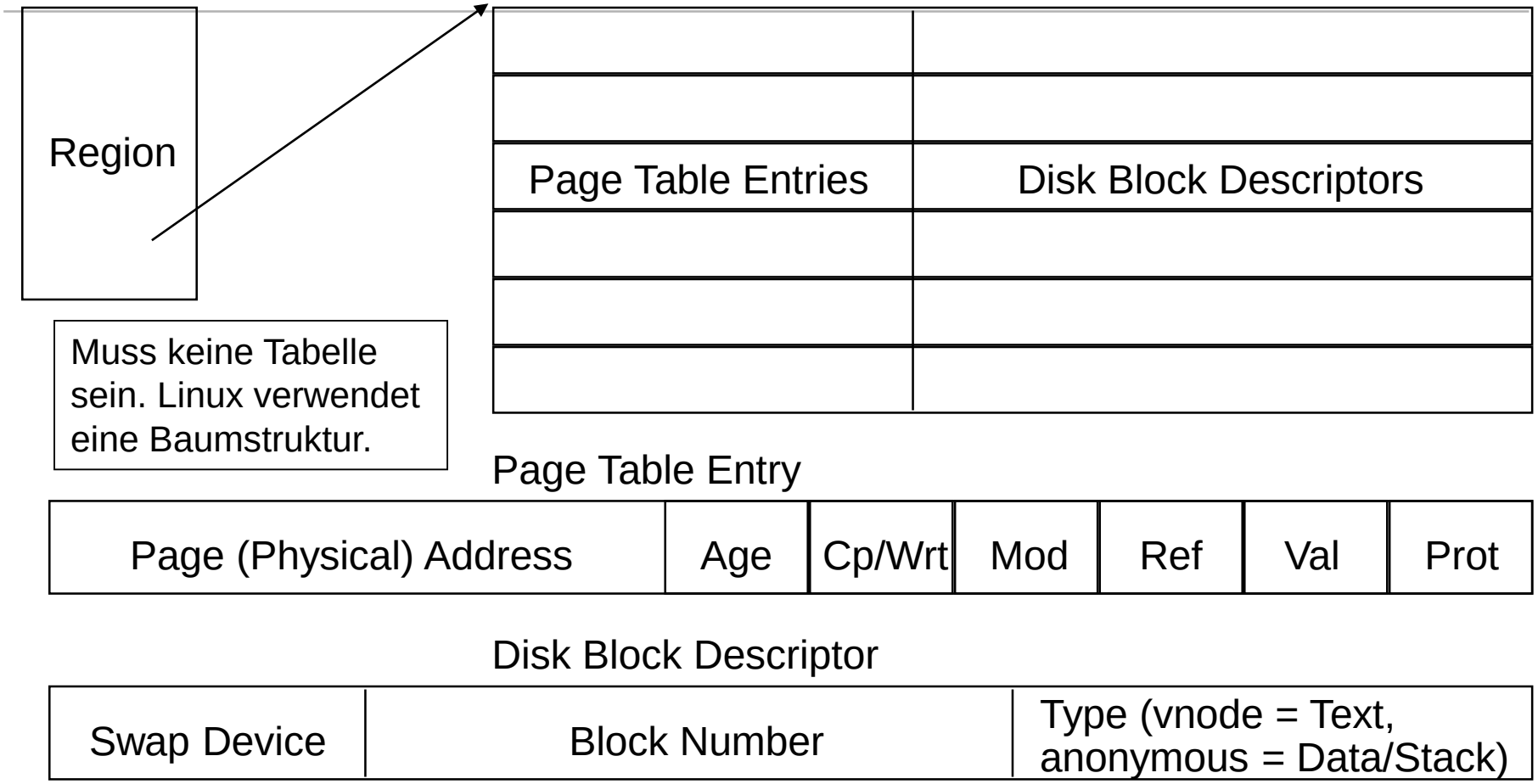
Demand Paging

- Anforderung: ein einzelner Prozess soll grösser sein dürfen, als der verfügbare Hauptspeicher
- Voraussetzungen für “demand paging”:
  - Hardware unterstützt seitenorientieres Speichermanagement (½ – 4 kB / Seite)
  - Wiederaufsetzbare CPU-Instruktionen (wenn Instruktionen über eine Seiten-grenze verlaufen)
- Beobachtung: Lokalität des Codes, nur etwa 10-15% des Programmcodes müssen zu jeder beliebigen Zeit im HS liegen (“working set”).
- Zugriff auf eine Seite ausserhalb des “working set” führt zu einem “page fault” (Seitenladefehler), dann wird die Seite nachgeladen → balance zwischen der Grösse des “working set” und der Anzahl “page faults”
- Optimierung des “demand paging” durch
  - “Reference bit” → ermöglicht nicht-lineares “working set”
