Vorlesung Betriebssysteme
Wintersemester 2017/2018

Prof. Dr. Lars-Olof Burchard
Hochschule Darmstadt

5. Praktikumsaufgabe (Caching)

Implementieren Sie in C++ einen persistenten Datenspeicher, in dem 1024 Byte lange Datenblöcke auf der Festplatte gespeichert werden können und zusätzlich einige der Blöcke zur Beschleunigung des Zugriffs im Hauptspeicher gelagert werden. Verwendete Blöcke sollen in den Hauptspeicher (also den Cache) verlagert werden wenn Sie sich dort noch nicht befinden. Implementieren Sie LRU zur Ersetzung von Speicherblöcken im Hauptspeicher.

Dabei soll auf einen Datenblock der Länge 1024 Bytes (vom Typ **`string`**) über einen von Ihrem Datenspeicher vergebenen Schlüssel (**`blockID`**) zugegriffen werden.

Dazu sollen die folgenden Funktionen bereitgestellt werden:

- **`void init_store(int total_blocks, int ram_blocks);`**

  Die Funktion initialisiert den Datenspeicher. Es wird Platz für insgesamt **`total_blocks`** Datenblöcke (zu je 1024 Byte) auf der Festplatte und **`ram_blocks`** Datenblöcke im Hauptspeicher allokiert.

- **`void cleanup_store();`**

  Löscht alle Datenbereiche und gibt alle vom Datenspeicher allokierten Speicherblöcke frei.

- **`int readCacheBlock(int blockID, string &buffer);`**

  Liest 1024 Byte aus dem Block mit der ID **`blockID`** in den bereitgestellten Puffer **`buffer`** ein. Liefert 0 im Erfolgs- und -1 im Fehlerfall (d.h. **`blockID`** existiert weder im Hauptspeicher noch auf Festplatte) zurück. Sollte der entsprechende Block sich nicht im Hauptspeicher sondern auf der Festplatte befinden, muss für diesen Block Platz im Hauptspeicher durch Auslagerung eines anderen Blocks mittels LRU geschaffen werden.

- **`int writeCacheBlock(int blockID, string buffer);`**

  Falls **`blockID`** den Wert -1 hat, wird **`buffer`** in einen neuen 1024 Byte großen Block im Datenspeicher kopiert. In diesem Fall liefert **`writeCacheBlock`** eine neue gültige **`blockID`** zurück. Sollte im gesamten Datenspeicher kein Platz mehr für den neuen Block sein (d.h. es existieren bereits **`total_blocks`** Datenblöcke), soll die Funktion -1 zurückliefern.

  Falls **`blockID`** den Wert eines existierenden Blocks hat, wird **`buffer`** in den entsprechenden Block geschrieben. Sollte der entsprechende Block sich nicht im Hauptspeicher sondern auf der Festplatte befinden muss für diesen neuen Block Platz im Hauptspeicher durch Auslagerung geschaffen werden.

- **`int freeCacheBlock(int blockID);`**

  Gibt einen Block mit ID **`blockID`** im Datenspeicher frei. Dazu müssen u.a. die zum Speichern verwendeten (falls vorhanden) Bereiche im Hauptspeicher bzw. auf der Festplatte freigegeben werden.

Schreiben Sie ein kleines Testprogramm, um Ihre Implementierung mit einer geeigneten Menge von Anforderungen zu testen. Dabei soll **`total_blocks > ram_blocks`** gelten.

Hinweis: Sie können die Speicherung auf der Festplatte beliebig implementieren, es soll nur funktionieren.