Musterlösung der Abschlussklausur

Betriebssysteme

22. Juli 2019

Name: _____

Vorname: _____

Matrikelnummer: _____

Mit meiner Unterschrift bestätige ich, dass ich die Klausur selbständig bearbeite und das ich mich gesund und prüfungsfähig fühle.
Mir ist bekannt, dass mit dem Erhalt der Aufgabenstellung die Klausur als angetreten gilt und bewertet wird.

Unterschrift: _____

- Schreiben Sie Ihre Lösungen auf die vorbereiteten Blätter. Eigenes Papier darf *nicht* verwendet werden.
- Als Hilfsmittel ist ein *selbständig vorbereitetes* und *handschriftlich einseitig beschriebenes DIN-A4-Blatt* zugelassen (keine Kopien!).
- Als Hilfsmittel ist ein *Taschenrechner* zugelassen.
- Verwenden Sie *keinen* Rotstift.
- Die Bearbeitungszeit beträgt *90 Minuten*.
- Schalten Sie Ihre Mobiltelefone aus.

Bewertung:

Aufgabe:	1	2	3	4	5	6	7	8	9	10	11	**Σ**	Note
Maximale Punkte:	8	14	8	4	10	8	4	8	10	9	7	90	—
Erreichte Punkte:													

1.0: 90.0-85.5, **1.3**: 85.0-81.0, **1.7**: 80.5-76.5, **2.0**: 76.0-72.0, **2.3**: 71.5-67.5,
2.7: 67.0-63.0, **3.0**: 62.5-58.5, **3.3**: 58.0-54.0, **3.7**: 53.5-49.5, **4.0**: 49.0-45.0, **5.0**: <45

Name: Vorname: Matr.Nr.:

Aufgabe 1)

Punkte: .....

Maximale Punkte: 1+1+2+2+2=8

a) Zu jedem Zeitpunkt kann nur ein einziges Programm laufen. Nennen Sie den passenden Fachbegriff für diese Betriebsart.

Einzelprogrammbetrieb (Singletasking).

b) Nennen Sie den Fachbegriff der quasi-parallelen Programm- bzw. Prozessausführung.

Mehrprogrammbetrieb oder Multitasking.

c) Beschreiben Sie den Aufbau eines monolithischen Kernels.

Monolithische Kerne enthalten...

- *Funktionen für Speicherverwaltung*
- *Funktionen für Prozessverwaltung*
- *Funktionen für Prozesskommunikation*
- *Gerätetreiber*
- *Dateisysteme-Treiber*

Außerhalb des Kerns befinden sich die Benutzerprozesse.

d) Beschreiben Sie den Aufbau eines minimalen Kerns (Mikrokernels).

Im Kern befinden sich üblicherweise nur:

- *Notwendigste Funktionen zur Speicher- und Prozessverwaltung*
- *Funktionen zur Synchronisation und Interprozesskommunikation*
- *Notwendigste Treiber (z.B. zum Systemstart)*

Gerätetreiber, Dateisysteme und Dienste (Server), befinden sich außerhalb des Kerns und laufen wie die Anwendungsprogramme im Benutzermodus

e) Beschreiben Sie den Aufbau eines hybriden Kernels.

Hybride Kerne sind ein Kompromiss zwischen monolithischen Kernen und minimalen Kernen. Die enthalten aus Geschwindigkeitsgründen Komponenten, die bei minimalen Kernen außerhalb des Kerns liegen. Es ist nicht festgelegt, welche Komponenten bei Systemen mit hybriden Kernen zusätzlich in den Kernel einkompiliert sind.

Name: Vorname: Matr.Nr.:

Aufgabe 2)

Punkte: .....

Maximale Punkte: 1+2+2+3+6=14

a) Welche zwei Gruppen von Ein- und Ausgabegeräten gibt es bezüglich der kleinsten Übertragungseinheit?

Zeichenorientierte Geräte und blockorientierte Geräte.

b) Vergleichen Sie die Arbeitsweise der Gruppen aus Teilaufgabe a).