  - “Age bits” → verbleibende Zet einer Seite im “working set”

→ Zwei Aufgaben für das Paging-Subsystem:

- Seitenalterung und Auslagerung/Löschung genügend alter Seiten
- Bearbeitung von “page faults”

Paged Memory I



M.J. Bach, The Design of the UNIX Operating System, Figure 9.13

Pause

Übung
(ca. 30 min.)

- Aufgabe(n) gemäss separatem Aufgabenblatt
- Lösungsansatz: Einzelarbeit oder Gruppen von max. 3 Personen
- Hilfsmittel: beliebig
- Besprechung möglicher Lösungen in der Klasse (es gibt meist nicht die eine «Musterlösung»)

Übungsbesprechung (ca. 15 min.)

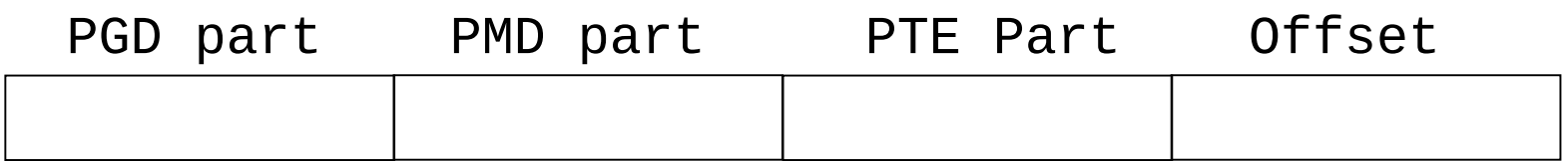
- Stellen Sie Ihre jeweilige Lösung der Klasse vor.
- Zeigen Sie auf, warum ihre Lösung korrekt, vollständig und effizient ist.
- Diskutieren Sie ggf. Design-Entscheide, Alternativen oder abweichende Lösungsansätze.
- Gibt es Unklarheiten? Stellen Sie Fragen.

