Probelösung II: Betriebssysteme 2025

Bearbeitungszeit 50 Minuten. Ohne Hilfsmittel.

Von 52 möglichen Punkten werden maximal 50 gewertet.

1. Aufgabe: Richtig oder Falsch — 8 Pkt

Sind die folgenden Aussagen richtig oder falsch?

In einer ausführbaren Datei steht eine Einsprungadresse.
☒ Richtig ☐ Falsch

Dynamische Bibliotheken müssen in jedem Adressraum an der gleichen Adresse liegen.
☐ Richtig ☒ Falsch

Instanzen können abwechselnd Bibliotheken oder Systemfunktionen nutzen, aber nicht beides gleichzeitig.
☐ Richtig *Systemfunktionen werden über Bibliotheken aufgerufen.* ☒ Falsch

Der Programmlader ist für das Erzeugen von Instanzen zuständig.
☒ Richtig ☐ Falsch

Der Prozesszustand *Aktiv* aus der Vorlesung entspricht *Running* im 5 State Process Model (5SPM).
☐ Richtig *Aktiv gibt es im 5SPM nicht.* ☒ Falsch
Die Recherche zum 5SPM ist klausurrelevant.

Wird die Bereitmenge mit einer *Multilevel Fallback Queue* (MLFQ) verwaltet, braucht man keine Leerlaufprozesse.
☐ Richtig ☒ Falsch
Die Recherche zur Multilevel Feedback *Queue ist klausurrelevant.*

Sowohl POSIX als auch NTFS (Windows) unterscheiden zwischen symbolischen Links auf Dateien und auf Verzeichnisse.
☐ Richtig *Die Recherche zu Links ist klausurrelevant.* ☒ Falsch

Ein Java–Thread darf `Object.wait` nicht in einem `synchronized`–Block aufrufen.
☐ Richtig ☒ Falsch
Die Recherche zur Synchronisation in Java ist klausurrelevant.

2. Aufgabe: Infrastruktur und Hauptspeicher — 12 Pkt

Diese Teilaufgaben repräsentieren Wissensabfragen allgemein.

a) Wieviele Schichten unterscheidet das einfache Wettstein'sche Schichtenmodell? — 1 Pkt

Drei Instanzenschichten oberhalb des dicken, fetten Strichs. Die Infrastruktur unterhalb bildet die vierte Schicht.

b) Wie hängen die Betriebsmodi eines Prozessors mit den Wettstein'schen Schichten zusammen? — 2 Pkt

Instanzen führt der Prozessor im Anwendungsmodus aus, die Infrastruktur im Systemmodus.

c) Wie hängen die Betriebsmodi eines Prozessors mit Rechen- und Ablaufumgebung zusammen? — 1 Pkt

Auf die Register der Rechenumgebung können Befehlsströme in beiden Betriebsmodi uneingeschränkt zugreifen, auf die Ablaufumgebung nur im Systemmodus.

d) Warum können Instanzen nicht auf interne Attribute von Kernobjekten zugreifen? — 3 Pkt

Kernobjekte liegen auf der Halde des Kerns.[0.5] Interne Attribute bietet die Kernschnittstelle nicht an.[0.5] MMUs verhindern direkte Zugriffe von Instanzen auf Speicher des Kerns.[0.5] Übersetzungstabellen für MMUs liegen im Speicher des Kerns[0.5] und sind in der Ablaufumgebung der Instanzen konfiguriert.[0.5] Weil Instanzen im Anwendungsmodus des Prozessors arbeiten, können sie ihre Übersetzungstabelle nicht ersetzen.[0.5] Der Prozessor wechselt nur durch Unterbrechungen[0.5] in den Systemmodus, zum Beispiel bei Kernaufrufen. Dann führt er aber Code des Kerns aus,[0.5] nicht Code einer Instanz.

Höchstens 3 Punkte erreichbar. Auch andere gute Erklärungen können zählen.

e) Welche Arten von Kernobjekten dienen zur Verwaltung von Hauptspeicher? — 1 Pkt

Adressräume, Einblendungen, Segmente

f) *Übersetzungstabelle, für Aufgabenstellung siehe Klausur*

Logisch	Physisch	Flags
0x02	0xa0	C L
0x03	0xa1	C L
0x04	0xa2	D L
0x06	0xa3	D S
0xfc	0xa4	D S
0xfd	0xa5	D S
0x10	0xa6	D S
0x11	0xa7	D S

