TI III WS 2013, Fr. 12-14

Lösung Übungsblatt 5

Christoph van Heteren-Frese (Matr.-Nr.: 4465677), Julien Stengel
Tutor: Ruhland, eingereicht am 12. Dezember 2013

Aufgabe 2

a)

Gesichtspunkt	Paging	Segmentierung
Müssen sich die Programmierer der angewandten Technik bewusst sein?	Nein	Ja
Wieviele lineare Adressräume werden benutzt?	1	Viele
Muss die komplette Seite bzw. das komplette Segment geladen werden?	Ja	Ja
Sind Programme beliebiger Größe ladbar?	Ja	Nein
Können Prozeduren und Daten auseinandergehalten werden und getrennt geschützt werden?	Nein	Ja
Können Tabellen deren Größe sich verändert leicht angepasst werden?	Nein	Ja
Ist die gemeinsame Benutzung von Prozeduren von verschiedenen Prozessen [geschickt] möglich?	Nein	Ja
Warum wurde diese Technik eingeführt?	Um größeren Adressraum zu erhalten ohne den physikalischen Speicher vergrößern zu müssen.	Um die Trennung zwischen Programm und Daten in unabhängigen Adressräumen zu ermöglichen und um gemeinsame Benutzung von Prozeduren und den Schutz von einzelnen Datenbereichen zu erlauben.

Abbildung 1: Vergleich von Paging und Segmentierung aus [**?**]

b)

c)

Aufgabe 4

a)

Der Operator `!=` (*not equal to*, Prioritätsstufe 9) bindet stärker als `=` (*direct assignment*, Prioritätsstufe 16). Daher müssen wie folgt Klammern gesetzt werden:

```
while ( (c = getchar ()) != EOF )
        putchar ( c );
```

b)

Es wird ein Zahlenpaar ausgegeben. Die zweite Zahl ist dabei die Quadratzahl des direkten Vorgängers der erste Zahl. Diese Ausgabe entspricht warscheinlich nicht der Intention des Programierers.

Begründung: Erscheint der Opperator `++` vor dem Operand, wird dessen Wert zunächst inkrementiert und erst dann in dem Ausdruck benutzt. Steht `++` hinter dem Operand, wird sein aktueller Wert erst im Ausdruck benutzt und anschließend inkrementiert.

c)

1. Der Wert von `len` wird inkrementiert.
   `->` bindet stärker (Prioritätsstufe 2) als `++` (Prioritätsstufe 3). Daher wird zuerst `p->len` ausgewertet (`p->len` ist äquivalent mit `(*p).len`) und somit auf die Variable `len` der Struktur `p` zugegriffen. Diese wird anschließend inkrementiert.

2. Hier wird erst der Zeiger `p` inkrementiert und dann auf das Element len der Struktur zugegriffen, auf den der nächste Zeiger zeigt.

3.

4. Der Wert von `str` wird inkrementiert.

5.

d)

e)

TI3 Übung 5

1.a)

First-Fit:

Eingehende Anforderung	Freier Speicherplatz 1	Freier Speicherplatz 2	Freier Speicherplatz 3
-	1024	512	2048
384	640	512	2048
640	0	512	2048
512	0	0	2048
2048	0	0	0

Rotating-First-Fit:

Eingehende Anforderung	Freier Speicherplatz 1	Freier Speicherplatz 2	Freier Speicherplatz 3
-	1024	512	2048
384	640	512	2048
640	0	512	2048
512	0	0	2048
2048	0	0	0

Best-Fit:

Eingehende Anforderung	Freier Speicherplatz 1	Freier Speicherplatz 2	Freier Speicherplatz 3
-	1024	512	2048
384	1024	128	2048
640	384	128	2048
512	384	128	1536
2048	384	128	1536

Die Anforderung 2048 kann in diesem Verfahren keinen genügend großen Speicher finden

Worst-Fit:

Eingehende Anforderung	Freier Speicherplatz 1	Freier Speicherplatz 2	Freier Speicherplatz 3
-	1024	512	2048
384	1024	512	1664
640	1024	512	1024
512	512	512	1024
2048	512	512	1024

Da bei gleich großen freien Speicherplätzen keinen speziellen Angaben gemacht wurden, wird zur Lösung der Aufgabe davon ausgegangen, dass der erste Speicher belegt wird. Die Anforderung 2048 kann in diesem Verfahren keinen genügend großen Speicher finden.

b)

First Fit:

Sobald First-Fit eine Möglichkeit hat die Anforderungen abzudecken, kann automatisch auch Rotating-First-Fit den Anforderungen gerecht werden, daher gibt es keine Spezifische Möglichkeit, die nur First-Fit lösen kann.

TI3 Übung 5

1.b)

Rotating-First-Fit:

Eingehende Anforderung	Freier Speicherplatz 1	Freier Speicherplatz 2	Freier Speicherplatz 3
-	2048	512	1024
2048	0	512	1024
640	0	512	384
384	0	512	0
512	0	0	0

Best-Fit:

Eingehende Anforderung	Freier Speicherplatz 1	Freier Speicherplatz 2	Freier Speicherplatz 3
-	1024	512	2048
2048	1024	512	0
640	384	512	0
384	0	512	0
512	0	0	0

Worst-Fit:

für Worst-Fit gibt es keine Möglichkeit, die die anderen Methoden nicht auch abdecken könnten.

3.a)

Die Anzahl der Speicherworte entsteht aus der Länge aller Segmente zusammengenommen. In diesem Fall ist die Länge in Speicherworten angegeben, daher ist die Anzahl der Speicherworte gleich der Gesamtlänge.

365+70+120+515+150 = 1220

Es stehen 1220 Speicherworte zur Verfügung.

b)

Die physikalische Adresse ergibt sich aus Basis + Offset, da jedoch kein Offset gegeben ist entspricht die physikalische Adresse der Basis.

Das kleinste Segment hat die Länge 70 und die dazugehörige Adresse ist 400.

das größte Segment hat die Länge 515 und die dazugehörige Adresse ist 1145.

c)

(erscheint viel zu leicht, jedoch ist dies der einzige logische Ansatz):

Die logische Adresse ergibt sich aus der Segmentnummer, dem Anfangspunkt des physikalischen Speichers und des Offsets. Da kein Offset gegeben, besteht die logische Adresse nur aus Segmentnummer und physikalischem Speicher. (Die 0 hilft dabei die Zahlen auf eine genormte Länge zu bringen)

1. 10762 2. 21145 3. 30146 4. 40485

Literatur

[] Andrew S Tanenbaum. *Moderne Betriebssysteme*. Hanser ; Prentice Hall Internat., Muenchen; Wien; London, 1994. ISBN 3446174729 9783446174726 0135178894 9780135178898.