Pause

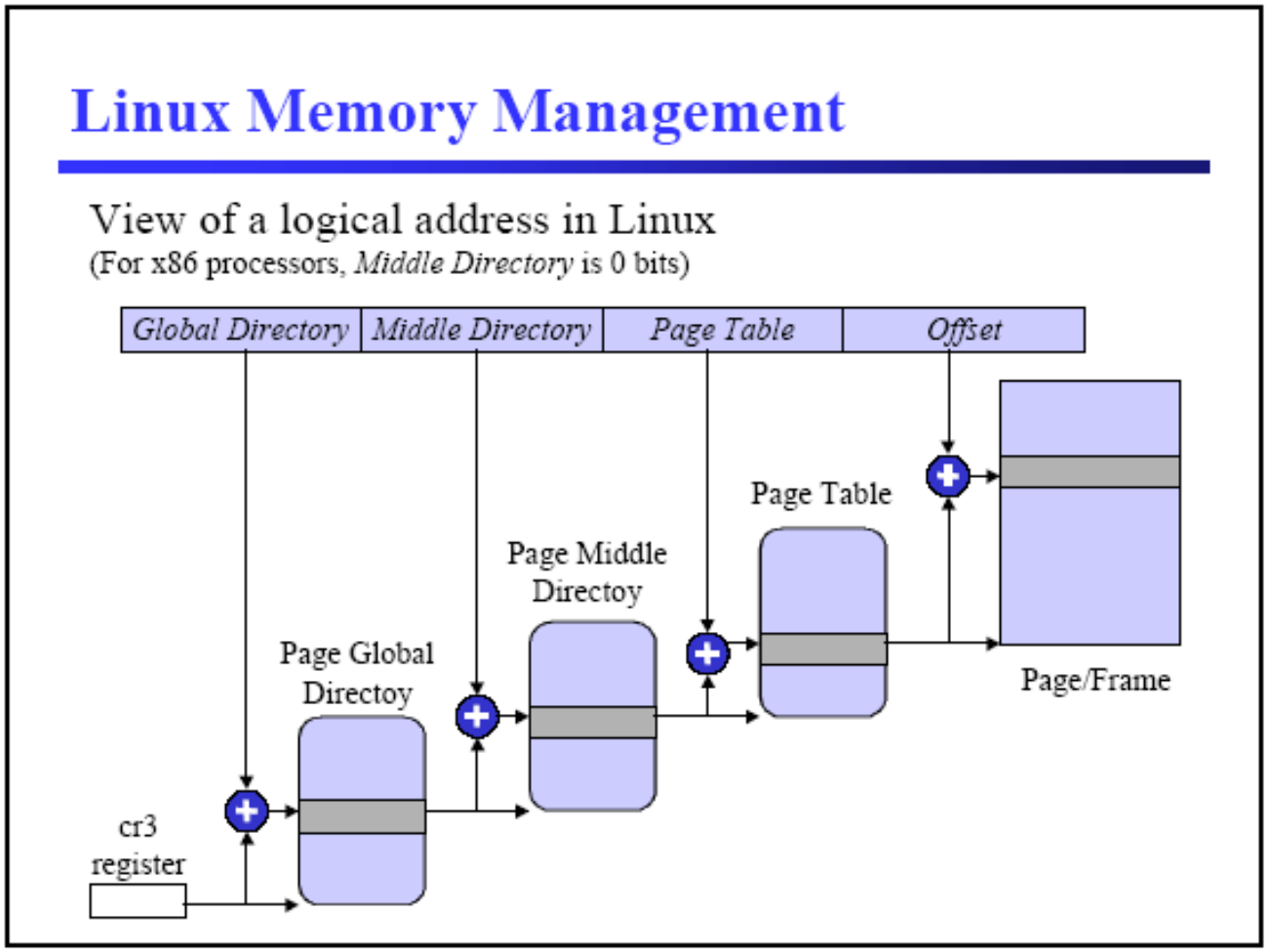
Paged Memory II

- Design und Layout der Datenstrukturen für die seiten-orientierte Speicherverwaltung variieren zwischen Unix- und Linux-Varianten und sind zudem von den Hardware-Eigenschaften anhängig.
- Beispiel: Linux verwendet eine 3-stufige Seitentabelle:
  - Page Directory pro Prozess und für den BS-Kern
  - Page Mid-level Directory (für 64 bit CPU-Architekturen)
  - Page Table, enthält Seitenbeschreibungen und Verweise auf den physische Speicherort

  und eine virtuelle Linux-Adresse enthält diese Elemente:



