Name: Matrikel-Nr.: Studiengang:

Klausur Betriebssysteme und Sicherheit, 03.08.2016

1	2	3	4	5	6	$\sum$
7	8	7	7	7	6	42

Alle Aussagen sind so ausführlich wie nötig, aber so knapp wie möglich zu begründen.

Aufgabe 1 – Echtzeit

7 Punkte

Gegenstand dieser Aufgabe ist die Einplanung einer Task-Menge T mit Tasks der Form $X = (e; p)$, beschrieben durch Ausführungszeit e und Periodenlänge p. Die relative Deadline d ist gleich der Periodenlänge ($d = p$). Der Scheduler des Systems kann laufende Jobs jederzeit unterbrechen, der dabei auftretende Overhead wird ignoriert. Alle Tasks sind zum Zeitpunkt $t = 0$ bereit. Es steht zur Ausführung genau ein Prozessor zur Verfügung.

Dazu werde folgendes Beispiel betrachtet: $T = \{T_1 = (2{,}5; 5), T_2 = (1; 3)\}$.

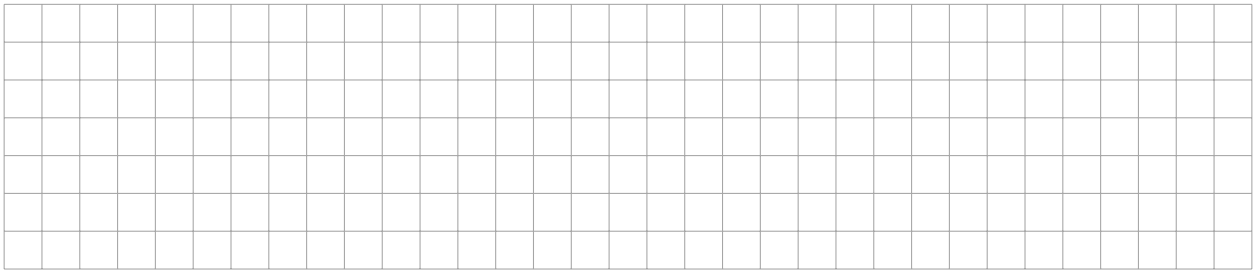
a) Diskutieren Sie die Einplanbarkeit der Taskmenge T mittels RMS anhand des zugehörigen Admission-Kriteriums.

HINWEIS: Nehmen Sie $\sqrt{2} = 1{,}41$ an. **1 P**

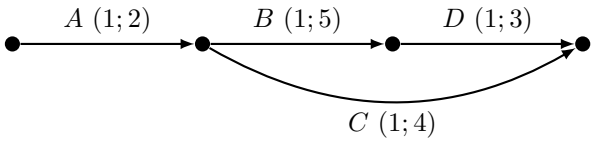
b) Prüfen Sie, ob die Taskmenge T mittels dynamischer Prioritätszuordnung einplanbar ist. Wählen Sie dazu ein Verfahren zur dynamischen Prioritätszuordnung. Wenden Sie dieses Verfahren auf die Taskmenge T an und erstellen Sie einen Ablaufplan. Ist Ihr Ablaufplan eine Begründung für die Einplanbarkeit mittels dynamischer Prioritätenzuordnung oder nicht? **3 P**

c) Welcher Task hat die höchste Priorität zum Zeitpunkt 9000 bei Einplanung mit dem von Ihnen in Teilaufgabe b) genannten Verfahren? **1 P**

Sofern erforderlich, können Sie diesen Bereich zum Zeichnen von Ablaufplänen für alle Teilaufgaben nutzen.



Das folgende Vorgangspfeilnetz repräsentiert eine Menge von einmalig auszuführenden Jobs $J = \{A,B,C,D\}$ in der Form $X = (e; D)$, wobei e die benötigte Ausführungszeit und D die absolute Deadline angibt:



Die gemeinsame Release-Zeit aller Jobs ist 0. Jeder Job wird erst zur Ausführung bereit, wenn alle seine Vorgänger beendet sind.

d) Ist die Jobmenge J mittels EDF einplanbar? **1 P**

e) Wie lang ist die kürzestmögliche Gesamtbearbeitungszeit für die Jobmenge J, wenn beliebig viele Prozessoren verwendet werden können? Warum ist keine weitere Verkürzung möglich? **1 P**

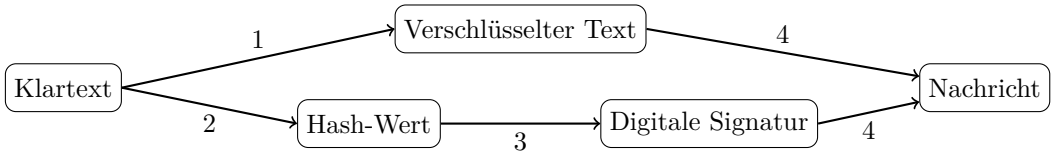
Sofern erforderlich, können Sie diesen Bereich zum Zeichnen von Ablaufplänen für alle Teilaufgaben nutzen.

