Fakultät IV Elektrotechnik und Informatik
Einführung in die Programmierung WS 2022/2023
Manfred Hauswirth
Damien Foucard / Uwe Kuehn / Aljoscha Meyer

Aufgabenblatt 1

Ausgabe: *18.10.2022 14:00*
Abgabe: *20.10.2022 22:00*

Thema: *Erste Schritte mit Git und C*

Abgabemodalitäten

1. Die Aufgaben des C-Kurses bauen aufeinander auf. Versuche daher die Aufgaben zeitnah zu bearbeiten.
2. Alle abzugebenden Quelltexte müssen ohne Warnungen und Fehler auf Deinem Rechner mit dem Befehl `clang -std=c11 -Wall -g` kompilieren.
3. Die Abgabe für den Quellcode erfolgt ausschließlich über unser Git im entsprechenden Branch. Nur wenn ein Ergebnis im ISIS-Kurs angezeigt wird, ist sichergestellt, dass die Abgabe erfolgt ist. Die Abgabe ist bestanden, wenn Du an Deinem Test einen grünen Haken siehst.
4. Du kannst bis zur Abgabefrist beliebig oft neue Versionen abgeben. Lies Dir die Hinweise der Tests genau durch, denn diese helfen Dir, Deine Abgabe zu korrigieren.
   Bitte beachte, dass ausschließlich die letzte Abgabe gewertet wird.
5. Die Abgabe erfolgt, sofern nicht anders angegeben, in folgendem Branch: `ckurs-b<xx>-a<yy>`, wobei `<xx>` durch die zweistellige Nummer des Aufgabenblattes und `<yy>` durch die entsprechende Nummer der Aufgabe zu ersetzen sind.
6. Gib für jede Aufgabe die Quellcodedatei(en) gemäß der Vorgabe ab. Im ISIS-Kurs werden zum Teil Vorgabedateien bereitgestellt. Nutze diese zur Lösung der Aufgaben.
7. Die Abgabefristen werden vom Server überwacht. Versuche, Deine Abgabe so früh wie möglich zu bearbeiten. Du minimierst so auch das Risiko, die Abgabefrist auf Grund von „technischen Schwierigkeiten“ zu versäumen. Eine Programmieraufgabe gilt als bestanden, wenn alle bewerteten Teilaufgaben bestanden sind.

Erste Schritte mit Git

Git ist ein freies verteiltes Versionsverwaltungssystem und kommt hauptsächlich bei der Entwicklung von Software zum Einsatz. Es protokolliert Änderungen an Dateien (z. B. Programmcode) in einem *Repository* und erlaubt dabei jederzeit Zugriff auf jeden der protokollierten Zustände (*Commits*). Änderungen am Programmcode sind somit immer nachvollziehbar. Softwareentwicklern ist es dadurch möglich, parallel und koordiniert an einem gemeinsamen Projekt zu arbeiten.

In dieser Lehrveranstaltung wirst Du Git einsetzen, um den Programmcode Deiner Hausaufgaben zu verwalten. Im Folgenden sind die dafür notwendigen Konzepte, Abläufe und Git-Befehle beschrieben. Um die Inhalte nachvollziehen zu können, ist es hilfreich, bereits mit den grundlegenden Befehlen in einem Terminal (Bewegen im Dateisystem: `cd`, `ls`, `pwd`; Dateioperationen: `cp`, `mv`, `touch`, `mkdir`) und der Verwendung eines Texteditors vertraut zu sein. Weiterführende Informationen zu Git findest Du online im Pro Git-Buch oder in den Handbüchern zu den einzelnen Git-Befehlen (`man git <befehl>`).

Git gibt es für alle gängigen Betriebssysteme. Eine ausführliche Anleitung, um Git für Dein jeweiliges System zu installieren, findest Du ebenfalls im Pro Git-Buch.

Falls Du zum ersten Mal mit Git arbeitest, müssen nach der Installation noch zwei Einstellungen vorgenommen werden, um Git Deinen Namen und Deine E-Mail-Adresse mitzuteilen. Führe dazu die folgenden zwei Befehle aus:

```
git config --global user.name '<Dein Name>'
git config --global user.email '<deine@mail.adresse>'
```

Der Name und die E-Mail-Adresse können an dieser Stelle frei gewählt werden. Wir empfehlen, dass Du Deine TU-E-Mail-Adresse verwendest.

