Tutorübung Grundlagen: Betriebssysteme und Systemsoftware

Moritz Beckel

München, 8. Februar 2023

Mittwoch 14:15-16:00 Uhr Online (https://bbb.in.tum.de/mor-6ij-iuw-ypm)

Zulip-Stream https://zulip.in.tum.de/#narrow/stream/1296-GBS-Mi-1400-A

Unterrichtsmaterialien findest du hier:

https://home.in.tum.de/~beckel/gbs

Folien wurden von mir selbst erstellt. Es besteht keine Garantie auf Korrektheit.

Aufgabe 1 Klausur 2020

Wir befinden uns auf einer fiktiven 16-bit little-endian Architektur (die Werte werden demnach in der umgekehrten Bytefolge interpretiert). Eine Speicheradresse ist somit 16 bit breit. Ein Char sei 8 bit breit, ein Integer sei 16 bit breit, ein Long 32 bit. Auf den Speicher kann beliebig byteweise zugegriffen werden, es wird kein Alignment enforced.

```
1  int* i = (int*) 0xca;
2
3  int* a = &i[2];
4  long b = *(long*)*(int*)*i;
5  char* c = ((char*)i)+0x6;
```

a) Wie viele Hex-Zeichen umfasst eine Speicheradresse im obigen Hexdump?

b) An der Adresse 0xf4 ist ein Pointer gespeichert. Geben Sie den Pointer in big-endian als Hexadezimalwert an.

c) Welcher Wert wird der Variablen a (Zeile 3) zugewiesen? Geben Sie den Wert als Hexadezimalwert in big-endian an.

d) Welcher Wert wird der Variablen b (Zeile 4) zugewiesen? Geben Sie den Wert als Hexadezimalwert in big-endian an.

e) Geben Sie den String an, der durch die Variable c (Zeile 5) referenziert wird. (Hinweis: Nutzen Sie man 7 ascii oder Abbildung 1.1)

```
0x0020  19 ab 4d 00 80 89 22 95  5b 81 31 00 3a 00 8b 89
0x0030  58 b7 96 01 0f 00 00 ed  10 48 24 01 f8 00 01 00
0x0040  d1 63 f5 04 b0 00 c6 2f  78 c7 e6 e1 a6 fe a3 98
0x0050  01 a3 01 00 31 b0 5f 40  51 00 45 ab 60 78 b0 2e
0x0060  04 4f fa c8 44 f1 ef 1d  14 45 a5 e9 e6 76 5b 50
0x0070  23 43 87 57 00 10 9c 01  27 4d 06 a2 4b 3c 4c 28
0x0080  d8 5a 83 fc 0c c9 dd 31  d1 3c ee 8a ff 6f c8 00
0x0090  96 bf 66 d1 a3 9c 27 a2  b8 c0 d9 65 1e ec 5e ff
0x00a0  75 b3 dc d4 f3 12 78 f9  0f e2 1c 09 bc 28 00 00
0x00b0  71 3f c8 b8 3b 6b a0 87  99 00 01 a2 04 47 41 02
0x00c0  b1 00 9a a5 17 b1 01 1a  7a 6e 3c 00 63 b9 be 3c
0x00d0  7a 65 72 6f 00 82 04 a9  ca e0 93 85 2e f4 5c 3d
0x00e0  2f 31 aa 8c 2c 29 f4 ad  21 a7 6b 49 5b a0 e3 91
0x00f0  bc e7 ca 44 85 9f fc 01  21 77 8e 00 e9 32 c6 69
0x0100  76 6f 69 64 00 00 60 c9  47 60 ad 00 42 ec a7 86
0x0110  6e 65 6c 6c 00 81 8d 2d  9b 26 a5 d0 4b f9 03 51
0x0120  92 00 86 92 00 10 af b3  f6 54 f0 f5 c6 00 c5 48
0x0130  f7 58 3b 19 72 38 fb df  be ab c7 7d 8a 46 6c 5b
0x0140  eb 16 80 ab ea 7e 73 14  6f c0 f7 35 7b 65 00 15
```

```
1  int* i = (int*) 0xca;
2
3  int* a = &i[2];
4  long b = *(long*)*(int*)*i;
5  char* c = ((char*)i)+0x6;
```

a) Wie viele Hex-Zeichen umfasst eine Speicheradresse im obigen Hexdump?
   - 4

b) An der Adresse 0xf4 ist ein Pointer gespeichert. Geben Sie den Pointer in big-endian als Hexadezimalwert an.
   - 0x 9f 85

c) Welcher Wert wird der Variablen a (Zeile 3) zugewiesen? Geben Sie den Wert als Hexadezimalwert in big-endian an.
   - 0x ce

d) Welcher Wert wird der Variablen b (Zeile 4) zugewiesen? Geben Sie den Wert als Hexadezimalwert in big-endian an.
   - 0x 8e 77 21

