Übung zum C/C++-Praktikum Fachgebiet Echtzeitsysteme

TECHNISCHE UNIVERSITÄT DARMSTADT

Cheatsheet für das C++-Praktikum

1 CodeLite Tutorial

1.1 Neues Projekt anlegen

Um ein neues Projekt anzulegen, wähle **Workspace → New project** im Menü. Wähle dann in der Kategorie **CPPP** je nach Aufgabenstellung die passende Projektvorlage aus. Gib den gewünschten **Projektnamen** ein. Alle anderen Optionen müssen nicht geändert werden.

1.2 Neue Dateien zum Projekt hinzufügen

Alle neu angelegten Projekte enthalten bereits eine **main.c** bzw. **main.cpp** und ein **Makefile**. Um eine neue Sourcecode-Datei zum Projekt hinzuzufügen, klicke mit der **rechten Maustaste** auf den Ordner **src** und wähle **Add a new file...**. Wähle einen passenden Dateityp aus und gib einen Dateinamen ein. Die Dateiendung (z.B. **.cpp**, **.h**) wird automatisch angehängt.

1.2.1 Neue Klassen zum Projekt hinzufügen

Eine Klasse wird meistens in zwei Dateien aufgeteilt: Eine **.cpp**- und eine **.hpp**-Datei. CodeLite bieten einen einfachen Weg um eine neue Klasse zu einem Projekt hinzuzufügen. Mit einem **Rechtsklick** auf den Ordner **src** und dem Menüpunkt **New Class...** wird der Assistent gestartet. Es muss nur noch der Klassenname eingegeben werden. Wenn nötig kann auch ein Namespace angegeben werden, dem die Klasse zugeordnet werden soll. Alle anderen Felder müssen nicht bearbeitet werden. Mit einem Klick auf **OK** werden die Dateien erstellt.

1.3 Projekt kompilieren und starten

Bevor ein C++-Programm gestartet werden kann, muss es kompiliert werden. Die Kompilierung wird durch ein **Makefile** gesteuert. Im Laufe des Praktikums wird es eine Einführung in Makefiles geben. Um ein Projekt zu kompilieren klicke auf das **Build-Symbol** in der Toolbar. Es sieht aus wie ein grünes Rechteck mit einem weißen Pfeil in der Mitte. Um das Programm auszuführen muss auf das **Zahnrad-Symbol** geklickt werden, welches sich links vom Build-Symbol befindet. Es öffnet sich ein Terminal in dem die Ausgabe des Programms angezeigt wird. Das Drücken der Enter-Taste schließt den Terminal.

1.3.1 Projekt debuggen

Du kannst dein Programm auch im Debug-Modus laufen lassen, um dir Schritt für Schritt den Ablauf anzusehen und mögliche Fehler zu finden. Zum Debuggen wählst du den Debug-Knopf (**grüner Pfeil**) und CodeLite wird automatisch in die Debug-Perspektive wechseln.

2 Arbeiten mit git

git ist ein Werkzeug zur Versionskontrolle bei der Softwareentwicklung. Es ermöglicht unter anderem ein verteiltes Arbeiten von mehreren Entwicklern an einem Projekt. Änderungen am Code werden dabei dokumentiert und gesichert. So ist es möglich, Änderungen an versionierten Dateien wieder rückgängig zu machen oder an mehreren Features gleichzeitig zu arbeiten.
Auch alle Unterlagen des Praktikums werden durch `git` verwaltet und sind auf `github.com` veröffentlicht. Der erste Teil der folgenden Einführung beschreibt, wie man die lokale Kopie der Unterlagen aktualisiert. Dabei wird der aktuelle Stand von `github.com` heruntergeladen. Im zweiten Teil wird beschrieben wie man eigene Projekte mit git verwaltet.

2.1 Praktikumsunterlagen aktualisieren

Alle Unterlagen und Materialien des Praktikums werden in zwei `git`-Projekten (auch **Repositories** oder **Repos** genannt) verwaltet. Eine lokale Kopie der Repos befindet sich in dem Ordner `~/CPPP/Repos`. Sie heißen **tud-cpp-exercises** und **tud-cpp-lecture**.

