technische universität
dortmund

Betriebssysteme

Probeklausur

http://ess.cs.tu-dortmund.de/DE/Teaching/SS2016/BS/

Olaf Spinczyk und **Pascal Libuschewski**

olaf.spinczyk@tu-dortmund.de
https://ess.cs.tu-dortmund.de/~os

AG Eingebettete Systemsoftware
Informatik 12, TU Dortmund

Ablauf

- Probeklausur (45 Minuten)
- Besprechung der Aufgaben
- Auswertung
- Weitere Hinweise zur Vorbereitung

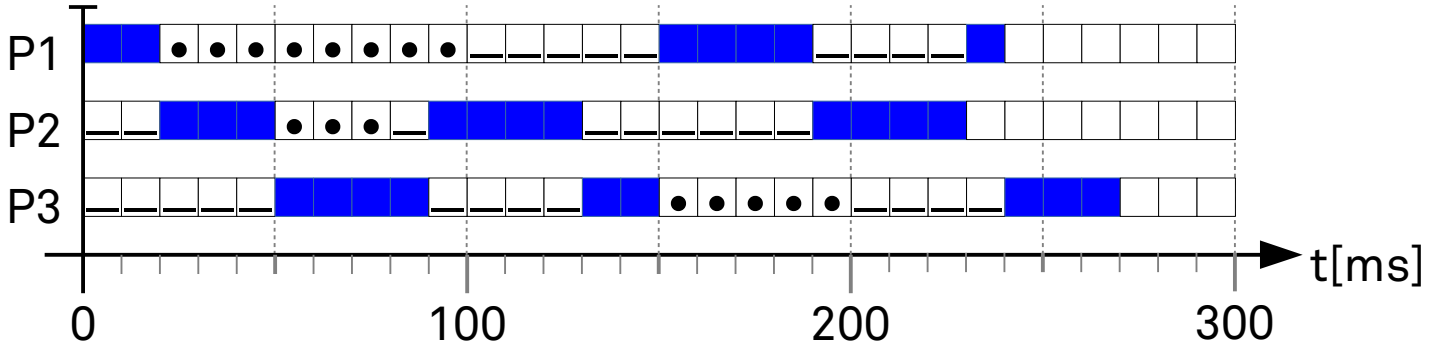
Probeklausur

... in (fast) allen Belangen realistisch:

- Art der Aufgaben
  - Auswahl aus dem **gesamten** Inhalt der Veranstaltung
    - Betriebssystemgrundlagen **und** UNIX-Systemprogrammierung in C
    - alle Vorlesungen und Übungen sind relevant
- Umfang
  - kürzer als das „Original“: 45 (statt 60) Minuten
- Durchführung
  - **keine Hilfsmittel** erlaubt (keine Spickzettel, Bücher, ...)
  - bitte **still arbeiten**
  - jeder für sich
- Die Klausur wird **nicht** eingesammelt.

1a) Round Robin (6 Punkte)

Prozess	P1	P2	P3
Bedienzeit	70 ms	110 ms	90 ms
E/A-Zeitpunkt	20 ms	30 ms	60 ms
E/A-Dauer	80 ms	30 ms	50 ms



Hinweise:

- Die Prozessorzeit wird in Zeitscheiben von 40 ms aufgeteilt
- Mit Ablauf der Zeitscheibe erfolgt ggfs. ein Prozesswechsel
- Unterbrochene Prozesse werden ans Ende der Bereitliste verdrängt
- Der nächste Prozess wird gemäß FCFS der Bereitliste entnommen

1b) Allgemeine Fragen

1. **“>” Shell-Operator (1.5 Punkte)** Angenommen der Befehl “echo Parameter” gibt in die Standardausgabe den Text aus dem Parameter:

   Was bewirkt der Befehl “echo “” > Datei”, falls Datei schon Inhalt enthält?

   – Der Inhalt der Datei wird gelöscht und die Datei enthält danach den String “”, also keine Zeichen.

1b) Allgemeine Fragen

2. **Prozesse vs. Funktionen (1.5 Punkte)** Erklären Sie kurz den Unterschied einer Prozesserzeugung und einem Funktionsaufruf.