e) Geben Sie den String an, der durch die Variable c (Zeile 5) referenziert wird. (Hinweis: Nutzen Sie man 7 ascii oder Abbildung 1.1)
   - zero

```
0x0020  19 ab 4d 00 80 89 22 95  5b 81 31 00 3a 00 8b 89
0x0030  58 b7 96 01 0f 00 00 ed  10 48 24 01 f8 00 01 00
0x0040  d1 63 f5 04 b0 00 c6 2f  78 c7 e6 e1 a6 fe a3 98
0x0050  01 a3 01 00 31 b0 5f 40  51 00 45 ab 60 78 b0 2e
0x0060  04 4f fa c8 44 f1 ef 1d  14 45 a5 e9 e6 76 5b 50
0x0070  23 43 87 57 00 10 9c 01  27 4d 06 a2 4b 3c 4c 28
0x0080  d8 5a 83 fc 0c c9 dd 31  d1 3c ee 8a ff 6f c8 00
0x0090  96 bf 66 d1 a3 9c 27 a2  b8 c0 d9 65 1e ec 5e ff
0x00a0  75 b3 dc d4 f3 12 78 f9  0f e2 1c 09 bc 28 00 00
0x00b0  71 3f c8 b8 3b 6b a0 87  99 00 01 a2 04 47 41 02
0x00c0  b1 00 9a a5 17 b1 01 1a  7a 6e 3c 00 63 b9 be 3c
0x00d0  7a 65 72 6f 00 82 04 a9  ca e0 93 85 2e f4 5c 3d
0x00e0  2f 31 aa 8c 2c 29 f4 ad  21 a7 6b 49 5b a0 e3 91
0x00f0  bc e7 ca 44 85 9f fc 01  21 77 8e 00 e9 32 c6 69
0x0100  76 6f 69 64 00 00 60 c9  47 60 ad 00 42 ec a7 86
0x0110  6e 65 6c 6c 00 81 8d 2d  9b 26 a5 d0 4b f9 03 51
0x0120  92 00 86 92 00 10 af b3  f6 54 f0 f5 c6 00 c5 48
0x0130  f7 58 3b 19 72 38 fb df  be ab c7 7d 8a 46 6c 5b
0x0140  eb 16 80 ab ea 7e 73 14  6f c0 f7 35 7b 65 00 15
```

Aufgabe 2 Klausur 2020

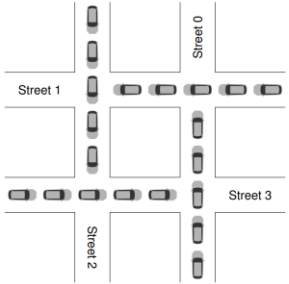


Abbildung 2.1: Deadlock

a) Benennen Sie die vier Deadlock-Bedingungen nach Coffman und beschreiben Sie, in welcher Form sich diese in Abbildung 2.1 zeigen.
b) Geben Sie eine konkrete Lösung zur Auflösung/Vermeidung des Deadlocks aus Abbildung 2.1 an und nennen Sie die Deadlock Bedingung, die dabei außer Kraft gesetzt wird.
c) Für welche Seiten (Pages) könnte es sinnvoll sein, dass sie von zwei Prozessen gleichzeitig genutzt werden (von Interprozesskommunikation abgesehen), wenn beide Prozesse auf dem gleichen Programm basieren?
d) Nennen und beschreiben Sie drei gängige Speicherschutzmaßnahmen auf Betriebssystemebene, die da Ausnutzen von Speicherfehlern (zum Beispiel: Buffer-Overflow, . . .) erschweren.

Aufgabe 2 Klausur 2020

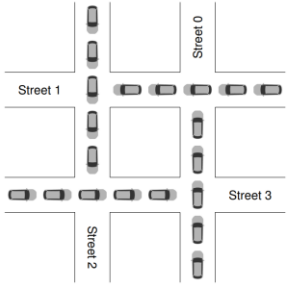


Abbildung 2.1: Deadlock

a) Benennen Sie die vier Deadlock-Bedingungen nach Coffman und beschreiben Sie, in welcher Form sich diese in Abbildung 2.1 zeigen.

- Hold and Wait / Belegen und Anfordern Fahrzeuge belegen eine Kreuzung bis sie weiterfahren können.
- No Preemption / Nicht entziehbar Fahrzeuge können nur selber eine Kreuzung verlassen und nicht abgeschleppt bzw. anderweitig entfernt werden
- Mutual Exclusion / Wechselseitiger Ausschluss Zu einem gegebenen Zeitpunkt kann stets nur ein Fahrzeug eine Kreuzung überqueren.
- Circular Wait / Zyklische Wartebedingung Die Straßen verbinden die vier Kreuzungen zyklisch.

Aufgabe 2 Klausur 2020

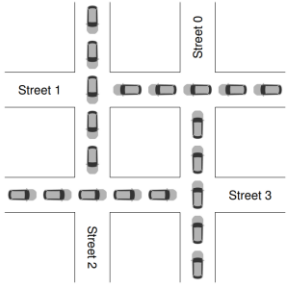


Abbildung 2.1: Deadlock

b) Geben Sie eine konkrete Lösung zur Auflösung/Vermeidung des Deadlocks aus Abbildung 2.1 an und nennen Sie die Deadlock Bedingung, die dabei außer Kraft gesetzt wird.

- Mutual Exclusion Eine Kreuzung wird durch eine Unterführung ersetzt, sodass Fahrzeuge beider Straßen unabhängig von einander die Kreuzung passieren können.

