FernUniversität in Hagen

Fakultät für
Mathematik und Informatik

Lehrgebiet Kooperative Systeme
Prof. Dr. Jörg M. Haake

Klausur zum Kurs

Betriebssysteme (1802)

am 18. September 2010

Klausurort:	
Vorname Name:	
Adresse:	
Matrikelnummer:	

Aufgabe	1	2	3	4	5	6	7	8	Summe
Punkte	30	20	25	35	30	15	30	15	200
bearbeitet									
erreichte Punkte									

Hinweise zur Bearbeitung der Klausur zum Kurs Betriebssysteme (1802)

Wir begrüßen Sie zur Klausur *Betriebssysteme* und bitten Sie, diese Hinweise vollständig und aufmerksam durchzulesen, bevor Sie mit der Bearbeitung der Aufgaben beginnen.

1. Prüfen Sie bitte die Vollständigkeit dieser Unterlagen:
   - Deckblatt, diese Hinweise und 8 Aufgaben auf den Seiten 1 bis 18
   - eine Teilnahmebescheinigung zur Vorlage beim Finanzamt
2. Bevor Sie mit der Bearbeitung der Aufgaben beginnen, tragen Sie bitte auf dem Deckblatt Name, Anschrift und Matrikelnummer ein.
3. Falls Sie eine Teilnahmebestätigung wünschen, füllen Sie diese bitte aus.
4. Schreiben Sie Ihre Lösungen bitte auf die Aufgabenblätter bzw. die dafür vorgesehenen Leerseiten und benutzen Sie auch die Rückseiten, wenn der Platz nicht reicht.
5. Auf jedes Blatt, auf dem sich Teile Ihrer Lösung befinden, schreiben Sie bitte oben Ihren Namen und Ihre Matrikelnummer.
6. Wenn Sie eine Prozedur oder ein Programm schreiben sollen, achten Sie auf eine klare Gliederung und eine *ausführliche* Kommentierung.
7. Wenn Sie Zahlenwerte ausrechnen sollen, skizzieren Sie auch den Rechenweg.
8. Als Hilfsmittel ist unbeschriebenes Konzeptpapier zugelassen.
9. Zum Bestehen der Klausur reichen 100 von 200 Punkten auf jeden Fall aus.

Wir wünschen Ihnen bei der Bearbeitung der Klausur viel Erfolg!

Aufgabe 1: Prozesse 5 + 5 + 5 + 5 + 5 + 5 Punkte

Geben Sie bitte auf jede Frage eine kurze Antwort.

1. Was bedeutet *Mehrprogrammbetrieb*? Welche zwei wesentlichen Probleme bei einem Mehrprogrammbetrieb müssen gelöst werden?

2. Was ist ein *Prozess*? Welche Informationen über einen Prozess sind für das Betriebssystem wichtig und wo werden sie gespeichert?

3. In welcher Datenstruktur werden Informationen über *alle vorhandenen Prozesse* abgelegt?

4. Wie wird die Anfangsadresse einer *Unterbrechungsroutine* gefunden?

5. Warum muss der *Kontext* eines gerade unterbrochenen Prozesses vor Ausführung der Unterbrechungsroutine gerettet werden?

6. Warum können die *Threads* eines Prozesses effizienter miteinander kommunizieren als separate Prozesse?

Aufgabe 2: **Scheduling** **5 + 5 + 5 + 5 Punkte**

Geben Sie bitte auf jede Frage eine kurze Antwort.

1. In einem *nicht-präemptiven System* ist es möglich, dass ein Prozess in eine Endlos-Schleife gerät und den Prozessor für sich allein behält. Kann so ein Problem auch in einem *präemptiven System* entstehen? Falls ja, unter welchen Umständen?

2. Wenn man die *SJF-* und die *FCFS-Strategie* als Priorität-Strategien auffasst, wie sind dann die *Prioritäten* festgelegt?

3. Wie funktioniert die *Round-Robin-Strategie*?

4. Welche *Scheduling-Strategien* kann man für *Benutzer-Threads* einsetzen?

Aufgabe 3: Hauptspeicher 5 + 5 + 5 + 5 + 5 Punkte

1. Warum sind *Systemaufrufe* notwendig für ein Betriebssystem?
2. Was ist der Unterschied zwischen *logischen* und *physischen* Adressen?
3. Wir nehmen an, dass in einem *Paging-System* eine virtuelle Adresse $v = (s, d)$ 32 Bit lang ist, wobei s die Seitennummer und d der Offset sind. Wie viele Seiten gibt es, wenn der Offset n Bit lang ist? Warum beeinflusst die Auswahl der Zahl n die *interne Fragmentierung* und den Speicherbedarf der *Seitentabelle*?
4. Die Strategien *Demand Paging* und *Prepaging* entscheiden, wann eine Seite in den Hauptspeicher geholt werden soll. Welche Strategie benötigt mehr Aufwand?
5. Warum wird die Auslagerungsstrategie *LRU* kaum implementiert?

