Georg-Simon-Ohm Fachhochschule **5. Semester Informatik**

Betriebssysteme **Schriftliche Prüfung WS 97/98**

Prüfer: Prof. Dr. Kern / Prof. Dr. Wienkop

Prüfungstermin: 28.1.98, 11 Uhr Dauer: 90 Minuten

Hilfsmittel: Vorlesungsmitschrift, Vorlesungsumdrucke und Taschenrechner

6 Aufgabenblätter

---

1. Aufgabe

In einem Unix-Betriebssystem sei an der Basisschnittstelle zum Dateiverwaltungssystem der Systemaufruf

seek(dateiidentifikator, startposition, distanz)

gegeben, mit dessen Hilfe byteweise die nächste Zugriffsposition in der durch *dateiidentifikator* spezifizierten Datei bestimmt wird, und zwar im (in Bytes gemessenen) Abstand *distanz* von der *startposition*. *startposition* kann dabei die mit Hilfe numerischer Konstanten codierten Werte „Dateianfang", „Dateiende" und „aktuelle Position" annehmen. Die drei Aufrufparameter seien durch ganzzahlige Variablen geeigneter Länge gegeben; *distanz* kann auch negativ sein.

Welche Fehlersituationen muß dieser Systemaufruf zurückmelden können? Denken Sie dabei u.a. an Probleme, die sich mit der Parameterversorgung, dem Dateityp und im Bereich des Dateiverwaltungssystems ergeben können, und listen Sie diese detailliert auf.

Lösung:

Kategorie Parameterversorgung:

- Wertebereich von startposition überschritten → „Parameterfehler"
- Dateianfang&negative Distanz ➔ „Positionierung vor Dateianfang"
- Dateiende & positive Distanz ➔ KEIN FEHLER, MSDOS: Lücke wird mit Daten gefüllt; Unix → dazwischenliegende Datenbereiche bleiben noch unallokiert

Kategorie Dateisystem:

- Dateiidentifikator ist keine Datei, sondern (unter Unix!) z.B. ein Zeichengerät, bei dem keine Positionierung erlaubt ist (etwa Drucker) ➔ „Positionierung nicht erlaubt"
- Dateiidentifier veraltet (z.B. Datei wurde bereits geschlossen) ➔ „File not open"

2. Aufgabe

a) Schreiben Sie eine Shell-Prozedur, die bei Aufruf je nach der jeweiligen Tageszeit die folgenden Kommentare ausgibt:

von 7:00:00 Uhr	–	8:59:59 Uhr	Guten Morgen
von 11:00:00 Uhr	–	12:59:59 Uhr	Mahlzeit
von 16:00:00 Uhr	–	18:00:00 Uhr	Feierabend

Zu allen anderen Zeiten soll nichts ausgegeben werden

Hinweis: – Bei der 18:00:00 Uhr-Angabe handelt es sich nicht um einen Tippfehler
– Verwenden Sie nur die in der Vorlesung behandelten Befehle!

Lösung:

```
set `date`;
case $4 in
        [78])               echo „Guten Morgen“;;
        1[12])              echo „Mahlzeit“;;
        1[67] | 18:00:00)   echo „Feierabend“;;
esac
```

b) Erstellen Sie ein Shell-Programm, das für die als Aufrufparameter übergebenen Namen im aktuellen Katalog der Reihe nach

- die Anzahl sämtlicher Dateien,
- die Anzahl sämtlicher ausführbarer Dateien,
- die Anzahl sämtlicher Unterkataloge

angibt! Beachten Sie dabei, daß das einschlägige Shell-Kommando auch Unterkataloge, bei denen in den Rechten "x" angegeben ist (hier: "search"-Recht), als "ausführbar" erkennt. Sie sollen aber nicht bei den ausführbaren Dateien mitgezählt werden!