Paged Memory III

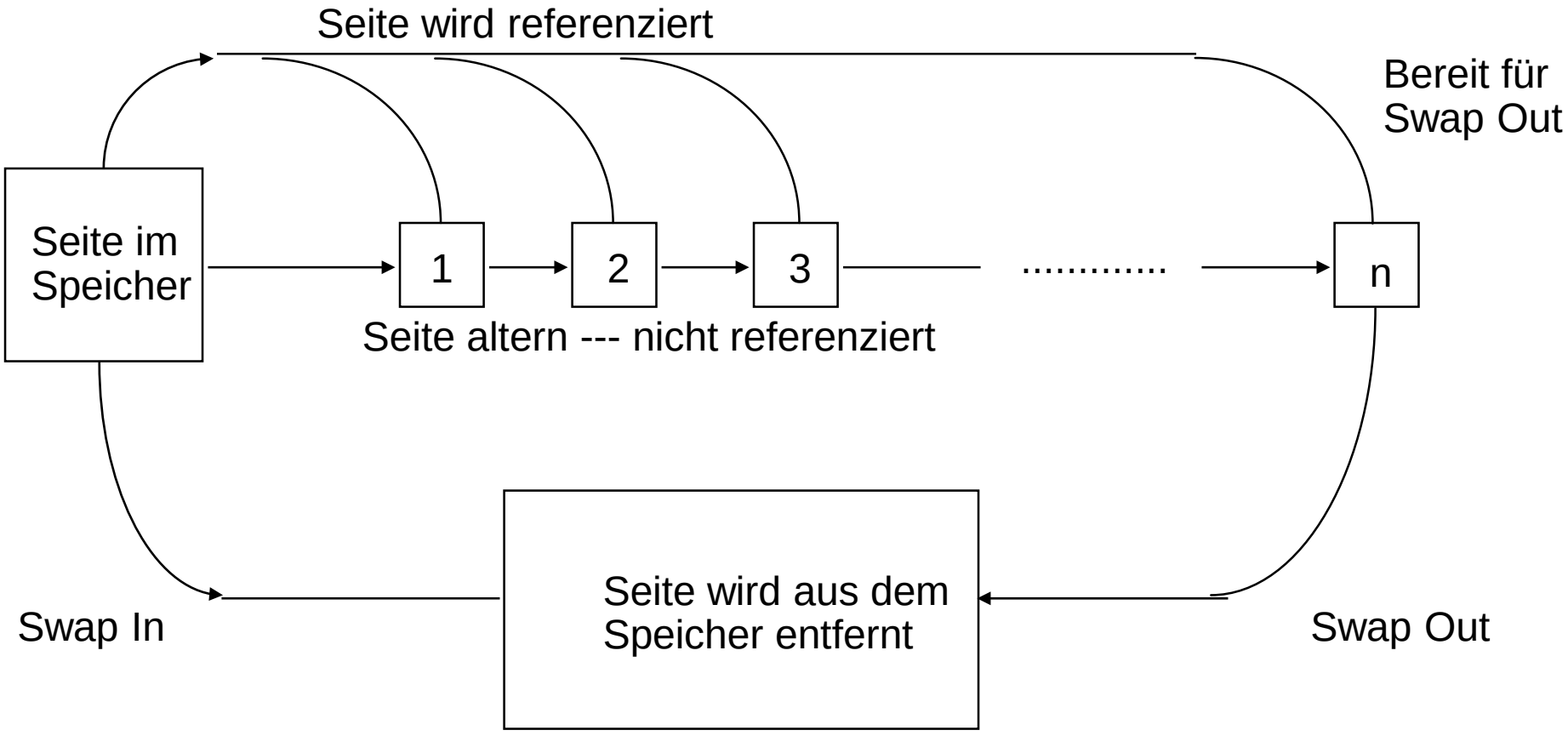


http://www.inf.fu-berlin.de/lehre/SS01/OS/Lectures/Lecture14.pdf

Paged Memory IV

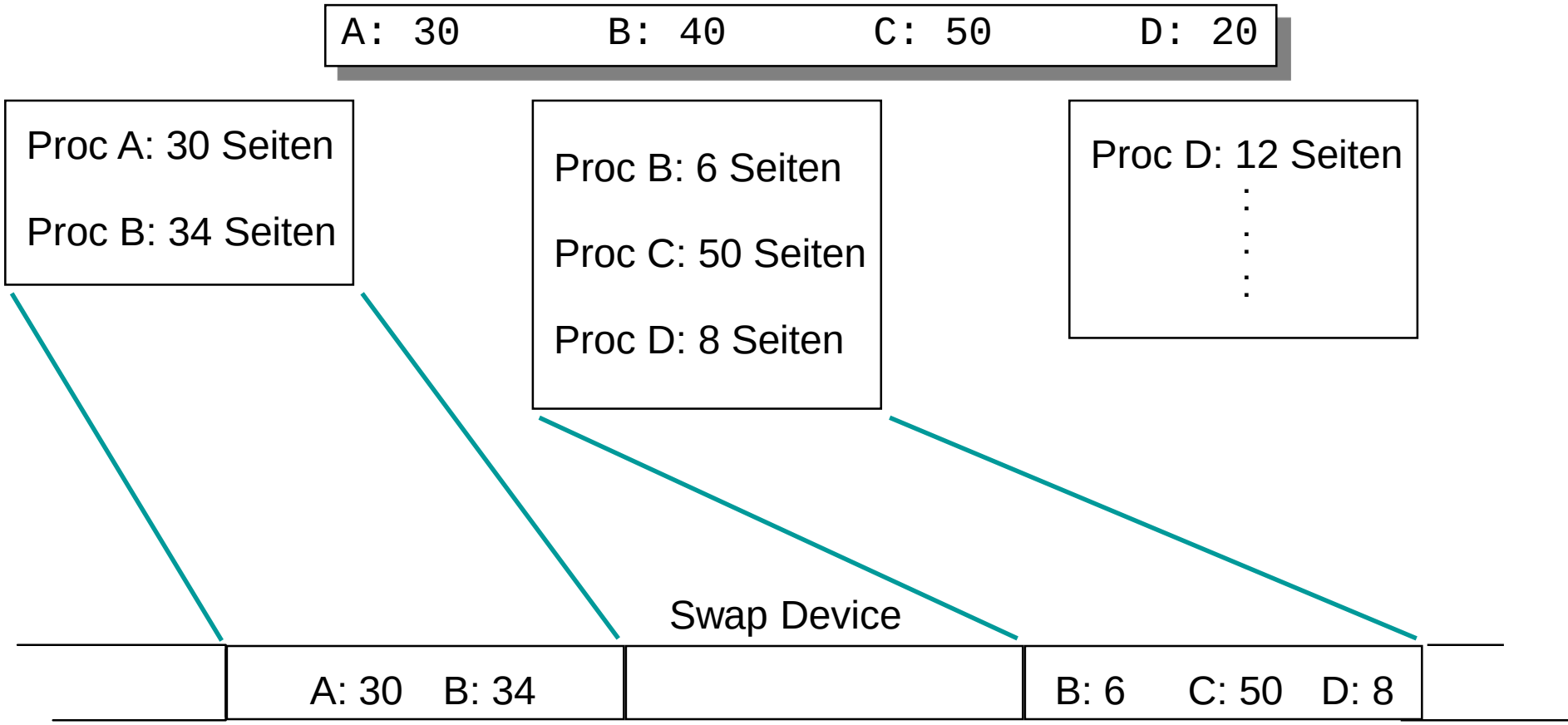
- Linux Kennzahlen: Page-Grösse 4 kByte
- Die ersten beiden MB des physischen Speichers sind reserviert für PC-interne Zwecke sowie Text und Daten des Betriebssystem-Kerns
- Der logische Prozess-Adressraum ist zweigeteilt:
  - 0x00000000 bis PAGE_OFFSET – 1 sind sowohl im User Mode als auch im Kernel Mode adressierbar
  - PAGE_OFFSET bis 0xffffffff ist nur im Kernel Mode adressierbar
  - PAGE_OFFSET ist meist 0xc0000000
- Kontinuierlich aufeinander folgende Pages (z.B. für Geräte mit fixen DMA) durch das „buddy system“.

Altern von Seiten: Der “Page Stealer”



