Dynamisches Speichermanagement

Proseminar C-Programmierung - Grundlagen und Konzepte

Timo Minartz

timo.minartz@informatik.uni-hamburg.de

Wissenschaftliches Rechnen
Fachbereich Informatik
Universität Hamburg

20-05-2011

informatik
die zukunft

Motivation

- Dynamische Datenstrukturen erstellen
- Speicherbereiche zwischen Gültigkeitsbereichen teilen
  - Reduzierung von Kopiervorgängen

1 Speicherkonzept

2 Speicherallozierung

3 Beispiel: Liste

4 Fehlerquellen

5 Speicherverwaltung

informatik
die zukunft

1 Speicherkonzept

2 Speicherallozierung

3 Beispiel: Liste

4 Fehlerquellen

5 Speicherverwaltung

informatik
die zukunft

Ein Programm besteht aus 4 Speicherbereichen

- Code - Maschinencode des Programms
- Daten - Statische und globale Variablen
- Stack - Funktionsaufrufe und lokale Variablen
- Heap (Freispeicher) - Dynamisch reservierter Speicher

1 Speicherkonzept

2 Speicherallozierung

3 Beispiel: Liste

4 Fehlerquellen

5 Speicherverwaltung

informatik
die zukunft

Speicherallozierung

- Reservierung von Speicherplatz im Freispeicher
- Speicherplatz wird durch das Betriebssystem zur Verfügung gestellt
- In der Regel erst bei der ersten Benutzung wirklich alloziert
- Sollte nach Benutzung wieder freigegeben werden
- Wird bei Programmende automatisch freigegeben

malloc und *free*

Syntax

```
1 void *malloc(size_t size);
2 void free(void *ptr);
```

malloc

- Alloziert *size* Bytes und gibt einen Zeiger auf das erste Element zurück
- Die Speicherwerte bleiben unverändert
- Bei *size == 0* oder Fehler wird *NULL* zurückgegeben

Beispiel

Beispiel

```
1 int * p_int = (int*) malloc(3 * sizeof(int));
2 for(int i=0; i<3; ++i) {
3     p_int[i] = i;
4 }
5 free(p_int);
```

calloc und *realloc*

Syntax

```
1 void *calloc(size_t nmemb, size_t size);
2 void *realloc(void *ptr, size_t size);
```

calloc

- Alloziert Speicher für *nmemb* Elemente der Größe *size*
- Setzt den Speicher explizit auf Null

realloc

- Ändert die Größe des allozierten Speicher
- Potentiell werden Daten umkopiert

1 Speicherkonzept

2 Speicherallozierung

3 Beispiel: Liste

4 Fehlerquellen

5 Speicherverwaltung

informatik
die zukunft

Einfügen und Löschen von Knoten

```
1  void add(struct Node * last, const char * data)
2  {
3      struct Node * newNode = malloc(sizeof(struct Node));
4      newNode->data = strdup(data);
5      newNode->next = NULL;
6      newNode->prev = last;
7      last->next = newNode;
8  }
9
10 void delete(struct Node * node)
11 {
12     node->prev->next = node->next;
13     node->next->prev = node->prev;
14     free(node->data);
15     free(node);
16 }
```