Aufgabe 4: Synchronisation 5 + 5 + 10 + 15 Punkte

Geben Sie bitte auf jede Frage eine kurze Antwort.

1. Unter welchen Problemen leiden die Synchronisationsverfahren mit *globalen Variablen*?

2. Wir verwenden jetzt *Semaphore*. Warum kann ein Prozess sich zu jedem Zeitpunkt höchstens in einer Semaphor-Warteschlange befinden?

3. Wozu benötigt ein *Monitor* die Funktionen `wait()` und `signal()`?

4. Warum verhindern die folgenden drei Strategien *Deadlocks*? Denken Sie an die vier notwendigen *Deadlock-Bedingungen*.

   (a) Wenn ein Prozess zusätzliche Betriebsmittel anfordert, muss er alle bisher reservierten Betriebsmittel zunächst freigeben und dann zusammen mit den zusätzlichen Betriebsmitteln erneut anfordern.

   (b) Jeder Prozess muss alle von ihm benötigten Ressourcen gleichzeitig im Voraus anfordern. Wenn sie verfügbar sind, dann werden sie ihm zugeteilt, sonst blockiert der Prozess.

   (c) Eine lineare Ordnung *„wichtiger als“* aller Betriebsmittel wird festgelegt. Ein Prozess kann nur noch weniger wichtige Betriebsmittel nachfordern.

Aufgabe 5: Diskscheduling 20 + 10 Punkte

Wir betrachten die Bewegungen eines Festplattenlesekopfs.

1. Wir nehmen an, dass die Zugriffsaufträge auf die *Zylindern*

   $$33, 72, 47, 8, 99, 74, 52, 75$$

   in dieser Reihenfolge in der Warteschlange vorliegen. Der Kopf steht gerade auf Zylinder Nummer 63.

   In welcher Reihenfolge werden die Strategien

   (a) *Shortest-Seek-Time-First* (SSTF),

   (b) *SCAN* (der Kopf bewegt sich gerade in Richtung absteigender Zylindernummern)

   die Zylinder besuchen?

2. Welche Strategie hat das Problem der *Starvation* und warum? Wie kann man diese Strategie modifizieren, damit das Problem vermieden wird?

Aufgabe 6: **Dateisysteme** **5 + 5 + 5 Punkte**

Wir betrachten ein *Dateisystem* mit *i-nodes*. Die Blockgröße ist 1 KByte, und eine Blockadresse ist 2 Byte (16 Bit) lang. Was ist die maximale Dateigröße, wenn der i-node wie folgt organisiert wird?

1. Der i-node enthält nur 12 direkte Adressen und keine indirekten.
2. Der i-node enthält 12 direkte und eine indirekte Adresse.
3. Der i-node enthält 12 direkte, eine indirekte und eine doppelt indirekte Adresse.

Aufgabe 7: Sicherheit 10 + 10 + 10 Punkte

Die beiden primären Ziele von Sicherheitsmechanismen in einem Rechnersystem sind die Kontrolle von Zugriffen auf das System und von Zugriffen auf Informationen innerhalb des Systems.

Antworten Sie bitte auf die folgenden Fragen so kurz wie möglich.

1. Was ist der Unterschied zwischen *Identifikation* und *Authentisierung*? Für welches der oben genannten Sicherheitsziele kann man diese Mechanismen einsetzen?

2. Was bedeuten *diskretionäre Zugriffs- und Informationsflusskontrollen*? Für welches der genannten Ziele kann man sie einsetzen?

3. Warum kann man mit diskretionären Zugriffskontrollen einen Angriff durch *Trojaner* nicht verhindern?

Aufgabe 8: Kommandosprachen 15 Punkte

Schreiben Sie ein `Makefile` für die folgende Anwendung.

Es gibt eine ausführbare Datei namens `prog`, die immer wieder neu gebunden wird, wenn sich eines der Objektmodule `a.o` und `b.o` geändert hat. Für das Binden wird das Kommando

```
cc a.o b.o -o prog
```

ausgeführt.
Die Objektmodule sind wiederum abhängig von den Quelldateien `a.c` und `b.c` und werden durch Kompilieren erzeugt mit den Befehlen

```
cc -o a.o a.c
cc -o b.o b.c
```