Aufgabe 2 Klausur 2020

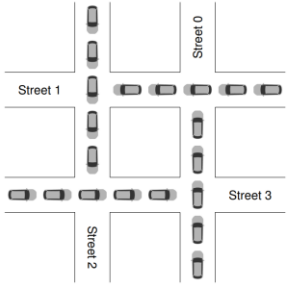


Abbildung 2.1: Deadlock

c) Für welche Seiten (Pages) könnte es sinnvoll sein, dass sie von zwei Prozessen gleichzeitig genutzt werden (von Interprozesskommunikation abgesehen), wenn beide Prozesse auf dem gleichen Programm basieren?

- Die Prozesse können sich alle read-only Seiten teilen (Code/Text-Segment, rodata Segment (.rodata), Bibliotheken, etc.)

Aufgabe 2 Klausur 2020

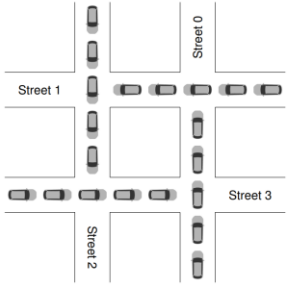


Abbildung 2.1: Deadlock

d) Nennen und beschreiben Sie drei gängige Speicherschutzmaßnahmen auf Betriebssystemebene, die da Ausnutzen von Speicherfehlern (zum Beispiel: Buffer-Overflow, . . .) erschweren.

- Stack Shielding / Stack Guards / Stack Canarieswerden vom Compiler zwischen der Return Adresse und lokalen Variablen auf dem Stack einer Funktion eingefügt. Das Betriebssystem überprüft die Integrität des Stack Canaries, bevor die Funktion returned.
- Data Execution Prevention (DEP). Der Stack wird vom Betriebssystem durch das NX-Bit als non-executable markiert und verhindert somit die Ausführung vom einem potenziell eingeschleusten Shellcode auf dem Stack.
- Address Space Layout Randomization (ASLR)randomisiert die Adresse des Stacks und erschwert dem Angreifer diesen zu finden.auch richtig: Position Independet Executable

Aufgabe 2 Klausur 2020

e) Ordnen Sie die folgenden Virtualisierungsmöglichkeiten den am besten passenden Szenarien zu. Tragen Sie dazu die Nummer der Virtualisierungstechnik bei dem jeweiligen Szenario ein.

1. Para Virtualization
2. Linux Container
3. Hardware-assisted virtualization
4. Binary Translation

a) Dedizierte Gerätetreiber kommunizieren mit dem Hypervisor durch sog. Hypercalls
b) Hypervisor interpretiert/emuliert (Teile des) Binärcode(s) der Gast-VM in Software
c) Nutzt Linux cgroups und namespaces zur Abschottung
d) Benutzung von Intel VT-x, AMD-V, etc.

Aufgabe 2 Klausur 2020

e) Ordnen Sie die folgenden Virtualisierungsmöglichkeiten den am besten passenden Szenarien zu. Tragen Sie dazu die Nummer der Virtualisierungstechnik bei dem jeweiligen Szenario ein. 1a, 2c, 3d, 4b

1. Para Virtualization
2. Linux Container
3. Hardware-assisted virtualization
4. Binary Translation

a) Dedizierte Gerätetreiber kommunizieren mit dem Hypervisor durch sog. Hypercalls
b) Hypervisor interpretiert/emuliert (Teile des) Binärcode(s) der Gast-VM in Software
c) Nutzt Linux cgroups und namespaces zur Abschottung
d) Benutzung von Intel VT-x, AMD-V, etc.

Aufgabe 3 Klausur 2020

Die 65536 = 0x10000 Bytes eines gegebenen virtuellen Speichers sollen mittels eines zweistufigen Paging- Verfahrens auf Kacheln der Größe 256 = 0x100 Bytes abgebildet werden. Insgesamt stehen 2048 = 0x800 Bytes vollständig adressierbaren physischen Hauptspeichers zur Verfügung. Nehmen Sie an, dass die nicht auf den Seitenoffset entfallenden Bits einer virtuellen Adresse gleichmäßig für die Auswahl der Einträge der ersten und der zweiten Ebene der Seitentabellen verwendet werden. Gehen Sie ferner davon aus, dass die Seitengröße der Kachelgröße (Page Size = Frame Size) entspricht. Vereinfachend nehmen wir an, dass in den Seitentabellen ausschließlich Adressinformationen (d.h. keine zusätzlichen Bits wie present, dirty, ...) gespeichert werden. Berechnen Sie auf Basis dieser Informationen die folgenden Werte:

Anzahl Bits einer physischen Adresse; Anzahl Bits einer virtuellen Adresse; Anzahl der Bits einer virtuellen Adresse, die auf das Offset innerhalb einer Seite entfallen; Anzahl der Bits einer virtuellen Adresse, die pro Paging-Stufe verwendet werden; Anzahl Seiten (Pages) im virtuellen Speicher bei Vollbelegung; Anzahl Kacheln (Frames) im physischen Speicher bei Vollbelegung; Anzahl Einträge in der Tabelle, die für den ersten Schritt der Adressübersetzung verwendet wird; Größe einer Tabelle, die für den zweiten Schritt der Adressübersetzung verwendet wird in Bits