Aufgabe 2 – Sicherheit

8 Punkte

Ein Chat-System zum Austausch von kurzen Nachrichten soll die Vertraulichkeit und Integrität der übertragenen Daten sicherstellen. Jeder Teilnehmer besitzt ein Schlüsselpaar aus einem privaten und einem öffentlichen Schlüssel. Für jede Kommunikationsbeziehung wird außerdem ein sicher ausgehandelter symmetrischer Schlüssel vorausgesetzt.

Das Chat-System verwendet das folgende Design:



1. Der Klartext der Nachricht wird mit dem symmetrischen Kommunikationsschlüssel verschlüsselt.
2. Parallel dazu wird die Nachricht von einer kryptografischen Hash-Funktion zusammengefasst.
3. Dieser Hash-Wert wird mit dem privaten Schlüssel des Senders in eine digitale Signatur umgewandelt.
4. Beide Nachrichtenteile werden zusammengefügt und über ein unsicheres Netz zum Empfänger gesendet.

a) Nennen Sie konkrete kryptografische Algorithmen, mit denen die Schritte 1, 2 und 3 jeweils umgesetzt werden können. **3 P**

b) Diskutieren Sie die Sicherheit des gegebenen Verfahrens unter den genannten Schutzzielen. Wie ließe sich die Sicherheit des Verfahrens verbessern? **3 P**

c) Wir nehmen an, dass durch Fortschritte bei Quantencomputern alle kryptografischen Algorithmen unsicher werden, die auf Faktorisierung oder diskreten Logarithmen beruhen. Nennen Sie einen kryptografische Algorithmus, der von dieser Entwicklung betroffen wäre und einen, der nicht betroffen wäre. **2 P**

Aufgabe 3 – Dateisysteme **7 Punkte**

In einem Unix-artigen Dateisystem werden alle Dateiinhalte und Metadaten in Blöcken auf dem Speichermedium organisiert. Die Größe eines Blocks beträgt 4 KiB.

a) Wie groß muss die Bitmap mit den belegt/frei-Bits für alle Blöcke (inklusive der für die Allokations-Bitmap selbst verwendeten Blöcke) sein, wenn das Dateisystem auf einem 4 GiB großen Speichermedium erzeugt wurde? **1 P**

b) Unix-Dateisysteme verwalten pro Datei jeweils ein sogenanntes „Inode“. Nennen Sie zwei verschiedene Arten von Metadaten, die in jedem Inode abgelegt sind. Erläutern Sie, welchen Vorteil Unix-Dateisysteme daraus ziehen, dass der Name einer Datei *nicht* im Inode gespeichert wird. **3 P**

Ein Anwendungsprogramm hat eine neue, 42 KiB große Datei erzeugt. Im Zuge dieser Operation wurden die nachfolgend aufgelisteten Metadatenblöcke im Buffer Cache des Betriebssystems modifiziert bzw. neu allokiert und zunächst gepuffert:

> 1 Superblock (SB), 1 Allokations-Block für Inodes (IA), 2 Allokationsblöcke für Blöcke (BA), 2 Inode-Blöcke (I_1 und I_2), 1 Verzeichnisblock (V), 1 Indirektionsblock mit Zeigern auf Datenblöcke (Z), 11 Datenblöcke (D)

Hinweis: I_1 enthält das neue Inode der angelegten Datei, I_2 das geänderte Inode des Elternverzeichnisses.

c) Geben Sie eine nach der Methode des „Synchronen Schreibens“ sichere Reihenfolge an, in der alle genannten Blöcke auf das Speichermedium geschrieben werden können, so dass das Dateisystem nach einem Absturz keine kritischen Inkonsistenzen aufweist. Markieren Sie in Ihrer Lösung die Blockschreiboperation(en) innerhalb der Sequenz, nach denen eine „Write Barrier“ für die Konsistenz nach einem Absturz *notwendig* ist. **3 P**

Aufgabe 4 – Virtueller Speicher **7 Punkte**

Gegeben sei eine 32-Bit-Architektur mit 2-stufiger Adressumsetzung. Die Tabelle der ersten Stufe heißt Seitenverzeichnis, die Tabellen der zweiten Stufe werden Seitentabellen genannt. Für jede Stufe werden 10 Bit der Adresse als Index verwendet, die übrigen 12 Bit sind das Offset. Die Einträge in den Tabellen sind jeweils 4 Byte groß.

a) Ein Prozess verwendet 5 Speicherseiten. Wie viel Speicherplatz wird zur Verwaltung dieses Adressraumes maximal benötigt? **1 P**

b) Welchen Nachteil hätte es nur eine 1-stufige Seitentabelle (20 Bit Index) für den Prozess aus a) zu verwenden? **1 P**

In den letzten 12 Bit der Einträge in den Tabellen werden folgende Zugriffsrechte und Optionen gespeichert:

- Eintrag ist vorhanden (p)
- Eintrag ist schreibbar (w)
- Eintrag ist als Copy-on-Write markiert (cow)

Für Zugriffe müssen die Rechte jeweils in beiden Stufen gesetzt sein.

