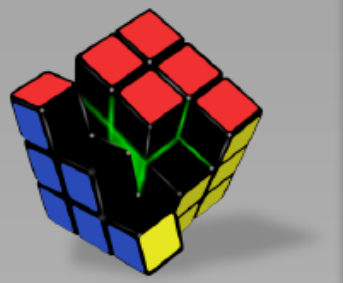


technische universität
dortmund

Betriebssysteme

Probeklausur

https://ess.cs.tu-dortmund.de/DE/Teaching/SS2017/BS/

Olaf Spinczyk

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

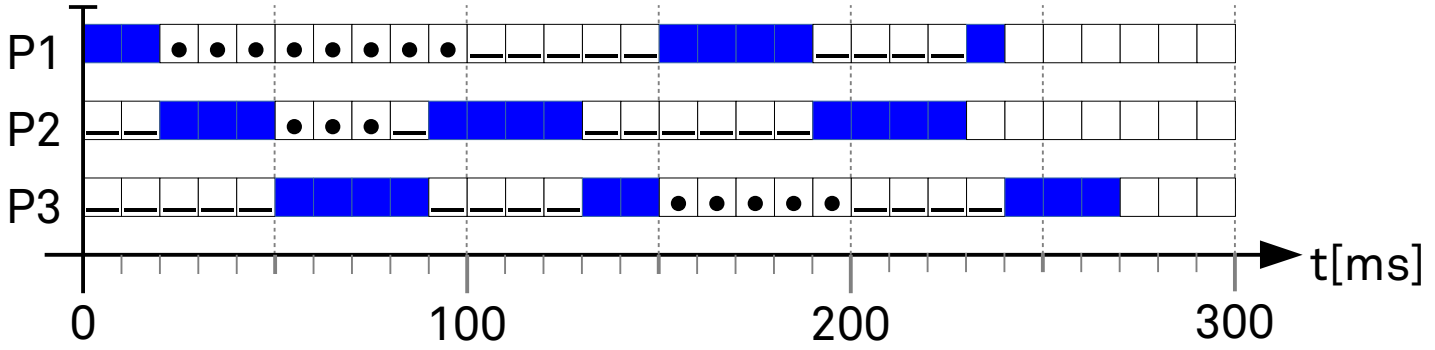
Probeklausur

... 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) Prozesserzeugung (7 Punkte)

- Was geben die folgenden Programme aus?

```
int pferd;

void* erzeugePferd(void* param)
{
    pferd++;
    printf("%d\n", pferd);
    return NULL;
}

int main(void) {
    pferd = 42;
    if (fork() == 0) {
        erzeugePferd(NULL);
    } else {
        erzeugePferd(NULL);
    }
    return 0;
}
```

43 43

```
int pferd;

void* erzeugePferd(void* param)
{
    pferd++;
    printf("%d\n", pferd);
    pthread_exit(NULL);
}

int main(void) {
    pferd = 42;
    pthread_t id;
    pthread_create(&id, NULL,
        erzeugePferd, NULL);
    erzeugePferd(NULL);
    return 0;
}
```

43 44

1b) Prozesserzeugung (7 Punkte)

- Wieso ist die Ausgabe unterschiedlich?
  - fork() erzeugt schwergewichtige Prozesse
    - Variable “pferd” liegt danach in unterschiedlichen Speicherbereichen
    - Änderungen sind unabhängig voneinander
  - pthread_create erzeugt leichtgewichtige POSIX-Threads
    - Variable “pferd” liegt im gemeinsam genutzten Datensegment
    - Beide Threads verändern die gleiche Variable

2a) Synchronisation (10 Punkte)

- Gemeinsamer Speicher, mehrere Leser, ein Schreiber

```
/* gem. Speicher */
Semaphore mutex = ?;
Semaphore wrt = ?;
int readcount = ?;
```

```
void schreiber() {
   ?
   schreibe_Daten();
   ?
}
```

```
void leser() {
   ?
   readcount = ?;
   if (readcount == ?) {
      ?
   }
   ?
   lese_Daten();
   ?
   readcount = ?;
   if (readcount == ?) {
      ?
   }
   ?
}
```

2a) Synchronisation (10 Punkte)