   - Prozesserzeugungen erzeugen eine Kopie des bereits laufenden Prozesses. Nach der Erzeugung laufen zwei Prozesse parallel weiter. Es wird der Adressraum nicht geteilt.
   - Ein Funktionsaufruf springt innerhalb einer Prozessausführung an eine Funktion, führt diese aus und springt anschließend an die aufrufende Funktion zurück. Es findet keine parallele Ausführung statt.

1b) Allgemeine Fragen

3. **Zombie-Prozesse (1.5 Punkte)** Warum muss das Betriebssystem auf das Abfragen von Zombie-Prozessen warten anstatt diese direkt aus der Prozesstabelle zu entfernen?
   - Evtl. wird der Rückgabewert der Prozesse noch benötigt und abgefragt.

2a) Erzeuger/Verbraucher (7 Punkte)

```
Semaphore mutex = 1
Semaphore available = 0

producer () {
   while (1) {
      Element e = produce_element ();
      P ( mutex );                          //wait(mutex)
      enqueue ( e );
      V ( mutex );                          //signal(mutex)
      V ( available );                      //signal(available)
   }
}
consumer () {
   while (1) {
      P ( available );                      //wait(available)
      P ( mutex );                          //wait(mutex)
      Element e = dequeue ( );
      V ( mutex );                          //signal(mutex)
      consume_element ( e );
   }
}
```

2b) Verklemmungen (6 Punkte)

- Nennen Sie stichpunktartig die drei Vorbedingungen, die erfüllt sein müssen, damit es überhaupt zu einer Verklemmung kommen kann, und erklären Sie diese jeweils kurz mit eigenen Worten.
  - Exklusive Belegung von Betriebsmitteln (mutual exclusion)
    - die umstrittenen Betriebsmittel sind **nur unteilbar nutzbar**
  - Nachforderung von Betriebsmitteln (hold and wait)
    - die umstrittenen Betriebsmittel sind **nur schrittweise belegbar**
  - Kein Entzug von Betriebsmitteln (no preemption)
    - die umstrittenen Betriebsmittel sind **nicht rückforderbar**

3a) Platzierung/Ersetzung (4 Punkte)

- Erläutern Sie den Unterschied zwischen der Platzierungsstrategie (*placement policy*) und der Ersetzungsstrategie (*replacement policy)*
  - Die Platzierungsstrategie bestimmt woher benötigter Speicher genommen wird. (z.B. zur Minimierung des Verschnitts)
    - First/Last Fit
    - Best Fit
    - Worst Fit
    - Buddy-Verfahren
  - Die Ersetzungsstrategie bestimmt welche Speicherinhalte verdrängt werden sollen, falls kein freier Speicher mehr zu Verfügung steht.
    - LRU – Least recently used
    - FIFO – First in First out
    - Second Chance

