Lösungshinweise zur Klausur zum Kurs 1802
Betriebssysteme im SS 98
Lehrgebiet Praktische Informatik VI

Aufgabe 1:

Unterschiede. In der Betriebssart Stapelbetrieb (off-line Betrieb) muß der Anwender zunächst eine Auftragsbeschreibung (Stapeljob, batch job) erstellen. Diese Beschreibung enthält die nacheinander auszuführenden Programme, deren Reihenfolge (falls es mehrere sind), alle benötigten Eingabedaten (diese müssen also alle vor der Ausführung des Jobs vorhanden sein) und weitere notwendige Informationen. Dieser Job wird dann an das System übergeben und i. a. zu einer durch das System festgelegten Zeit ausgeführt. Der Auftraggeber hat i. a. während der Ausführung keinen Kontakt zum Rechner, sodaß keinerlei Kommunikationsmöglichkeiten zwischen den gestarteten Programmen und den Auftraggebern bestehen. Am Ende eines Stapeljobs werden i. a. die Ausgaben des Auftrags und/oder ein Protokoll ausgedruckt.

In der Betriebsart Hintergrundausführung (einer Zwischenform zwischen Dialog- und Stapelbetrieb) wird ein Programm oder eine Kommandoprozedur interaktiv gestartet und dann „im Hintergrund“ ausgeführt, während die interaktiven Programme im Dialogbetrieb weiter „im Vordergrund” ablaufen. Der wesentliche Unterschied zur Stapelausführung besteht darin, daß Hintergrundprozesse i. w. wie Dialogprozesse behandelt werden, um die Kommunikationsmöglichkeiten des Hintergrundprozesses mit den anderen Prozessen sicherzustellen. Hintergrundprozesse werden meist von Dialogprozessen aus (sofort) gestartet, die dann „im Vordergrund” weiterlaufen. In solchen Fällen muß der Dialogprozeß benachrichtigt werden, wenn ein von ihm gestarteter Hintergrundprozeß beendet worden ist.

Motivation der Betriebsarten und weitere Unterschiede. Der Stapelbetrieb ist hauptsächlich durch das Ziel motiviert, die vorhandenen Ressourcen eines Rechnersystems möglichst hoch auszulasten (insbes. ist das wirtschaftlich notwendig, wenn die Ressourcen sehr teuer in der Anschaffung waren). Die Ablaufsteuerung in Systemen mit Stapelbetrieb ist darauf ausgerichtet, die Ausführung der Jobs so zu steuern, daß alle vorhandenen Ressourcen optimal ausgelastet werden. Die detaillierte Beschreibung der Aufträge vor ihrer Ausführung wird zum Beispiel für die Ablaufsteuerung der Aufträge entsprechend ausgenutzt. Der Hintergrundbetrieb ist dadurch motiviert, daß man die Möglichkeit bekommt, eine Aufgabe von mehreren parallel ausgeführten und miteinander kommunizierenden Prozessen lösen zu lassen. Oft hat man einen interaktiven Prozeß (der also direkt mit den Ein- und Ausgabegeräten verbunden ist) und mehrere Hintergrundprozesse, die vom Interaktiven gesteuert werden (Fenstersysteme; Serverprozesse). Aus der Sicht der Ablaufsteuerung müssen Hintergrundprozesse wie Dialogprozesse behandelt werden.

Aufgabe 2:

a) Die Plattengröße ist das Produkt aus Anzahl der Sektoren (x) und der Blockgröße. In unserem Fall gilt also: 4 GB = $x \times 0,5$ kB. Daraus folgt: $x = \frac{4\,\text{GB}}{0.5\,\text{kB}}$ = 8 M = 8 x 2^{20}. Es gibt also 8 M Sektoren und man braucht 23 bit um eine Sektornummer zu speichern. Auf volle Bytes aufgerundet braucht man 3 Bytes.

