Hochschule RheinMain
Studiengang Medieninformatik
Sommersemester 2018
Prof. Dr. Wolfgang Weitz

Betriebssysteme und Rechnerarchitektur

Übungsblatt drei

30.04.2018

Aufgabe 1:

Wie wir wissen, sind aller guten Dinge drei. Dies gilt demnach auch für Felder einer Struct wie dieser (vgl. Datei „`database.h`“ im read.MI):

```
typedef struct dbrec {
  char key[DB_KEYLEN];
  char cat[DB_CATLEN];
  char value[DB_VALLEN];
} DBRecord;
```

Wir möchten nun ein C-Modul schreiben, welches eine kleine Datenbank in einer Binärdatei aus solchen Sätzen verwaltet (keine Textdatei - structs werden komplett und direkt als Bytefolge aus dem Speicher in die Datei als Sätze fester Länge geschrieben).

Bitte implementieren Sie die folgenden Funktionen in einem C-Modul `database.c`:

- `int db_list(const char *path, int outfd,`
  `int (*filter)(DBRecord *rec, const void *data), const void *data)`
  liest die Datenbank, die unter dem übergebenen Dateipfad `path` abgespeichert ist, und schreibt ihren Inhalt als menschlesbare Tabelle auf den übergebenen Ausgabe-Filedescriptor `outfd`. Zeilentrenner ist das Newline-Zeichen `'\n'`.
  Optional kann eine Filterfunktion `filter` mitgegeben werden, die zu einem übergebenen Record `rec` einen Wahrheitswert zurückliefert. Der Aufrufer kann in `data` beliebige Zusatzinformationen mitgeben, die `filter` durchgereicht werden und die er bei seiner Arbeit verwenden kann, solange sie nicht verändert werden. Benötigt `filter` keine Zusatzinfos, kann für `data` der NULL-Zeiger übergeben werden.
  Liefert `filter` für ein `DBRecord` „true“, so wird es ausgegeben, sonst nicht. Wenn `filter` `NULL` ist, werden alle Sätze ausgegeben.
  Die Ausgabe der drei Felder eines `DBRecord` soll dreispaltig erfolgen, die Spaltenlängen

von Schlüssel und Kategorie richten sich immer den Inhalten der o.g. DB-Konstanten, das letzte Feld (`value`) wird in seiner natürlichen Länge ausgegeben (nicht aufgefüllt). Spalten sind durch senkrechte Striche „|“ voneinander getrennt. Rückgabewert ist die Anzahl der ausgegebenen Datenbanksätze oder -1 im Fehlerfall. Über geeignete Wahl von `outfd` können Sie Ihre Tabelle mit dem Datenbankinhalt also jederzeit auf die Standardausgabe schreiben lassen, z.B. zu Testzwecken.

- `int db_search(const char *filepath, int start, DBRecord *record)`
  sucht in der Datenbankdatei mit dem übergebenen Dateipfad, beginnend bei einschließlich dem Satz mit Indexnummer `start`, den ersten Satz, dessen `key`-Feld bzw. dessen `cat`-Feld den Werten im übergebenen `record` entspricht. Ein Leerstring in `key` oder `cat` matcht jeden Inhalt (keine Sucheinschränkung). Vom ersten gefundenen Satz in der Datenbank wird der Inhalt in `record` kopiert. Rückgabewert ist die Indexnummer des Treffer-Satzes, die Zählung beginnt bei 0. Wird kein passender Satz gefunden, ist der Rückgabewert -1. Im Fehlerfall ist der Rückgabewert -42.
- `int db_get(const char *filepath, int index, DBRecord *result)`
  liest den Datenbanksatz mit Index-Nummer `index` in den `DBRecord` ein, auf den `result` zeigt. Rückgabewert ist 0, falls das geklappt hat, ansonsten -1.
- `int db_put(const char *filepath, int index, const DBRecord *record)`
  überschreibt den Satz mit der Index-Nummer `index` mit dem Inhalt des übergebenen Satzes, auf das `record` verweist. Ist `index` größer als die Index-Nummer des letzten Satzes oder ist `index`==-1, so wird der referenzierte Satz am Ende der Datenbankdatei angehängt. Rückgabewert ist 0 für ok und -1 für Fehler.
- `int db_update(const char *filepath, const DBRecord *new)`
  fügt in die Datenbank filepath einen neuen Satz `new` hinzu. Existiert bereits ein Satz mit den gleichen Werten für `key` und `cat`, so wird dessen `value` aktualisiert (der Satz wird überschrieben und behält seine Index-Nummer). Existiert die gegebene `key`/`cat`-Kombination noch nicht in der Datenbank, so wird ein entsprechender neuer Satz hinten angehängt. Rückgabewert ist in beiden Fällen die Index-Nummer des aktualisierten bzw. hinzugefügten Satzes bzw. -1 im Fehlerfall.
- `int db_del(const char *filepath, int index)`
  löscht aus der Datenbank den Satz mit der Indexnummer `index`, sofern ein solcher existiert. Rückgabewert ist 0, falls das geglückt ist, ansonsten -1.
  Hinweis: Der zu löschende Satz soll nicht nur überschrieben oder markiert werden, er soll verschwinden. Da Sie keine Löcher in eine Datei schneiden können, kopieren Sie die Ursprungsdatei bitte in eine Zwischendatei um und lassen Sie dabei die zu löschenden Sätze aus. Benennen Sie dann diese Zwischendatei wieder zum ursprünglichen Dateinamen zurück, als wäre nichts gewesen. Sie können dazu die Systemfunktion „`rename()`“ (`man 2 rename`) verwenden.

