Vorlesung Betriebssystemarchitektur WS 2004/05

Aufgabenblatt 5 vom 6. Januar 2005

(Vorstellung der Lösungen bei den Tutoren bis zum 20. Januar 2005)

Aufgabe 5.1: (40 Punkte)
Schreiben Sie in der Programmiersprache C ein Programm, welches eine dateibasierende Datenbank implementiert. Konkret geht es um eine Geburtstagsdatenbank. Es gibt eine Tabelle mit zwei Spalten "Name" und "Geburtstag". Ihr Programm soll zwei Funktionen unterstützen: Lesen und Schreiben. Beim Lesen soll zu einem angegebenen Namen der Geburtstag ausgeben werden. Die Schreib-Funktion soll unter dem angegebenen Namen den Geburtstag speichern. Beispiel:

Schreiben:
```
$ aufg51  db.dat  Finja  20.12.2002
$
```

Lesen:
```
$ aufg51  db.dat  Finja
Name: Finja  Geburtstag: 20.12.2002
$
```

Die Daten sollen in einer Datei (im Beispiel: db.dat) gespeichert werden. Die Datei ist in 48-Byte große Datenbank-Einträge aufgeteilt. Die Einträge haben folgenden Aufbau:

```
struct db_entry
{
    char name[32];
    char geburtstag[16];
};
```

Der Name und der Geburtstag sollen als 0-terminierte Zeichenkette gespeichert werden. Die maximale Länge des Names beträgt 31 Zeichen, die des Geburtstages 15 Zeichen. Die Position der Datenbank-Einträge in der Datei wird durch eine vorgegebene Hash-Funktion bestimmt (`http://www.dcl.hpi.uni-potsdam.de/uebung`).

Schreibvorgang: Es wird der Hash-Wert x des Namens bestimmt. Dann wird der momentane Dateizeiger auf den x-ten Datenbank-Eintrag in der Datei positioniert (Byte-Position x*48). Durch eine Leseoperation wird festgestellt, ob an dieser Position schon ein Eintrag vorhanden ist. Wenn nicht (alle Bytes sind 0 oder Dateiende wurde überschritten), dann wird der Eintrag an dieser Stelle gespeichert. Wenn bereits ein Eintrag vorhanden ist, wird überprüft, ob der Name identisch ist. In diesem Fall wird an dieser Stelle der neue Geburtstags-Eintrag gespeichert. Wenn der Name nicht identisch ist, dann wird der Dateizeiger auf den nachfolgenden Datenbank-Eintrag gesetzt und obige Prozedur wiederholt (Lineare Sondierung bei Hash-Kollision).

Lesevorgang: Es wird der Hash-Wert x des gesuchten Namens bestimmt. Dann wird der momentane Dateizeiger auf den x-ten Datenbank-Eintrag in der Datei positioniert. Nun wird durch eine Leseoperation festgestellt, ob an dieser Position ein Eintrag vorhanden ist. Wenn nicht, dann gibt es zu diesem Namen keinen Datenbankeintrag. Wenn ein Eintrag vorhanden ist, wird zunächst der Name des Eintrags mit dem gesuchten Namen verglichen. Sind beide gleich, ist der gesuchte Eintrag gefunden und kann ausgegeben werden. Wenn der Name nicht identisch ist, dann wird der Dateizeiger auf den nachfolgenden Datenbank-Eintrag gesetzt und obige Prozedur wiederholt.

Sie können das Programm unter Linux oder Windows entwickeln. Es sollen die Funktionen *open(2), read(2), write(2)* und *lseek(2)* benutzt werden (auch unter Windows). Achten Sie auf eine gründliche Fehlerbehandlung und fehlerhafte Eingaben.Verpacken Sie alle Source-Code-Dateien inklusive eines Makefiles in eine ZIP-Datei und schicken Sie diese ZIP-Datei als Attachment mindestens **48 Stunden** vor dem Tutoriumstermin an `bs@hpi.uni-potsdam.de` mit dem Betreff

`AUFGABE=5.1 GRUPPE=`<Ihre Übungsgruppennummer> `OS=WIN32` oder
`AUFGABE=5.1 GRUPPE=`<Ihre Übungsgruppennummer> `OS=LINUX` je nach Betriebssystem.

Die ausführbaren Dateien sollen `aufg51.exe` (Windows) bzw. `aufg51` (Linux) heißen.