Aufgabe 3 Klausur 2020

Anzahl Bits einer physischen Adresse: 11
Anzahl Bits einer virtuellen Adresse: 16
Anzahl der Bits einer virtuellen Adresse, die auf das Offset innerhalb einer Seite entfallen: 8
Anzahl der Bits einer virtuellen Adresse, die pro Paging-Stufe verwendet werden: 4
Anzahl Seiten (Pages) im virtuellen Speicher bei Vollbelegung: 256
Anzahl Kacheln (Frames) im physischen Speicher bei Vollbelegung: 8
Anzahl Einträge in der Tabelle, die für den ersten Schritt der Adressübersetzung verwendet wird: $2^4 = 16$
Größe einer Tabelle, die für den zweiten Schritt der Adressübersetzung verwendet wird in Bits: $2^4 * 3$ Bit = 48 Bit

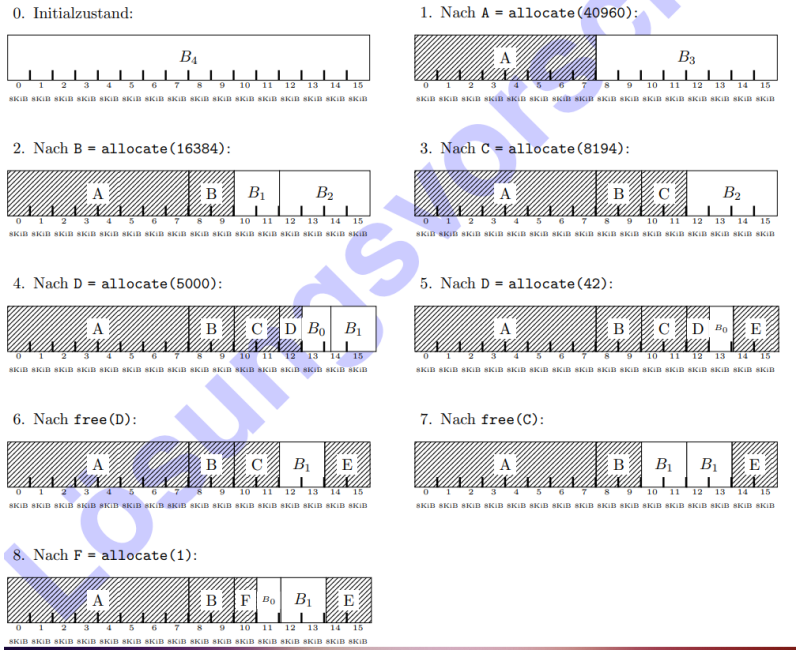
Aufgabe 5 Klausur 2020

a) Gegeben sei der Buddy-Allocator-Algorithmus gemäß Vorlesung, sowie der Pseudocode des folgenden Programms. Gehen Sie von einer minimalen Blockgröße (B0) von 8192 Bytes (8 KiB) und einer Gesamtspeichergröße von 131072 Bytes (128 KiB) aus.

```
1 A = allocate(40960);
2 B = allocate(16384);
3 C = allocate(8194);
4 D = allocate(5000);
5 E = allocate(9000);
6 free(D);
7 free(C);
8 F = allocate(1);
```

Zeichnen Sie die gesamte Belegung des Speichers nach der Ausführung jedes Statements aus dem obigen Pseudocode auf. Kennzeichnen Sie belegte Blöcke mit dem Buchstaben der Belegung, lassen Sie unbelegte Blöcke frei. Trennen Sie Blöcke mittels senkrechter Striche. Geben Sie bei jeder von ihnen verwendeten Vorlage an, welche Zeile des Pseudocodes sie abbildet.

Aufgabe 5 Klausur 2020



Aufgabe 5 Klausur 2020

b) Ein Least Recently Used (LRU) Seitenersetzungsalgorithmus wird auf einem System mit 4 Kacheln (f0-f3) und 9 Seiten (0-8) verwendet. Zu beginn sind alle Kacheln (f0-f3) leer. Geben Sie die Seiten (in Reihenfolge des Auftretens) an, welche einen Pagefault auslösen.

Zugriffsreihenfolge: 0, 8, 4, 2, 2, 5, 0, 2, 1 Pagefaults (in Reihenfolge des Auftretens):

Aufgabe 5 Klausur 2020

b) Ein Least Recently Used (LRU) Seitenersetzungsalgorithmus wird auf einem System mit 4 Kacheln (f0-f3) und 9 Seiten (0-8) verwendet. Zu beginn sind alle Kacheln (f0-f3) leer. Geben Sie die Seiten (in Reihenfolge des Auftretens) an, welche einen Pagefault auslösen.