- Gemeinsamer Speicher, mehrere Leser, ein Schreiber

```
/* gem. Speicher */
Semaphore mutex = 1;
Semaphore wrt = 1;
int readcount = 0;
```

```
void schreiber() {
   wait(&wrt);
   schreibe_Daten();
   signal(&wrt);
}
```

```
void leser() {
   wait(&mutex);
   readcount = readcount + 1;
   if (readcount == 1) {
      wait(&wrt);
   }
   signal(&mutex);
   lese_Daten();
   wait(&mutex);
   readcount = readcount – 1;
   if (readcount == 0) {
      signal(&wrt);
   }
   signal(&mutex);
}
```

2b) Verklemmungen (4,5 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**

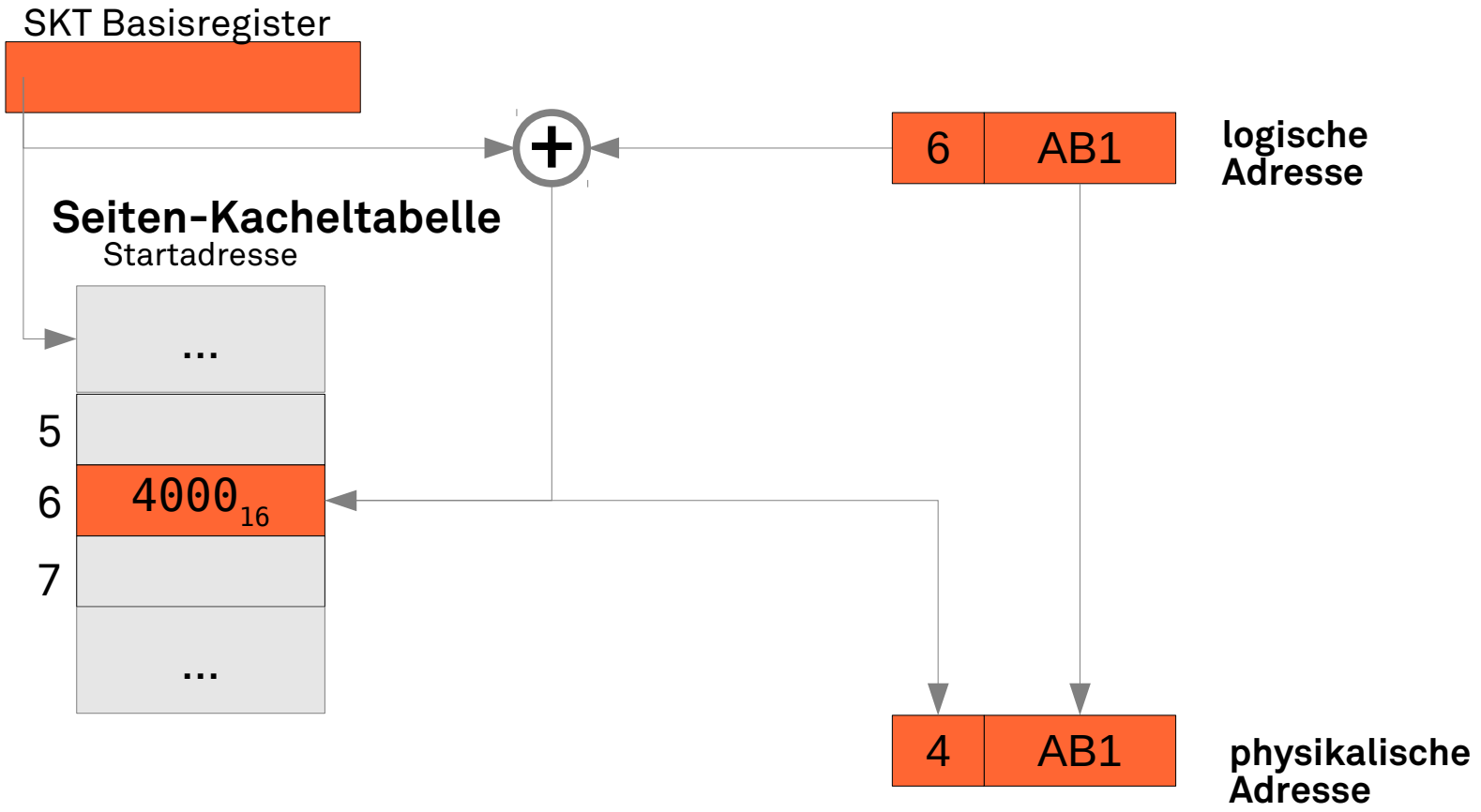
3a) Adressabbildung (4 Punkte)