Zeichenorientierte Geräte: Bei Ankunft/Anforderung jedes einzelnes Zeichens wird immer mit der CPU kommuniziert.

Blockorientierte Geräte: Datenübertragung findet erst statt, wenn ein kompletter Block (z.B. 1-4 kB) vorliegt.

c) Nennen Sie für jede Gruppe aus Teilaufgabe a) zwei Beispiele.

Zeichenorientierte Geräte: Maus, Tastatur, Drucker, Terminal, Magnetband...

Blockorientierte Geräte: Festplatte, SSD, CD-/DVD-Laufwerk, Disketten-Laufwerk...

d) Nennen Sie drei Möglichkeiten, wie Prozesse Daten von Ein- und Ausgabegeräten lesen können.

- *Busy Waiting (geschäftiges bzw. aktives Warten)*
- *Interrupt-gesteuert*
- *Direct Memory Access (DMA)*

e) Nennen Sie für jede Möglichkeit aus Teilaufgabe d) jeweils einen Vorteil und einen Nachteil.

- *Busy Waiting (geschäftiges bzw. aktives Warten)*
  - *Vorteile: Keine zusätzliche Hardware nötig*
  - *Nachteile: Belastet die CPU, verlangsamt die gleichzeitige Abarbeitung mehrerer Prozesse*
- *Interrupt-gesteuert*
  - *Vorteile: Die CPU wird nicht blockiert, gleichzeitige Abarbeitung mehrerer Prozesse wird nicht verlangsamt*
  - *Nachteile: Zusätzliche Hardware (Interrupt-Controller) ist nötig*
- *Direct Memory Access (DMA)*
  - *Vorteile: Vollständige Entlastung der CPU, gleichzeitige Abarbeitung mehrerer Prozesse wird nicht verlangsamt*
  - *Nachteile: Zusätzliche Hardware (DMA-Controller) ist nötig*

Name: Vorname: Matr.Nr.:

Aufgabe 3)

Punkte: .....

Maximale Punkte: 2+2+2+2=8

Auf einer Festplatte befinden sich folgende Informationen:

```
IBM Travelstar                                  MODEL: DBCA-204860 E182115 T
RATED: 5V 500mA                               MADE IN THAILAND BY IBM STORAGE
P/N: 21L9510   4090 MB                                                16NOV99
FRU: 22L0018   MLC:F41941                    (7944 CYL. 16 HEADS. 63 SEC/T)
```

a) Berechnen Sie die Kapazität einer Oberfläche einer Scheibe der Festplatte.
(Bei der Lösung muss der Rechenweg angegeben sein!)

Hinweis: Die Anzahl der Zylinder (`CYL`) ist identisch mit der Anzahl der Spuren („Tracks“) pro Scheibe. Die Größe der Sektoren (`SEC`) ist 512 Byte.

Lösung: Die Kapazität einer Oberfläche einer Scheibe = Anzahl der Zylinder ∗ Anzahl der Sektoren pro Spur bzw. Track (`spt`) ∗ Bytes pro Sektor bzw. Block.

$7.944 * 63 * 512 = 256.241.664$ Byte

b) Berechnen Sie die Größe einer Spur der Festplatte.
(Bei der Lösung muss der Rechenweg angegeben sein!)

Lösung: Größe einer Spur = Anzahl der Sektoren pro Spur bzw. Track (`spt`) ∗ Bytes pro Sektor bzw. Block.

$63 * 512 = 32.256$ Byte

c) Berechnen Sie die Gesamtkapazität der Festplatte.
(Bei der Lösung muss der Rechenweg angegeben sein!)

*Lösung: Gesamtkapazität = Kapazität einer Scheibe * Anzahl der Köpfe (`heads`).*

$256.241.664 * 16 = 4.099.866.624$ Byte

d) Wie viele Scheiben hat die Festplatte? *Hinweis: Jede Scheibe hat zwei Oberflächen.*
(Begründen Sie ihre Antwort!)

