

## Aufgabe 3: halde (14.0 Punkte)

In dieser Aufgabe soll eine einfache Freispeicherverwaltung implementiert werden, welche die Funktionen **malloc(3)**, **calloc(3)**, **realloc(3)** und **free(3)** aus der Standard-C-Bibliothek ersetzt. Die Verwaltung des Speichers der Größe 1 MiB erfolgt mit Hilfe einer einfach verketteten Liste. Die einzelnen Listenelemente, die die Größe des verwalteten Speicherbereichs beinhalten, werden jeweils am Anfang des dazugehörigen Speicherbereiches abgelegt.

### a) Makefile (2 Punkte)

Erstellen Sie ein Makefile, welches die Targets **test** und **test-ref** unterstützt. Das Target **test** erzeugt aus dem Testfall (**test.c**) und Ihrer Implementierung der Freispeicherverwaltung (**halde.c**) die ausführbare Datei **test**. Das Target **test-ref** erzeugt aus dem Testfall und der von uns bereitgestellten Freispeicherverwaltung (**halde-ref.o**) die ausführbare Datei **test-ref**. Greifen Sie dabei stets auf Zwischenprodukte (z. B. **halde.o**) zurück. Das Makefile soll ohne eingebaute Regeln funktionieren (**make(1)** mit den Optionen **-rR** starten).

### b) Testfall für malloc() und free() (2 Punkte)

Implementieren Sie einen Testfall für die Freispeicherverwaltung in der Datei **test.c**. Dieser soll **mindestens** aus vier aufeinanderfolgenden **malloc()**-Aufrufen, der Freigabe der angeforderten Speicherbereiche und weiteren vier **malloc()**-Aufrufen bestehen. Am Ende des Testfalles sollen alle angeforderten Speicherbereiche wieder mit **free()** freigegeben werden. Nach jedem **malloc()**- und **free()**-Aufruf soll die Funktion **printList()** aufgerufen werden, die den internen Zustand der Freispeicherliste ausgibt. Vergleichen Sie bereits während der Entwicklung Ihrer Freispeicherverwaltung die Ausgabe der Programme **test** und **test-ref**. Die Ausgabe muss nicht exakt übereinstimmen. Es ist ausreichend, wenn die Anzahl der angezeigten Listenelemente genau und die Gesamtmenge des freien Speichers ungefähr übereinstimmt.

Mit Hilfe des Aufrufs **make test test-ref** wird der Testfall für beide Implementierungen der Freispeicherverwaltung übersetzt.

**Achtung:** Ein funktionierender Testfall ist kein Garant für eine funktionierende Freispeicherverwaltung.

### c) Funktionen malloc() und free() (7 Punkte)

Die Funktion **malloc()** sucht in der Freispeicherliste den ersten Speicherbereich, der für die angeforderte Speichermenge groß genug ist, und entfernt ihn aus der Freispeicherliste. Ist der Speicherbereich größer als benötigt und verbleibt **genügend** Rest, so wird dieser Speicherbereich geteilt und der Rest wird mit Hilfe eines neuen Listenelementes in die Freispeicherliste eingehängt. Im herausgenommenen Listenelement wird statt eines **next**-Zeigers eine *Magic Number* mit dem Wert **0xbaadf00d** eingetragen. Der von **malloc()** zurückgelieferte Zeiger zeigt auf die Nutzdaten hinter dem Listenelement.

Die Funktion **free()** hängt den freizugebenden Speicherbereich wieder vorne in die Freispeicherliste ein, **ohne** ihn mit gegebenenfalls vorhandenen benachbarten freien Bereichen zu verschmelzen. Vor dem Einhängen muss die *Magic Number* überprüft werden. Schlägt die Überprüfung fehl, so soll das Programm durch den Aufruf der Funktion **abort(3)** abgebrochen werden.

### d) Funktionen realloc() und calloc() (3 Punkte)

Die Funktion **realloc()** ist auf **malloc() + memcpy() + free()** abzubilden. Die Funktion **calloc()** verwendet **malloc()** zur Anforderung eines Speicherbereichs in der passenden Größe und initialisiert ihn mit **0x0**.

### e) Unvollständige Checkliste

Die folgenden Punkte der **unvollständigen** Checkliste sollten Sie vor der finalen Abgabe zwingend abarbeiten:

- Die Funktionen **malloc(3)**, **calloc(3)**, **realloc(3)** und **free(3)** weisen das in den Manpages beschriebene Verhalten auf, auch in den genannten Grenzfällen (z.B. **free(NULL)**).
- Die **errno(3)** wird im Fehlerfall korrekt gesetzt.
- Die Anforderung eines Speicherbereiches der Größe (1 MiB - Größe eines Listenelementes) ist erfolgreich.

### Hinweise zur Aufgabe:

- Erforderliche Dateien: **halde.c**, Makefile, **test.c**
- Hilfreiche *Manual-Pages*: **abort(3)**, **calloc(3)**, **free(3)**, **malloc(3)**, **memcpy(3)**, **memset(3)**, **realloc(3)**
- Im Verzeichnis **/proj/i4sp1/pub/aufgabe3/** befinden sich die Dateien **halde.{c,h}**, **halde-ref.o** und **test.c**. Kopieren Sie sich diese Dateien mit Hilfe des Skriptes **/proj/i4sp1/bin/copy-public-files-for** in Ihr Projektverzeichnis und implementieren Sie die fehlenden Funktionen und Definitionen in der Datei **halde.c** und **test.c**.
- Die Funktion **printList()** gibt für jedes Listenelement die Position im Adressraum (**addr**), den Offset innerhalb der 1 MiB (**offset**) und die eingetragene Größe (**size**) auf den Standardfehlerkanal aus.

### Hinweise zur Abgabe:

Bearbeitung: Zweiergruppen

Bearbeitungszeit: 11 Werkstage

Abgabetermin: 17:30 Uhr