- In einem System mit Seitenadressierung (paging) befindet sich die Seitenkacheltabelle im unten angegebenen Zustand. Die Adresslänge beträgt 16 Bit, wovon 12 Bit für den Offset innerhalb der Seite verwendet werden (Seitengröße: 4096 Bytes). Bilden Sie die logischen Adressen 6AB1 und F1B7 auf ihre physikalischen Adressen ab. (Hinweis: Eine Hexadezimalziffer stellt immer genau vier Bit der Adresse dar.)

Seitennummer	Startadresse
…	…
6	4000_{16}
…	…
15	5000_{16}

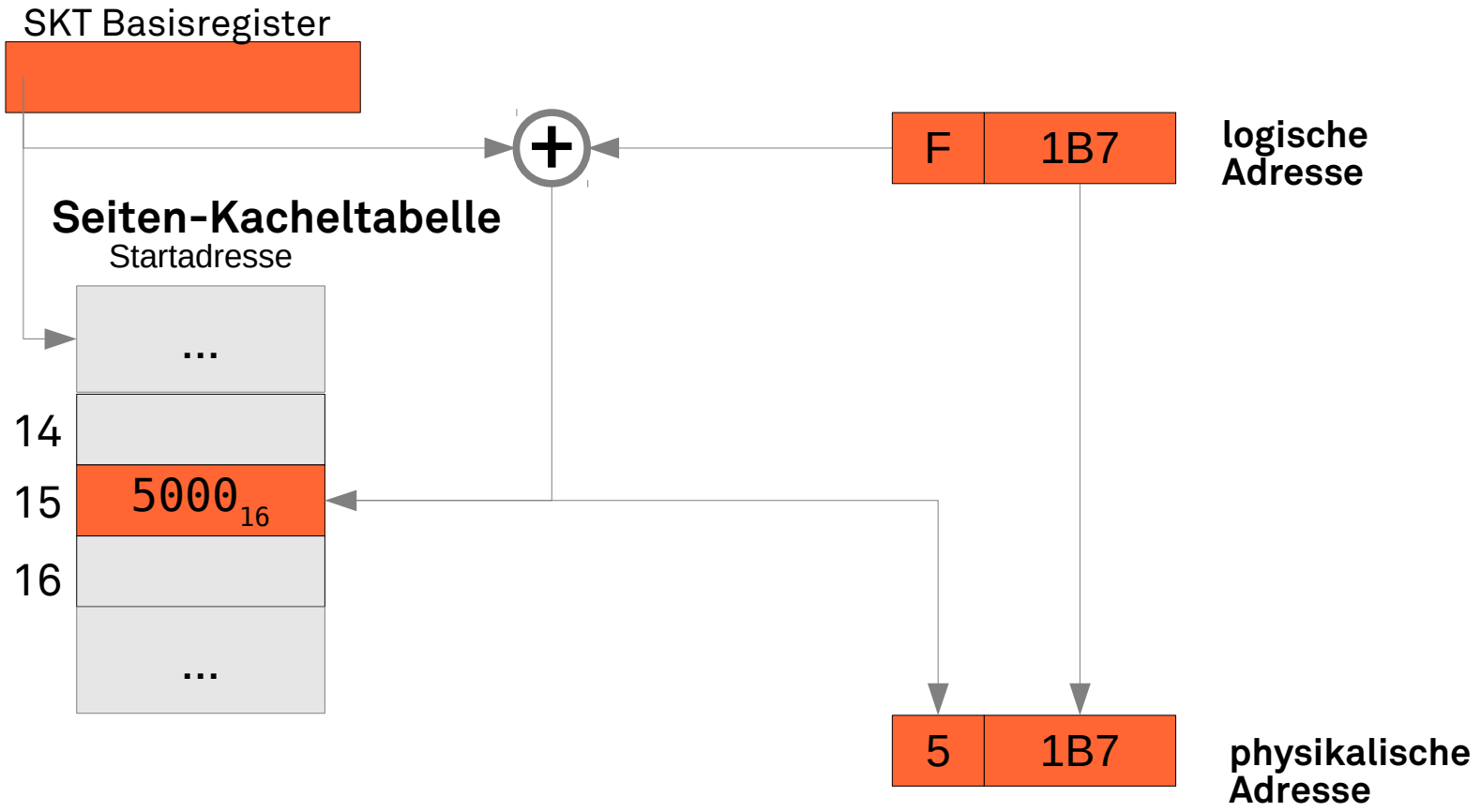
3a) Adressabbildung (4 Punkte)

