

## Aufgabe 3: halde (12.0 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 freien Speichers erfolgt mit Hilfe einer einfach verketteten Liste. Die einzelnen Listenelemente, die die Größe des verwalteten Speicherbereichs beinhalten, werden am Anfang des dazugehörigen Speicherbereiches abgelegt.

Im Verzeichnis `/proj/i4sp1/pub/aufgabe3/` befinden sich die Dateien `halde.{c,h}`. 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`.

### a) Funktionen `malloc()` und `free()`

Die Halde soll einen statisch allokierten Speicherbereich der Größe 128 MiB verwalten. Ein Nachfordern von Speicher vom Betriebssystem ist in dieser Aufgabe nicht vorgesehen.

Die Funktion `malloc()` sucht in der Freispeicherliste den ersten Speicherbereich, der für den geforderten Speicherbedarf groß genug ist (*first-fit*), 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 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.

### b) Funktionen `realloc()` und `calloc()`

Die Funktion `realloc()` ist auf `malloc() + memcpy() + free()` abzubilden. Der existierende Bereich wird weder vergrößert noch verkleinert.

Die Funktion `calloc()` verwendet `malloc()` zur Anforderung eines Speicherbereichs in der passenden Größe und initialisiert den Speicherbereich mit `0x0`.

### c) Makefile

Erstellen Sie ein Makefile, welches die beiden Targets `wsort` und `lilo` unterstützt. Die beiden Targets erzeugen jeweils aus Ihren Lösungen der vorangegangenen Aufgaben ausführbare Dateien, die die in dieser Aufgabe implementierte Speicher-verwaltung verwenden. Dabei werden alle nötigen Dateien im Verzeichnis `aufgabe3` erwartet. Die Dateien `wsort.c` und `lilo.c` sollen dazu gegebenenfalls von **make(1)** aus den entsprechenden Aufgabenverzeichnissen kopiert werden.

Wird an **make(1)** kein Argument übergeben, so sollen beide ausführbare Dateien (`wsort` und `lilo`) erzeugt werden. Das Makefile soll außerdem die speziellen Targets `all` und `clean` unterstützen.

### d) 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.
- ☐ Die **errno(3)** wird im Fehlerfall korrekt gesetzt.
- ☐ Die Lösung der Aufgabe `lilo` funktioniert mit dem `halde`-Modul.
- ☐ Die Lösung der Aufgabe `wsort` kann zumindest die Dateien `wlist0` und `wlist05` sortieren.
- ☐ Die Anforderung eines Speicherbereiches der Größe *128 MiB - Größe eines Listenelementes* ist erfolgreich.

### Hinweise zur Aufgabe:

- Erforderliche Dateien: `halde.c`, `Makefile`
- Hilfreiche *Manual-Pages*: **abort(3)**, **calloc(3)**, **free(3)**, **malloc(3)**, **memcpy(3)**, **memset(3)**, **realloc(3)**
- Sollten Sie keine Lösungen zu den Aufgaben `lilo` und `wsort` haben, so können Sie zum Abarbeiten der Checkliste auf die Dateien `lilo.o` und `wsort.o` im Verzeichnis `/proj/i4sp1/pub/aufgabe3/` zurückgreifen.

### Hinweise zur Abgabe:

Bearbeitung: Zweiergruppen

Bearbeitungszeit: 9 Werktage

Abgabezeit: 17:30 Uhr