Zum Aktualisieren wird folgendermaßen vorgegangen:

- Öffne einen Terminal, z.B. durch die Verknüpfung auf dem Desktop.
- Wechsle in das Repo, das aktualisiert werden soll. Dazu wird der Befehl **cp <Pfad>** verwendet. Bei der Eingabe des Pfades können Datei- und Ordnernamen durch Drücken der Tab-Taste vervollständigt werden.

  ```
  cppp@cppp-VirtualBox:~$ cd CPPP/Repos/tud-cpp-exercises
  ```

- Der Befehl **git pull** überprüft das Repository auf Aktualisierungen und lädt diese bei Bedarf herunter.

  ```
  cppp@cppp-VirtualBox:~/CPPP/Repos/tud-cpp-exercises$ git pull
  ```

2.2 Eigene Repositories

Jetzt folgt ein Überblick über das Arbeiten mit `git`. Vorweg sei gesagt dass `git` ein sehr mächtiges und umfangreiches Werkzeug ist. Die hier beschriebenen Schritte bilden die absolute Grundlage im Umgang mit `git`. Eine umfassende Dokumentation gibt es auf der Projektseite **git-scm.com**.
Alle Befehle werden im Hauptordner des Projekts ausgeführt. Wechsle also nach dem Starten des Terminals zuerst in den Projektordner.

- Zuerst muss ein neues Repository angelegt werden. Dies geschieht mit dem Befehl **git init**.

  ```
  cppp@cppp-VirtualBox:~/MeinProjekt$ git init
  ```

- Nun werden die Dateien, die mit `git` verwaltet werden sollen, zum Repository hinzugefügt. Im Beispiel sind das die Dateien `main.cpp`, `class.cpp` und `class.hpp`.

  ```
  cppp@cppp-VirtualBox:~/MeinProjekt$ git add main.cpp class.cpp class.hpp
  ```

- Sobald das Bearbeiten der Dateien abgeschlossen ist kann der aktuelle Stand gesichert werden. Einen solchen Sicherungspunkt nennt man **commit**. Folgender Befehl sichert den aktuellen Stand aller Dateien, die seit dem letzten commit verändert wurden:

  ```
  cppp@cppp-VirtualBox:~/MeinProjekt$ git commit -a
  ```

  Es öffnet sich ein Texteditor. Hier gibt man eine kurze Beschreibung der Änderungen ein. Zeilen mit einer Raute am Anfang werden dabei ignoriert. Ist man fertig drückt man **Ctrl+O** zum Speichern und **Ctrl+X** zum Schließen des Editors.

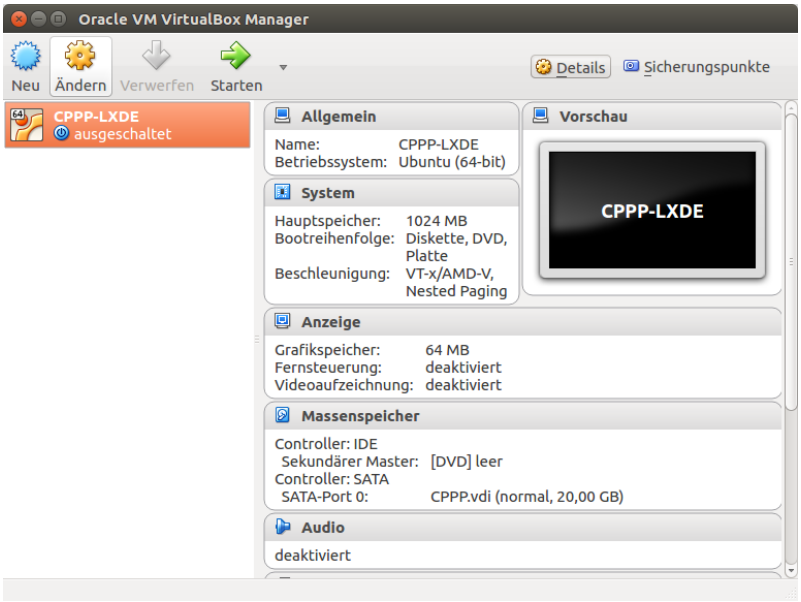
- `git` verfolgt, ob sich der Inhalt einer Datei ändert. Einen Überblick über alle geänderten Dateien liefert der Befehlt **git status**. **git log** liefert eine Übersicht über alle vergangenen Commits.

  ```
  cppp@cppp-VirtualBox:~/MeinProjekt$ git status
  cppp@cppp-VirtualBox:~/MeinProjekt$ git log
  ```