Wenn die logische Geometrie mit der physischen Geometrie übereinstimmt, dann hat die Festplatte wegen der 16 Köpfe auch 16 Oberflächen und damit 8 Scheiben.

Name: Vorname: Matr.Nr.:

Aufgabe 4)

Punkte: .....

Maximale Punkte: 4

Kreuzen Sie bei jeder Aussage an, ob die Aussage wahr oder falsch ist.

a) Real Mode ist für Multitasking-Systeme geeignet.

☐ Wahr ☒ Falsch

b) Beim Protected Mode läuft jeder Prozess in seiner eigenen, von anderen Prozessen abgeschotteten Kopie des physischen Adressraums.

☒ Wahr ☐ Falsch

c) Bei statischer Partitionierung entsteht interne Fragmentierung.

☒ Wahr ☐ Falsch

d) Bei dynamischer Partitionierung ist externe Fragmentierung unmöglich.

☐ Wahr ☒ Falsch

e) Beim Paging haben alle Seiten die gleiche Länge.

☒ Wahr ☐ Falsch

f) Ein Vorteil langer Seiten beim Paging ist geringe interne Fragmentierung.

☐ Wahr ☒ Falsch

g) Ein Nachteil kurzer Seiten beim Paging ist, das die Seitentabelle sehr groß werden kann.

☒ Wahr ☐ Falsch

h) Die MMU übersetzt beim Paging logische Speicheradressen mit der Seitentabelle in physische Adressen.

☒ Wahr ☐ Falsch

Name: Vorname: Matr.Nr.:

Aufgabe 5)

Punkte: .....

Maximale Punkte: 10

a) Geben Sie an, welche Informationen ein Inode speichert.

Speichert die Verwaltungsdaten (Metadaten) einer Datei, außer dem Dateinamen.

b) Nennen Sie drei Beispiele für Metadaten im Dateisystem.

Metadaten sind u.a. Dateigröße, UID/GID, Zugriffsrechte und Datum.

c) Beschreiben Sie, was ein Cluster im Dateisystem ist.

Dateisysteme adressieren Cluster und nicht Blöcke des Datenträgers. Jede Datei belegt eine ganzzahlige Menge an Clustern.

d) Beschreiben Sie, wie ein UNIX-Dateisystem (z.B. ext2/3), das keine Extents verwendet, mehr als 12 Cluster adressiert.

Durch indirekte Adressierung über zusätzliche Cluster, die ausschließlich Cluster-Nummern enthalten.

e) Beschreiben Sie, wie Verzeichnisse bei Linux-Dateisystemen technisch realisiert sind.

Verzeichnisse sind nur Text-Dateien, die die Namen und Pfade von Dateien enthalten.

f) Die meisten Betriebssystemen arbeiten nach dem Prinzip...

☒ Write-Back ☐ Write-Through

g) `/home/<benutzername>/Mail/inbox/` ist ein...

☒ Absoluter Pfadname ☐ Relativer Pfadname

h) Nennen Sie die Information, die der Bootsektor eines Dateisystems speichert.

Im Bootsektor liegen ausführbarer Maschinencode („Boot-Loader"), der das Betriebssystem starten soll und Informationen über das Dateisystem.

i) Nennen Sie die Information, die der Superblock eines Dateisystems speichert.

Er enthält Informationen über das Dateisystem, z.B. Anzahl der Inodes und Cluster.

j) Erklären Sie warum manche Dateisysteme (z.B. ext2/3) die Cluster des Dateisystems zu Blockgruppen zusammenfassen.

Die Inodes (Metadaten) liegen dadurch physisch nahe bei den Clustern, die sie adressieren.

Name: Vorname: Matr.Nr.:

Aufgabe 6)

Punkte: .....

Maximale Punkte: 2+1+1+3+1=8

a) Beschreiben Sie, was die Dateizuordnungstabelle bzw. File Allocation Table (FAT) ist und welche Informationen diese enthält.

