Praktikumsaufgabe 2

Fassung A (für schwächere Studenten)

Lernziele

Wiederholung und weiteres Verständnis der in EPR/OOA behandelten C/C++-Programmierung. Erlernen des Konzepts des Datenbankzugriffs über ein natives Call Level Interfaces. Kennenlernen des Transaktionsbegriffs.

Vorbereitung

Folgende Informationen müssen Sie zum *Antestat* bereithalten:

- Wie können in einem C++-Programm die Funktionsdefinitionen auf mehrere Sourcedateien verteilt werden? Welche Schritte werden bis zur Erstellung des Executables durchlaufen? (Wiederholung EPR)
- Wie funktioniert Compilieren und Linken mit dem *gcc*? Wie werden Bibliotheken verwendet? ([1] Kap. 12)
- Wie kann die Projektverwaltung mit *make* und einem *Makefile* automatisiert werden? ([1] Kap. 12) Erläutern Sie das bereitgestellte Makefile. Wie ruft man es auf?
- Wie kann man in einem C++-Programm die Kommandozeilenargumente auslesen? Legen Sie Beispielcode für das Auslesen der Argumente vor. ([2] Kap. 11.5)
- Wie wandelt man einen C-String (*char**) in einen STL-String um? Wie hängt man an einen STL-String einen C-String an? Wie kann man eine C Stringfunktion (z.B. *sprintf*) mit einem STL-String als Argument auf? Wieso vermeidet man dadurch den berüchtigten Buffer-Overflow Fehler? (Wiederholung OOA)
- Wie kann man eine Datei in der Programmiersprache C++ zeilenweise einlesen und jede Zeile in die durch ein Trennzeichen getrennten Felder zerlegen? (Hinweis: *fgets()* und *strtok()* bzw. *strsep()*) Legen Sie Beispielcode vor, der einen String in seine Felder zerlegt. Alternativ können Sie auch den Tokenizer aus der OOA-Vorlesung verwenden, wobei Sie für die Liste den STL-Container *list* oder *vector* verwenden.
- Legen Sie ein schematisches Flussdiagramm für Ihr Hauptprogramm vor, aus dem der grobe Programmablauf mit den einzelnen Schritten deutlich wird. (Bemerkung: der Ablauf soll klar werden, deshalb bitte nicht auf die Ebene der Programmvariablen heruntergehen, sondern beschreibende Texte verwenden!)

Aufgabe

Es soll ein Programm “dbimp” geschrieben werden, das Daten aus einer Datei in die Datenbanktabelle *hersteller* einspielt. Der Datenbankzugriff soll über die Bibliothek *libpq* erfolgen. Dazu müssen Sie

die folgenden Funktionen implementieren, die aus *main()* aufgerufen werden, und deren Prototypen in der bereitgestellten Datei *db.h* deklariert sind:

- *db_login* - Datenbanklogin
- *db_logout* - Datenbanklogout
- *db_begin*, *db_commit*, *db_rollback* - Transaktionsbefehle
- *db_findhnr* - Gibt zurück, ob Herstellernummer schon vorhanden
- *db_insert* - Einfügen Datensatz
- *db_delete* - Löscht kompletten Tabelleninhalt

Kommandozeilenoptionen Die Programme sollen folgendermaßen aufgerufen werden:

```
Usage:
        dbimp  [options] <infile>
Options:
        -del  delete table contents before import
        -u  database user
        -p  password
        -h  database host
        -d  database name
```

Die Reihenfolge der Optionen ist egal. Bei fehlerhaftem Aufruf (kein *infile* oder unbekannte, mit '-' beginnende Option) wird die obige Meldung ausgegeben und abgebrochen.

Zieltabelle und Dateiformat Die Zieltabelle *hersteller* müssen Sie von Hand per SQL anlegen mit folgenden Feldern:

Feld	hnr#	name	plz	ort
Typ	varchar(3)	varchar(30)	varchar(5)	varchar(30)

Die Import-Datei enthält pro Zeile einen Datensatz, wobei die einzelnen Felder durch `;` getrennt sind. Die Felder stehen in der Reihenfolge *hnr, name, plz, ort*.

Funktionalität *dbimp* soll sich wie folgt verhalten:

- Der ganze Import erfolgt in einer Transaktion: bei Erfolg *commit* und bei einem Fehler Abbruch und *rollback*.
- Vor dem *insert* wird anhand des Schlüsselfelds überprüft, ob der Datensatz schon in der Datenbank vorhanden ist. Wenn ja, wird der Satz nicht importiert.
- Wenn über die Option *-del* gewünscht, wird vor dem Import der Tabelleninhalt gelöscht (innerhalb der Transaktion).

Am Ende gibt das Programm eine Importstatistik aus mit der Gesamtzahl der gelesenen Datensätze und der Anzahl der davon importierten Sätze.

Test Sie können Ihr Programm überprüfen anhand der Testdaten [5]. Hier die Sollergebnisse beginnend mit einer leeren Tabelle *hersteller*:

Kommando	Datensätze/ davon importiert	Anzahl Tabellensätze nach dem Import
dbimp data1	3/3	3
dbimp data2	2/3	5
dbimp -del data2	3/3	3
dbimp data3	Abbruch wegen Fehler in Zeile 2 von *data3*	3

Hinweise zur C++-Programmierung

- Ein Makefile für diese Aufgabe finden Sie als *Makefile.2a* auf der Webseite zu dieser Veranstaltung.
- Um das Makefile portabel zu halten, verwendet es das Programm *pg_config*, das die Verzeichnisse der Include/Library-Files über entsprechende Optionen zurückgibt.
- Komplette Zeilen können Sie einlesen mit der Funktion *fgets()*. Zum Zerlegen der Inputzeilen können Sie *strtok()* oder -besser- *strsep()* verwenden. Alternativ können Sie auch den Tokenizer aus der OOA-Vorlesung verwenden, wobei Sie für die Liste den STL-Container *list* oder *vector* verwenden.
- Um Buffer-Overflows zu vermeiden, verwenden Sie nach Möglichkeit die STL-Klasse *string*. Diese kann man auch an die libpq-Funktionen übergeben, die ein *const char** als Argument erwarten (wie?).
- Zum Debuggen können Sie den "Data Display Debugger" *ddd* verwenden. *ddd* ist ein grafisches Frontend zum *gdb*. Im unteren Fenster des *ddd* können Sie auch direkt Kommandos an den *gdb* absetzen, ohne sich durch unzählige Menüs hangeln zu müssen. Nützliche *gdb*-Kommandos sind *run arg1 arg2 ...* (Startet Programm mit angegebenen Argumenten), *print var* (druckt Inhalt der Variablen *var* aus) und *list file:line* (öffnet direkt Datei *file* und springt an Zeile *line*).

Referenzen

[1] Welsh, Kaufmann: *Linux - Wegweiser zur Installation&Anwendung.* Semesterapparat (`TWR Wels`)

[2] Karlheinz Zeiner: *Programmieren Lernen mit C.* Das Lehrbuch aus dem Fach EPR

[3] Hartwig: *PostgreSQL Professionell und Praxisnah.* Semesterapparat (`TWY Hart`)

[4] The PostgreSQL Global Development Group:
PostgreSQL 9.1.8 Dokumentation. `http://www.postgresql.org/docs/` (2013)
Kapitel "Client Interfaces, libpq"

[5] Die Testdaten unter *http://informatik.hsnr.de/~dalitz/data/lehre/DBS/aufg2dat.tar* können Sie mit dem Befehl *tar xf ...* entpacken. Dort liegt auch die Header-Datei *db.h* und das *Makefile.2a*.