b) In der FAT steht für jeden Sektor ein Eintrag mit der Sektornummer des Folgesektors. Es gibt 8 M Einträge mit einer Größe von je 3 B. Also braucht die FAT 8 M x 3 B = 24MB Platz auf der Platte.

c)
- Ganz ohne indirekte Sektoradressen hat man nur die 10 Sektoradressen, die direkt im i-node stehen. Damit können $10 \text{ x } 0,5\,\text{kB} = 5\,\text{kB}$ große Dateien verwaltet werden.
- In einem Block der Größe 0.5 kB können 512 B/3 B = $170,\bar{6}$ Sektoradressen der Größe 3 B stehen. Auf die nächste ganze Zahl abgerundet erhält man somit 170 Sektoradressen, die in einem Sektor Platz haben.

  Ohne zwei- und dreifach indirekte Sektoradressen kann man also Dateien der Größe $5\,\text{kB} + 170 \text{ x } 0,5\,\text{kB} = 5\,\text{kB} + 85\,\text{kB} = 90\,\text{kB}$ verwalten.
- Ohne dreifach indirekte Sektoradressen kann man Dateien der Größe $90\,\text{kB} + (170^2) \text{ x } 0,5\,\text{kB} = 90\,\text{kB} + 14\,450\,\text{kB} = 14\,540\,\text{kB} \approx 14,2$ MB verwalten.
- Ohne Beschränkungen kann man Dateien der Größe

$$\begin{aligned} 14\,540\,\text{kB} + (170^3) \text{ x } 0,5\,\text{kB} &= 14\,540\,\text{kB} + 2\,456\,500\,\text{kB} \\ &= 2471040\,\text{kB} \\ &\approx 2413,1\,\text{MB} \\ &\approx 2,36\,\text{GB} \end{aligned}$$

verwalten.

d) Bei der größten möglichen Datei werden die folgenden Anzahlen an Blöcken für Verwaltungsinformationen belegt.

1	Block einfach indirekt
$1 + 170 = 171$	Blöcke zweifach indirekt
$171 + 170^2 = 29071$	Blöcke dreifach indirekt
29243	Blöcke insgesamt

Da jeder Block eine Größe von 512 Bytes hat sind das insgesamt $14\,621,5\,\text{kB} \approx 14,27\,\text{MB}$.

Aufgabe 3:

a) Im Skript werden die folgenden Prozeßzustände unterschieden:

erzeugt: Der Prozeß ist gerade neu entstanden.

rechnend: Der Prozessor ist dem Prozeß zugeteilt und arbeitet die Anweisungen des zugehörigen Programms ab.

bereit: Der Prozeß ist rechenbereit, hat aber den Prozessor nicht zugeteilt bekommen. Er wartet nur darauf, daß der Prozessor für ihn frei wird.

blockiert: Der Prozeß wartet auf irgendein Ereignis (z. B. Benutzereingabe).

beendet: Das Ende der Ausführung des zugehörigen Programms ist erreicht.

Abbildung 1 zeigt die möglichen Zustandsübergänge.

1. Der Benutzer hat einen neuen Prozeß gestartet.
2. Dem Prozeß wurden vom Betriebssystem alle benötigten Resourcen zugeteilt.
3. Der Prozeß erhält den Prozessor und wird ausgeführt.