Initialisieren eines Git-Repository

Erstelle ein Verzeichnis mit beliebigem Namen (`mkdir <name>`) und wechsel in dieses Verzeichnis (`cd <name>`). Mit dem Befehl `git init` kannst Du nun in diesem Verzeichnis ein lokales Repository initialisieren. Dabei wird ein Unterverzeichnis[1] `.git` angelegt, welches das Grundgerüst des Repository beinhaltet. Die Dateien in diesem Unterverzeichnis werden durch Git selbst verwaltet und sollten im Regelfall nicht angepasst werden.

Der Status des Repository kann mit dem Befehl `git status` abgefragt werden. Dieser gibt unter anderem an, in welchem Branch Du Dich im Repository befindest. Für das eben erstellte Repository sollte das der `master` oder `main`-Branch sein. Ebenso zeigt der Status, ob geänderte Dateien im Repository sind. Da dieses Repository noch leer ist, werden keine Änderungen angezeigt.

Versionierung von Dateien

Im Verzeichnis des Repository können Ordner und Dateien angelegt werden, um diese später in einem zweiten Schritt unter Versionskontrolle zu stellen. Erzeuge dazu zuerst eine leere Textdatei im Verzeichnis des Repository (`touch datei.txt`). Die eben erstellte Datei befindet sich zwar im Verzeichnis des Repository, jedoch noch nicht im eigentlichen Repository. Sie wurde also noch nicht dem Versionsverwaltungssystem hinzugefügt. Der Befehl `git status` sollte deshalb auch die Datei unter `Untracked files` auflisten.

Um die Datei nun tatsächlich dem Repository hinzuzufügen, führe `git add datei.txt` aus. Der Befehl `git status` listet daraufhin die neu hinzugefügte Datei als Änderung am Repository auf (`new file: datei.txt`). Für diese Änderung kann nun ein Commit angelegt werden, welcher die Änderungen im Repository protokolliert. Jedem Commit muss mit dem Parameter `-m` eine benutzerdefinierte Nachricht angehangen werden, welche die Änderung beschreibt. Führe den Befehl `git commit -m 'Neue Datei datei.txt hinzugefügt'` aus, um einen Commit anzulegen.

[1] Bei Unix-artigen Betriebssystemen sind Verzeichnisse, welche mit einem Punkt beginnen, versteckte Verzeichnisse. Diese können mit dem Befehl `ls -a` angezeigt werden.

Durch den mehrfachen Aufruf von `git add` können einzelne Änderungen auch zu einem Änderungssatz zusammengefasst werden. Der Befehl `git add <datei>` fügt dabei nicht nur neue Dateien, sondern auch Änderungen an bestehenden Dateien zu einem Änderungssatz hinzu.

Führe die unten stehenden Schritte durch, um Dich mit den Befehlen und Abläufen vertraut zu machen. Vergleiche nach jedem Schritt die Ausgabe von `git status` mit Deinen Erwartungen.

1. Erstelle eine neue Datei im Verzeichnis Deines Repository.
2. Füge die neue Datei dem Repository hinzu (`git add <datei>`).
3. Ändere den Inhalt der bisher leeren Datei `datei.txt` und schaue Dir die Änderungen an (`git diff`).
4. Füge die Änderung von `datei.txt` zum Änderungssatz hinzu (`git add datei.txt`).
5. Erstelle einen Commit, der alle bisherigen Änderungen (neue Datei und Änderung an `datei.txt`) enthält (`git commit -m '<Nachricht>'`).

Der Zustand (Snapshot) des Repository zum Zeitpunkt des Commits ist durch eine Prüfsumme (Hash[2]) eindeutig identifizierbar. Anhand dieser Prüfsumme kann Git den Zustand wiederherstellen oder mit dem anderer Commits vergleichen. Der Befehl `git log` zeigt Dir die Abfolge der Commits im aktuell aktiven Branch Deines Repository.

