1 Prozesse und Scheduling (10,5 Punkte)

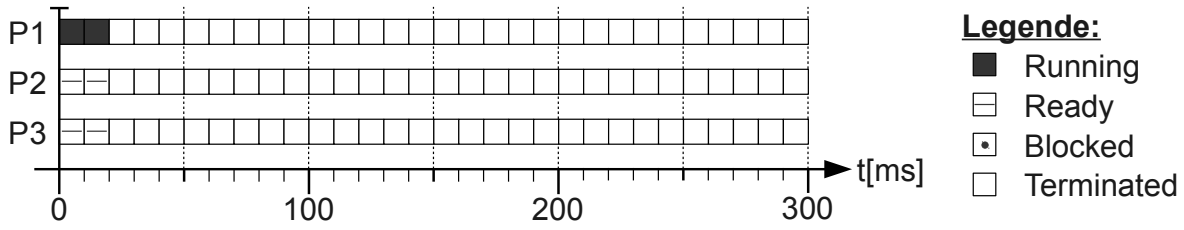
a) Round Robin (6 Punkte) Ein Betriebssystem verwaltet drei Prozesse P1, P2 und P3. Die Prozesse treffen in dieser Reihenfolge im System ein und sind alle zum Zeitpunkt t=0 rechenbereit. Die Bedienzeiten der Prozesse und die Zeitpunkte und Zeitdauer von E/A-Operationen, sind in der folgenden Tabelle angegeben:

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

Bei der „Bedienzeit“ handelt es sich um die reine Rechenzeit. Für die Gesamtzeit kommt noch die Zeitdauer von den E/A-Operationen hinzu. Der E/A-Zeitpunkt ist relativ zur Bedienzeit angegeben. Prozess P1 hat also seine E/A-Operationen nachdem er 20 ms gerechnet hat, Prozess P2 wird blockiert nachdem er seine ersten 30 ms gerechnet hat, etc.

Zeichnen Sie in das folgende Gantt-Diagramm ein, wie die drei Prozesse P1, P2 und P3 abgearbeitet werden würden, wenn das Scheduling nach der *„Round Robin“*-Strategie vorgenommen wird. Die gewählte Zeitscheibe beträgt *40 ms*. Jeder Prozess führt *genau einen* E/A-Vorgang durch. Zeitpunkt und Zeitdauer sind in der Tabelle angegeben. Die Prozessumschaltzeit kann vernachlässigt werden. Markieren Sie in dem folgenden Diagramm die Prozesszustände entsprechend der Legende.

Hinweis: Die ersten 20 ms sind bereits fertig ausgefüllt.



b) Allgemeine Fragen (insgesamt 4,5 Punkte)

1. „>“ **Shell-Operator (1,5 Punkte)** Angenommen der Befehl „echo Parameter“ gibt in die Standardausgabe den Text aus dem übergebenen Parameter aus:
   Was bewirkt der Befehl „`echo "" > Datei`“, falls `Datei` schon einen Inhalt enthält?

   ---

2. **Prozesse vs. Funktionen (1,5 Punkte)** Erklären Sie kurz den Unterschied einer Prozesserzeugung und einem Funktionsaufruf!

   ---

3. **Zombie-Prozesse (1,5 Punkte)** Warum muss das Betriebssystem auf das Abfragen von Zombie-Prozesse warten, anstatt diese direkt aus der Prozesstabelle zu entfernen?

   ---

2 Synchronisation und Verklemmungen (13 Punkte)

a) Erzeuger-/Verbraucher-Problem (7 Punkte) Im folgenden Pseudocode-Abschnitt soll eine Erzeuger-Funktion, producer(), Elemente mittels produce_element() erzeugen und diese mittels enqueue() in eine gemeinsam genutzte Warteschlange einfügen. Eine Verbraucher-Funktion, consumer(), soll vorhandene Elemente mit dequeue() aus der Warteschlange nehmen und diese mit consume_element() verbrauchen. Die beiden Funktionen werden potentiell gleichzeitig ausgeführt. Synchronisieren Sie die beiden Funktionen mittels *einseitiger Synchronisation* und beachten Sie dabei, dass die Warteschlange zwar beliebig viele Elemente aufnehmen kann, die Operationen enqueue() und dequeue() aber selbst kritisch sind und nicht gleichzeitig ausgeführt werden dürfen. Der Verbraucher soll nur dann ein Element aus der Warteschlange nehmen, wenn eines verfügbar ist.