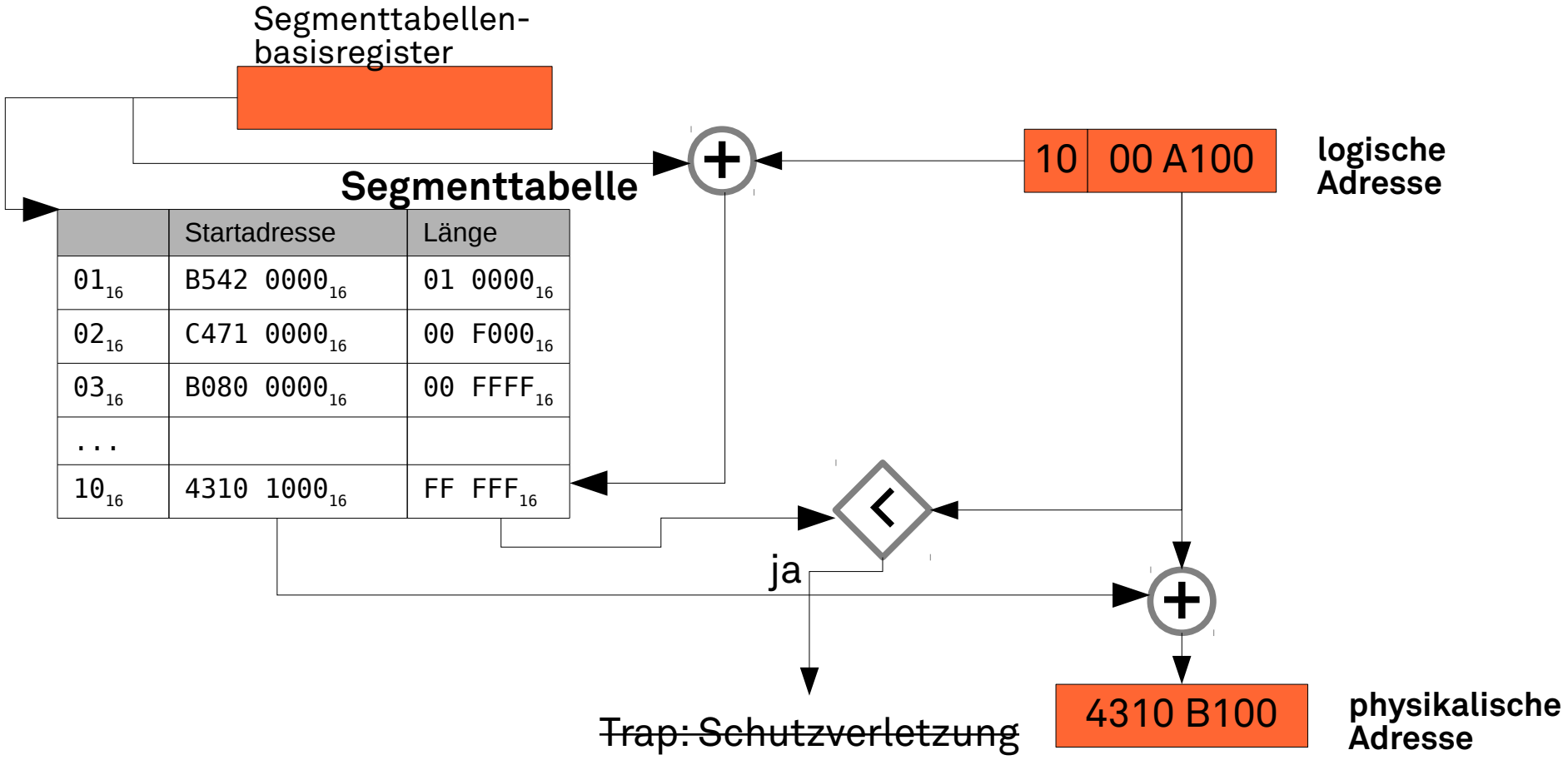
3b) Speichersegmentierung (4 Punkte)

- Geben Sie für die Speicheranfragen 0x1000 A100 und 0x030B 5000 die physikalische Adresse unter Anwendung des Speichersegmentierungsverfahrens an.
  Die höchstwertigen 8 Bit der logischen Adresse geben die Position innerhalb der Segementtabelle an. Löst eine Speicheranfrage eine Zugriffsverletzung aus, so machen Sie dies bitte kenntlich.

	Startadresse	Länge
01_{16}	$B542\ 0000_{16}$	$01\ 0000_{16}$
02_{16}	$C471\ 0000_{16}$	$00\ F000_{16}$
03_{16}	$B080\ 0000_{16}$	$00\ FFFF_{16}$
...		
10_{16}	$4310\ 1000_{16}$	$FF\ FFF_{16}$

3b) Speichersegmentierung (4 Punkte)