Verwalten von Branches

Mit Hilfe von Branches können verschiedene Entwicklungen parallel betrieben und später wieder zusammengefügt werden (*Merge*). In der Lehrveranstaltung nutzen wir Branches jedoch ausschließlich zur Trennung der einzelnen Abgaben. Ein neuer Branch wird immer vom letzten Commit (Zustand) des aktiven Branches abgezweigt. Mit dem Befehl `git branch <branch-name>` kann ein Branch erzeugt und mit `git branch -d <branch-name>` wieder gelöscht werden. Eine Übersicht aller verfügbaren Branches erhält man mit `git branch -a`. Den jeweils aktiven Branch kann man mit dem Befehl `git checkout <branch-name>` wechseln.

Führe folgende Schritte aus, um Dich mit den Befehlen und Abläufen vertraut zu machen. Prüfe nach jedem Schritt mit `git status`, in welchem Branch Du Dich befindest.

1. Stelle sicher, dass Du Dich im `master` oder `main`-Branch Deines Repository befindest (`git status`).
2. Erstelle einen neuen Branch (`git branch <branch-name>`).
3. Wechsel in den neuen Branch (`git checkout <branch-name>`).
4. Führe eine Änderung am Repository durch und erzeuge einen Commit. Lasse Dir das Commit-Log ausgeben (`git log`).
5. Wechsel zurück in den `master` oder `main`-Branch. Lasse Dir das Commit-Log ausgeben und vergleiche es mit der vorherigen Ausgabe.

Arbeiten mit verteilten Repositories

Bisher haben wir nur auf in einem lokalen Repository gearbeitet. Git erlaubt aber auch das Teilen von Repositories und damit die Kollaboration an einem Projekt. Für diese Lehrveranstaltung nutzen wir persönliche Repositories, welche von der Gitlab-Plattform der TU-Berlin (https://git.tu-berlin.de) bereit gestellt werden. Dort findest Du auch Dein eigenes Repository mit dem Namen Deines TUB-Accounts in der `introprog-ws22` Gruppe. Beachte, dass dieses Repository erst nach Durchführen des Check-ins in unserem ISIS-Kurs zur Verfügung steht. Danach ist das Repository unter folgender URL erreichbar:

`https://git.tu-berlin.de/introprog-ws22/<TUB-Account>`

Ein Repository klonen: Mit dem Befehl `git clone <repository-url>` legst Du eine lokale Kopie des Repository an. Beim Aufruf des Befehls wirst Du gegebenenfalls nach Deinem TUB-Nutzeraccount gefragt. Um den Accountnamen und das Password nicht immer eingeben zu müssen, kann der Zugang per SSH[3] eingerichtet werden. Nach dem Klonen findest Du das Repository in dem dabei neu erstellten Ordner `<TUB-Account>`. Dort kannst Du nun Deine Abgabendateien versionieren und Branches verwalten.

Änderungen übertragen: Vorerst arbeitest Du nur auf der lokalen Kopie Deines Repository. Um Commits und Branches an die Gitlab-Plattform zu übertragen, führe den Befehl `git push` aus. Falls Du lokal einen neuen Branch erzeugt hast und dieser auf der Gitlab-Plattform noch nicht existiert, muss dieser zuerst mit `git push --set-upstream origin <branch-name>` auf der Plattform angelegt werden.

Führe folgende Schritte durch, um Dich mit den Befehlen und Abläufen vertraut zu machen. Gleichzeitig bereitest Du dabei die Abgabe für die erste Aufgabe vor.

1. Klone Dein persönliches Repository
   (`git clone https://git.tu-berlin.de/introprog-ws22/<TUB-Account>`).
2. Erstelle einen neuen Branch (`git branch ckurs-b01-a01`).
3. Wechsel in den neuen Branch (`git checkout ckurs-b01-a01`).
4. Übertrage den Branch an das Repository auf der Gitlab-Plattform (`git push --set-upstream origin ckurs-b01-a01`).
5. Melde Dich auf https://git.tu-berlin.de an und mache Dich mit der Oberfläche von Gitlab vertraut. Finde Dein persönliches Repository und prüfe, ob der Branch `ckurs-b01-a01` angelegt wurde.