Lösung:

```
# Shell-Programme WS 97/98
# 17.1.98
# Teil b), funktioniert so.

dateien=0
xdateien=0
kat=0
for i
do
   if test -f $i
   then
      dateien=`expr $dateien + 1`
   fi
   if test \( -x $i \) -a \( ! -d $i \)
   then
      xdateien=`expr $xdateien + 1`
   fi
   if test -d $i
   then
      kat=`expr $kat + 1`
   fi
done
echo "$dateien Dateien, davon $xdateien ausfuehrbare Dateien, \
und $kat Unterkataloge vorhanden."
```

3. Aufgabe

Wenn man unter MSDOS von einer Festplatte z.B. 100 Dateien jeweils der Größe 2 KByte auf eine Diskette kopiert, dann dauert dieser Vorgang spürbar länger als in der umgekehrten Richtung (Diskette → Festplatte).

a) Begründen Sie, warum dies so ist

b) Verwendet man ein Programm, welches mehrere Dateien zu einer Großen zusammenfaßt (auch ohne Komprimierung), so ergibt sich beim Kopieren auf die Diskette eine deutliche Beschleunigung des Kopiervorgangs. Begründen Sie bitte auch diesen Sachverhalt.

Lösung:

a) Beim Kopieren der vielen Dateien auf die Diskette muß der Schreiblesekopf nach dem Schreiben jeder Datei wieder zur Directoryspur und zu beiden FAT-Bereichen auf der Diskette zurückkehren und die gemachten Änderungen an diesen drei Stellen nachtragen. Da Kopfbewegungen des Diskettenlaufwerks deutlich langsamer sind, als die eines Festplattenlaufwerks, ergeben sich spürbare Verzögerungen, insbesondere hervorgerufen bei den letzten geschriebenen Dateien.

b) Beim Zusammenfassen vieler Dateien zu einem Archiv entfällt natürlich das einzelne Kopfpositionieren nach jedem Dateischreiben, so daß eine Vielzahl von Spurwechseln entfällt.

4. Aufgabe

Gegeben sei folgende Prozeßmenge

Prozeß	Ankunftszeit	Rechenzeitbedarf
A	0	5
B	4	3
C	7	2
D	9	2

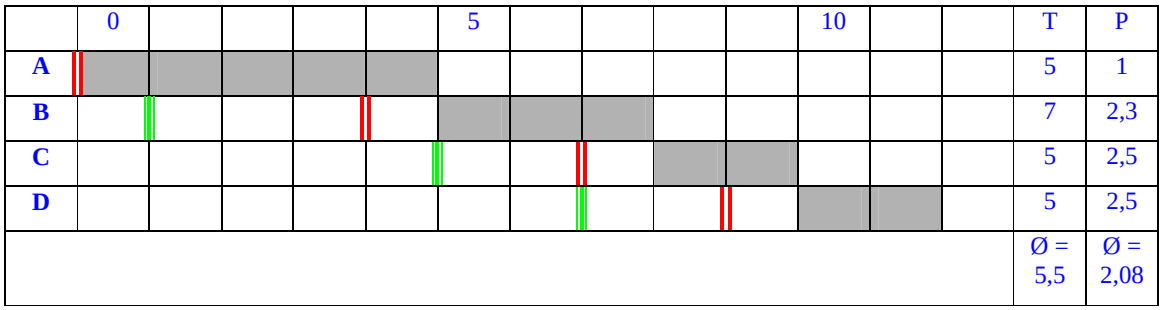
Bei Verwendung des Ablaufplanungsverfahrens FCFS ergibt sich natürlich als Beendigungsreihenfolge der Prozesse A-B-C-D. Geben Sie eine Prozeßmenge an, in der die Ankunftszeiten der Prozesse B, C und D gegenüber obiger Aufstellung **minimal** verschoben sind, so daß die Prozesse je nach verwendetem Ablaufplanungsverfahren **eindeutig** in folgenden Reihenfolgen **beendet** werden:

Bei FCFS: A-B-C-D
Bei SPN: A-C-D-B
Bei PSPN: B-C-D-A

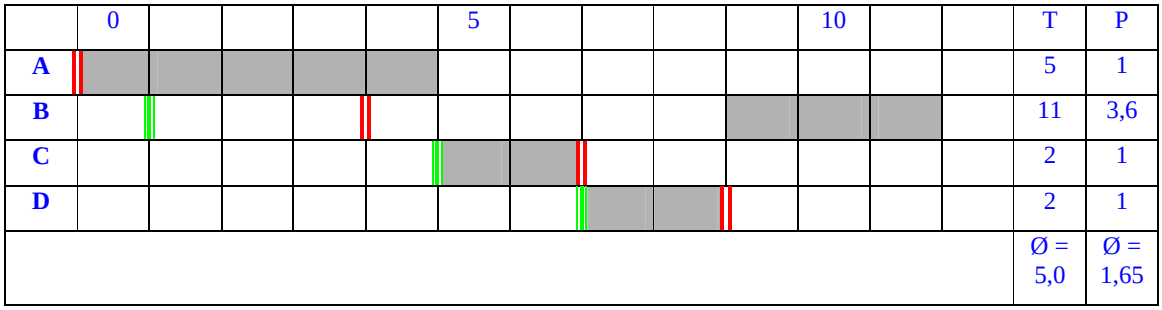
Geben Sie jeweils das Ablaufdiagramm an und berechnen Sie die Durchschnittswerte für T und P.