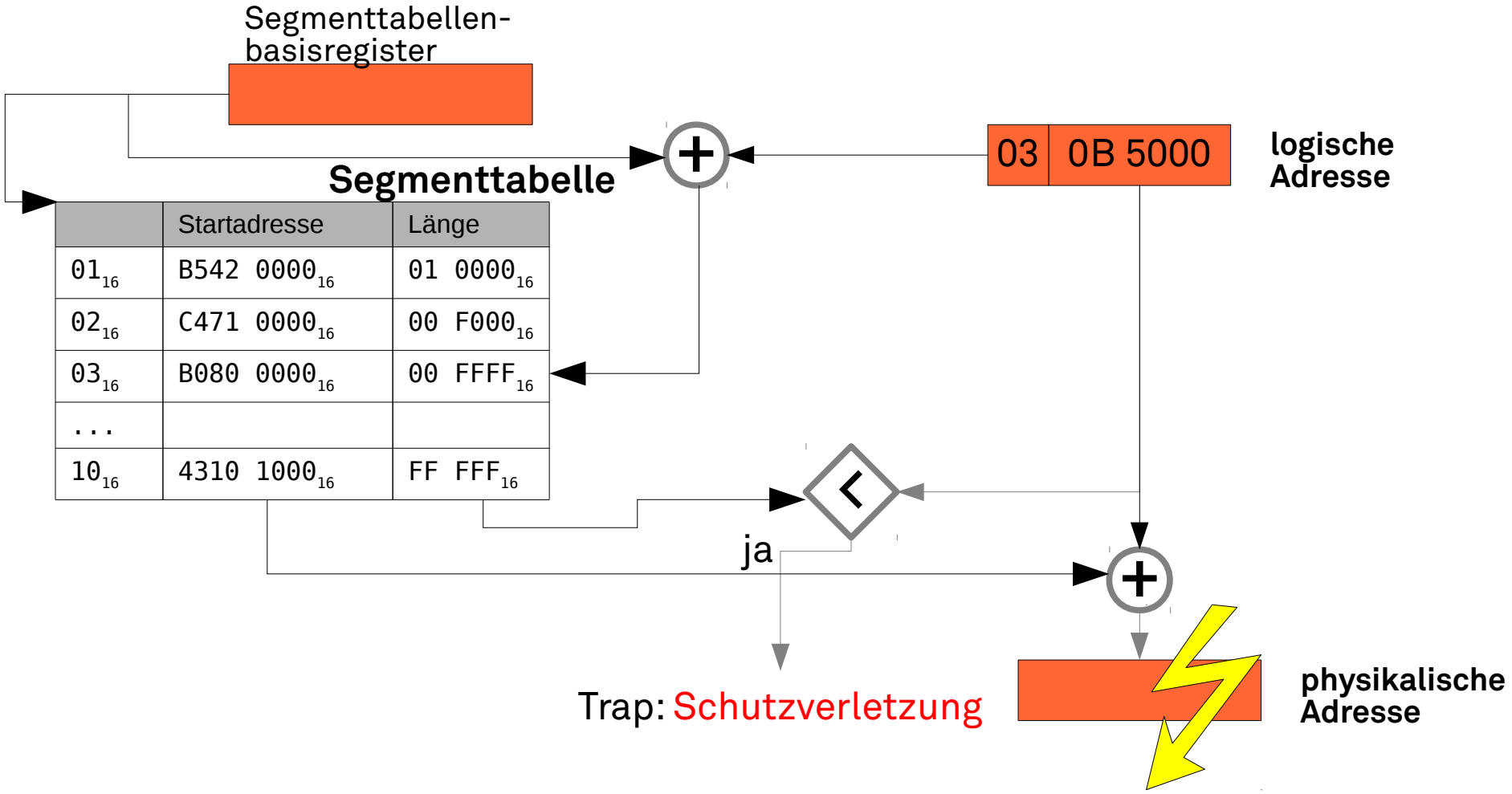
- Anfrage: `0x1000 A100`



	Startadresse	Länge
01_{16}	B542 0000_{16}	01 0000_{16}
02_{16}	C471 0000_{16}	00 $F000_{16}$
03_{16}	B080 0000_{16}	00 $FFFF_{16}$
...		
10_{16}	4310 1000_{16}	FF FFF_{16}

3b) Speichersegmentierung (4 Punkte)

- Anfrage: `0x030B 5000`



3c) Buddy-Verfahren (4 Punkte)

- **Szenario 1:** Prozess C belegt 3 MiB (aufgerundet 4 MiB)

0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
A	A	A	A							B	B				

Lösung:

0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
A	A	A	A					C	C	B	B				

3c) Buddy-Verfahren (4 Punkte)

- **Szenario 2:** Prozess D belegt 12 MiB (aufgerundet 16 MiB)

0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
A	A														

Lösung:

0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
A	A							D	D	D	D	D	D	D	D

3c) Buddy-Verfahren (4 Punkte)

- **Szenario 3:** Prozess E belegt 14 MiB (aufgerundet 16 MiB)

0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
		B	B									A	A		

Belegung ist nicht möglich

Lösung:

0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
		B	B									A	A		

4 MiB | 8 MiB | 8 MiB | 4 MiB

3c) Buddy-Verfahren (4 Punkte)

