Einführung

I. Vorbereitung

Damit Sie die Aufgaben aus dem GIP1-Praktikum bearbeiten können, benötigen Sie Papier und Bleistift für Notizen und Skizzen, einen Texteditor (mit Syntax-Highlighting), einen C-Compiler und die Versionsverwaltung *Git*. Sorgen Sie dafür, dass Sie zum ersten Praktikumstermin die benötigte Software korrekt installiert haben.

I.1. Installation eines Editors

Für das GIP1-Praktikum benötigen Sie einen Texteditor mit Syntax-Highlighting für C-Code. Verwenden Sie für das GIP1-Praktikum zunächst *keine* IDE. Diese würde Ihnen die Sicht auf die zugrunde liegenden Vorgänge nehmen. Ein Verständnis dafür ist allerdings immens wichtig und hilft Ihnen *später*, eine IDE effizient zu nutzen. Sollten Sie noch keinen geeigneten Editor installiert haben, können Sie unter Linux, macOS und Windows den freien Editor Pulsar (`https://pulsar-edit.dev`) verwenden.

I.2. Installation des C-Compilers

Für das GIP1-Praktikum benötigen Sie einen C-Compiler. Hier bietet sich der *GCC* an.
Diesen werden Sie aus der Kommandozeile bzw. aus dem Terminal bedienen. Verwenden Sie *keine* IDE – siehe oben.

I.2.1 Linux

Installieren Sie *GCC* als Teil der Entwicklungswerkzeuge und -bibliotheken mithilfe des Paketmanagers Ihrer Distribution.

I.2.2 macOS

Installieren Sie den Compiler als Teil der *Xcode Command Line Tools*. Diese können Sie aus dem Terminal mit dem Befehl `xcode-select --install` installieren. Genaugenommen installieren Sie die Apple-Version des Compilers *clang*. Dieser funktioniert aber für das Praktikum genauso wie der *GCC*.

I.2.3 Windows

Installieren Sie das *w64devkit* von `http://mingw-w64.org`. Über diese Seite gelangen Sie zum entsprechenden Download auf GitHub. Wählen Sie die aktuellste Version – für 64-Bit nicht als *i686*. Entpacken Sie das erhaltene Zip-Archiv in einen neuen Ordner in Ihr *Programme*-Verzeichnis. Der entpackte Ordner enthält ein Unterverzeichnis `bin`. Fügen Sie den gesamten Pfad dieses Unterverzeichnisses zu Ihrer Umgebungsvariable `PATH` hinzu.

I.3. Installation von Git

Im GIP1-Praktikum lernen Sie die Grundzüge der Quellcodeverwaltung mit Versionskontrolle durch *Git* kennen. Dazu benötigen Sie Git-Werkzeuge, die Sie aus der Kommandozeile bedienen. Verwenden Sie auch hier zunächst keinen GUI-Client – es gilt dasselbe wie für die IDE.

I.3.1 Linux

Falls noch nicht vorhanden, installieren Sie Git mithilfe des Paktemanagers Ihrer Distribution.

I.3.2 macOS

Falls noch nicht vorhanden laden und installieren Sie Git als Teil der Xcode Kommandozeilenwerkzeuge oder über Homebrew.

I.3.3 Windows

Falls noch nicht vorhanden laden und installieren Sie das Paket *Git for Windows Setup* in 64-bit von `https://git-scm.com/download/win`.
Wählen Sie während der Installation *Notepad* als Standard-Editor für Git. Verwenden Sie als Standard-Namen für neue Repositories *main*. Achten Sie bei der Installation darauf, dass Sie Git von der Kommandozeile starten können
Stellen Sie weiterhin sicher, dass Sie Zeilenumbrüche im Windows-Format auschecken aber im Unix-Format einchecken. Linux, macOS und Windows verwenden unterschiedliche Zeichen für einen Zeilenumbruch. Mit dieser

Einstellung können Sie mit Ihren Kommilitonen zusammenarbeiten, auch wenn diese ein anderes Betriebssystem verwenden. Mehr zum Hintergrund: `https://en.wikipedia.org/wiki/Carriage_return`, `https://en.wikipedia.org/wiki/Newline#Representation`.
Wählen Sie MinTTY als Terminal-Emulator.

I.3.4 für alle Betriebssysteme nach der Installation

Tragen Sie nach der Installation Ihren Namen und Ihre E-Mail-Adresse in die globale Git-Konfiguration ein.

II. HINTERGRUNDINFORMATIONEN

In diesem Praktikum machen Sie sich zunächst mit den Grundzügen der Versionsverwaltung mit *Git* vertraut. In einem zweiten Schritt, erstellen Sie mit Editor und Compiler eine Konsolenanwendung in der Programmiersprache C. Sie starten Ihre Anwendung von der Konsole und sehen sich den Rückgabewert an. Dann ergänzen Sie die Konsolenanwendung um eine Ausgabe.

