1 Prozesse und Scheduling (12 Punkte)

a) UNIX Shell-Operatoren (insgesamt 4 Punkte)

1. **„|“ Operator (1,5 Punkte)** Beschreiben Sie die Funktionsweise des „|“ Operators.

2. **„|“ Operator – Beispiel (1 Punkt)** Geben Sie für die Einsatzmöglichkeiten des „|“ Operators ein Beispiel an.

3. **„<“ Operator (1,5 Punkte)** Beschreiben Sie die Funktionsweise des „<“ Operators.

b) Prozesse (2 Punkte) Erklären Sie knapp in eigenen Worten den Unterschied zwischen den Begriffen *Programm* und *Prozess*.

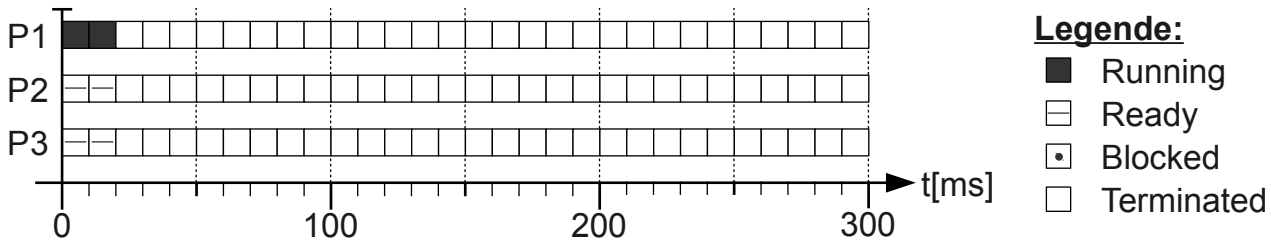
c) Round Robin (6 Punkte) Ein Betriebssystem verwaltet drei Prozesse P1, P2 und P3. Die Prozesse treffen in dieser Reihenfolge im System ein und sind alle zum Zeitpunkt t=0 rechenbereit. Die Bedienzeiten (in ms) der Prozesse und die Zeitpunkte von E/A-Operationen, relativ zur Bedienzeit*, sind in der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

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

*: Bei der „Bedienzeit“ handelt es sich um die reine Rechenzeit. Prozess P1 hat seinen ersten E/A-Zugriff nachdem er 20 Zeiteinheiten gerechnet hat. Prozess P2 wird blockiert nachdem er seine ersten 30 Zeiteinheiten 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 *40ms*. Jeder Prozess führt *genau einen* E/A-Vorgang durch. Zeitpunkt und Länge sind in der Tabelle angegeben. Die Prozessumschaltzeit kann vernachlässigt werden. Markieren Sie in dem folgenden Diagramm die Prozesszustände entsprechend der Legende.

Hinweis: Die ersten beiden Zeiteinheiten sind bereits fertig ausgefüllt.



2 Synchronisation und Verklemmungen (15 Punkte)

a) 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.

b) Synchronisation (3 Punkte) Welche der folgenden Aussagen zu Synchronisation und Verklemmungen treffen zu? Kreuzen Sie bei **J** (ja, zutreffend) oder **N** (nein, nicht zutreffend) an!

J	N	
		Semaphoren basieren auf dem Prinzip des aktiven Wartens.
		Mit Semaphoren kann gegenseitiger Ausschluss realisiert werden.
		Der Bäckerei-Algorithmus ist ein Verfahren zur Vermeidung von Verklemmungen.
		Wenn Betriebsmittel nicht exklusiv belegt werden, können keine Verklemmungen entstehen.
		Prozesse in einem *Deadlock* befinden sich im Prozesszustand *blockiert*.
		Unterbrechungsanforderungen zählen zu den wiederverwendbaren Betriebsmitteln.

c) Synchronisation (3 Punkte) Betrachten Sie das folgende Codefragment, in dem eine Schlossvariable implementiert wird (Datentyp `Lock` mit Operationen `acquire` und `release`). Warum ist dieses *naive* Verfahren zur Synchronisierung von Prozessen (bei präemptivem Scheduling) nicht geeignet? Welche gängige Betriebssystem-Abtraktion sollte man stattdessen verwenden?

```
1  /* Schlossvariable (Initialwert 0) */
2  typedef unsigned char Lock;
3
4  /* Kritischen Abschnitt betreten */
5  void acquire (Lock *lock) {
6      while (*lock);
7      *lock = 1;
8  }
9
10 /* Kritischen Abschnitt wieder verlassen */
11 void release (Lock *lock) {
12     *lock = 0;
13 }
```

d) Verklemmungs-Arten (3 Punkte) Erklären Sie stichpunktartig den Unterschied zwischen einem *Deadlock* und einem *Livelock*. Weshalb sind Deadlocks das „geringere Übel“?

3 Speicherverwaltung und Virtueller-Speicher (8 Punkte)

a) 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										

b) Verschnitt (4 Punkte) Bei den behandelten Speicherverwaltungs-Platzierungsstrategien kann es zu *externem* und *internem Verschnitt* kommen. Erklären Sie die beiden Begriffe stichpunktartig und nennen Sie jeweils eine Platzierungsstrategie (nicht zwei Mal dieselbe!), bei der das Problem auftritt.

4 Ein-/Ausgabe und Dateisysteme (10 Punkte)

a) Dateioperationen (6 Punkte) Beantworten Sie die folgenden Fragen zu dem kurzen C-Programm. #include der Systemheader und Fehlerabfragen wurden der Einfachheit halber weggelassen. Gehen Sie von einer fehlerfreien Abarbeitung aller Systemaufrufe aus.

Die Datei `datei.txt` hat folgenden Inhalt:

```
Live␣for␣today!
```

```
1  int main() {
2      struct stat stats;
3      FILE *filestream;
4      char *block="**##";
5
6      stat("datei.txt", &stats);
7      if(!S_ISREG(stats.st_mode)) return 1;
8
9      filestream = fopen("datei.txt", "r+");
10     fwrite(block, 2, 1, filestream);
11     fseek(filestream, -5, SEEK_END);
12     fwrite(block + 2, 2, 1, filestream);
13     fclose(filestream);
14
15     return 0;
16 }
```

1. stat (2 Punkte) Wozu dienen die Zeilen 6 und 7?

2. fseek (2 Punkte) Beschreiben Sie die genaue Funktion des `fseek`-Aufrufs in Zeile 11.

3. Textdatei (2 Punkte) Geben Sie an, was nach der Ausführung des Programms in der Datei `datei.txt` steht.

b) E/A-Scheduling (4 Punkte) Erläutern Sie die E/A-Scheduling-Verfahren „Elevator“ und „Shortest Seek Time First“ (SSTF). Welches Problem besteht bei SSTF im Gegensatz zu „Elevator“? Erklären Sie beispielhaft wie es zu dem Problem kommen kann.