- **Szenario 4:** Prozess F belegt 7 MiB (aufgerundet 8 MiB)

0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
A	A	A	A	B	B										

Lösung:

0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
A	A	A	A	B	B			F	F	F	F				

4a) Block-Buffer-Cache (3 Punkte)

- Nennen und erläutern Sie drei Ereignisse, die das Rückschreiben des Block-Buffer-Caches auslösen.
  - Wenn kein freier Puffer mehr vorhanden ist
  - Bei Aufruf des Systemaufrufes sync()
  - Regelmäßig vom System
  - Nach jedem Schreibaufruf im Modus O_SYNC

4b) Kontinuierliche Speicherung (2 Punkte)

- Datei wird in Blöcken mit aufsteigenenden Blocknummern gespeichert
- Vorteile:
  - Zugriff auf alle Blöcke mit minimaler Positionierungszeit
  - Schneller direkter Zugriff auf bestimmte Dateipositionen
  - Gut geeignet für nicht-modifizierbare Datenträger (z.B. optische Medien)
- Nachteile:
  - Aufwändiges Finden von freiem, aufeinanderfolgendem Speicherplatz
  - Fragmentierungsproblem
  - Dateigröße von neuen Dateien oft nicht im Voraus bekannt
  - Erweitern bestehender Daten komplex
  - Umkopieren notwendig, wenn hinter Daten kein freier Platz ist

4c) IO-Scheduling (4,5 Punkte)

$$L_1 = \{1,4,7,2\} \quad L_2 = \{3,6,0\} \quad L_3 = \{5,2\}$$

Sofort bekannt

Nach 3 Ops bekannt

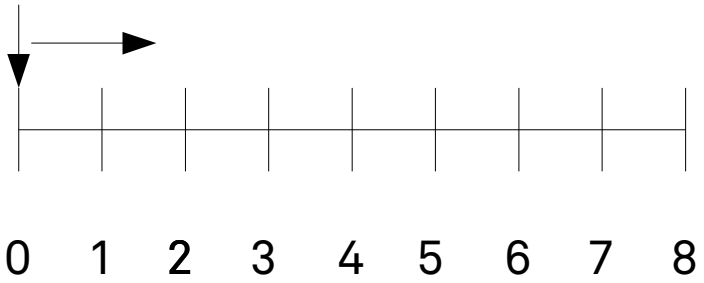
Nach 6 Ops bekannt

- Bitte tragen Sie hier die Reihenfolge der gelesenen Spuren für einen I/O-Scheduler, der nach der **Fahrstuhl (Elevator)** Strategie arbeitet, ein:

4c) IO-Scheduling (4,5 Punkte)

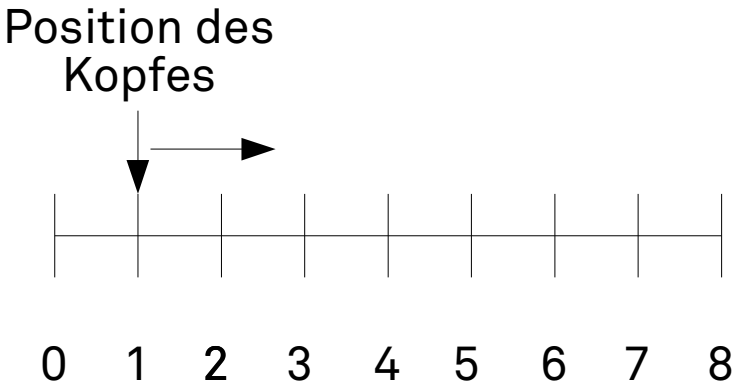
T = 0 I/O-Anfragen:
1, 4, 7, 2

Position des
Kopfes



4c) IO-Scheduling (4,5 Punkte)