Erklären Sie Ihrem Tutor das Programm und führen Sie es vor. Wie groß wird die Datenbank-Datei, wenn Sie einen Geburtstag unter dem Namen "Bernhard" speichern? Wieviel Platz wird dafür auf der Festplatte wirklich verbraucht? (Linux: du(1), Windows: Datei komprimieren)

Aufgabe 5.2: (40 Punkte)

Schreiben Sie eine C-Funktion `char *list(const char* path)` welche den Inhalt eines angegebenen Verzeichnisses `path` als eine formatierte Zeichenkette (`char*`) zurückgibt. Die Ausgabe soll `ls -la` unter Unix entsprechen und als Antwort auf das LIST-Kommandos des zu programmierenden FTP-Servers im Übungsblatt 6 dienen. Zur Vereinfachung brauchen die Einträge nicht sortiert, die Dateizugriffrechte, der Hardlink-Zähler, sowie Besitzer und Gruppe nicht den originalen Werten entsprechen. Die Werte für Größe in Byte, Datum des letzten Schreibzugriffs und der Name des Verzeichniseintrags sollen den originalen Werten entsprechen. Im Unterschied zu `ls -la` soll die Zeitangabe des letzten Schreibszugriffs immer dem Format `'Monatname<SPC<>Tag des Monats<SPC>Jahr'` entsprechen. Verzeichnisse müssen von anderen Einträge unterschieden werden (`d`). Die Ausgabe des speziellen Verzeichnisses "." soll unterdrückt werden. Die 1. Zeile mit der Angabe der Größe des Verzeichnisses in Kilobyte in der Form: `total XXXX` ist optional. Als Trennzeichen zwischen den Informationen zu einem Verzeichniseintrag werden nur Leerzeichen verwendet. Zur Verwendung als Ausgabe des FTP-Kommados LIST wird eine Zeile mit *Carriage Return* und *New Line* (`\r\n`) Steuerzeichen beendet. Ein Listing sieht dann folgendermaßen aus:

```
drwx------ 1  root  None     0  Dec 13  2003  ..[\r\n]
-rwx------ 1  root  None  5643  Mar 24  2004  dummy.txt[\r\n]
-rwx------ 1  root  None  2100  Feb 10  2004  dummy1.txt[\r\n]
drwx------ 1  root  None     0  Apr  1  2004  subfolder[\r\n]
-rwx------ 1  root  None  2543  Mar 24  2004  dummy2.txt[\r\n]
```

Sie können das Programm unter Linux oder Windows entwickeln. Ein Fehler wird durch den Rückgabewert NULL angezeigt. Die Fehlermeldung soll in der statischen Variablen `static char* errormsg` in `list.c` enthalten sein. Achten Sie auf eine gründliche Fehlerbehandlung und aussagekräfige Fehlermeldungen. Lesen Sie sich dazu die Dokumentation zu den Bibliotheksfunktionen *opendir(3), readdir(3), stat(2), localtime(3)* durch bzw. suchen Sie in der MSDN-Bibliothek nach *FindFirstFile*, *FindNextFile*, und *FileTimeToSystemTime*. Verpacken Sie die Source-Code-Datei `list.c` mit der Funktion `list` und eventuell benötigte Dateien in die ZIP-Datei `aufg52.zip` und schicken Sie diese Datei als Attachment mindestens **24 Stunden** vor dem Tutoriumstermin an `bs@hpi.uni-potsdam.de` mit dem Betreff
`AUFGABE=5.2 GRUPPE=`<Ihre Übungsgruppennummer> `OS=WIN32` oder
`AUFGABE=5.2 GRUPPE=`<Ihre Übungsgruppennummer> `OS=LINUX` je nach Betriebssystem.

Erklären Sie Ihrem Tutor die Funktion und führen diese mit einem entsprechenden Testprogramm vor.

Aufgabe 5.3: (20 Punkte)

Gegeben sei eine Datei, die im unkomprimierten Zustand 49 Cluster für die Daten im NTFS-Dateisystem belegt. Die Dateisystemdatenbank (MFT) enthält für jede Datei einen Eintrag für die durch die Daten belegten Cluster (VCN-LCN Mapping). Cluster 123 sei der erste durch die Daten belegte Cluster im Dateisystem. Folgende Cluster-Bereiche im Dateisystem stehen zur Verfügung 123-138, 260-279, 393-417. Es sind Fragmente von 4-16 Clustern möglich, wobei die Fragmente möglichst groß sein sollen. Geben Sie die Belegung (im MFT-Format (VCN-LCN-Cluster)) der Cluster im Dateisystem für die gegebene Datei an und stellen Sie diese Ihrem Tutor in Papierform vor.

a) Die Datei soll unkomprimiert gespeichert werden.
b) Die folgenden Cluster-Bereiche der Datei lassen sich auf jeweils einen Cluster komprimieren: 0-2, 12-14, 33-37, 43-47.
c) Die folgenden Bereiche der Datei enthalten Nullen (*sparse data*): 12-26, 31- 40

- Was bedeuten die Begriffe MFT, VCN, LCN, und Run.
- Wie viel Platzgewinn ergibt sich durch die Komprimierung?