- Logische Adresse: $6AB1_{16}$



3a) Adressabbildung (4 Punkte)

- Logische Adresse: $F1B7_{16}$



3b) 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				

3b) 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

3b) 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

3b) 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 aufeinander folgenden 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

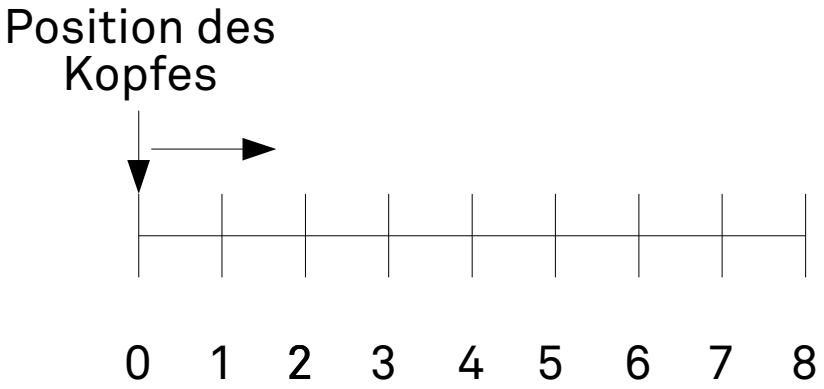
Nach 3 Ops bekannt

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:

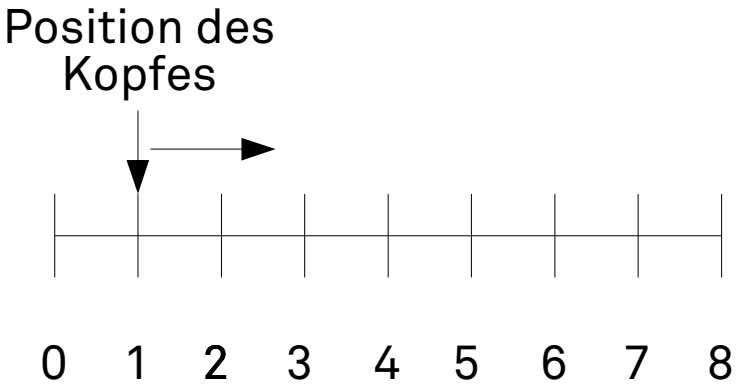
4c) IO-Scheduling (4,5 Punkte)