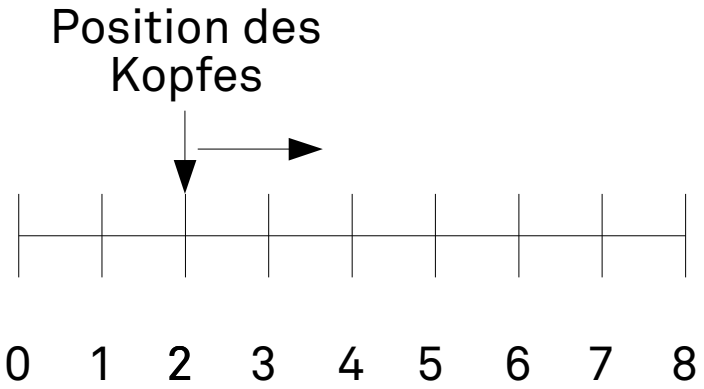
T = 1 I/O-Anfragen:
4, 7, 2



1								

4c) IO-Scheduling (4,5 Punkte)

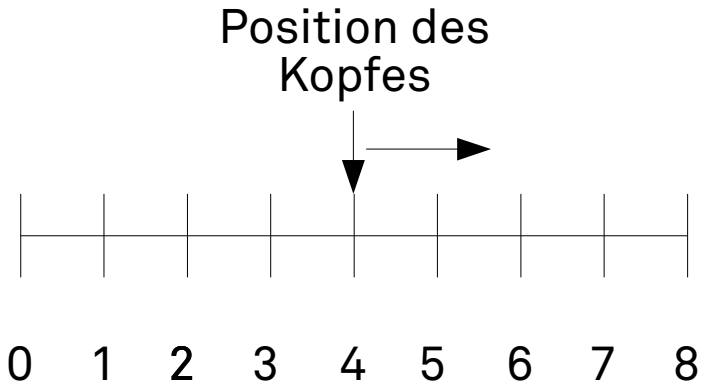
T = 2 I/O-Anfragen:
4, 7



1	2							

4c) IO-Scheduling (4,5 Punkte)

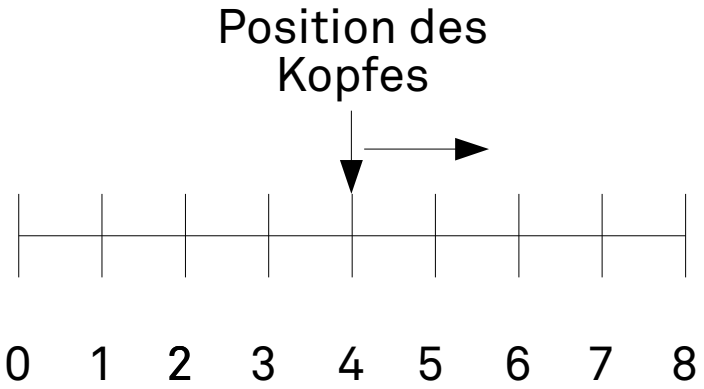
T = 3 I/O-Anfragen:
7



1	2	4						

4c) IO-Scheduling (4,5 Punkte)

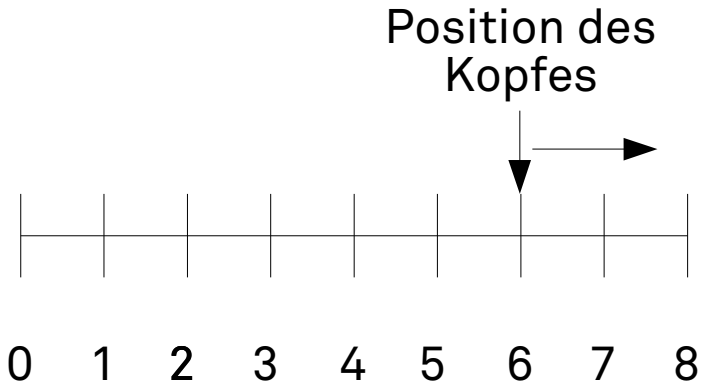
T = 3 I/O-Anfragen:
7, 3, 6, 0



1	2	4						

4c) IO-Scheduling (4,5 Punkte)