informatik die zukunft

1 Speicherkonzept

2 Speicherallozierung

3 Beispiel: Liste

4 Fehlerquellen

5 Speicherverwaltung

informatik
die zukunft

Typische Fehler

- Laufzeitfehler
- Fernwirkung
- Reproduzierbarkeit
- Kausalitätsprinzip

1 Speicherkonzept

2 Speicherallozierung

3 Beispiel: Liste

4 Fehlerquellen

5 Speicherverwaltung

informatik
die zukunft

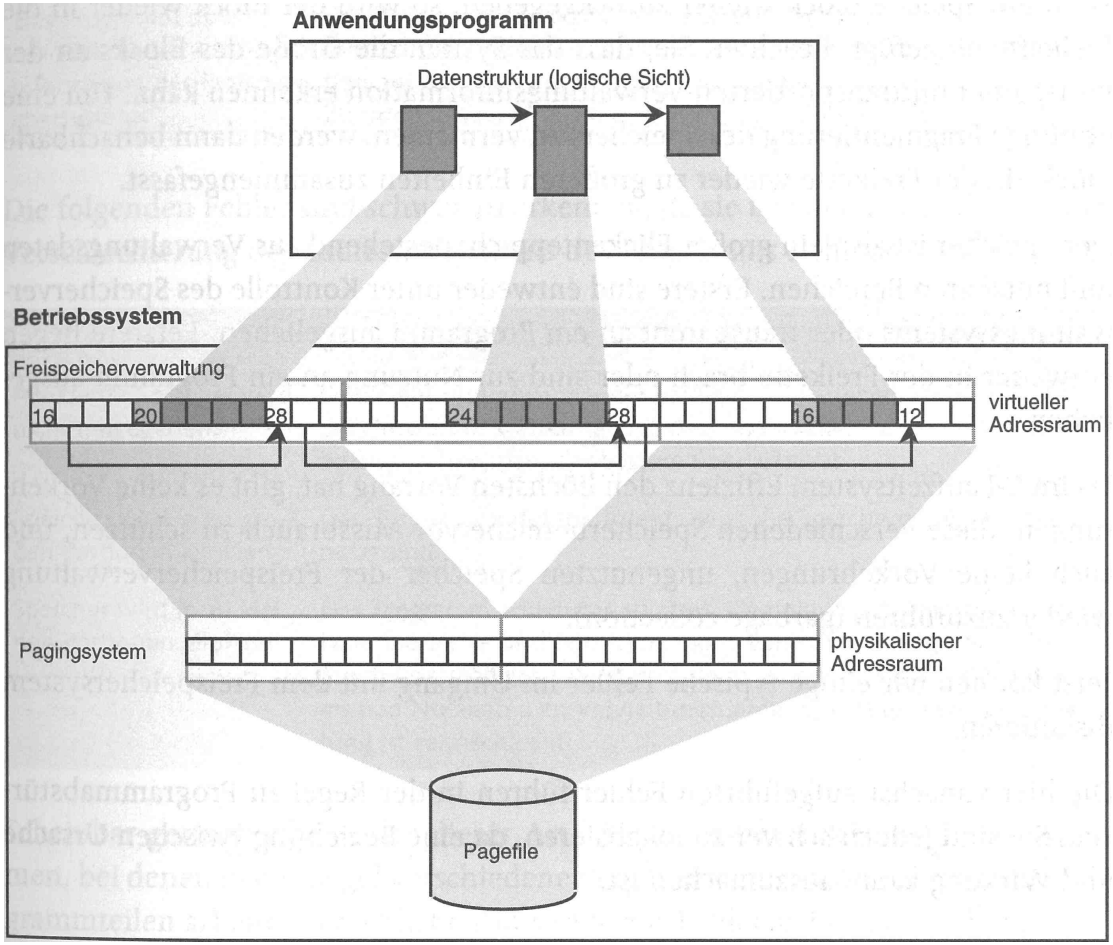

Figure: Freispeicherverwaltung [Kaiser and Kecher(2010)]

Hauptspeicherverwaltung im Betriebssystem

- Freier Speicher als Liste von Speicherblöcken fester Größe (Seiten)
- Seiten beinhalten Verwaltungsinformationen und Daten
- Finden von freien Seiten für einen Speicherblock:
  - First-Fit, Next-Fit, Best-Fit, Worst-Fit
  - Quick-Fit, Buddy-Verfahren

Allozierung von Speicherblöcken durch *malloc*

Seitenverwaltung

- Seiten werden aus dem Hauptspeicher in einen Puffer geladen
- Ein Teil der Seiten wird in Blöcke einer festen Größe (2^x) unterteilt

Allozierung von Blöcken $<$ Seitengröße

- Allozierung eines Blockes der nächst besten Größe
- $\Rightarrow$ potentieller Verschnitt

Allozierung von Speicherblöcken durch *malloc* (2)

Allozierung von Blöcken >= Seitengröße

- "bessere" Allozierung
- Allozierung ganzer Seiten
- Allozierung von partiellen Seiten falls nötig
- $\Rightarrow$ mehr Aufwand bei der Allozierung
- Aber größere Blöcke seltener, werden aber häuftig genutzt

Alternativen der Allozierung

memalign

- Speicher wird "ausgerichtet"
- Startadresse kann variiert werden

valloc

- *memalign* mit Seitengröße

posix_memalign() ersetzt *memalign* und *valloc*

weitere Hilfsmittel

memcheck()

Konsistenzprüfung der Freispeicherverwaltung

kmalloc()

- Spezialimplementierung, die in der Regel direkt zusammenhängenden physikalischen Speicher alloziert
- Kann über Parameter besonderen Speicher allozieren: blockierend, explizit, . . .

Slice Allocator (GLIB)

- Effizientere Verwaltung der Seiten
- keine statische Unterteilung
- keine Unterteilung in 2^x Blöcke

Zusammenfassung

- Dynamische Datenstrukturen werden im Freispeicher alloziert
  - Mit *malloc*,*calloc* und *realloc*
- Datenstrukturen sollten wieder freigegeben werden
  - Mit *free*
- Effiziente Realisierung der Allozierung hängt von Betriebssystem ab
- Es gibt viele Fehlerquellen, die berücksichtigt werden müssen

Literatur


Gnu.org.
The gnu c library.


U. Kaiser and C. Kecher.
C/C++ - Das umfassende Lehrbuch, volume 4.
Galileo Computing, 2010.
ISBN 3-89842-839-7.

informatik
die zukunft