Die folgende Tabellenbelegung sei gegeben. Dabei sind alle Zahlen hexadezimal zu verstehen. Bei Bedarf steht weiterhin der freie Rahmen Nummer `0x412` zur Verfügung.

Seitenverzeichnis

Rahmen-Nr.: `100`

0	-	–
1	370	p, w
2	108	p
...		
1cd	40090	p, w

Seitentabelle

Rahmen-Nr.: `108`

0	-	–
1	371	p
2	491	–

Seitentabelle

Rahmen-Nr.: `370`

0	411	p
1	375	p, cow
2	423	p, w

c) Geben Sie für die folgenden Zugriffe an, ob sie gültig oder ungültig sind. Begründen Sie dabei warum Zugriffe ungültig sind. Geben Sie außerdem für gültige Zugriffe die physische Adresse an, auf die das Programm zugreift. **4 P**

1. Lesen von `0x00801001`:

2. Schreiben des Wertes `0x00000000` an `0x00401000`:

3. Lesen von `0x00802002`:

4. Schreiben des Wertes `0xdeadbeef` an `0x00400033`:

d) Geben Sie an, welche Änderungen an den Seitentabellen sich aus den gültigen Zugriffen aus c) ergeben. **1 P**

Aufgabe 5 – Unix

7 Punkte

In einem Unix-ähnlichen System existieren im Verzeichnis `/tmp` die ausführbaren Dateien `Alpha` und `Omega`. Den beiden Programmen liegt folgender C-Code zugrunde:

Alpha

```
1  int main() {
2     int pid;
3     pid = fork();
4
5     if (pid < 0) { printf("F"); }
6     else {
7        if (pid == 0) { exec("/tmp/Omega"); }
8        else          { unlink("/tmp/Omega"); }
9     }
10    printf("A");
11    return 0;
12 }
```

Omega

```
1  int main() {
2     printf("M");
3     return 0;
4  }
```

a) Nennen Sie zwei mögliche Ursachen dafür, dass die Funktion `exec` fehlschlägt. **1 P**

b) Das Programm `Alpha` wird nun ausgeführt. Der aktuelle Benutzer verfügt dabei über alle erforderlichen Rechte an den beteiligten Dateien und Verzeichnissen. Skizzieren Sie kurz *alle* möglichen Programmabläufe und geben Sie jeweils an, welche Ausgaben dabei auf der Konsole erscheinen.

HINWEIS: Beachten Sie, dass Systemaufrufe fehlschlagen können. **5 P**

c) Welche Auswirkungen hat es, wenn innerhalb des Quelltextes von `Alpha` in den Zeilen 7 und 8 „Omega“ durch „Alpha“ ersetzt wird? **1 P**

Aufgabe 6 – Synchronisation

6 Punkte

Gegeben sei die folgende Implementierung eines Ticket-Locks, welche genutzt wird um kritische Abschnitte abzusichern.

```c
struct lock_t {
   int next_ticket;
   int cur_ticket;
};
```

```c
void init(struct lock_t *l) {
   l->next_ticket = 0;
   l->cur_ticket  = 0;
}
```

```c
void lock(struct lock_t *l) {
   int my_ticket =
    ↪ xadd(&l->next_ticket, 1);

   while (l->cur_ticket != my_ticket) {
      /* wait */
   }
}
```

```c
void unlock(struct lock_t *l) {
   l->cur_ticket++;
}
```

Hinweis: Der Aufruf `xadd(&a, b)` erhöht die Variable `a` atomar um den Wert `b`, und gibt den ursprünglichen Wert von `a` zurück. Wegen des atomaren Zugriffs kann ein anderer Thread die Variable `a` während dieses Vorgangs nicht verändern.

Das Lock wird wie folgt verwendet:

Initialisierung

```c
lock_t l;
init(&l);
```

Verwendung

```c
lock(&l);
/* Kritischer Abschnitt */
unlock(&l);
```

a) Ist diese Implementierung eines Locks korrekt? Begründen Sie Ihre Aussage. **1 P**

b) Geben Sie die Werte von `l->next_ticket` und `l->cur_ticket` nach der vollständigen Ausführung der nebenstehenden Befehlsfolge an. **1 P**

```c
lock_t l;
init(&l);

lock(&l);
unlock(&l);
lock(&l);
```

c) Ist die Implementierung dieses Locks fair? Begründen Sie Ihre Antwort. **2 P**

d) Welches Problem kann auftreten, wenn für die Variablen `next_ticket` und `cur_ticket` aus Platzgründen nur ein 4-Bit großer Datentyp verwendet wird? **1 P**

e) Kann die `lock`-Funktion durch die folgende Implementierung ersetzt werden? Begründen Sie Ihre Antwort. **1 P**

```
1  void lock2(struct lock_t *l) {
2     int my_ticket;
3
4     my_ticket = l->next_ticket;
5     xadd(&l->next_ticket, 1);
6
7     while (l->cur_ticket != my_ticket) {
8        /* wait */
9     }
10 }
```