Zugriffsreihenfolge: 0, 8, 4, 2, 2, 5, 0, 2, 1 Pagefaults (in Reihenfolge des Auftretens):

- 0, 8, 4, 2, 5, 0, 1

Anfrage	f_0, t	f_1, t	f_2, t	f_3, t	Pagefault?
0	**0,1**				**Ja**
8	0,1	**8,2**			**Ja**
4	0,1	8,2	**4,3**		**Ja**
2	0,1	8,2	5,3	**2,4**	**Ja**
2	0,1	8,2	5,3	**2,5**	Nein
5	**5,8**	8,2	5,3	2,5	**Ja**
0	5,8	**0,8**	5,3	2,6	**Ja**
2	5,8	0,8	5,3	**2,8**	Nein
1	5,8	0,8	**1,9**	2,8	**Ja**

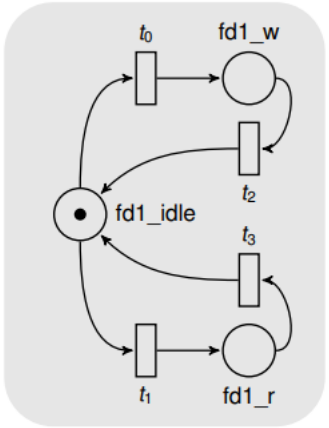
Aufgabe 6 Klausur 2020

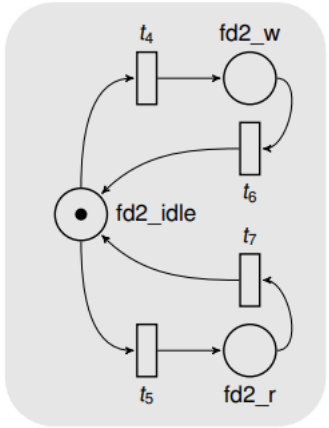
Gegeben ist folgendes natürlichzahliges Petrinetz. Das Petrinetz zeigt drei Datei Deskriptoren (fd1, fd2 und fd3), die gleichzeitig versuchen auf eine Datei lesend und schreibend zuzugreifen.
Dabei ist die Benennung der Stellen wie folgt:

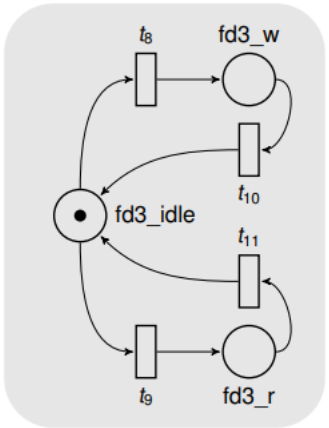
- fdi_idle: Der i-te Datei Descriptor wird gerade nicht verwendet.
- fdi_w: Der i-te Datei Descriptor greift gerade auf die Datei schreibend zu (write).
- fdi_r: Der i-te Datei Descriptor greift gerade auf die Datei lesend zu (read).

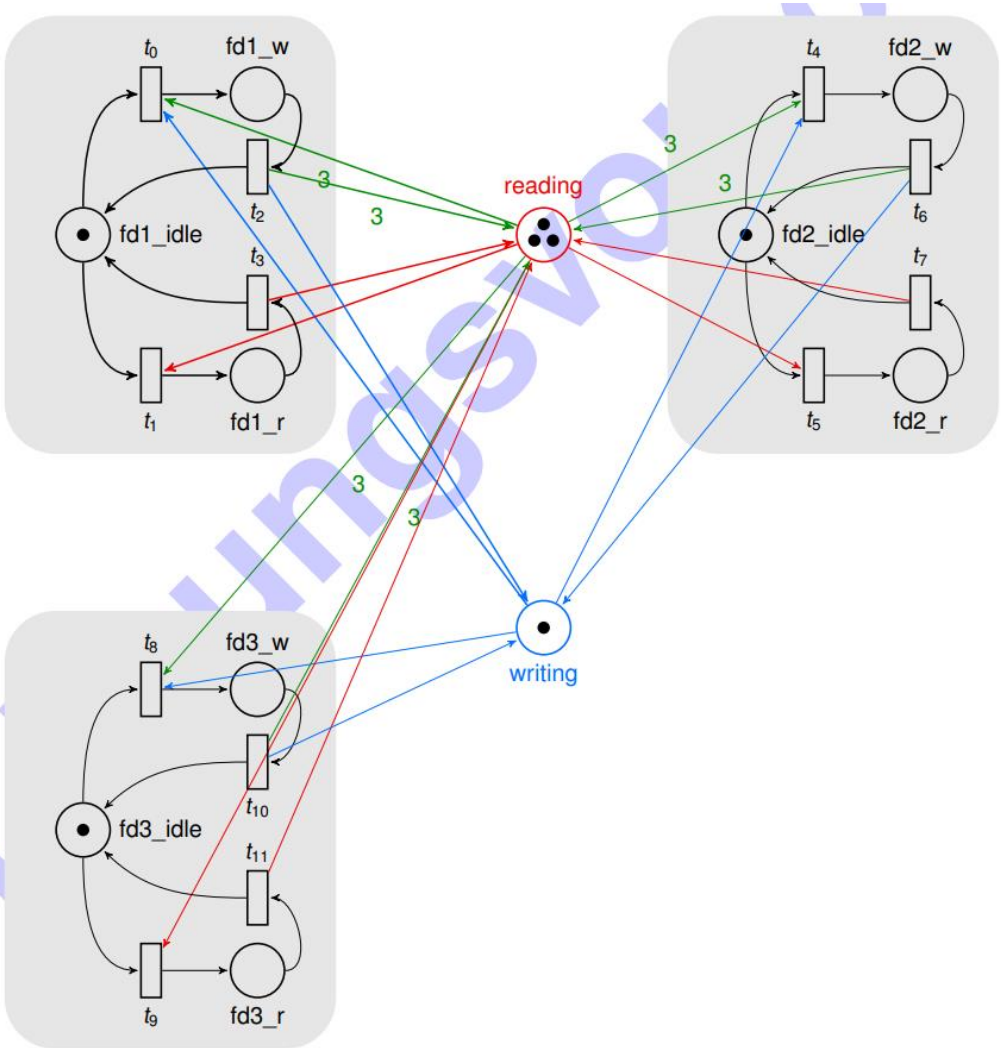
Erweitern Sie das gegebene Petri-Netz so, dass es alle der folgenden Anforderungen erfüllt. Alle hinzugefügten Erweiterungen müssen benannt werden und ohne das Verwenden weiterer Kapazitätsangaben an den Stellen umgesetzt werden.

