Hochschule Aalen/Stg. IN Klausur im Fach Betriebssysteme Prüfer: Dr. Werthebach
Datum: 17.02.2023 Zeit: 9.00 Uhr Raum: AH-1.01 (UG) Dauer: 120 min.
Püfungsnummern (Studiengang): 43302 (DS), 50302 (ETI), 57302 (IN), 73206 (DPD)

Vor- und Nachname / Mat.-Nr. _____ / _____

Ergebnis Aufgabe	*1:* ____	*/ 5*	
	2: ____	*/ 3*	
	3: ____	*/ 11*	
	4: ____	*/ 8*	
	5: ____	*/ 14*	
	6: ____	*/ 9*	
	7: ____	*/ 9*	
	8: ____	*/ 10*	
	9: ____	*/ 8*	
	10: ____	*/ 4*	
	11: ____	*/ 12*	
	12: ____	*/ 7*	
	____	*/ 100*	*Summe*

Aufgabe 1: Zeichnen Sie ein Blockschaltbild einer einfachen Von-Neumann-Rechnerarchitektur bestehend aus CPU, Hauptspeicher, Massenspeicher und Bussystemen. Benennen Sie alle Komponenten. Erklären Sie Memory-Mapped Files. ***5 Punkte***

Hinweis: Brauchen Sie für **Antworten mehr Platz**, dann nutzen Sie auch die **Rückseiten**.

Aufgabe 2: Betriebssysteme steuern und verwalten Betriebsmittel. Nennen Sie drei Betriebsmittel.

3 Punkte

Aufgabe 3: Zeichnen Sie das Prozesszustandsdiagramm von UNIX. Welcher Zustand ist besonders auffällig? Erklären Sie Sinn und Zweck dieses auffälligen Zustands.

11 Punkte

Aufgabe 4: In einem Rechenzentrum sind 2 Tapes, 2 Plotter, 2 Drucker und 1 3D-Drucker zur gemeinsamen Nutzung freigegeben: E = (2 2 2 1). Es gebe 5 Benutzerprozesse, die bereits Ressourcen nutzen und auch noch weitere Ressourcen belegen wollen. Führen Sie den aus der Vorlesung bekannten Algorithmus zur Verklemmungserkennung durch.

Die Belegungsmatrix B und die Wünschematrix C sind wie folgt:

$$B = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

8 Punkte

Aufgabe 5: Wir hatten uns u.a. die Verfahren „First Fit“ und „Worst Fit“ für die Zuordnung von Hauptspeicher an Prozesse angesehen. **Erklären** Sie diese Verfahren. Zeichnen Sie die Ergebnisse beider Verfahren direkt in die u.a. Grafiken (**jeweils ab der 2ten Zeile**) für
Start A: 18; **Start** B: 4; **Start** C: 4; **Ende** B; **Start** D: 5; **Start** E: 12 ein.
Hinweis: Die z.Z. bereits belegten Speicherblöcke haben alle die Größe 1 (graue Felder).

14 Punkte

Erklärung First Fit:

Ereignis	10	4	20	18	7	9	12	5	Ggf. Bem.
Start A 18									

Erklärung Worst Fit:

10	4	20	18	7	9	12	5

Aufgabe 6: Führen Sie die Berechnungen für das Buddy-System mit folgenden Ereignissen durch:

Start A: 212; **Start** B: 99; **Start** C: 62; **Ende** A; **Start** D: 128; **Start** E: 120;
Ende B; **Ende** E; **Start** F: 256.

Hinweis: es stehen insgesamt 512 Blöcke zur Verfügung.

9 Punkte

Ereignis	512	Ggf. Bem.
Start A 212		

Aufgabe 7: (a) Betrachten wir 2stufiges paging für 32 Bit Speicheradresssen mit einem Offset von 12 Bit und jeweils 10 Bit für Seitenadressen pro Stufe. Zeichnen Sie alle benötigten Seitentabellen und Seiten inklusive der Verweise für einen kleinen Prozess, der 5.000 zusammenhängende Wörter belegt. Nutzen Sie ggfs. sinnvolle Abkürzungen wie "...".
(b) Wieviel Speicherplatz wird für die Tabellen benötigt und wieviel Speicherplatz für den Prozess auf seinen Seiten?
(c) Was ist der Hauptvorteil einer zweistufigen Seitentabelle gegenüber einer einstufigen Seitentabelle bei 32 Bit Wortbreite? ***9 Punkte***

Aufgabe 8: Zeichnen Sie die Page Faults (**Quadrate, Kreise**) sowie die Seitenzahlen der Referenzfolge 0, 1, 40, 31, 30, 1, 0, 1, 0, 400, 30, 3, 31, 1, 30, ... in die Vorlagen ein.

