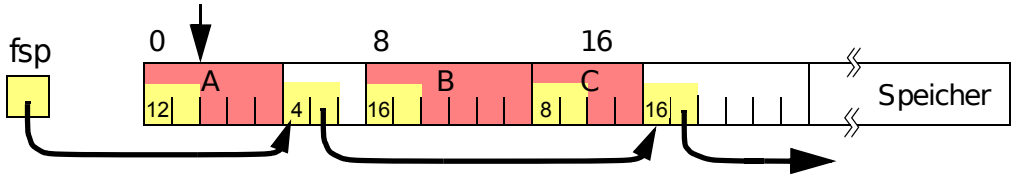
Aufgabe 3: halde (12.0 Punkte)

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 Freispeicherverwaltung soll den freien Speicher in einer einfach verketteten Liste verwalten. Am Anfang eines freien Speicherbereichs steht jeweils eine Verwaltungsstruktur mit der Größe des Speicherbereichs und einem Zeiger auf den nächsten freien Bereich. Ein Zeiger (hier *fsp* genannt) zeigt auf den Anfang der Liste.



Im Verzeichnis `/proj/i4sp1/pub/aufgabe3/` befinden sich die Dateien `halde.{c,h}` und `Makefile`. Kopieren Sie sich die 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) Speicher initialisieren, belegen und freigeben

Implementieren Sie die Funktionen `malloc()` und `free()`. Die Halde soll einen statisch allokierten Speicherbereich der Größe 1 MiB verwalten. Ein Nachfordern von mehr Speicher vom Betriebssystem ist in dieser Aufgabe nicht vorgesehen. Die Funktion `malloc()` sucht den ersten freien Speicherbereich in der Freispeicherliste, der für den geforderten Speicherbedarf und die Verwaltungsstruktur groß genug ist (*first-fit*). Ist der Speicherbereich größer als benötigt und verbleibt genügend Rest, so ist dieser Rest mit einer neuen Verwaltungsstruktur am Anfang wieder in die Freispeicherliste einzuhängen. In die Verwaltungsstruktur vor dem belegten Speicherbereich wird die Größe des Bereichs und statt des *next*-Zeigers eine *Magic Number* mit dem Wert `0xbaadf00d` eingetragen. Der zurückgelieferte Zeiger zeigt auf die Nutzdaten hinter der Verwaltungsstruktur, wie in der Abbildung für den Bereich *A* eingezeichnet.

Die Funktion `free()` hängt einen freigegebenen Speicherbereich wieder vorne in die Freispeicherliste ein, **ohne** ihn mit gegebenenfalls vorhandenen benachbarten freien Bereichen zu verschmelzen. Vor dem Einhängen ist die *Magic Number* zu überprüfen. Schlägt die Überprüfung fehl, so soll das Programm abgebrochen werden (**abort(3)**).

b) Speicher vergrößern und verkleinern

Nun sollen die Funktionen `realloc()` und `calloc()` implementiert werden (**memcpy(3)**, **memset(3)**). `realloc()` ist hierbei grundsätzlich auf `malloc()` + `memcpy()` + `free()` abzubilden. Der existierende Bereich wird nicht vergrößert oder verkleinert.

c) Testen

Mit Hilfe des vorgegebenen Makefiles können die beiden Testprogramme `{simple,extended}-test` erzeugt werden. Testen Sie damit Ihre Implementierung.

Hinweise zur Aufgabe:

- Erforderliche Dateien: `halde.c`
- Die Funktionen **malloc(3)**, **calloc(3)**, **realloc(3)** und **free(3)** müssen das in den Manpages beschriebene Verhalten aufweisen. Denken Sie auch an das Setzen von **errno(3)** im Fehlerfall!
- Zum Testen können Sie das `halde`-Modul mit Ihrer `wsort`-Implementierung aus Aufgabe 2 binden. Das Sortieren wird aber nur mit Listen funktionieren, für die die Speicherbeschränkung von 1 MiB ausreichend ist.

Hinweise zur Abgabe:

Bearbeitung: Zweiergruppen
Bearbeitungszeit: 7 Werktage
Abgabezeit: 17:30 Uhr