Die FAT (Dateizuordnungstabelle) ist eine Tabelle fester Größe. Für jeden Cluster des Dateisystems existiert in der FAT ein Eintrag mit folgenden Informationen über den Cluster:

- *Cluster ist frei oder das Medium an dieser Stelle beschädigt.*
- *Cluster ist von einer Datei belegt und enthält die Adresse des nächsten Clusters, der zu dieser Datei gehört bzw. ist der letzte Cluster der Datei.*

b) Beschreiben Sie die Aufgabe des Journals bei Journaling-Dateisystemen.

Im Journal werden Schreibzugriffe gesammelt, bevor sie durchgeführt werden.

c) Nennen Sie einen Vorteil von Journaling-Dateisystemen gegenüber Dateisystemen ohne Journal.

Nach einem Absturz müssen nur diejenigen Dateien (Cluster) und Metadaten überprüft werden, die im Journal stehen.

d) Nennen Sie die drei Werte, die zum Speichern eines Extents nötig sind.

Start (Clusternummer) des Bereichs (Extents) in der Datei.

Größe des Bereichs in der Datei (in Clustern).

Nummer des ersten Clusters auf dem Speichergerät.

e) Beschreiben Sie den Vorteil des Einsatzes von Extents gegenüber direkter Adressierung der Cluster.

Statt vieler einzelner Clusternummern sind nur 3 Werte nötig. Vorteil: Weniger Verwaltungsaufwand.

Name: Vorname: Matr.Nr.:

Aufgabe 7)

Punkte: .....

Maximale Punkte: 4

a) Beschreiben Sie, was das Defragmentieren macht.

Logisch zusammengehörende Cluster von Dateien werden räumlich beieinander auf dem Speichermedium angeordnet.

b) Beschreiben Sie welche Art der Datenverarbeitung durch Defragmentieren maximal beschleunigt wird.

Eine zusammenhängende Anordnung beschleunigt das fortlaufende Vorwärtslesen der Daten maximal, da keine Warte- und Suchzeiten mehr vorkommen können.

c) Beschreiben Sie in welchen Szenario das Defragmentieren sinnvoll ist.

Wenn die Suchzeiten groß sind.

d) Ist das Defragmentieren bei SSDs sinnvoll? (Begründen Sie ihre Antwort.)

Es ist nicht sinnvoll. Bei SSDs spielt die Position der Cluster keine Rolle für die Zugriffsgeschwindigkeit. Zudem haben die Speicherzellen von SSDs eine eingeschränkte Anzahl an Schreib-/Löschzyklen.

Name: Vorname: Matr.Nr.:

Aufgabe 8)

Punkte: .....

Maximale Punkte: 8

a) Beschreiben Sie, was der Systemaufruf `fork()` macht.

Ruft ein Prozess `fork()` auf, wird eine identische Kopie als neuer Prozess gestartet.

b) Beschreiben Sie, was der Systemaufruf `exec()` macht.

Der Systemaufruf `exec()` ersetzt einen Prozess durch einen anderen.

c) Erklären Sie, was `init` ist.

`init` ist der erste Prozess unter Linux/UNIX. Er hat die PID 1. Alle laufenden Prozesse stammen von `init` ab. `init` ist der Vater aller Prozesse.

d) Nennen Sie den Unterschied eines Kindprozess vom Elternprozess kurz nach der Erzeugung.

Die PID und die Speicherbereiche.

e) Beschreiben, Sie was passiert, wenn ein Elternprozess vor dem Kindprozess beendet wird.

`init` adoptiert den Kind-Prozess. Die PPID des Kind-Prozesses hat dann den Wert 1.

f) Nennen Sie den Inhalt des Textsegments.

Den ausführbaren Programmcode (Maschinencode).

g) Nennen Sie den Inhalt des Heap.

Konstanten und Variablen die außerhalb von Funktionen deklariert sind.

h) Nennen Sie den Inhalt des Stack.

