Übungen zu Betriebssysteme

Ü3 – Aufgabe: lilo

Sommersemester 2023

Henriette Hofmeier, Manuel Vögele, Benedict Herzog, Timo Hönig

Bochum Operating Systems and System Software Group (BOSS)

Faculty of Computer Science

RUHR UNIVERSITÄT BOCHUM

RUB

Aufgabe: lilo

- Aufgabe: Einfach verkettete FIFO-Liste, die Ganzzahlen (`int`) verwaltet

→ Dynamische wachsende Datenstrukturen (bspw. verkettete Listen) notwendig, wenn die Anzahl der Einträge a-priori unbekannt ist

Aufgabe: lilo

- Aufgabe: Einfach verkettete FIFO-Liste, die Ganzzahlen (`int`) verwaltet

→ Dynamische wachsende Datenstrukturen (bspw. verkettete Listen) notwendig, wenn die Anzahl der Einträge a-priori unbekannt ist

- Lernziele:
  - Dynamische Speicherverwaltung verstehen & gezielt einsetzen
  - Umgang mit ”einfachen” Zeigern
  - Nutzung von zusammengesetzen Datentypen (Strukturen) und ggf. Typaliasen

```
struct listelement {
  int value;
  struct listelement *next;
};
typedef struct listelement listelement; // optional
```

lilo - Dynamische Speicherverwaltung

- lilo: Kopf der Liste als globale Variable
- Anlegen eines Listenelementes mittels `malloc(3)`

```
struct listelement *newElement;
newElement = malloc(sizeof(struct listelement));
if(newElement == NULL) { /* Fehlerbehandlung */ }

// [...]

free(newElement);
```

  - Zurückgegebener Speicher hat undefinierten/zufälligen Wert
  - Inititialisierung muss per Hand erfolgen

- **Allokierter Speicher muss wieder freigegeben werden** → `free(3)`

lilo - Dynamische Speicherverwaltung

- lilo: Kopf der Liste als globale Variable
- Anlegen eines Listenelementes mittels `malloc(3)`

```
struct listelement *newElement;
newElement = malloc(sizeof(struct listelement));
if(newElement == NULL) { /* Fehlerbehandlung */ }

// [...]

free(newElement);
```

  - Zurückgegebener Speicher hat undefinierten/zufälligen Wert
  - Inititialisierung muss per Hand erfolgen
- **Allokierter Speicher muss wieder freigegeben werden** → `free(3)`

Achtung: Speicherverwaltung in C ist umständlich und fehleranfällig!
Macht euch Gedanken, bevor ihr ”mal schnell” was programmiert.

lilo - Schnittstelle nach außen

- Nur folgende Funktionen zu implementieren
  - `insertElement()`: Fügt einen neuen, nicht-negativen Wert in die Liste ein, wenn dieser noch nicht vorhanden ist. Tritt ein Fehler auf, wird -1 zurückgegeben. Ansonsten wird der eingefügte Wert zurückgegeben.

lilo - Schnittstelle nach außen

- Nur folgende Funktionen zu implementieren
  - `insertElement()`: Fügt einen neuen, nicht-negativen Wert in die Liste ein, wenn dieser noch nicht vorhanden ist. Tritt ein Fehler auf, wird -1 zurückgegeben. Ansonsten wird der eingefügte Wert zurückgegeben.
  - `removeElement()`: Entfernt den ältesten Wert in der Liste und gibt diesen zurück. Ist die Liste leer, wird -1 zurückgeliefert.

lilo - Schnittstelle nach außen

- Nur folgende Funktionen zu implementieren
  - `insertElement()`: Fügt einen neuen, nicht-negativen Wert in die Liste ein, wenn dieser noch nicht vorhanden ist. Tritt ein Fehler auf, wird -1 zurückgegeben. Ansonsten wird der eingefügte Wert zurückgegeben.
  - `removeElement()`: Entfernt den ältesten Wert in der Liste und gibt diesen zurück. Ist die Liste leer, wird -1 zurückgeliefert.
- Keinerlei Listen-Funktionalität in der `main()`-Funktion
  - **Alle** für die Verwaltung der Liste notwendigen Operationen werden in `insertElement()` und `removeElement()` ausgeführt
  - Allerdings: Erweitern der `main()` zum Testen erlaubt und **erwünscht**

lilo - Schnittstelle nach außen

- Nur folgende Funktionen zu implementieren
  - `insertElement()`: Fügt einen neuen, nicht-negativen Wert in die Liste ein, wenn dieser noch nicht vorhanden ist. Tritt ein Fehler auf, wird -1 zurückgegeben. Ansonsten wird der eingefügte Wert zurückgegeben.
  - `removeElement()`: Entfernt den ältesten Wert in der Liste und gibt diesen zurück. Ist die Liste leer, wird -1 zurückgeliefert.
- Keinerlei Listen-Funktionalität in der `main()`-Funktion
  - **Alle** für die Verwaltung der Liste notwendigen Operationen werden in `insertElement()` und `removeElement()` ausgeführt
  - Allerdings: Erweitern der `main()` zum Testen erlaubt und **erwünscht**
- Sollte bei der Ausführung einer verwendeten Funktion (z. B. `malloc(3)`) ein Fehler auftreten, sind keine Fehlermeldungen auszugeben. Im Fehlerfall wird -1 zurück gegeben.