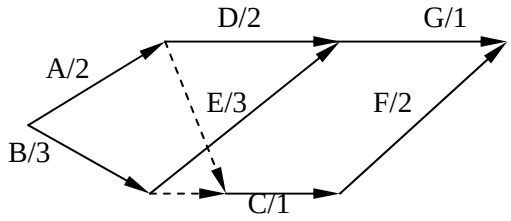
Lösungen zu ausgewählten Aufgaben des Gebiets Betriebssysteme

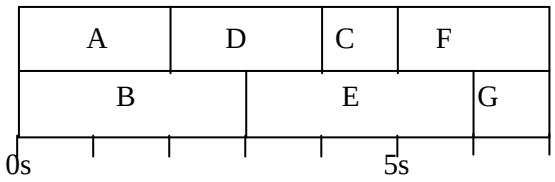
H7.a) FIFO: Rahmen 2 (früheste Ladezeit, 120)
LRU: Rahmen 3 (entfernteste Referenzzeit, 207)
SC: Rahmen-Ordnung nach FIFO: 2 0 3 1
zugehörige R-Bits: 1 0 0 1.
Damit wird R-Bit von Rahmen 2 gelöscht, Rahmen nach hinten gehängt und so die in Rahmen 0 befindliche Seite verdrängt.

b) Rahmen 3: Seite modifiziert, ohne referenziert zu sein. Mögliche Erklärung: Seite wurde zur Zeit 207 referenziert und modifiziert; zur Zeit 230 wurde die in Rahmen 1 befindliche Seite S eingelagert, der dafür erforderliche Rahmen wurde nach SC bestimmt (Speicher war voll), dabei wurde R-Bit von Rahmen 3 gelöscht (dazu muß S später als zur Zeit 160 eingelagert und die vorher in Rahmen 1 befindliche Seite längere Zeit nicht benutzt worden sein!).

Q2. ***Vorgangspfeilnetz:***



Ablaufplan:



Durchsatz D ist offenbar maximal, da minimale Gesamtbearbeitungszeit (keine Stillstandszeiten, maximale Prozessorauslastung von 100%). Weiter gilt:

$$\overline{t_v} = \overline{t_w} + \overline{t_b} = \frac{2+4+5+3+6}{7}s + \frac{14}{7}s = \frac{34}{7}s; \qquad D = \frac{7\ \text{Prozesse}}{7s} = 1s^{-1}.$$

Einzig (sinnvoll) möglicher anderer Ablauf (C statt E bei t = 3) führt zwar zu derselben mittleren Verweilzeit, aber zu einer größeren Gesamtbearbeitungszeit (nachrechnen!) → beide Optimierungsziele gleichzeitig erreicht (ist in der Regel nicht möglich!).

Modellierung durch ein M/M/1/∞-System:

Aufgabenstellung entnehmbar: $\mu = 1/\overline{T_b} = 0{,}5s^{-1}$; 1Bedienungskanal; Warteraum unbegrenzt (da keine anderslautende Einschränkung).

Gegenargumente: über Ankunftsprozeß ist nichts bekannt (nicht einmal Mittelwerte)!!!;
es ist nicht erkennbar (sogar fraglich), ob Bedienungszeit exponentiell verteilt ist;
Abhängigkeiten können in M/M/1/∞-System nicht berücksichtigt werden.

S9. ***Einordnung:*** Sicherheit (in Betriebssystemen).

Erklärung: Access Control List, Zugriffssteuerliste. Liste, die jedem Objekt (z.B. Datei) zugeordnet ist und (nur) diejenigen Subjekte (z.B. Prozesse) mit ihren Zugriffsrechten enthält, die auf das jeweilige Objekt zugreifen dürfen.

Bedeutung: ist eine mögliche Realisierung einer Schutzmatrix (eine andere Möglichkeit ist die Verwendung von Capabilities, Zugriffsberechtigungen), vergleichbar einer Gästeliste.

Vorteil: einfach implementierbar; relativ sicher, da bei den Objekten gespeichert; Rechteänderung einfach (insbes. Zurückziehen von Rechten). Nachteil: uneffizient/unbrauchbar in verteilten Systemen.

Vorkommen: vereinfacht in Unix, nur 3 „Subjekte“: Nutzer, Gruppe, Rest.

X2. Vorteil: höhere Sicherheit (gegenüber Eindringlingen)

Nachteil: Aufwand! vor allem Zeit für Plattenzugriffe; Wiederherstellung bei unbeabsichtigtem Löschen unmöglich.

T9. ***Bezeichnungen:*** Puffer: L: *linkeHälfte* R: *rechteHälfte*
Zustände: 00 (L und R leer), 01 (L leer, R voll), 10, 11 (entsprechend)
codiert als 0 1 2 3

Zustandsübergänge: Prozeß A: 0 fülle L → 2 Prozeß B: 1 leere R → 0
1 fülle L → 3 2 leere L → 0
2 fülle R → 3 3 leere L → 1

Vereinbarungen:

int zustand = 0	Zustand
sem_t *leer = new_sem(2)	Semaphore zur Lösung des in der Aufgabe enthaltenen
sem_t *voll = new_sem(0)	Erzeuger-/Verbraucher-Problems
sem_t *mutex = new_sem(1)	binärer Semaphor zum Schutz der Variablen zustand

Prozeß A:

```
{while (1)  {
  P(leer);
  if (zustand == 2)  {
          fülleR();
          P(mutex);
          zustand = zustand+1;
          V(mutex);
  }
  else    {
          fülleL();
          P(mutex);
          zustand = zustand+2;
          V(mutex);
  }
  V(voll);
 }
}
```

Prozeß B:

```
{while (1)  {
  P(voll);
  if (zustand == 1)  {
          leereR();
          P(mutex);
          zustand = zustand-1;
          V(mutex);
  }
  else    {
          leereL();
          P(mutex);
          zustand = zustand-2;
          V(mutex);
  }
  V(leer);
  }
}
```

Korrektheit durch ein (repräsentatives!) Ablaufbeispiel demonstrieren

Gültigkeitsdauer: A fülle unmittelbar nacheinander beide Pufferhälften. Danach wechseln sich B und A ständig ab, so daß nur noch L geleert und wieder gefüllt wird. Daten in R können damit ihr Verfallsdatum überschreiten.

V5. ***Sicherer Zustand:*** kein Verklemmungszustand, und es gibt eine Folge von Zustandsübergängen des Prozeßsystems, die zur Beendigung aller Prozesse führt, ohne daß eine Verklemmung auftritt.

Beispiel unsicherer Zustand:
2 Prozesse A, B;
4 Betriebsmittel-Exemplare.

Zustand:

Prozeß	maximal gefordert	belegt
A	3	1
B	4	2
frei		1

Beide Prozesse brauchen noch 2 BM-Exemplare, es ist jedoch nur noch 1 Exemplar frei. In dem Moment, in dem einer der beiden Prozesse die ausstehenden Exemplare fordert, liegt ein Verklemmungszustand vor.

Gegenmaßnahme beispielsweise: Bank-Algorithmus. Nachteile: Aufwand (Overhead), vor allem aber: in realen Prozeßsystemen sind die Maximalforderungen i.a. nicht bekannt.