# Testat Systeme I

## Besprechung

Stefan Disch

Lehrstuhl für Betriebssysteme
Institut für Informatik
Universität Freiburg

WS 2005/2006

# Aufgabe 1 Teil a)

Für diese Aufgabe wird angenommen, daß Sie auf einem Dateisystem arbeiten, dass das in der Vorlesung vorgestellte I-Node-Konzept verwendet.

Sie besitzen zwei Verzeichnisse A und B. Angenommen, Sie erstellen in A ein Unterverzeichnis C und im Verzeichnis B einen Hardlink namens D auf C. (Das ist nur theoretisch möglich, da praktisch jedes Betriebssystem Hardlinks auf Verzeichnisse verbietet; den Grund werden Sie hier kennenlernen)

Wenn Sie ausgehend vom Verzeichnis B in D wechseln und dann wieder eine Verzeichnisebene nach oben gehen, in welchem Verzeichnis sind Sie dann? Begründen Sie kurz Ihre Antwort.

Beachten Sie hierbei, daß ein Verzeichnis lediglich eine spezielle Datei ist, die in Tabellenform alle enthaltenen Dateien/Verzeichnisse zusammen mit der entsprechenden I-Node-Nummer speichert; zusätzlich wird die I-Node-Nummer des Vaterverzeichnisses „..“ und des aktuellen Verzeichnisses „.“ gespeichert.

Lösung:

Die Einträge von C und D zeigen auf den gleichen inode, der auf einen Sektor zeigt, in dem unter .. die inode-Nummer von A steht. Daher landet man in A.

# Aufgabe 1 Teil b)

Angenommen, Sie löschen C und dann A; danach wechseln sie in Verzeichnis B und von dort aus in D. Welches Problem tritt auf, wenn Sie nun versuchen, eine Verzeichnisebene nach oben zu gehen?

Lösung:

Wenn C gelöscht wird, existiert D weiter, zeigt aber nach wie vor auf A als Vaterverzeichnis, selbst wenn A gelöscht wurde. Wenn man also von D aus versucht, in .. zu wechseln, müsste man in A landen, was aber nicht mehr existiert.

# Aufgabe 1 Teil c)

Sie haben eine Datei Z, die 1,5 KB an Daten enthält. Weiterhin haben Sie 2 symbolische Links und 3 Hardlinks, die auf Z verweisen. Wieviel Plattenplatz benötigen Z und seine Links, wenn die Blockgröße dieses Dateisystems 1024 Byte beträgt? Beachten Sie auch den von den I-Nodes verbrauchten Speicherplatz, vernachlässigen Sie aber die Einträge in den Verzeichnissen. Begründen Sie Ihr Ergebnis.

## Lösung

| | |
|---|---|
| Datei | 2 Blöcke für Daten und einen für den I-Node: 3KB |
| Hardlinks | keinen Platzverbrauch (nur Verzeichniseintrag): 0KB |
| Softlinks | 2·(1 I-Node + 1 Datenblock): 4 KB |

Daraus ergibt sich ein Gesamtbedarf von 7 KB.

Hinweis: Bei einer Blockgröße von 1 KB benötigen 1,5 KB an Daten immer 2 KB (2 Blöcke) auf der Festplatte! Ein Block kann nicht von mehreren Dateien genutzt werden.

# Aufgabe 1 Teil d)

Warum kann man über Dateisystemgrenzen hinweg nur symbolische Links und keine Hardlinks verwenden?

Lösung:

- Wenn das möglich wäre, müsste man zusätzlich zum I-Node abspeichern, in welchem Dateisystem/in welcher Partition das Ziel liegt. Ansonsten ist keine eindeutige Zuordnung möglich.
- Bei einem umount würde das Ziel eines Hardlinks evtl. verloren gehen, es ist aber nicht vorgesehen, dass der zugehörige I-Node nicht erreichbar ist (im Gegensatz zu Softlinks).

# Aufgabe 2 Teil a)

In der Vorlesung wurden drei Konzepte der Realisierung von Dateien in einem Dateisystem vorgestellt:

- Zusammenhängende Belegung
- Verkettete Listen
- I-Nodes

Skizzieren Sie die prinzipiellen Arbeitsweisen der Konzepte „Zusammenhängende Belegung“ und „Verkettete Listen“.

Lösung: siehe Skript.

# Aufgabe 2 Teil b)

Skizzieren Sie die prinzipielle Arbeitsweise des I-Node-Konzeptes und erläutern Sie kurz die Struktur.

Lösung: siehe Skript.

# Aufgabe 2 Teil c)

Erläutern Sie jeweils kurz die Vor- und Nachteile jedes Konzeptes. Betrachten Sie hierbei insbesondere die Effizenz beim wahlfreien Zugriff, den Verbrauch an Hauptspeicher und die Probleme beim Löschen und Vergrößern von Dateien.

Lösung: siehe Skript.

# Aufgabe 2 Teil d)

In welchen Anwendungsgebieten werden die einzelnen Konzepte verwendet?

Lösung: siehe Skript.

# Aufgabe 2 Teil a)

Erläutern Sie den Begriff *Kontextwechsel*. Welche Teile des Betriebssystems sind daran beteiligt? Wie unterscheidet sich dies bei *präemptiven* und *nicht präemptiven Multitasking*?

Lösung: siehe Skript.