Lösung:

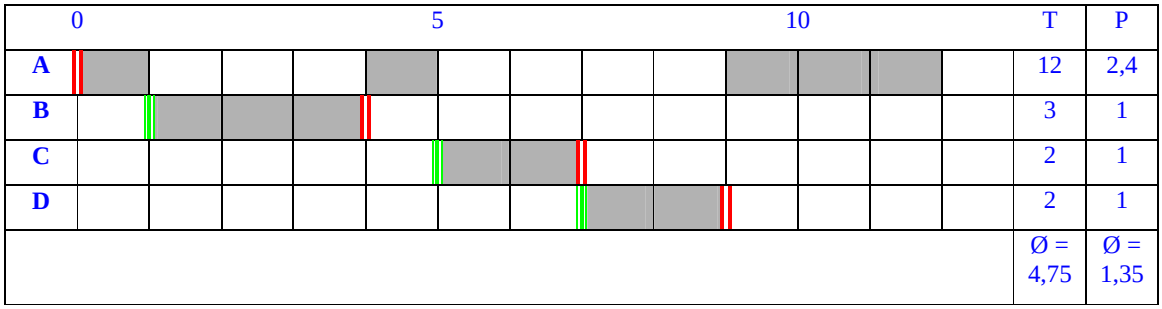
FCFS



SPN



PSPN

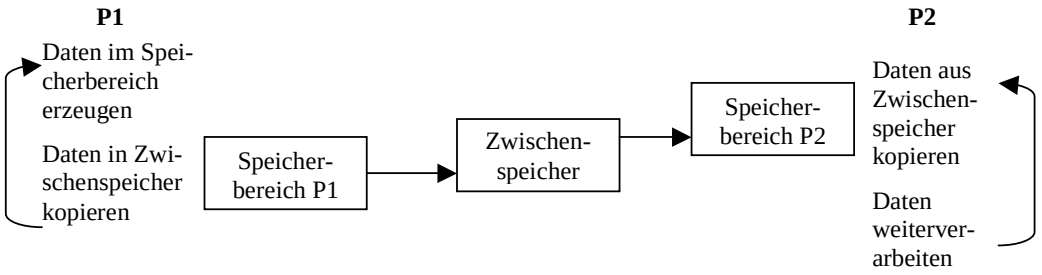


5. Aufgabe

Gegeben sei ein Prozeßsystem mit den zwei Prozessen P1 und P2. P1 produziert in einer Schleife immer wieder Datenpakete, welche P2 weiterverarbeiten soll. Es muß jedoch sichergestellt werden, daß P1 immer kontinuierlich arbeiten kann – d.h. daß P1 nie länger als für einen kurzen Zeitraum blockiert wird.

P2 läuft i.a. langsamer ab als P1. Daher werden nicht alle Datenpakete weiterverarbeitet, sondern nur solche, die gerade zu einem Zeitpunkt fertig werden, wenn P2 wieder aufnahmebereit ist (jeweils ein Schnappschuß)! Wichtig ist dabei, daß Datenpakete jeweils entweder vollständig und konsistent von P1 nach P2 übergeben werden oder gar nicht!

Damit der langsamere Prozeß P2 nicht den schnellen Prozeß P1 „ausbremst“, werden die Daten in jedem Zyklus von P1 nach dem Erzeugen in einen Zwischenpuffer kopiert, aus dem P2 – sofern aufnahmebereit – dann lesen kann! Das folgende Bild zeigt diese Anordnung:



Der Aufwand für das Kopieren der Daten ist bei beiden Prozessen gegenüber dem Erzeugen/Verarbeiten der Daten niedrig. Speicherschutzaspekte beim Kopieren können hier unberücksichtigt bleiben.