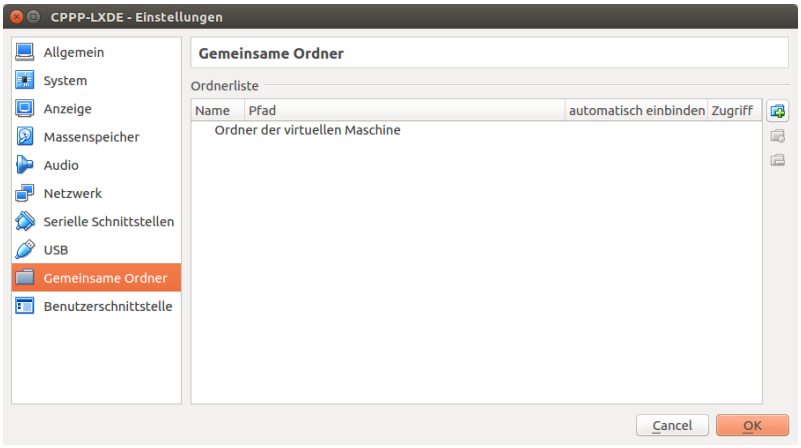
3 Shared Folder in VirtualBox

Manchmal ist es praktisch, eine Datei vom Host-Rechner zur VM oder umgekehrt zu kopieren, bspw. um deine Ergebnisse zu sichern. Dazu bietet VirtualBox sogenannte **Shared Folder** an. Um einen Shared Folder einzurichten sind folgende Schritte nötig:

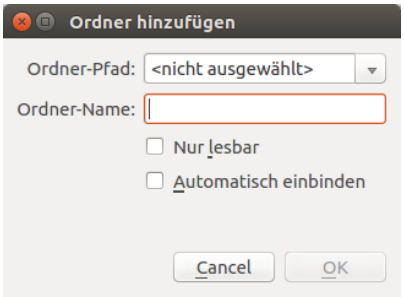
- Wähle die VM aus und klicke auf **Ändern**.



- Wähle links im Menü **Gemeinsame Ordner** aus. Klicke dann auf das Ordnersymbol mit dem grünen Plus, um einen neuen Shared Folder anzulegen.

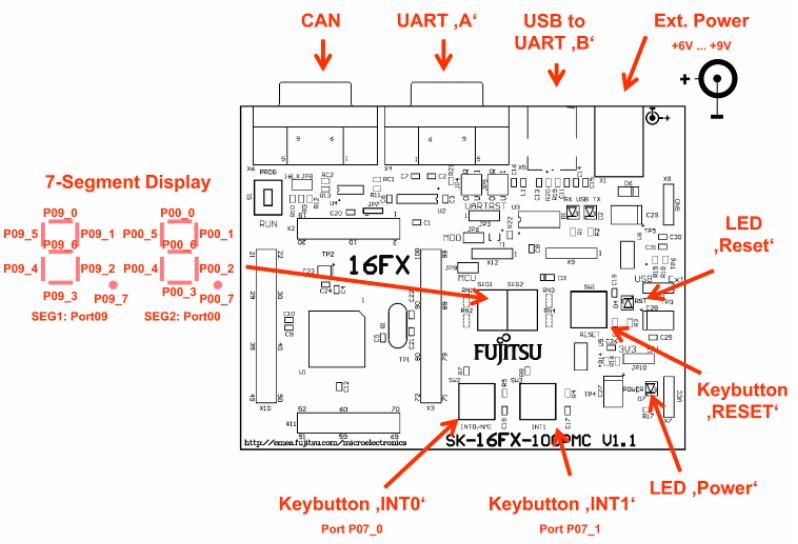


- Wähle den Ordner auf dem Host-PC aus und gib einen Namen ein. Setzt ein Häkchen bei **Automatisch einbinden**.



- Starte jetzt die VM neu. Im Dateimanager ist der Shared Folder in der linken Leiste zu sehen.

4 Mikrocontroller



4.1 Minimalprogramm

Folgendes Programm bleibt unendlich lange in der `for`-Schleife und führt bei jedem Durchgang eine Warteoperation durch. Die `#include`-Anweisung ist notwendig, um das Programm auf dem Mikrocontroller zu verwenden. Es bietet Zugang zur Hardware, als auch Systemfunktionen wie zum Beispiel die Warteoperation.

```
#include "mb96348hs.h"
void main(void) {
   for (;;) {
      __wait_nop();
   }
}
```