a) Es dürfen maximal drei der unten gegebenen Datei Deskriptoren gleichzeitig lesend auf die Datei zugreifen.
b) Es darf maximal einer der unten gegebenen Datei Deskriptoren gleichzeitig schreibend auf die Datei zugreifen.
c) Erweitern Sie ihre Lösung so, dass immer entweder geschrieben oder gelesen werden kann. Beides zusammen soll nicht erlaubt sein. Es muss dabei kein faires Netz erzeugt werden.


t_0
fd1_w
t_2
fd1_idle
t_3
t_1
fd1_r

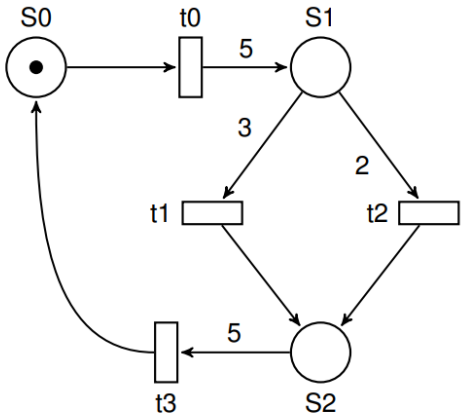

t_4
fd2_w
t_6
fd2_idle
t_7
t_5
fd2_r


t_8
fd3_w
t_{10}
fd3_idle
t_{11}
t_9
fd3_r


t_0
fd1_w
t_2
t_3
fd1_idle
t_1
fd1_r
3
3
reading
3
3
t_4
fd2_w
t_6
t_7
fd2_idle
t_5
fd2_r
3
3
writing
t_8
fd3_w
t_{10}
t_{11}
fd3_idle
t_9
fd3_r

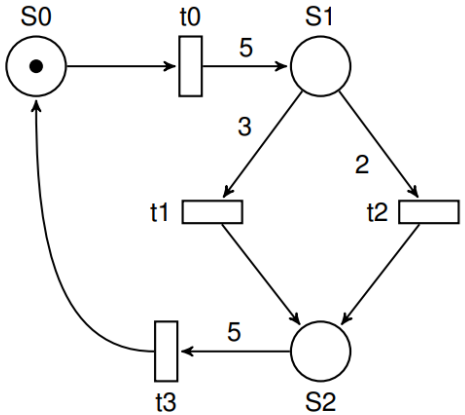
Aufgabe 6 Klausur 2020

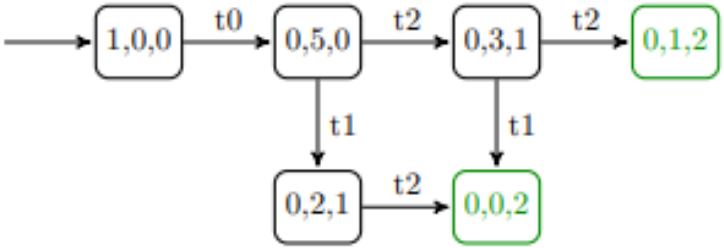
d)* Geben Sie den Erreichbarkeitsgraphen zu folgendem Petrinetz an.
e) Ist das Petri-Netz aus d) verklemmungsfrei? Begründen Sie.



Aufgabe 6 Klausur 2020

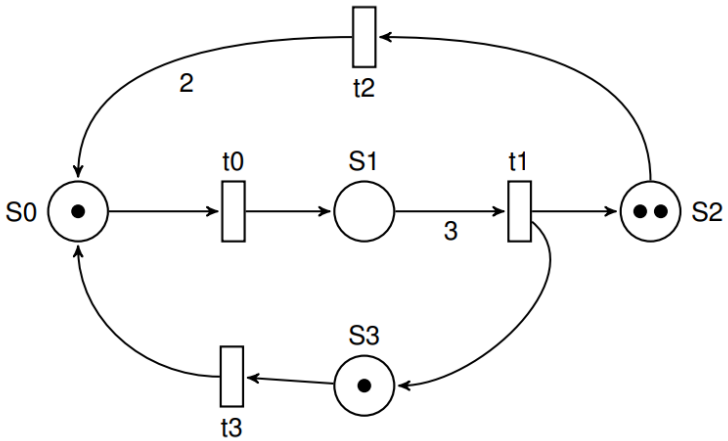
d)* Geben Sie den Erreichbarkeitsgraphen zu folgendem Petrinetz an.
e) Ist das Petri-Netz aus d) verklemmungsfrei? Begründen Sie.
- Nein, siehe Erreichbarkeitsgraph