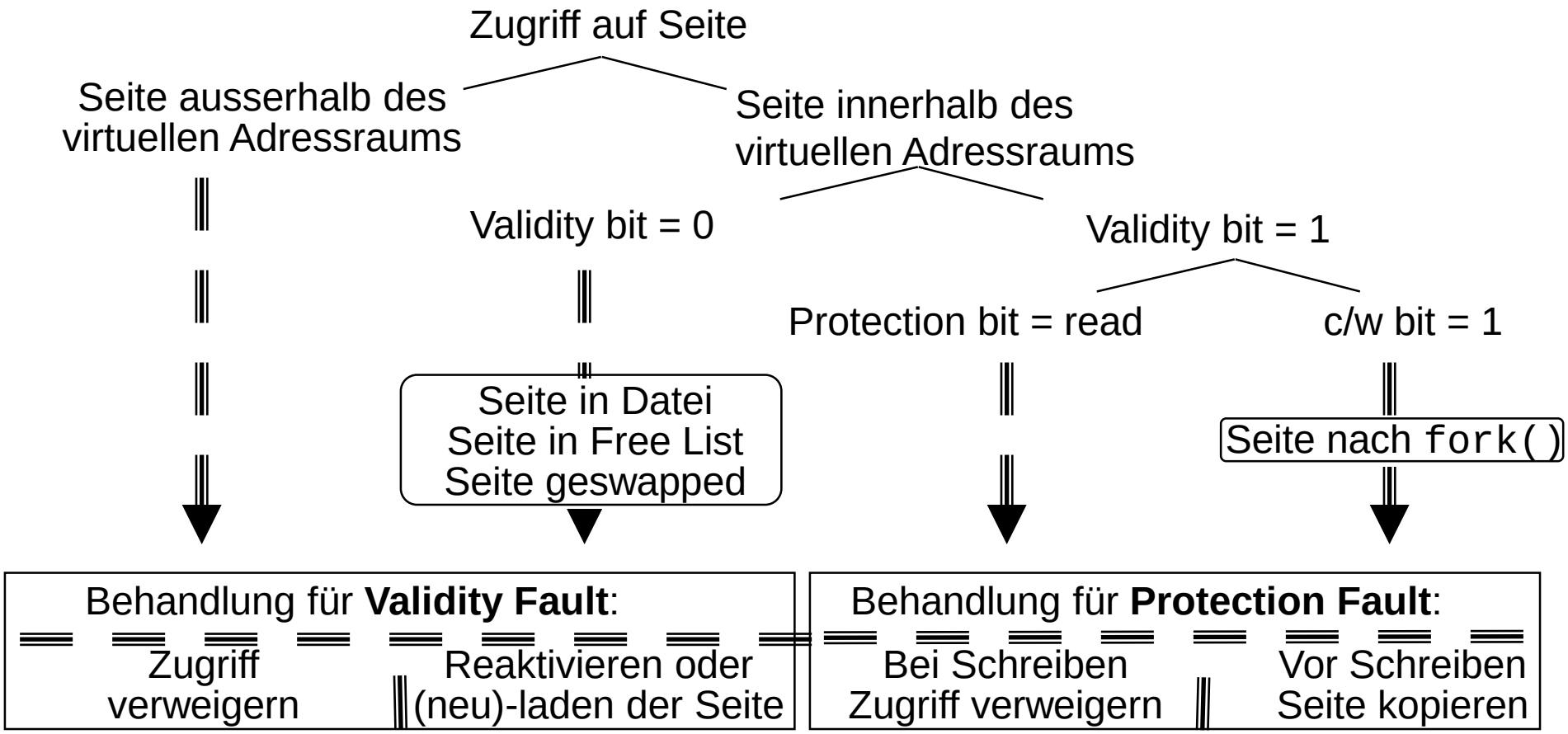
M.J. Bach, The Design of the UNIX Operating System, Figure 9.18