Skizzieren Sie in (PASCAL oder C-ähnlichem Pseudocode) diese Anordnung und achten Sie unter Verwendung von binären Semaphoren auf Einhaltung der folgenden Bedingungen:

- konsistentes Kopieren der Daten in den / aus dem Zwischenspeicher
- zusätzlich: P2 soll mit seinem Kopieren/Verarbeiten nicht eher beginnen als sicher neue, von P1 erzeugte Daten konsistent im Zwischenspeicher vorliegen. Aktives Warten soll dabei vermieden werden.

Hinweis:
Bei binären Semaphoren bewirken auch mehrere V()-Aufrufe nur ein Setzen des Semaphorzählers auf 1!

Lösung:

```
bin_semaphore   Speicher = 1,
                Warten = 0;
P1()
{
        while (1)
        {
                erzeuge_Daten_im Speicherbereich();
                P(Speicher);
                kopiere_Daten_zum_Zwischenspeicher();
                V(Speicher);
                V(Warten);
        }
}

P2()
{
        while (1)
        {
                P(warten);
                P(Speicher);
                kopiere_Daten_aus_dem_Zwischenspeicher();
                V(Speicher);
                Verarbeite_Daten();
        }
}
```

6. Aufgabe

Gegeben sei folgendes Prozeßsystem mit den angegebenen Speicheranforderungen/-freigaben

Prozeß	Ankunftszeit	Rechenzeitbedarf	Speicheranforderungen /-freigaben
A	0	5	(1) +1: Forderung 100 (2) +2: Forderung 50 (3) +3: Freigabe von (1) (4) +5: Freigabe von (2)
B	1	4	(1) +1: Forderung 200 (2) +3: Freigabe von (1)
C	2	4	(1) +1: Forderung 150 (2) +2: Forderung 100 (3) +3: Freigabe von (1) (4) +4: Freigabe von (2)

(Bem.: »+1: Forderung 100« bedeutet beispielsweise, daß dieser Prozeß nach tatsächlicher Rechenzeit von einer Einheit 100 Bytes Speicher anfordert, jedoch ganz kurz **vor** einem Zeitscheibenende!)

Die Freispeicherliste habe folgenden Anfangszustand:

Adresse	100	700
Größe	200	250

Als Schedulingstrategie wird Round Robin (RR) mit einem Zeitquantum von 2 verwendet. Sollte eine Speicheranforderung nicht erfüllbar sein, so wird der betreffende Prozeß vom Betriebssystem solange blockiert, bis die Freispeicherverwaltung einen entsprechend großen Block zur Verfügung stellen kann. Der nächste Prozeß auf der Bereitliste kommt dadurch mit einem vollen Zeitquantum an die Reihe. Nach Zuteilung eines entsprechend großen Speicherblocks wird dieser Prozeß als erster in die Bereitliste eingeordnet. Ein evtl. noch rechnender Prozeß wird hierdurch jedoch nicht verdrängt. Kann die Anforderung erfüllt werden, so rechnet der Prozeß hingegen ohne Blockierung weiter.

Zeigen Sie den Ablauf der drei Prozesse mit ihren Speicheranforderungen und –freigaben bei Verwendung des Speicherzuteilungsverfahrens „First-Fit“. Geben Sie hierzu eine Auflistung der Speicheraktionen im zeitlichen Ablauf, das Ablaufdiagramm sowie die Parameter T und P und deren Durchschnitte an.

Bei der Zuteilung von Speicher gehen Sie bitte wie folgt vor: ein Speicherbereich mit Adresse a und Länge L wird bei angeforderter Größe x in einen Restblock (a, L-x) und einen zugeteilten Block (a+L-x, x) aufgeteilt.

Lösung: (First fit)

0					5					10						T	P
A																12	2,4
B																8	2
C																11	2,75
																Ø = 10,3	Ø = 2,38

100/200		**700/250**	**Aktion**
100/100			A100 → 200/100
100/50			A50 → 150/50
		700/50	B200 → 750/200
			C150 → Block
100/50	200/100	700/50	A: Freigabe (1)
		700/250	B: Freigabe (1)
		700/100	C150 → 800/150
	200/0		C100 → 200/100
		700/250	C: Freigabe (1)
100/100			A: Freigabe (2)
100/200			C: Freigabe (2)

VIEL ERFOLG!