0.5 Punkte pro Zeile

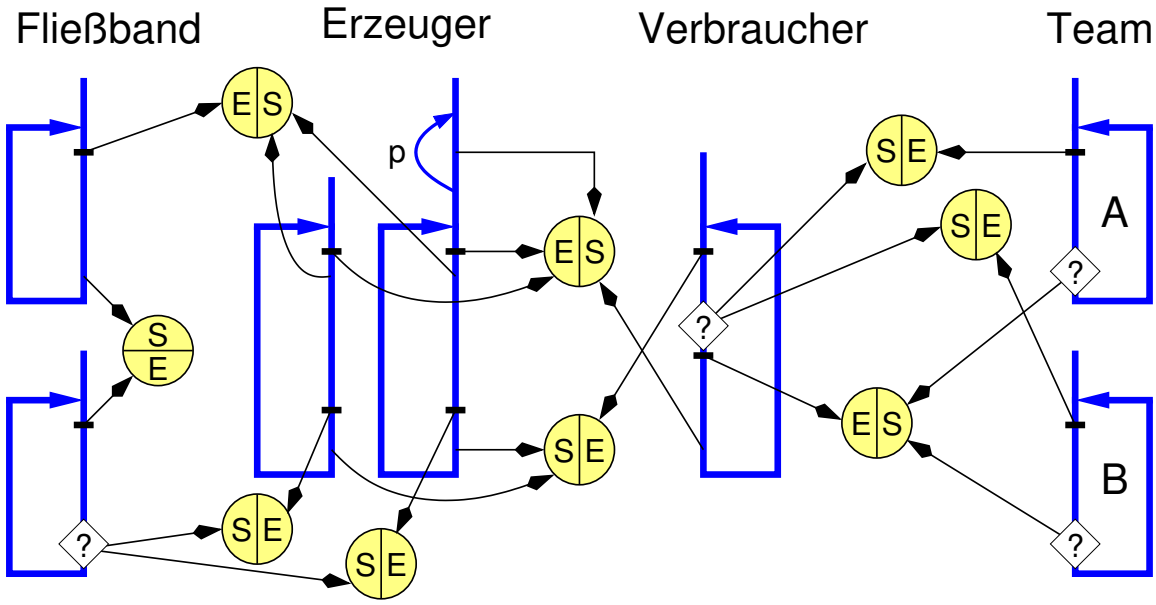
-1 für Adressen statt Seitennummern

3. Aufgabe: Interaktionsdiagramm

8 Pkt

Erstellen Sie ein Interaktionsdiagramm nach der folgenden Beschreibung:

In einem Erzeuger–Verbraucher–System kreisen p Nachrichten. Die Initialisierung übernimmt der Erzeuger. Er ist mit insgesamt zwei Prozessen reproduziert. Während des Erzeugens schicken sie einen asynchronen Auftrag an ein zweistufiges, offenes Fließband (*links*) und empfangen die Antwort. Der Verbraucher schickt beim Verbrauchen einen Auftrag an ein Team (*rechts*) und empfängt die Antwort sofort. Das Team bearbeitet ohne Verteiler zwei Arten von Aufträgen.

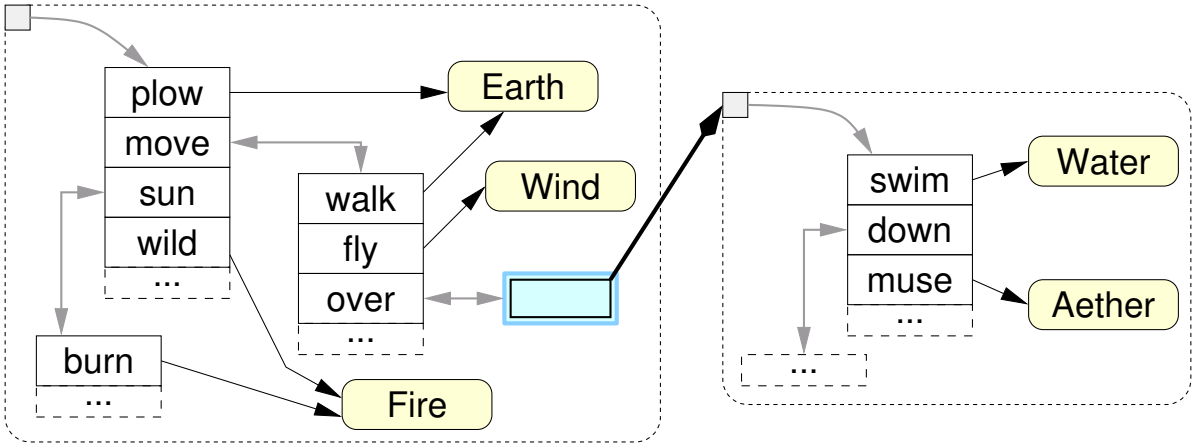


Punkte: E/V 2, Init 1, Repro 1, asyn. Auftrag 1, Fließband 1, Team 2

4. Aufgabe: Dateien und Verzeichnisse

8 Pkt

Die folgende Abbildung zeigt zwei Dateisysteme mit insgesamt fünf Dateien. Das zweite ist beim ersten mit dem Pfad `/move/over` eingehängt. Die Dateiinhalte bestehen jeweils aus einem Wort: Earth, Wind, Fire, Water, Aether. Geben Sie Pfade ohne Umwege an, also ohne überflüssige `../` oder `./`



a) Welcher absolute Pfad führt zur Datei, in der „Wind“ steht? 1 Pkt

`/move/fly`

b) Welche absoluten Pfade führen zur Datei, in der „Earth“ steht? 1 Pkt

`/plow` *und* `/move/walk`

c) Welche relativen Pfade führen aus dem Verzeichnis `move` zu „Fire“? 1 Pkt

`../wild` *und* `../sun/burn`

d) Welcher relative Pfad führt aus `sun` zu „Water“? 1 Pkt

`../move/over/swim`

e) Erklären Sie die Unterschiede zwischen Hard Links, die mit einem relativen bzw. absoluten Pfad angelegt wurden. 2 Pkt