Kommandozeilenargumente des Programmaufrufs, Umgebungsvariablen, Aufrufparameter und Rücksprungadressen der Funktionen, lokale Variablen der Funktionen.

Name: Vorname: Matr.Nr.:

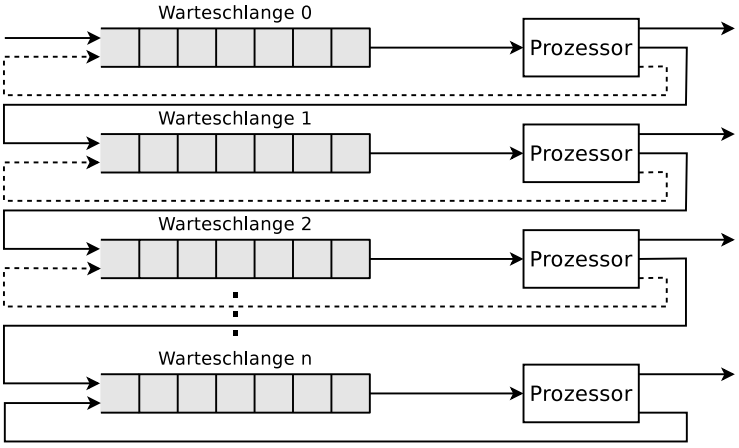
Aufgabe 9)

Punkte: .....

Maximale Punkte: 6+2+2=10

a) Beschreiben Sie (gerne auch mithilfe einer Abbildung) wie das Multilevel-Feedback-Scheduling funktioniert.

Es arbeitet mit mehreren Warteschlangen. Jede Warteschlange hat eine andere Priorität oder Zeitmultiplex. Jeder neue Prozess kommt in die oberste Warteschlange und hat damit die höchste Priorität. Innerhalb jeder Warteschlange wird Round Robin eingesetzt. Gibt ein Prozess die CPU freiwillig wieder ab, wird er wieder in die selbe Warteschlange eingereiht. Hat ein Prozess seine volle Zeitscheibe genutzt, kommt er in die nächst tiefere Warteschlange mit einer niedrigeren Priorität.



Quelle: William Stallings. Betriebssysteme. Pearson Studium. 2003

b) Nennen Sie <u>vier</u> Schedulingverfahren, die fair sind.

First Come First Served, *Round Robin mit Zeitquantum*, *Earliest Deadline First*, *Fair-Share*, *Multilevel-Feedback-Scheduling*.

c) Nennen Sie <u>vier</u> Schedulingverfahren, bei denen die CPU-Laufzeit (= *Rechenzeit*) der Prozesse <u>nicht</u> bekannt sein muss.
(Hinweis: Es sind also nur solche Schedulingverfahren gesucht, die unter realistischen Bedingungen eingesetzt werden können.)

Prioritätengesteuertes Scheduling, *First Come First Served*, *Round Robin mit Zeitquantum*, *Earliest Deadline First*, *Fair-Share*, *Multilevel-Feedback-Scheduling*.

Name: Vorname: Matr.Nr.:

Aufgabe 10)

Punkte: .....

Maximale Punkte: 2+7=9

a) Kreuzen Sie vier Bedingungen an, die gleichzeitig erfüllt sein müssen, damit ein Deadlock entstehen kann?

- ☐ Rekursive Funktionsaufrufe
- ☒ Wechselseitiger Ausschluss
- ☐ Häufige Funktionsaufrufe
- ☐ Geschachtelte `for`-Schleifen
- ☒ Ununterbrechbarkeit
- ☒ Anforderung weiterer Betriebsmittel
- ☐ > 128 Prozesse im Zustand `blockiert`
- ☐ Iterative Programmierung
- ☒ Zyklische Wartebedingung
- ☐ Warteschlangen

b) Kommt es zum Deadlock?
Führen Sie die Deadlock-Erkennung mit Matrizen durch.