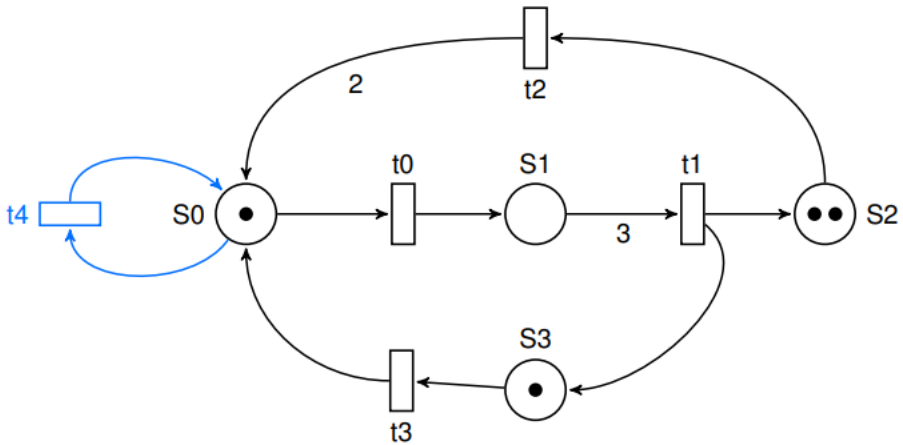
Aufgabe 6 Klausur 2020

f)* Erweitern Sie das folgende Petrinetz so, dass es unfair ist. Falls es schon unfair ist, umkreisen Sie den dafür verantwortlichen Teil des Netzes.



Aufgabe 6 Klausur 2020

f)* Erweitern Sie das folgende Petrinetz so, dass es unfair ist. Falls es schon unfair ist, umkreisen Sie den dafür verantwortlichen Teil des Netzes.



Aufgabe 7 Klausur 2020

Ein Finanzdienstleister ermöglicht seinen Kunden, an einer Wertpapierbörse gehandelte Unternehmensanteile zu kaufen und zu verkaufen. Für jeden Vorgang (Kauf oder Verkauf) wird hierfür beim Finanzdienstleister ein Prozess erstellt, dessen Ausführung sich in vier Phasen einteilen lässt:

1. Abfrage des Kontostands des Kunden in der Datenbank des Finanzdienstleisters
2. Abfrage des Börsenkurses über die Webschnittstelle der Wertpapierbörse
3. Verrechnung des (Ver-)Kaufpreises mit dem Kontostand und Berechnung einer kryptographischen Prüfsumme für den Vorgang.
4. Eintragung des Vorgangs (inkl. Prüfsumme) und neuen Kontostands in die Datenbank des Finanzdienstleisters

Um die Systeme des Finanzdienstleisters möglichst effizient auszulasten, sollen die Prozesse die CPU freiwillig abgeben, wenn diese gerade nicht akut benötigt wird.

Aufgabe 7 Klausur 2020

a) Welche Phase eignet sich besonders gut für diese freiwillige Abgabe? Begründen Sie Ihre Antwort kurz.
b) Der Scheduler in diesem System verwaltet eine Liste der neu angekommenen Prozesse, um diesen bei der nächsten regulären Schedulingentscheidung die CPU für ihre erste Phase zuzuteilen. Die Ankunft eines neuen Prozesses führt nicht zur Unterbrechung des aktuell rechnenden Prozesses. Schildern Sie, wie sich hieraus ein Problem ergeben kann
c) Gegeben sei eine Implementierung des Round Robin Schedulingverfahrens, die eine Liste mit Pointern auf PCBs verwaltet. Das Zeitquantum ist hierbei hardcoded, kann also nicht verändert werden. Wie kann rechenlastigen Prozessen dennoch mehr Rechenzeit eingeräumt werden?
(d) Nennen Sie zwei unterbrechende Scheduling-Strategien.)
e) Beschreiben Sie stichpunktartig den Ablauf eines Kontextwechsels in vier Schritten. Welche Komponente des Betriebssystems übernimmt diese Aufgabe?
f) Threads werden des Öfteren als leichtgewichtige Prozesse beschrieben. Nennen Sie zwei Aspekte, die diese Bezeichnung rechtfertigen.

Aufgabe 7 Klausur 2020

a) Welche Phase eignet sich besonders gut für diese freiwillige Abgabe? Begründen Sie Ihre Antwort kurz.

- Phase 1 - Abfrage des Kontostands: Datenbankzugriff ist IO
- Phase 2 - Abfrage der Webschnittstelle: Netzwerkzugriff dauert i.A. am längsten.
- Phase 3 - Nicht sinnvoll: Berechnung der Prüfsumme braucht CPU.
- Phase 4 - Bei wirklich guter Begründung:. Da der Prozess danach eh terminiert, ist diese Phase sonst nicht besonders gut geeignet für freiwillige Abgabe.