Legen Sie dazu zwei geeignet benannte Semaphore an, initialisieren Sie diese mit einen passenden Wert. Setzen Sie anschließend an den richtigen Stellen im Code die Semaphor-Operationen P() und V() ein (z.B. `P(MeinSemaphor1)`). Die Funktionen P(), V(), produce_element(), consume_element(), enqueue() und dequeue() können als gegeben angesehen werden und müssen *nicht* implementiert werden! Statt P() und V() dürfen Sie Alternativ auch die Operationen wait() und signal() bzw. sem_wait() und sem_post() verwenden. Alle Funktionen dürfen ohne Fehlerbehandlung verwendet werden.

Namen und Initialwerte der Semaphoren: [ ] = [ ]
[ ] = [ ]

```
producer() {
    while (1) {
        Element e = produce_element();



        enqueue(e);









   }
}

consumer() {
    while (1) {









        Element e = dequeue();



        consume_element(e);
    }
}
```

b) Verklemmungen (6 Punkte) Wenn eine geschlossene Kette wechselseitig wartender Prozesse existiert (*circular wait*, also ein Zyklus im Betriebsmittelbelegungsgraphen), liegt eine Verklemmung vor. 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.

3 Speicherverwaltung und Virtueller-Speicher (12 Punkte)

a) Platzierungs- & Ersetzungsstrategie (4 Punkte) Erläutern Sie den Unterschied zwischen der Platzierungsstrategie (*placement policy*) und der Ersetzungsstrategie (*replacement policy*)!

b) 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 Segmenttabelle an. Löst eine Speicheranfrage eine Zugriffsverletzung aus, so machen Sie dies bitte kenntlich.

Segmenttabelle:

	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 $FFFF_{16}$

logische Adresse: *0x1000 A100*$_{16}$
→ physikalische Adresse: ☐

logische Adresse: *0x030B 5000*$_{16}$
→ physikalische Adresse: ☐

c) Buddy-Verfahren (4 Punkte) Dynamische Speicherverwaltung nach dem *Buddy*-Verfahren: Die jeweils zweite Zeile der folgenden Szenarien zeigt die momentane Speicherbelegung des Speichers der Größe 32 MiB. Ergänzen Sie die folgenden Tabellen um Markierungen für die vorgegebenen Anfragen.
Hinweis: Falls eine Belegung/Freigabe *nicht* erfüllt werden kann, kennzeichnen Sie die betreffende Zeile geeignet.

Szenario 1: Prozess C belegt 3 MiB

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

Szenario 2: Prozess D belegt 12 MiB

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

Szenario 3: Prozess E belegt 14 MiB

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

Szenario 4: Prozess F belegt 7 MiB

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

4 Ein-/Ausgabe und Dateisysteme (9,5 Punkte)

a) Block-Buffer-Cache (3 Punkte) Nennen und erläutern Sie drei Ereignisse, die das Rückschreiben des Block-Buffer-Caches auslösen.

b) Kontinuierliche Speicherung (2 Punkte) Nennen Sie je einen Vor- und einen Nachteil der kontinuierlichen Speicherung von Dateien.

c) I/O-Scheduling (4,5 Punkte) Gegeben sei ein Plattenspeicher mit 8 Spuren. Der dazugehörige I/O-Scheduler bekommt immer wieder Leseaufträge für eine bestimmte Spuren. Die Leseaufträge in L_1 sind dem I/O-Scheduler bereits bekannt. Nach drei bearbeiteten Aufträgen erhält er die Aufträge in L_2. Nach weiteren drei (d.h. nach insgesamt 6) bearbeiteten Aufträgen erhält er die Aufträge in L_3. Zu Beginn befinde sich der Schreib-/Lesekopf über Spur 0.
$L_1 = \{1, 4, 7, 2\}, L_2 = \{3, 6, 0\}, L_3 = \{5, 2\}$
Der I/O-Scheduler arbeitet nach der **Fahrstuhlstrategie** (Elevator). Bitte tragen Sie die Reihenfolge der gelesenen Spuren ein.