Auslagern von Seiten



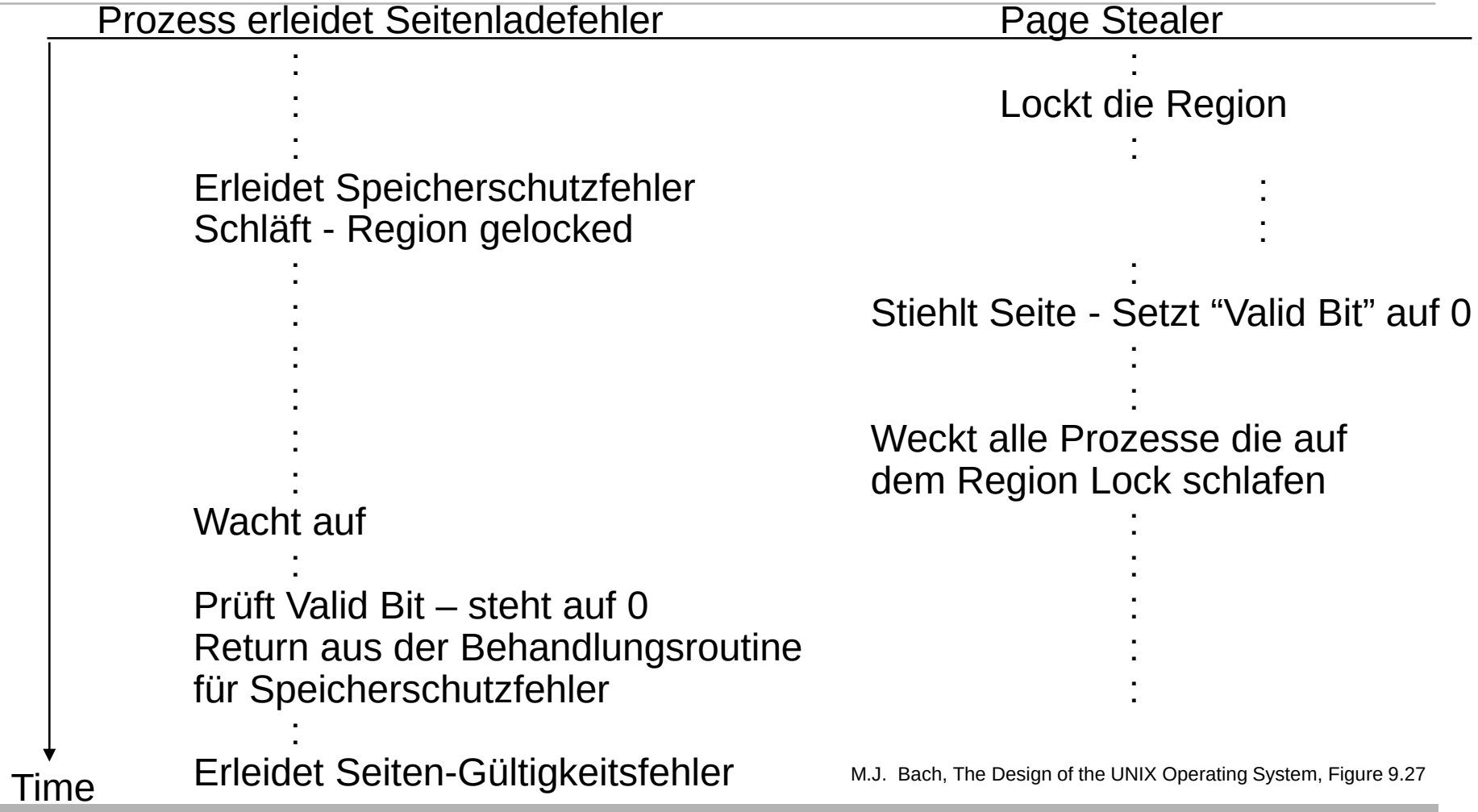
M.J. Bach, The Design of the UNIX Operating System, Figure 9.20

Behandlung von Fehlerzuständen



(Quelle: C. Lanz, TIK, ETHZ)

Speicherfehler Abhängigkeiten



M.J. Bach, The Design of the UNIX Operating System, Figure 9.27

Sekundärspeicher-verwaltung und -zuteilung

- Quota-System: Es gibt für Systemadministratoren mehrere Möglichkeiten, Limits für den Plattenplatz, den ein Benutzer oder eine Gruppe verbrauchen kann, oder die Anzahl der Dateien, die angelegt werden dürfen, festzulegen. Die Limits können auf dem Plattenplatz (Block-Quotas) oder der Anzahl der Dateien (Inode-Quotas) oder einer Kombination basieren. Jedes dieser Limits wird weiterhin in zwei Kategorien geteilt: Hardlimits und Softlimits.
  - Ein Hardlimit kann nicht überschritten werden. Hat der Benutzer das Hardlimit erreicht, so kann er auf dem betreffenden Dateisystem keinen weiteren Platz mehr beanspruchen und alle Schreib-/Kopierversuche schlagen fehl.
  - Im Gegensatz dazu können Softlimits für eine befristete Zeit überschritten werden. Hat der Benutzer das Softlimit über die Frist hinaus überschritten, so wird das Softlimit in ein Hardlimit umgewandelt und der Benutzer kann keinen weiteren Platz mehr beanspruchen. Wenn er einmal das Softlimit unterschreitet, wird die Frist wieder zurückgesetzt.