Aufgabe 7 Klausur 2020

b) Der Scheduler in diesem System verwaltet eine Liste der neu angekommenen Prozesse, um diesen bei der nächsten regulären Schedulingentscheidung die CPU für ihre erste Phase zuzuteilen. Die Ankunft eines neuen Prozesses führt nicht zur Unterbrechung des aktuell rechnenden Prozesses. Schildern Sie, wie sich hieraus ein Problem ergeben kann

- Wenn sehr viele Kunden innerhalb kurzer Zeit eine Aktion starten, müssen die alten Prozesse lange warten bis sie wieder dran sind oder verhungern gar. Wenn Prozesse sehr lange nicht drankommen, kann sich der Börsenkurs zwischenzeitlich stark geändert haben

Aufgabe 7 Klausur 2020

c) Gegeben sei eine Implementierung des Round Robin Schedulingverfahrens, die eine Liste mit Pointern auf PCBs verwaltet. Das Zeitquantum ist hierbei hardcoded, kann also nicht verändert werden. Wie kann rechenlastigen Prozessen dennoch mehr Rechenzeit eingeräumt werden?

- Pointer mehrfach in Queue einfügen

Aufgabe 7 Klausur 2020

e) Beschreiben Sie stichpunktartig den Ablauf eines Kontextwechsels in vier Schritten. Welche Komponente des Betriebssystems übernimmt diese Aufgabe?

- Die Aufgabe wird vom Dispatcher übernommen

1. Ändert den Zustand des rechnenden Prozesses zu wartend oder rechenbereit
2. Sichert den Kontext des zuvor rechnenden Prozesses/Threads im PCB
3. Lädt den Kontext des rechenbereiten Prozesses/Threads
4. Ändert den Zustand des rechenbereiten Prozesses zu rechnend.

Aufgabe 7 Klausur 2020

f) Threads werden des Öfteren als leichtgewichtige Prozesse beschrieben. Nennen Sie zwei Aspekte, die diese Bezeichnung rechtfertigen.

- Prozesse: Eigener Adressraum, eigener Heap, eigenes Codesegment, bei Prozess(kontext)wechsel teurer Dispatch nötig (austausch des PCB)
- Threads: Teilen sich Adressraum, Heap, Codesegment des Prozesses, bei Thread(kontext)wechsel bleibt PCB gleich

Aufgabe 7 Klausur 2020

g)* Gegeben sei folgender, unvollständiger Schedule der Prozesse P1 bis P4, die folgende Eigenschaften haben:

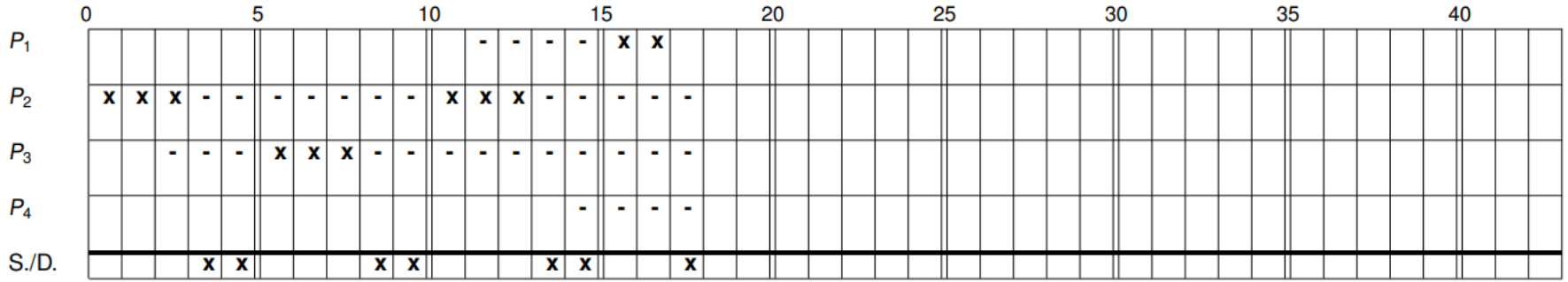
Prozess	Ankunftszeit	Benötigte Rechenzeit
P_1	11	2
P_2	0	14
P_3	2	4
P_4	14	5

Verwendet werde ein Round Robin Verfahren, dessen Parameter Sie dem Schedule entnehmen können. Wird der Scheduler aktiv, so benötigt er eine Zeiteinheit. Ein Kontextwechsel benötige eine weitere Zeiteinheit (beides in Zeile S./D.). Vervollständigen Sie den Schedule, indem Sie Wartezeiten mit einem - und Rechenzeiten mit einem x markieren. Nutzen Sie den zweiten Vordruck, falls Sie Fehler im ersten Vordruck korrigieren möchten. Streichen Sie deutlich durch, was nicht gewertet werden soll (sonst keine Wertung möglich)

Aufgabe 7 Klausur 2020

g)* Gegeben sei folgender, unvollständiger Schedule der Prozesse P1 bis P4, die folgende Eigenschaften haben:

Prozess	Ankunftszeit	Benötigte Rechenzeit
P_1	11	2
P_2	0	14
P_3	2	4
P_4	14	5



Aufgabe 7 Klausur 2020

g)* Gegeben sei folgender, unvollständiger Schedule der Prozesse P1 bis P4, die folgende Eigenschaften haben:

Prozess	Ankunftszeit	Benötigte Rechenzeit
P_1	11	2
P_2	0	14
P_3	2	4
P_4	14	5