$$\text{Ressourcenvektor} = \begin{pmatrix} 4 & 8 & 6 & 6 & 5 \end{pmatrix}$$

$$\text{Belegungsmatrix} = \begin{bmatrix} 0 & 2 & 1 & 0 & 0 \\ 2 & 3 & 1 & 0 & 4 \\ 1 & 0 & 2 & 1 & 1 \end{bmatrix} \qquad \text{Anforderungsmatrix} = \begin{bmatrix} 3 & 3 & 2 & 4 & 5 \\ 0 & 3 & 1 & 4 & 0 \\ 0 & 2 & 3 & 5 & 4 \end{bmatrix}$$

Aus dem Ressourcenvektor und der Belegungsmatrix ergibt sich der Ressourcenrestvektor.

$$\textit{Ressourcenrestvektor} = \begin{pmatrix} 1 & 3 & 2 & 5 & 0 \end{pmatrix}$$

Nur Prozess 2 kann bei diesem Ressourcenrestvektor laufen. Folgender Ressourcenrestvektor ergibt sich, wenn Prozess 2 beendet ist und seine Ressourcen freigegeben hat.

$$\textit{Ressourcenrestvektor} = \begin{pmatrix} 3 & 6 & 3 & 5 & 4 \end{pmatrix}$$

Nur Prozess 3 kann bei diesem Ressourcenrestvektor laufen. Folgender Ressourcenrestvektor ergibt sich, wenn Prozess 3 beendet ist und seine Ressourcen freigegeben hat.

$$\textit{Ressourcenrestvektor} = \begin{pmatrix} 4 & 6 & 5 & 6 & 5 \end{pmatrix}$$

Nun kann Prozess 1 laufen.

Es kommt nicht zum Deadlock.

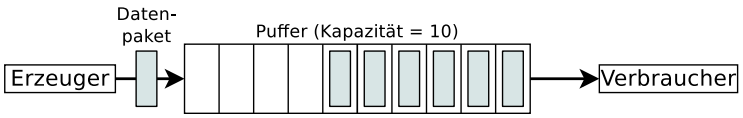
Name: Vorname: Matr.Nr.:

Aufgabe 11)

Punkte: .....

Maximale Punkte: 7

- Ein Erzeuger schreibt Daten in den Puffer und der Verbraucher entfernt diese.
- Gegenseitiger Ausschluss ist nötig, um Inkonsistenzen zu vermeiden.
- Ist der Puffer voll, muss der Erzeuger blockieren.
- Ist der Puffer leer, muss der Verbraucher blockieren.



Synchronisieren Sie die beiden Prozesse, indem Sie die nötigen Semaphoren erzeugen, diese mit Startwerten versehen und Semaphor-Operationen einfügen.

```
typedef int semaphore;              // Semaphore sind von Typ Integer
semaphore voll  = 0;                // zählt die belegten Plätze im Puffer
semaphore leer  = 10;               // zählt die freien Plätze im Puffer
semaphore mutex = 1;                // steuert Zugriff auf kritische Bereiche

void erzeuger (void) {
  int daten;

  while (TRUE) {                    // Endlosschleife
    erzeugeDatenpaket(daten);       // erzeuge Datenpaket
    P(leer);                        // Zähler "leere Plätze" erniedrigen
    P(mutex);                       // in kritischen Bereich eintreten
    einfuegenDatenpaket(daten);     // Datenpaket in Puffer schreiben
    V(mutex);                       // kritischen Bereich verlassen
    V(voll);                        // Zähler für volle Plätze erhöhen
  }
}

void verbraucher (void) {
  int daten;

  while (TRUE) {                    // Endlosschleife
    P(voll);                        // Zähler "volle Plätze" erniedrigen
    P(mutex);                       // in kritischen Bereich eintreten
    entferneDatenpaket(daten);      // Datenpaket aus dem Puffer holen
    V(mutex);                       // kritischen Bereich verlassen
    V(leer);                        // Zähler für leere Plätze erhoehen
    verbraucheDatenpaket(daten);    // Datenpaket nutzen
  }
}
```