Testing und Debugging: Übung 03

Abgabetermin: siehe Canvas Kurs

1 Coverage Tool für gcc

Auch für C/C++ Code, der mit *gcc* kompiliert wurde, gibt es Tools, mit Hilfe derer man, ganz ähnlich wie mit *coverage3* aus der Vorlesung, Test coverage mit verschiedenen Metriken durchführen kann.

DIE ZUSÄTZLICHEN TOOLS HEISSEN:

- *gcov*: Dieses Tool erzeugt die Test Coverage Metrik als Textdatei (Quelldatei.c.cov)
- *lcov*: Wrapper für *gcov* der Dateien erzeugt die in html-Dateien umgewandelt werden können.
- *genhtml*: Erzeugt aus der Ausgabe von *lcov* ein Verzeichnis mit den html-Dateien.

MAN GEHT DABEI WIE FOLGT VOR UM EINE HTML AUSGABE ZU ERZEUGEN:

1. Kompilieren des Quellcodes mit speziellen Compiler- (und ggf. Linker-) Flags.
   - Es wird zu jeder Quelldatei eine gleichlautende Datei mit der Endung *.gcno* erstellt.
2. Aufruf von *lcov* mit diversen Schaltern um Informationen aus einem vorangegangenem *lcov*-Lauf zurück zu setzten.
3. Aufrufen des Tests für den eine Überdeckung erstellt werden soll.
   - Es wird zu jeder beteiligten Quelldatei eine gleichlautende Datei mit der Endung *.gcda* erstellt.
4. Aufruf von *lcov* mit diversen Schaltern um die Informationen aus den *.gcda* Dateien zusammen zu tragen.
   - Es wird ein *.info* Datei für diesen Testlauf erstellt.
5. Zurück zu 2. für einen neuen Test.
6. Aufruf von *genhtml* mit den Ausgabedateien von *lcov* und dem Namen des Verzeichnisses für die Ausgabe der html-Dateien und zusätzlichen Schaltern.

Tipp
Den Namen der *.gcno* und *.gcda* Datei kann man nicht beeinflussen, sie beruhen immer auf dem Basisnamen [a] der Quelldatei!
Verzichten Sie beim Kompilieren möglichst auf Optimizerflags (mit Ausnahme *-O0*) damit zu beobachtender Code nicht wegoptimiert wird.

[a] Name ohne Endung

1.1 Arbeiten mit *lcov* (80 Punkte)

Tipp
Bearbeiten Sie diese Aufgabe in der virtuellen Maschine in einem Unterverzeichnis von `~/project`. In der virtuellen Maschine sind bereits alle benötigten Tools installiert und jedes Unterverzeichnis von `~/project` kann vom Hostrechner aus im Browser unter http://127.0.0.1:8001/project betrachtet werden.

Dateien für diese Benutzung von lcov

examples/example.c	Quelldateien
examples/methods/iterate.c	Quelldateien
examples/methods/gauss.c	Quelldateien
examples/iterate.h	Headerdateien
examples/gauss.h	Headerdateien
examples/Makefile	Makefile zum Erzeugen des kompletten Beispiels
examples/descriptions.txt	Beschreibungstext der Tests

Wechsel Sie in das Verzeichnis *example* und rufen Sie *make output* auf.

Lesen Sie sich den kompletten Verlauf des Makevorgangs durch.

Betrachten Sie das Resultat (*example/output/index.html*) im Browser und schauen Sie sich die einzelnen Unterseiten an.