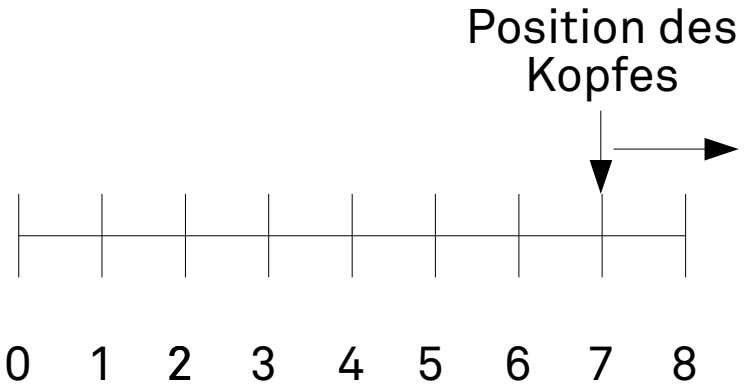
T = 4 I/O-Anfragen:
7, 3, 0



1	2	4	6					

4c) IO-Scheduling (4,5 Punkte)

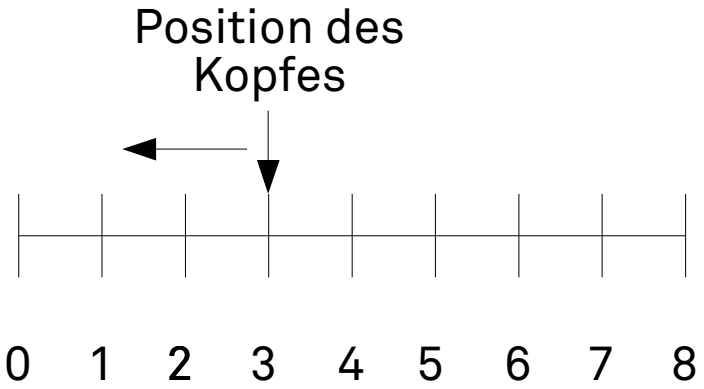
T = 5 I/O-Anfragen:
3, 0



1	2	4	6	7				

4c) IO-Scheduling (4,5 Punkte)

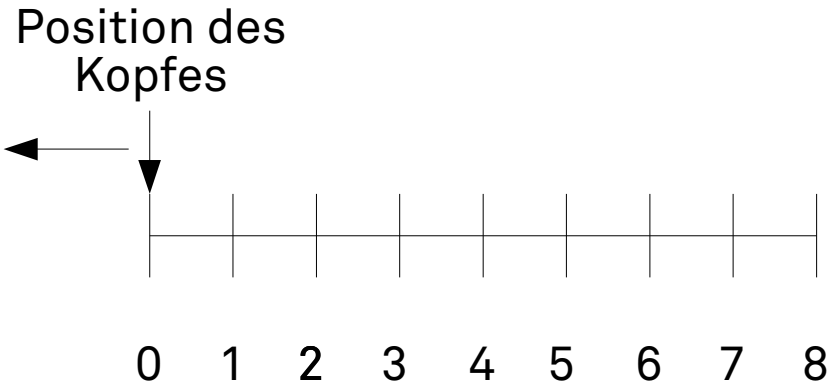
T = 6 I/O-Anfragen:
0



1	2	4	6	7	3			

4c) IO-Scheduling (4,5 Punkte)

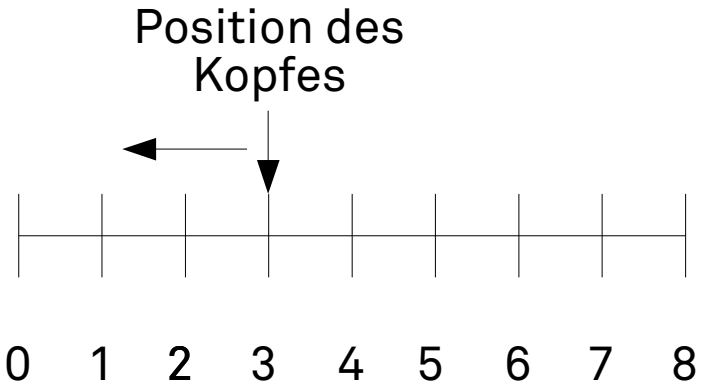
T = 7 I/O-Anfragen:
5, 2



1	2	4	6	7	3	2		

4c) IO-Scheduling (4,5 Punkte)