# Aufgabe 2 Teil b)

Es seien folgende parallel ausgeführte Prozesse gegeben:

```
/* P1 */
{
  A;
  B;
}
```

```
/* P2 */
{
  C;
  D;
}
```

Die Anweisungen `A`, `B`, `C`, `D` sind atomar, d.h. wenn sie zur Ausführung kommen, werden sie immer komplett abgearbeitet. Geben Sie alle möglichen Abarbeitungsfolgen der Anweisungen an, die durch Scheduling erzeugt werden können. Gehen Sie davon aus, dass beide Prozesse gleichzeitig gestartet wurden und es keine Abhängigkeiten zwischen den Anweisungen gibt.

# Aufgabe 2 Teil c)

In der Vorlesung haben Sie mehrere Software-Lösungsversuche zum wechselseitigen Ausschluß kennengelernt. Der erste betrachtete Vorschlag lautete wie folgt:

```
/* Prozess 0 */
wiederhole {
  solange (turn ≠ 0) tue nichts;
  /* kritischer Abschnitt */
  turn := 1;
  /* nichtkrit. Abschnitt */
}
```

```
/* Prozess 1 */
wiederhole {
  solange (turn ≠ 1) tue nichts;
  /* kritischer Abschnitt */
  turn := 0;
  /* nichtkrit. Abschnitt */
}
```

Beweisen Sie, daß dieses Verfahren den wechselseitigen Ausschluß garantiert. Nehmen Sie hierbei an, daß in den kritischen und nichtkritischen Abschnitten keine Zuweisungen an `turn` vorkommen und daß die benötigte Rechenzeit in den kritischen und nichtkritischen Abschnitten durch eine Konstante zeitlich nach oben begrenzt ist.

Lösung: siehe Skript.

# Aufgabe 2 Teil d)

Es gibt fünf Forderungen an den wechselseitigen Ausschluß:

1. Höchstens ein Prozess ist im kritischen Abschnitt.
2. Jeder Prozess hält sich nur endliche Zeit im kritischen Abschnitt auf.
3. Wenn ein Prozess in den kritischen Abschnitt will, so muss er nur endliche Zeit darauf warten.
4. Wenn kein Prozess im kritischen Abschnitt ist, so wird ein interessierter Prozess ohne Verzögerung akzeptiert.
5. Alles funktioniert unabhängig von der relativen Ausführungsgeschwindigkeit der Prozesse.

Welche dieser Forderungen erfüllt obiger Vorschlag und welche nicht? Begründen Sie Ihre Aussagen.

Lösung: siehe Skript.

# Aufgabe 4 Teil a)

Welches Problem taucht bei *festen Prioritäten* sowohl bei *Petersons Algorithmus* als auch bei der *Hardwarelösung* auf? Geben Sie ein kleines Beispiel an.

Lösung:

Nachteil: aktives Warten

1. Prozess 0 hat hohe Priorität, ist aber blockiert.
2. Prozess 1 betritt kritischen Abschnitt.
3. Prozess 0 wird wieder rechenbereit und erhält Prozessor wegen hoher Priorität.
4. Prozess 0 möchte in den kritischen Abschnitt und betritt Schleife für *aktives Warten*. $\Rightarrow$**Deadlock**

# Aufgabe 4 Teil b)

Was ist der Unterschied zwischen einem *schwachen* und *starken* Semaphor?

Lösung:

**schwach** Reihenfolge der Abarbeitung der Anfragen ist nicht festgelegt.

**stark** Reihenfolge des Eintreffens in der Warteschlange wird berücksichtigt.

# Aufgabe 4 Teil c)

Geben Sie eine Lösung für das Problem der *speisenden Philosophen* an. Sie dürfen Semaphoren verwenden. Benutzen Sie das unten angeführte Programmgerüst und kommentieren Sie Ihren Code. **Hinweis:** Es reicht völlig aus, wenn immer nur ein Philosoph essen darf!
Es stehen Ihnen folgende Methoden zur Verfügung:

- `nimmRechteGabel()`
- `nimmLinkeGabel()`
- `legeRechteGabelWeg()`
- `legeLinkeGabelWeg()`

# Aufgabe 5 Teil a)

Zwei Prozesse wollen auf vier Ressourcen `A`, `B`, `C` und `D` zugreifen. Folgende Tabelle zeigt, in welcher Reihenfolge die Prozesse Ressourcen anfragen bzw. freigeben; Befehle, die zwischen den Anforderungen und Freigaben stehen, vernachlässigen wir hier.

| /* Prozess 0 */ | /* Prozess 1 */ |
| --- | --- |
| 1: Anforderung A | 1: Anforderung A |
| 2: Anforderung B | 2: Anforderung D |
| 3: Anforderung C | 3: Anforderung B |
| 4: Freigabe B | 4: Freigabe A |
| 5: Freigabe A | 5: Anforderung C |
| 6: Anforderung D | 6: Freigabe B |
| 7: Freigabe D | 7: Freigabe C |
| 8: Freigabe C | 8: Freigabe D |

Zeichnen Sie in untenstehende Grafik die Bereiche ein, in der beide Prozesse auf eine Ressource zugreifen würden. Die horizontale Achse repräsentiert den Programmfortschritt von Prozess 0 und die vertikale Achse repräsentiert den Programmfortschritt von Prozess 1. Die mit einer Nummer versehenen horizontalen und vertikalen Linien sind die Zeitpunkte in der Programmausführung, an der die entsprechend numerierte Zeile des Prozesses ausgeführt wird.

# Aufgabe 5 Teil b)

Kann es in diesem Szenario zu einem Deadlock kommen? Wenn ja, markieren Sie den Deadlock und zeichnen Sie den Bereich ein, in dem der Deadlock unvermeidlich ist. Begründen Sie kurz Ihre Aussagen.