Das folgende Beispiel zeigt die Ausgabe von `quota -v` für einen Benutzer, der Quota-Limits auf zwei Dateisystemen besitzt:

```
Disk quotas for user test (uid 1002):
     Filesystem  usage   quota   limit   grace   files   quota   limit   grace
           /usr      65*      50      75   5days       7      50      60
       /usr/var       0      50      75               0      50      60
```

Im Dateisystem `/usr` liegt der Benutzer momentan 15 Kilobytes über dem Softlimit von 50 Kilobytes und hat noch 5 Tage seiner Frist übrig. Der Stern `*` zeigt an, dass der Benutzer sein Limit überschritten hat.

Scheduling-Strategien

- Funktionsweise
  - Fairness / Regeleinhaltung bezüglich der Zuteilung von Betriebsmitteln an Prozesse gemäss definierter Kriterien
  - Vermeidung von „Starvation“ und „Deadlocks“
- Vor- und Nachteile
  - Durchschnittliche Qualität für alle ist ggf. suboptimal
  - Komplexität / Overhead des Scheduling
- Einsatzgebiete
  - Echtzeitsysteme mit „harten“ Garantien
  - Möglichst unterbruchsfreie Batchverarbeitung
  - Interaktives Mehrbenutzersystem

Scheduling in Unix

- Unix verwendet typischerweise 3 Prioritätsklassen (virtual sched.):
  - Realtime: Scheduling (oft mit fixen Prioritäten), siehe “5-RealTimeOS.pdf ”
  - System: Geschlossene Scheduling-Klasse für Systemprozesse, inklusive dem Null Prozess (oder init Prozess) zum Konsumieren nicht verwendeter CPU-Zeit
  - Time-shared: Für alle Benutzerprozesse
- Prioritäts-basiertes Scheduling:
  - “Round robin” für alle Prozesse im Zustand “ready to run”
  - Initiale Priorität pro Klasse (siehe “nice”), Linux hat 140 Klassen pro CPU
  - Verminderte Prozesspriorität mit steigendem Ressourcenverbrauch
  - Verminderung der Priorität gemäss Laufzeit (via den “clock handler”), gemäss Formel `counter = priority + counter/2`
- Scheduling-Strategien:
  - Kleine, schnelle Prozesse präferieren (z.B. Shell) für interaktive Systeme
  - Ressourcen-intensiven Prozessen alle benötigen Ressourcen geben: (1) schnellere Beendigung & Freigabe und (2) Beedigung im Zeitlimit (Batch)

Konfiguration am Beispiel Linux

- Swapping beeinflussen: `sysctl -a | grep swappiness` (0 nur bei Memory-Überlauf, 100 aggressiv, 60 Default)
- Scheduling beeinflussen: `sysctl -a | grep kernel.sched`
- Page Faults anzeigen:
  - `ps –o min_flt -o maj_flt 1` (Major Fault = Disk Zugriff, Minor Fault = Seite neu erzeugen, für Prozess 1 (init))
  - `top`, dann „F“ für Auswahl Spalten, dann „u“ für Page Faults, dann Return = neue Spalte
  - Paket sysstat installieren, dann `sar –B 1 10` (Page Fault Statistiken 1 mal pro Sekunde, 10 Sekunden lang.

Zusammenfassung der Lektion 8 und Hausaufgabe

- Anforderungen an die Ressourcenverwaltung im Betriebssystem.
- Korrekte Identifikation und Priorisierung von Art und Umfang der verfügbaren Ressourcen.
- Grundlegende Scheduling-Strategien im Betriebssystem.
- Bewerten, vergleichen und fallweises Anwenden von Scheduling-Strategien.

- Hausaufgabe:
  - Repetieren Sie den Stoff dieser Lektion.
  - Studieren Sie Kapitel 8 und 9 des Dokuments „bsyI.pdf“.
  - Studieren Sie das Dokument „8-RealTimeOS.pdf“.