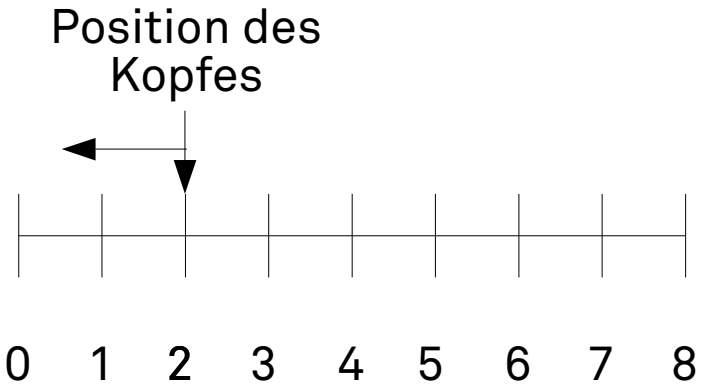
T = 6 I/O-Anfragen:
0, 5, 2



1	2	4	6	7	3			

4c) IO-Scheduling (4,5 Punkte)

T = 7 I/O-Anfragen:
0, 5

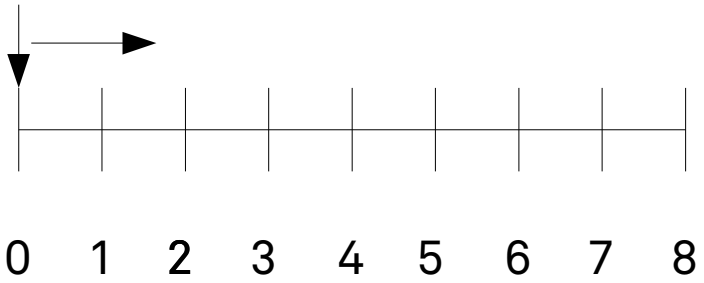


1	2	4	6	7	3	2		

4c) IO-Scheduling (4,5 Punkte)

T = 8 I/O-Anfragen:
5,

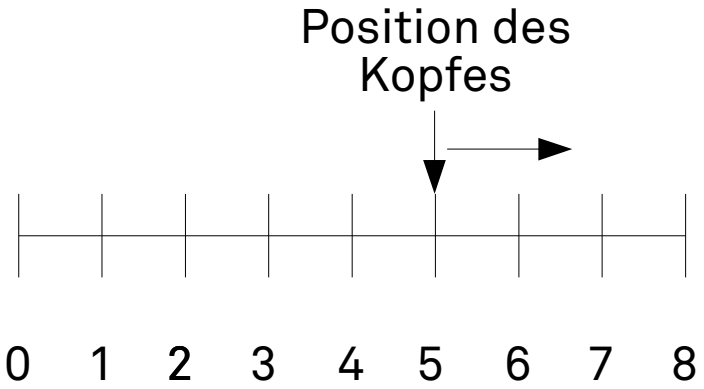
Position des
Kopfes



1	2	4	6	7	3	2	0	

4c) IO-Scheduling (4,5 Punkte)

T = 9 I/O-Anfragen:



1	2	4	6	7	3	2	0	5

Auswertung

- Bitte schnell einmal die Punkte zusammenzählen …
- Notenspiegel:

Punkte	Note
38,5–45	1
33,5–38	2
28–33	3
22,5–27,5	4
0–22	5

Weitere Hinweise zur Vorbereitung

- Inhalt der Folien lernen
  - Klassifizieren: Was muss ich **lernen**? Was muss ich **begreifen**?
- Übungsaufgaben verstehen, C und UNIX „können“
  - AsSESS-System bleibt mindestens bis zur Klausur offen
    - bei Fragen zur Korrektur melden
  - Am besten die Aufgaben noch einmal lösen
  - Optionale Zusatzaufgaben bearbeiten
- Literatur zur Lehrveranstaltung durchlesen
- BS-Forum nutzen

Empfohlene Literatur

[1] A. Silberschatz et al. *Operating System Concepts*. Wiley, 2004. ISBN 978-0471694663

[2] A. Tanenbaum: Modern Operating Systems (2nd ed.). Prentice Hall, 2001. ISBN 0-13-031358-0

[3] B. W. Kernighan, D. M. Ritchie. *The C Programming Language*. Prentice-Hall, 1988. ISBN 0-13-110362-8 (paperback) 0-13-110370-9 (hardback)

[4] R. Stevens, *Advanced Programming in the UNIX Environment*, Addison-Wesley, 2005. ISBN 978-0201433074

Viel Erfolg bei der Klausur!