Aufgabe 2:
Nun wollen wir unsere Bibliothek auch benutzen. Schreiben Sie dazu bitte ein separates C-Modul, das die `main()`-Funktion beherbergt. Betrachten wir dazu eine Aufgabe mit Tieren.

Wer länger unterwegs ist, kann seine Zier- und Nutzfische bei einem oberbayerischen Fischverwahrungsunternehmen abgeben. Die Verwaltung der Gastfische soll nun durch ein neues IT-System unterstützt werden. Die zugehörige Datei `fischfile.dat` besteht aus Datensätzen des oben gezeigten Aufbaus, wobei „`key`“ der Name des Fischbesitzers ist, „`cat`“ ironischerweise der Name des Fisches und „`value`“ die Unterbringung des Fisches (z.B. Name des Aquariums).

Das zu erstellende Verwaltungsprogramm wird mit einem Kommandowort als erstem Parameter und (je nach Kommando) evtl. weiteren Parametern aufgerufen, wie folgende Beispiele zeigen:

- `a.out` **`list`**
  gibt eine Liste mit dem gesamten Dateiinhalt tabellarisch aus
- `a.out` **`list`** `meier`
  gibt eine Liste mit Daten zu den Fischen des Besitzers „meier“ aus
  (bitte verwenden Sie Ihre db_list() mit einer geeigneten Filterfunktion)
- `a.out` **`search`** `substr`
  gibt eine Liste aller der Datenbanksätze aus, in wo „substr“ als Teilzeichenkette im Besitzernamen oder Fischnamen auftritt.
- `a.out` **`add`** `schmidt glupschi`
  fügt einen neuen Datensatz für den Fisch „glupschi“ des Besitzers „schmidt“ mit der Standard-Unterbringung „Gruppenaquarium“ hinzu. Armer Fisch.
- `a.out` **`add`** `schmidt glupschi BlubberSuite`
  fügt einen neuen Datensatz für den Fisch „glupschi“ des Besitzers „schmidt“ mit der Unterbringung „BlubberSuite“ hinzu. Wenn die Datenbank schon einen glupschi von Besitzer schmidt enthält, wird der Wert im bestehenden Satz ausgegeben und danach auf den neuen aktualisiert.
- `a.out` **`update`** `huber Loftiquarium`
  aktualisiert für alle Fische des Besitzers huber die Unterbringung auf „Loftiquarium“ (glückliche Fische, das Loftiquarium hat einen eigenen Pool...).
- `a.out` **`delete`** `meier`
  löscht alle Sätze von „meier“.
- `a.out` **`delete`** `meier fluppi`
  löscht Satz für Fisch „fluppi“ von Besitzer „meier“ (aber nicht z.B. fluppis anderer Leute).

Bitte testen Sie Ihre Implementierung gründlich und beglücken Sie den Benutzer ggf. mit sinnvollen Fehlermeldungen.