Aufgaben

1. *Welche Metriken wurden hier verwendet? (15 Punkt)*

```
.
.
.
```

2. *Betrachten Sie im Browser die Ergebnisse von* examples/methodes/gauss.c. *Welche Ergebnisse liefen die Metriken (in Prozent und in Hit/Total)? (15 Punkte)*

```
.
.
.
```

3. *Betrachten Sie im Browser die Ergebnisse von* examples/methodes/iterate.c *(die Seite mit dem Quellcode und den Markierungen der Metriken). Welche zusätzliche Informationen über das Line Coverage im Quellcode angezeigt, die das Python Tool nicht angezeigt hat? (10 Punkte)*

```
.
.
.
```

4. *Betrachten Sie das Makefile. Welche Compilerflags sind für die Benutzung des Coverage Tools unbedingt zu setzten? (Überflüssige Flags geben Punktabzug!) (10 Punkte)*

```
.
.
.
```

5. *Mit welchen Schaltern wurde* lcov *aufgerufen und was bewirken sie? (20 Punkte)*

```
.
.
.
.
.
.
.
```

6. *Mit welchen Optionen wurde* genhtml *aufgerufen und was bewirken sie? (10 Punkte)*

```
.
.
.
.
.
.
.
```

2 Programmieraufgabe: Branchcoverage (100 Punkte)

In der Datei *uebung03.py* ist ein 8 Bit Volladdierer mit zwei Hilfsfunktionen Implementiert.

Dateien für die Programmieraufgabe

Grading/__init__.py	Hinweisdatei für Python: Das Verzeichnis ist ein Modul
Grading/Grading.py	Quellcode der Testklasse, die in *run_test.py* verwendet wird
uebung03.py *	In dieser Datei müssen Sie die Funktionen `testFullBrachCoverage_X()` gemäß Aufgabenstellung implementieren

Die mit [*] markierten Dateien müssen **bearbeitet** und **abgegeben** werden.

Spezifikation: `add8` ist ein 8bit Volladdierer mit Übertrag.

Die Werte a0-a7, b0-b7 und c0 können den Wert True oder False haben. (Testen auf andere Eingaben ist nicht nötig!) a0-a7 stellen die Binärschreibweise der Zahl a dar. b0-b7 stellen die Binärschreibweise der Zahl b dar. c0 ist der Übertrag von einem etwaig vorangestellten Addierer.

Ausgegeben wird die Summe der Zahlen a und b in Binärform. C8 ist dabei das Übertragsbit, falls die Summe den Darstellungsbereich von 8 Bit Zahlen überschreitet.

`split` wandelt eine Integer Zahl in eine 8-Bit Binärzahl um. (Testen auf andere Eingaben ist nicht nötig!)

`glue` wandelt eine 8-Bit Binärzahl mit etwaigem Übertrag in eine Dezimalzahl um. b0-b7 und c können True oder False sein. (Testen auf andere Eingaben ist nicht nötig!)

Ihre Aufgabe

Implementieren Sie die Tests für die Funktionen `add8`, `split` und `glue`, so dass für *uebung03.py* (die Ergebnisse des Gradingmoduls können Sie ignorierten) vollständiges Branchcoverage erreicht wird.
Implementieren Sie für jeden Test eine eigene Funktion nach dem Namensschema `testFullBranchCoverage_X` wobei X eine fortlaufende Zahl ist.
Die Hilfsfunktionen `split` und `glue` können für die Tests von `add8` mitverwendet werden.
Denken Sie daran, dass Sie, alle neu geschriebenen Testfunktionen am Ende der Datei, so wie die schon bereitgestellten Funktionen, auch aufrufen müssen.
Benutzen Sie das in der Vorlesung vorgestellte Coverage Tool um Ihre Lösung zu überprüfen.

Bewertung Bei korrekter Implementation sollte der Aufruf von *python3 uebung03.py* folgende Ausgabe erzeugen (wobei natürlich Ihre Gruppennummer und Name und zugehörige Matrikelnummer Ihrer Gruppenmitglieder angezeigt werden muss); es dürfen **keine AssertionErrors** erscheinen:

```
====================
Aufgabe:       01
Gruppennummer: 99

Matrikelnummer, Name
--------------------
        99999, Sebastian Stigler
====================
```

Dies ist die Mindestvoraussetzung für die Bewertung. Ist diese nicht erfüllt erhalten Sie überhaupt keine Punkte für die Programmieraufgabe!

Für jeden Prozentpunkt im Brangecoverage erhalten Sie einen Punkt.

Viel Erfolg