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



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

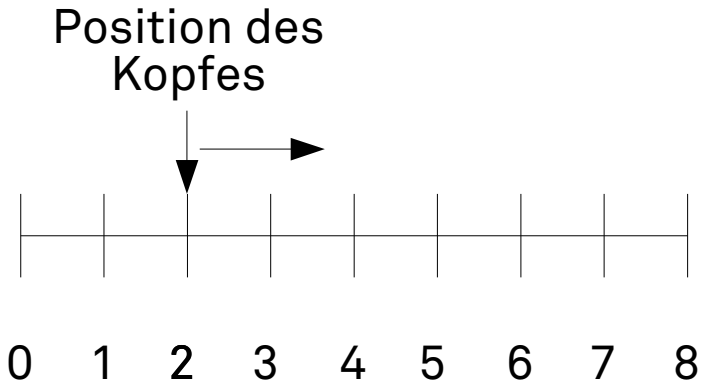
T = 1 I/O-Anfragen:
4, 7, 2



1								

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

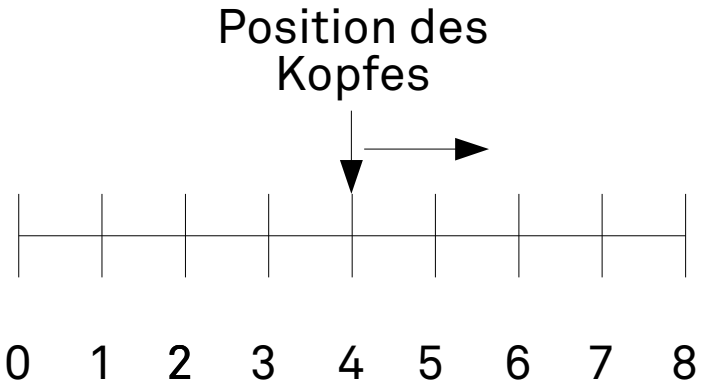
T = 2 I/O-Anfragen:
4, 7



1	2							

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

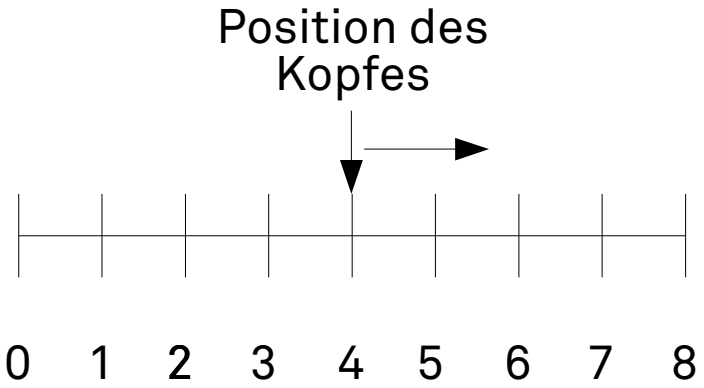
T = 3 I/O-Anfragen:
7



1	2	4						

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

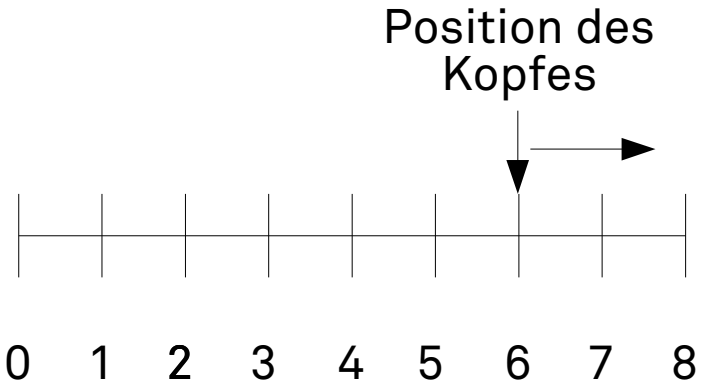
T = 3 I/O-Anfragen:
7, 3, 6, 0



1	2	4						

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

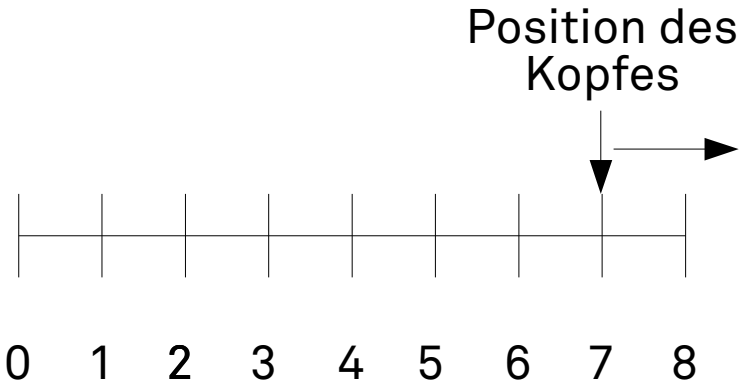
T = 4 I/O-Anfragen:
7, 3, 0