Aufgabe 1 Hallo Osiris

Die erste Aufgabe soll Dich mit dem Workflow unserer Abgabeplattform vertraut machen. Dieser Ablauf ist in jeder Programmierhausaufgabe ähnlich. Die Abgaben erfolgen immer in Deinem persönlichen Repository:

`https://git.tu-berlin.de/introprog-ws22/<TUB-Account>`

Jede Aufgabe muss in einem separaten Branch eingereicht werden. Das Schema der Branch-Bezeichnung findest Du in den Abgabemodalitäten. Für diese Aufgabe lautet der Branch `ckurs-b01-a01`. Stelle für Deine Abgaben sicher, dass Du neue Branches immer vom `main`-Branch abzweigst.

[2] Beispiel: `5303a671af5b28a9c095024ffd630aca3d6c9ce1`, Kurzform: `5303a671`

[3] https://docs.gitlab.com/ee/ssh/

Eine Datei abgeben: Erstelle im Abgabe-Branch dieser Aufgabe (`ckurs-b01-a01`) in Deinem persönlichen Repository eine Textdatei `ckurs_blatt01_hallo_osiris.txt` mit beliebigem Inhalt. Protokolliere die Änderung mit einem Commit (`git add` und `git commit`) und übertrage (`git push`) diese an das Repository der Gitlab-Plattform.

Für die erste Aufgabe ist hier noch mal der vollständige Ablauf beschrieben. Die Schritte 1-3 können übersprungen werden, wenn Du erfolgreich *Erste Schritte mit Git* durchgeführt hast.

1. Klone Dein persönliches Repository
   (`git clone https://git.tu-berlin.de/introprog-ws22/<TUB-Account>`) und wechsle in das erzeugte Verzeichnis.
2. Erstelle einen neuen Branch (`git branch ckurs-b01-a01`).
3. Wechsel in den neuen Branch (`git checkout ckurs-b01-a01`).
4. Erstelle eine Textdatei (`ckurs_blatt01_hallo_osiris.txt`) mit beliebigem Inhalt im Verzeichnis Deines Repository.
5. Füge diese Änderung zum Repository hinzu (`git add ckurs_blatt01_hallo_osiris.txt`).
6. Erstelle einen Commit für diese Änderung (`git commit -m 'Leere Abgabedatei hinzugefügt'`).
7. Übertrage die Änderungen und den neuen Branch an das Repository auf der Gitlab-Plattform (`git push --set-upstream origin ckurs-b01-a01`).

Schaue Dir in unserem ISIS-Kurs das Ergebnis und das Feedback zu Deiner Abgabe an. Je nach Auslastung der Abgabeplattform kann es wenige Sekunden dauern, bis das Ergebnis angezeigt wird. Lese das Feedback aufmerksam durch und mache alle notwendigen Änderungen an der Textdatei `ckurs_blatt01_hallo_osiris.txt`, um die Abgabe zu bestehen. Die Abgabe erfüllt alle Anforderungen und gilt damit als bestanden, wenn sie mit einem doppelten grünen Haken versehen ist.

Aufgabe 2 Hallo Welt!

Schreibe ein C-Programm, welches auf dem Terminal mittels `printf` den Text “Hallo, <Name>!” ausgibt. Ersetze dabei <Name> durch Deinen TUB-Login. Die Ausgabe soll auf einer separaten Zeile erfolgen.

Eine beispielhafte Benutzung des Programms liefe ab wie in Listing 1 gezeigt:

Listing 1: Kompilieren und Aufruf des Programms

```
> clang -std=c11 -Wall -g ckurs_blatt01_aufgabe02.c -o ckurs_blatt01_aufgabe02
> ./ckurs_blatt01_aufgabe02
Hallo, hans99!
```

Gehe sicher, dass Deine Aufgabe dabei die folgenden Bedingungen erfüllt:

- Die Ein- und Ausgabebibliothek `stdio.h` wird geladen.
- Der Dateiname ist `ckurs_blatt01_aufgabe02.c`.
- Die Funktion `main` ist definiert.
- `printf` wird zur Ausgabe von Text benutzt.
- Die Datei kompiliert ohne Fehler und Warnungen beim Aufruf von:
  `clang -std=c11 -Wall -g ckurs_blatt01_aufgabe02.c -o ckurs_blatt01_aufgabe02`

Füge Deine Lösung als Datei `ckurs_blatt01_aufgabe02.c` im Abgabebranch `ckurs-b01-a02` in Dein persönliches Repository ein und übertrage die Lösung an die Abgabeplattform.