Übungsblatt 2:
C++ Libraries, Git & CMake

Sommersemester 2016

Programmiertechniken

Prof. Dr. L. Pollet

J. Greitemann, D. Hügel, J. Nespolo, T. Pfeffer

---

1) C++ Projekt mit mehreren Dateien

In dieser Aufgabe sollen Sie ein kleines Programm schreiben, dass Temperaturdaten einliest und die Durchschnittstemperatur, sowie die Standardabweichung ausgibt.

(a) Schreiben Sie eine `.cpp` Datei, die eine Funktion `void add(double)` bereitstellt mit der Daten gefüttert werden können und eine Funktion `double mean()`, die den Mittelwert der bereits gefütterten Daten ausgibt. Erstellen Sie einen Header `stat.hpp`, der diese Statistikfunktionalität nach außen beschreibt. Schreiben Sie dann den Anwendercode in eine separate `.cpp` Datei mit der `main` Funktion, die Eingaben des `stdin` abgreift, diese mit `add` füttert und schließlich die Durchschnittstemperatur ausgibt.

Tipp: Sie können Eingaben vom `stdin` mit

```
double x;
while(std::cin >> x) { ... }
```

einlesen und somit direkt als `double` parsen. Wenn Sie das Programm mit `./avg_temp` starten, können Sie eine Zahl eingeben, Return drücken und die nächste Zahl eingeben. Beenden Sie dann die Eingabe durch drücken von `Ctrl+D`. Dies sendet das *end-of-file* (`EOF`) Steuerzeichen.

(b) Testen Sie ihr Programm anhand der Tagesdurchschnittstemperaturen des Jahres 2015 in München, die Sie unter `https://db.tt/kET7HUlZ` herunterladen können. Die Jahresdurchschnittstemperatur sollte 11.1 ℃ betragen.

Tipp: Mit `./avg_temp < munich_2015.txt` können Sie die Daten direkt an das Programm übergeben.

(c) Legen Sie ein Git-Repository für dieses Projekt an und committen Sie den aktuellen Stand. Nur Quellcode, nicht aber Objektdateien oder ausführbare Programme, sollten von der Versionsverwaltung erfasst werden. Relative Pfade zu Dateien, die Git ignorieren soll, schreiben Sie in eine Datei `.gitignore`.

(d) Nun wollen wir zusätzlich die Varianz berechnen und als Standardabweichung ausgeben. Legen Sie einen *Branch* an und wechseln Sie in diesen:

```
git branch variance
git checkout variance
```

Nehmen Sie die nötigen Änderungen vor und erstellen Sie einen neuen Commit für diese.

Größere Projekte werden häufig von mehreren Personen entwickelt. Neue Features werden in *feature branches* entwickelt bis sie fertig sind und ein *merge* in den `master` Branch stattfindet. Wir wollen mit dem *merge* hier noch etwas warten und wechseln zurück in den `master` Branch und arbeiten zunächst damit weiter:

```
git checkout master
```

2) Erstellen einer *static library* von Hand

Die Statistikfunktionalität des soeben entwickelten Programms wollen wir nun vom Anwendercode trennen und in eine statisch gelinkte Library auslagern.

(a) Verwenden Sie `ar` (und indirekt `ranlib`) um eine static library `libstat.a` zu erzeugen.

(b) Kompilieren Sie nun den Anwendercode erneut und linken Sie diesen gegen die Library um das ausführbare Programm auf diesem Weg zu erstellen.

(c) Löschen Sie ihre Library wieder. Funktioniert ihr Programm immer noch? Warum?

Sorgen Sie dafür, dass am Ende dieser Aufgabe ihr Verzeichnis wieder sauber ist, d.h. keine von Git nicht erfassten Änderungen vorliegen.

3) Verwenden von *CMake*; Git Merges

CMake ist ein Werkzeug um den Kompilierungsprozess zu vereinfachen und plattformunabhängig zu gestalten. Wir wollen CMake jetzt dazu bringen die obige manuelle Erzeugung von Library und Programm für uns zu übernehmen.

(a) Schreiben Sie eine `CMakeLists.txt` Datei, mit der Sie CMake anweisen eine statisch gelinkte Library (`add_library`) und eine ausführbare Datei (`add_executable`) zu erzeugen und Letztere gegen Erstere zu linken (`target_link_libraries`). Orientieren Sie sich dazu am CMake Tutorial [1]. Führen Sie CMake aus und kompilieren Sie das Projekt mit `make`.

Tipp: Sie können mit `make VERBOSE=1` im Detail sehen wie genau der Compiler, `ar`, `ranlib` und der Linker aufgerufen werden.

(b) Erstellen Sie einen Git-Commit mit den in (a) vorgenommenen Änderungen auf dem `master` Branch.

(c) Sie entscheiden sich nun, dass das in 1 (d) entwickelte Feature fertig ist. Rufen Sie

```
git merge master variance
```

auf. Da der `master` sich seit des Abzweigens von `variance` geändert hat, erstellt Git einen sog. *merge commit*.

Pro-Tipp: Merge commits geben ein historisch korrektes Abbild des Entwicklungsprozesses wieder, führen aber zu zusätzlichen Commits und verkomplizieren so die commit history. Wenn ihr Feature-Branch noch nicht publiziert wurde (mit `git push`), können Sie stattdessen mit

```
git rebase master variance
```

die Geschichte ihres Features-Branches neu schreiben, d.h. die in `variance` aufgenommenen Änderungen auf den aktuellen `master` abspielen. Wenn Sie danach den Merge durchführen, erkennt Git, dass kein Merge-Commit nötig ist und führt einen sog. *Fast-forward merge* durch. Für weitere Informationen, lesen Sie

```
man git-merge
man git-rebase
```

(d) Ändern Sie ihre `CMakeLists.txt` so ab, dass statt einer statisch gelinkten Library `libstat.a` eine dynamisch gelinkte `libstat.so` erzeugt wird. Dafür müssen Sie lediglich den `SHARED` Modifier in den Aufruf von `add_library`) einfügen. Hier sehen Sie wie hilfreich CMake sein kann. Verwenden Sie wiederum `make VERBOSE=1` um zu sehen was hinter den Kulissen geschieht. Testen Sie das resultierende Programm. Löschen Sie die shared library und versuchen Sie es erneut. Was beobachten Sie nun?

(e) *(sehr fortgeschritten)* Gliedern Sie die Library komplett aus dem Anwendungsprojekt aus und machen Sie sie installierbar, d.h. fügen Sie `install`-Anweisungen zu `CMakeLists.txt` hinzu um die Library, den Header und ein Konfigurationsscript `stat-config.cmake` in die entsprechenden Installationsverzeichnisse zu kopieren. Verwenden Sie dann `find_package` im Anwendungsprojekt um die installierte Library ohne weiteres Dazutun des Users auf der Zielplattform zu finden.

Literatur

[1] CMake Tutorial, `https://cmake.org/cmake-tutorial/`

[2] The Git Book, Chapter 2: Git Basics,
`https://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository`

[3] Ein nützliches *hands-on* Tutorial zu Libraries in C++ mit vielen hilfreichen Zusatzinformationen,
`http://www.bogotobogo.com/cplusplus/libraries.php`