(a) für die **Optimale Strategie**

RAM														
RAM														
RAM														
Disk														
Disk														
Disk														
Disk														

(b) für die **FiFo Second Chance Strategie**

RAM														
RAM														
RAM														
Disk														
Disk														
Disk														
Disk														

10 Punkte

Aufgabe 9: Schreiben Sie ein Shellskripts zum Raten von 6 Lottozahlen zwischen 1 und 49 (ohne Zusatzzahl). Bei mehr als 2 richtigen Zahlen soll dem Anwender gratuliert werden. Zusätzlich werden die Lottozahlen und die getippten Zahlen ausgegeben.

8 ***Punkte***

Aufgabe 10: Beim Anlegen einer Linux-Partition kann man über die i-node Dichte festlegen wie viele i-node Einträge zur Verfügung stehen.
(a) Welche Auswirkung hat eine kleine bzw. große i-node Dichte?
(b) Welches sehr wichtige Attribut wird nicht im Bereich für die Metainformationen in einer i-node gespeichert?
(c) Am Ende jeder i-node gibt es Einträge zu den Clustern einer Datei. Wie klein kann die kleinste Datei bzw. die größte Datei sein, wenn die Clustergröße 1 kByte groß ist?

4 Punkte

Aufgabe 11: Beim Programmieren mit Semaphoren hatten wir uns den Quellcode zum Initialisieren und zum Zugriff auf Semaphore wie folgt angesehen: **12 Punkte**

```
01 int initsem (key_t semkey) { // initsem.c
02  int status = 0, semid;

03  if (( semid = semget (semkey, 1, SEMPERM|IPC_CREAT|IPC_EXCL)) == -1 ) {
04    if ( errno == EEXIST ) {
05      semid = semget (semkey, 1, 0);
      }
    }
06   else {    /* if created ... */
07    semun arg;
08    arg.val = 1;
09    status = semctl (semid, 0, SETVAL, arg);
    }

10  if (semid == -1 || status == -1) {
11    perror ("initsem failed");
12    return (-1);
    }
    /* all okay */
13  return (semid);
   }
```

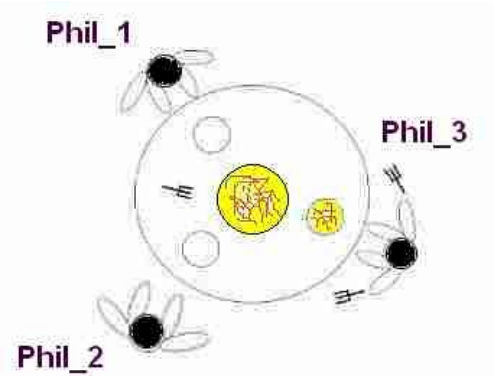
```
01 int p (int semid) {  // p.c
02  struct sembuf p_buf;

03  p_buf.sem_num = 0;
04  p_buf.sem_op = -1;
05  p_buf.sem_flg = SEM_UNDO;

06  if (semop (semid, &p_buf, 1) == -1){
07    perror ("p(semid) failed");
08    exit (1);
    }
09  return (0);
   }
```

(a) Erklären Sie den Quellcode jeweils zeilenweise.
(b) Was ändert sich bei `int v (int semid)` gegenüber `int p (int semid)`?

Aufgabe 12: 3 Philosophen sitzen um einen runden Tisch. Jeder hat einen eigenen Teller vor sich und in der Mitte des Tisches steht ein großer Topf mit Spaghetti. Zwischen je zwei kleinen Tellern liegt anfänglich eine Gabel.



Die Philosophen denken, werden hungrig oder essen. Essen kann man nur, wenn man die Gabel links und die Gabel rechts vom eigenen Teller aufnehmen konnte. Im Zustand hungrig versucht jeder Philosoph seine beiden Gabel zu ergattern. Nachdem er gegessen hat, legt er beide Gabeln zurück und fängt an zu denken. Später wird er wieder hungrig usw.

(a) Formulieren Sie C-Code für die Philosophen, um das Problem zu simulieren. Nutzen Sie Semaphore, um den Zugriff auf Ressourcen zu schützen. In diesem Teil der Aufgabe schützen Sie den Zugriff auf den großen Topf. Wer diesen ergattert, kann seine Gabeln aufnehmen und essen. Alle anderen sind später an der Reihe.

(b) Schon bei der Erweiterung auf vier Philosophen ist der Ansatz aus (a) nicht optimal, da jetzt theoretisch immer 2 Philosophen essen könnten. Ein neuer Ansatz, die Gabeln über Semaphore zu schützen, führt schnell zu einem Deadlock. Welchen Deadlock erkennen Sie?

(c) Beschreiben Sie einen besser funktionierenden Ansatz für den Fall, dass ein Philosoph essen kann, wenn er beide Gabeln aufnehmen konnte. ***7 Punkte***

Viel Erfolg!