1	2	4	6					

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

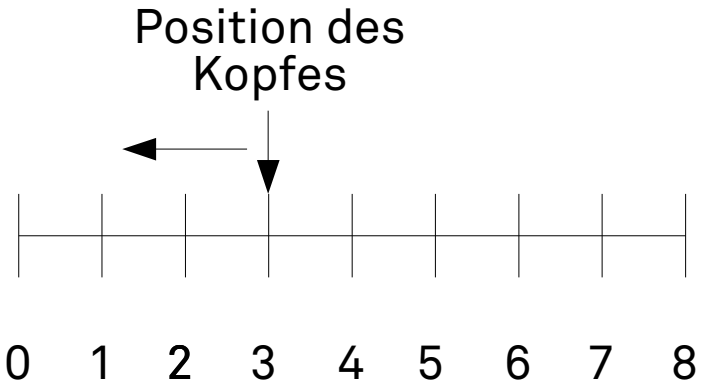
T = 5 I/O-Anfragen:
3, 0



1	2	4	6	7				

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

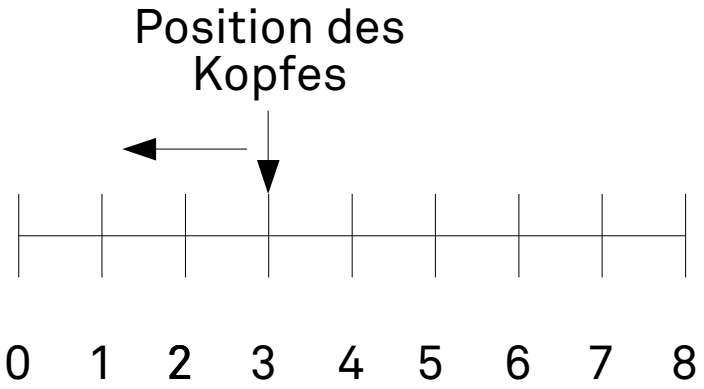
T = 6 I/O-Anfragen:
0



1	2	4	6	7	3			

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

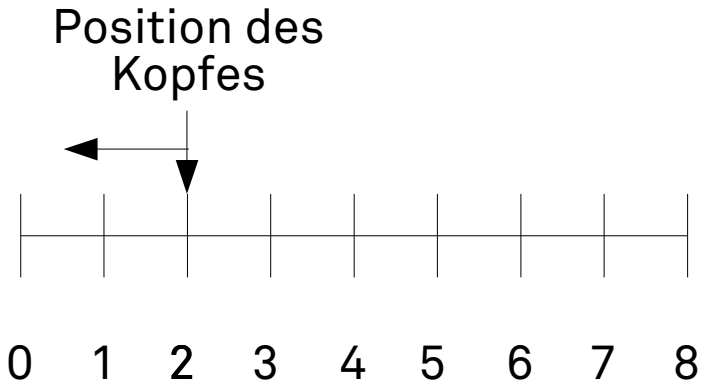
T = 6 I/O-Anfragen:
0, 5, 2



1	2	4	6	7	3			

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

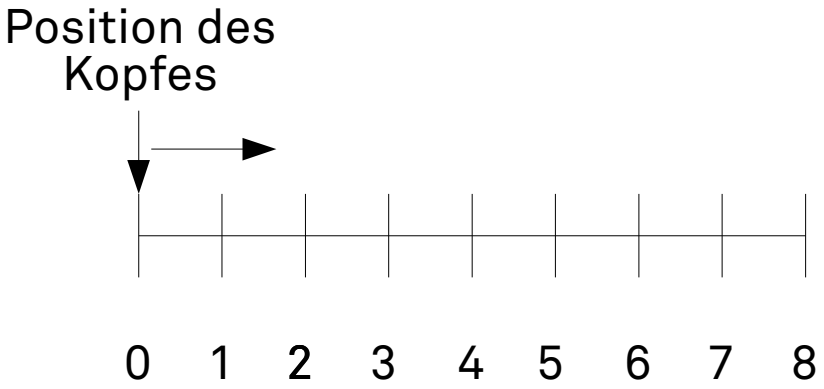
T = 7 I/O-Anfragen:
0, 5



1	2	4	6	7	3	2		

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

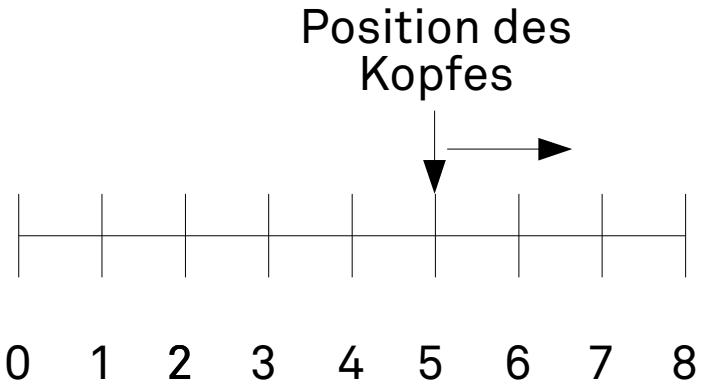
T = 8 I/O-Anfragen:
5



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!