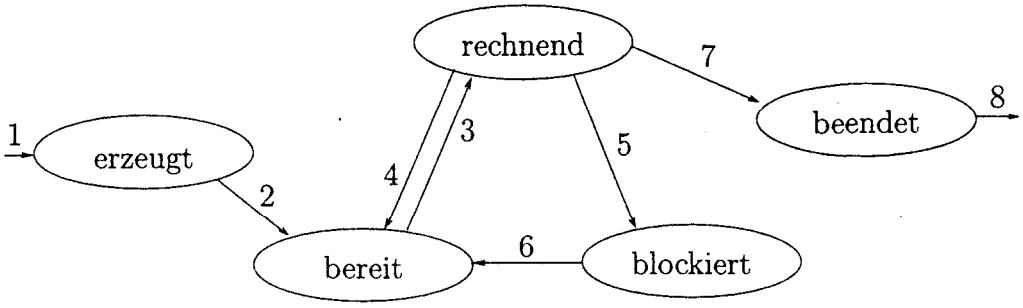

Abbildung 1: Zustandsübergänge

4. Der Prozeß gibt den Prozessor ab, bzw. der Prozessor wird dem Prozeß entzogen.
5. Der Prozeß muß auf ein Ereignis warten.
6. Das Ereignis auf das der Prozeß wartet ist eingetreten, der Prozeß könnte weiter rechnen.
7. Der Prozeß ist am Ende des Programms angekommen.

b) Prozesse haben ihre komplett eigene Umgebung, die sie mit keinem anderen Prozeß teilen. Zu dieser Umgebung gehören:

- Speicherbereich mit Programmsegment, Stecksegment und Datensegment.
- Registersatz inkl. Programmzähler.
- geöffnete Dateien.

Zugriffe aus einem Prozeß in die Umgebung eines anderen Prozesses können und werden vom Betriebssystem verhindert.

Threads können ihre Umgebung mit anderen threads teilen. Sie benutzen dann einen gemeinsamen Speicherbereich und unterscheiden sich nur durch unterschiedliche Stacks und Register. Zugriffe aus einem thread in die Umgebung eines anderen threads können nicht mehr vom Betriebssystem erkannt werden (außer die threads gehören zu unterschiedlichen Prozessen).

c) Im Kurstext wurden

- einfachere parallelisierung von Programmen auf Mehrprozessorrechnern,
- Gerätetreiber für langsame Geräte und
- verteilte (Client/Server) Systeme

vorgestellt.

Aufgabe 4:

a) Wenn ein Prozeß während seiner Ausführung auf eine logische Adresse zugreifen will, deren zugehörige physische Adresse zu einer Seite gehört die gerade nicht eingelagert ist, spricht man von einem page fault. In so einem Fall wird eine exception (Ausnahmebehandlung) ausgelöst. Der Prozeß wird unterbrochen, und das Betriebssystem übernimmt die Kontrolle. Das Betriebssystem sorgt dann dafür, da5 die fehlende Seite in den Hauptspeicher geladen wird. Bei Rechnern mit DMA kann gleichzeitig ein anderer Prozeß in den Zustand rechnend übergehen. Wenn die Seite geladen ist, wird der auslösende Prozeß wieder in den Zustand bereit versetzt und kann die CPU wieder zugeteilt bekommen.

b) Unter trashing versteht man den Systemzustand, der eintritt, wenn überwiegend Seitenfehler ausgelöst werden. Das System ist dann überlastet, und das Betriebssystem sollte einige Prozesse stillegen, d. h. alle Seitenrahmen dieses Prozesses für andere Prozesse zur Verfügung stellen. Dadurch sollten die anderen Prozesse weniger Seitenfehler erzeugen, und die CPU-Auslastung sollte steigen.

Aufgabe 5:

Die Idee bei der Lösung dieser Aufgabe besteht darin, zwei weitere binäre Semaphore zu benutzen. Eins der Semaphore dient dazu, den Zugriff auf die Zählervariable zu synchronisieren. Das zweite Semaphor braucht man, damit dort Prozesse blockieren können, die das allgemeine Semaphor noch nicht passieren dürfen.

```
TYPE  binSemaphor = (* egal *);
      allSemaphor = RECORD
                      zaehler: INTEGER;
                      mutex,
                      warten  : binSemaphor;
                    END; (* RECORD *)

PROCEDURE initn(     n    : INTEGER;
                VAR sem :  allSemaphor ; )
BEGIN
   sem.zaehler := n;
   init1(1, sem.mutex);
   init1(0, sem.warten);
END;  (* initn *)


PROCEDURE Pn( VAR sem : allSemaphor );
BEGIN
   P1(sem.mutex); (* exklusiven Zugriff auf Zaehler sichern *)
   sem.zaehler  := sem.zaehler - 1;
   if  sem.zaehler < 0
   then V1(sem.mutex); (* schon n Prozesse im kritischen Abschnitt,
                          aber man muss anderen Prozessen erlauben,
                          den Zaehler wieder zu erhoehen *)
        P1(sem.warten);(* blockieren, bis jemand den Zaehler
                          wieder erhoeht *)
   else V1(sem.mutex);(* Zaehler wieder freigeben *)
   endif;
END; (* Pn *)


PROCEDURE Vn( VAR sem : allSemaphor );
BEGIN
   P1(sem.mutex); (* exklusiven Zugriff auf Zaehler sichern *)
   sem.zaehler := sem.zaehler + 1;
   if sem.zaehler <= 0
   then V1(sem.warten); (* blockierten Prozeß wieder aufwecken *)
```

```
   endif;
   V1(sem.mutex); (* Zaehler wieder freigeben *)
END; (* V *)
```

Aufgabe 6:

Benutzerverwaltung: Das System führt eine Liste der *Personen*, die das System benutzen dürfen; diese Personen werden als Benutzer bezeichnet. Für jeden registrierten Benutzer werden alle Angaben gespeichert, die für die systeminterne Verwaltung oder für den Betrieb des Rechners. erforderlich sind. Zu diesen Daten gehören beispielsweise der Benutzername, ein interner Benutzeridentifizierer (Benutzer-Nummer), Authentisierungsdaten und das Benutzerprofil. Diese Daten werden fast alle durch den Systemadministrator eingetragen; manche (unkritische) Daten können vom Benutzer direkt geändert werden oder werden als Seiteneffekt anderer Operationen geändert (z.B. werden die Authentisierungsdaten geändert, wenn der Benutzer sein Paßwort ändert).

Identifikation: Jede Person, die das System benutzen will, muß sich zunächst als registrierter Benutzer zu erkennen geben. Identifikation und Authentisierung werden i. a. in einer sogenannten login-Prozedur zusammengefaßt, die ein Benutzer erfolgreich durchlaufen muß, wenn er das System benutzen will. Die erste Angabe, die der Benutzer im Rahmen der login-Prozedur macht, ist der Benutzername (Identifikation). Das Betriebssystem überprüft anhand der gespeicherten Benutzerdaten, ob es sich um einen registrierten und zugelassenen Benutzer handelt.

Authentisierung: Nach der Identifikation muß die Person durch weitere Angaben beweisen? daß sie wirklich der im Rahmen der Identifikation angegebene Bennutzer ist. Benutzernamen sind in der Regel nicht geheim; sie stehen beispielsweise oft auf Deckblättern von Papier-Ausgaben. Der zweite Schritt innerhalb der login-Prozedur (Authentisierung) besteht daher in der Angabe von Beweisen durch den Benutzer dafür, daß er auch wirklich der angegebene Benutzer ist. Das Betriebssystem überprüft diese Angaben wieder anhand der gespeicherten Benutzerdaten. Oft wird die Eingabe eines Paßworts als Beweis verlangt.

Aufgabe 7:

Das Schließen und ggf. das Neueröffnen der ererbten Standard-E/A-Geräte durch das aufgerufene Programm hat keinen Einfluß auf das aufrufende Programm bezüglich dessen Standard-E/A-Geräte. In diesem Fall werden nur prozeßlokale Verweise auf Dateikontrollblöcke verändert.

Falls das aufrufende Programm jedoch ein spezielles Verhalten des aufgerufenen Programms bezüglich dessen Standard-E/A-Geräte voraussetzt, kann es nicht erkennen, ob sich das aufgerufene erwartungsgemäß verhält. Zum Beispiel könnte das aufrufende Programm eine spezielle Ausgabeumlenkung für das aufgerufene Programm installieren, welche dieses unerwarteterweise wieder aufhebt.