Es gibt keinen Unterschied. Hard Links zeigen mittels einer internen Kennung auf eine Datei. Der Pfad zum Ziel wird schon beim Anlegen des Hard Links aufgelöst und spielt danach keine Rolle mehr.

f) Wie kann man auf Dateien in einem Verzeichnis zugreifen, das gerade durch einen Mount Point verdeckt ist? *Zwei Möglichkeiten.* 2 Pkt

- Den Mount Point entfernen. Dann sind die Dateien wieder sichtbar.
- Über Hard Links, die vor dem Anlegen des Mount Points in nicht verdeckten Verzeichnissen erzeugt wurden.

5. Aufgabe: Interaktion — Türsteher

16 Pkt

Ein Kernobjekt vom Typ „Türsteher“ begrenzt die Anzahl der Prozesse, die sich gleichzeitig in einem lastkritischen Bereich befinden. Der Türsteher unterscheidet „erwachsene“ und „minderjährige“ Prozesse. Erwachsene Prozesse betreten und verlassen den Bereich explizit, ihre Anzahl ist beschränkt. Minderjährige Prozesse zählen nicht mit, dürfen den Bereich aber auch nicht alleine betreten. Sie kommen nur in Begleitung eines erwachsenen Prozesses hinein. Der Kernobjekttyp bietet dazu folgende Operationen:

`enterSyn`: ein erwachsener Prozess betritt den gesperrten Bereich

`tailgateSyn`: ein minderjähriger Prozess wartet auf Einlass eines erwachsenen
klappt nur, wenn schon ein erwachsener ansteht, also wartet

`leaveAsyn`: ein erwachsener Prozess verlässt den gesperrten Bereich

`setCapacity`: legt die erlaubte Anzahl erwachsener Prozesse im Bereich fest

Die Implementierung eines Türstehers verwendet folgende Attribute:

Z: aktuelle Anzahl erwachsener Prozesse im gesperrten Bereich, initial 0

K: erlaubte Anzahl erwachsener Prozesse im gesperrten Bereich, initial 1

WE: Wartemenge für erwachsene Prozesse, initial leer

WM: Wartemenge für minderjährige Prozesse, initial leer

Lösungen auf der nächsten Seite

a) Beschreiben Sie den Ablauf von `enterSyn`. Minderjährige Prozesse spielen hier noch keine Rolle. 3 Pkt

- $Z < K$, d.h. Aufrufer darf eintreten? +1

Ja: Z um 1 erhöhen +1

Nein: Aufrufer blockieren und in WE legen +1

Falls der aufrufende Prozess sofort eintritt, ist WE leer. Gemäß Aufgabenstellung kann dann kein minderjähriger Prozess in WM liegen.

b) Beschreiben Sie den Ablauf von `tailgateSyn`. Falls gerade kein erwachsener Prozess wartet, scheitert der Aufruf mit einem Fehler. 3 Pkt

- WE leer? +1

Ja: Abbruch mit Fehler +1

Nein: Aufrufer blockieren und in WM legen +1

Könnten minderjährige Prozesse jederzeit blockieren, müsste `enterSyn` sie mit einlassen. So aber ist WM leer wenn WE leer ist, also auch bei $Z < K$.

c) Beschreiben Sie den Ablauf von `leaveAsyn`. Bedenken Sie, dass sich K seit dem Aufruf von `enterSyn` geändert haben könnte. Erhält ein erwachsener Prozess Eintritt, schleichen alle wartenden minderjährigen Prozesse mit hinein. 5 Pkt

- $Z \leq K$ (*bzw.* $Z = K$) und WE nicht leer? +2

Ja:
- einen Prozess aus WE nehmen und deblockieren +1
- alle Prozesse aus WM nehmen und deblockieren +1

Nein: Z um 1 vermindern +1

Verlässt ein erwachsener Prozess den gesperrten Bereich, kann höchstens ein anderer eintreten. Bei $Z < K$, also wenn mehr Prozesse eintreten dürften, ist WE leer. Das gilt auch bei Änderungen der Schranke K, sofern `setCapacity` richtig funktioniert.

d) Beschreiben Sie den Ablauf von `setCapacity`. Der neue Wert gilt sofort, auch wenn Z dadurch vorübergehend größer als K sein sollte. 5 Pkt

- K auf neuen Wert setzen +0.5
- solange $Z < K$ und WE nicht leer: +2
  - einen Prozess aus WE nehmen und deblockieren +1
  - alle Prozesse aus WM nehmen und deblockieren +1
  - Z um 1 erhöhen +0.5