II.1. Git-Repository

Git ist eine verteilte Quellcodeverwaltung. Damit können Änderungen an Dateien erfasst und einzelne Zwischenschritte (*commit*) abgelegt und wiederhergestellt werden. Grundsätzlich kann von jedem abgelegten Zwischenschritt abgezweigt werden (*branch*). Durch die Verteilung des *Repositories* – des Aufbewahrungsortes der Änderungshistorie – können mehrere Entwickler gleichzeitig an verschiedenen Zweigen arbeiten und die Änderungen anschließend zusammenführen (*merge*). Weiterhin benötigt *Git* dadurch keinen zentralen Server; das komplette Repository liegt lokal bei jedem einzelnen Entwickler. Geeignete Server können allerdings als gemeinsame Austausch-Plattform dienen.
In diesem Praktikum legen Sie auf einem Git-Server ein Projekt an, klonen dieses auf Ihren lokalen Rechner, legen eine Textdatei unter Versionsverwaltung an und stellen diese Ihren Kommilitoninnen und Kommilitonen zur Verfügung. Änderungen legen Sie nacheinander ab. Verzweigungen werden Sie noch nicht nutzen.
Machen Sie sich mit den Grundzügen der Anwendung von Git vertraut. Gute Tutorials finden Sie beispielsweise hier: `https://www.atlassian.com/git/tutorials`

II.2. Konsolenanwendung

Eine *Konsolenanwendung* ist ein Programm, das von der Kommandozeile aufgerufen werden kann und das – wenn überhaupt – lediglich ein Text-basiertes Interface zur Verfügung stellt. Die Bedienung erfolgt über die Tastatur.
Konsolenanwendungen geben am Ende immer eine Ganzzahl an die Kommandozeile zurück, die für den Benutzer zunächst nicht sichtbar ist, die aber ausgewertet werden kann. Diese Ganzzahl informiert in der Regel über aufgetretene Fehler. Der Rückgabewert `0` signalisiert üblicherweise „keine Fehler“. Die Bedeutung anderer Rückgabewerte ist anwendungsabhängig.
Unter Linux – zumindest in den Shells `bash`, `ksh`, `tcsh` und `sh` – ist der zuletzt zurückgegebene Wert in der Variable `?` gespeichert. Er kann durch folgendes Kommando ausgegeben werden:

```
$ echo $?
```

In der Windows-Kommandozeile `cmd` ist der zuletzt zurückgegebene Wert in der Variable `errorlevel` gespeichert. Er kann durch folgendes Kommando ausgegeben werden:

```
> echo %errorlevel%
```

Zunächst nutzen Sie den Rückgabewert gewissermaßen als „Ausgabe“ Ihres Programms. Im abschließenden Teil greifen Sie der Vorlesung etwas vor und nutzen bereits in diesem Praktikum die Standardausgabe.

III. AUFGABEN – TEIL 1

1. Hello Me

In dieser Aufgabe erstellen Sie ein Projekt auf einem Git-Server, klonen dieses auf Ihren lokalen Rechner, verändern Dateien und stellen diese Ihren Kommilitoninnen und Kommilitonen zur Verfügung.

(a) Stellen Sie sicher, dass Sie einen Nutzer-Account auf einem geeigneten Git-Server haben. Studierende des Fachbereichs Wirtschaft und Informationstechnik der Westfälischen Hochschule können (und sollten) den hochschuleigenen Server `https://git.w-hs.de` nutzen. Hier melden Sie sich mit Ihrer Hochschulkennung an. Kommerzielle Alternativen sind z.B. `https://gitlab.com`, `https://github.com` und `https://bitbucket.org`.

(b) Legen Sie auf dem Git-Server ein eigenes Projekt für diese Aufgabe an und fügen Sie Ihre Gruppenmitglieder diesem Projekt hinzu. Verwenden Sie auf dem hochschuleigenen Server einen Projektnamen nach folgendem Muster – dann können Sie es später einfacher zuordnen:

```
GIP1-2023-[Gruppe]-[Praktikumsnummer]-[Aufgabennummer]-[Durchgang]
```

(c) Klonen Sie das Projektrepository auf Ihren lokalen Rechner (`clone`).

(d) Legen Sie in Ihrer lokalen Kopie des Repositories eine neue Textdatei an. Füllen Sie diese Textdatei mit einem kurzen Steckbrief von Ihnen.

(e) Stellen Sie die Textdatei unter Versionskontrolle (`add`, `commit`) und veröffentlichen Sie diese auf dem Server (`push`).

(f) Reihum holen sich die übrigen Gruppenmitglieder die vorhandenen Steckbriefe vom Server (`pull`) und ergänzen eigene Steckbriefe.

IV. AUFGABEN – TEIL 2

2. Hello 42

In dieser Aufgabe erstellen Sie eine Konsolenanwendung, die lediglich einen einzelnen Wert zurückgibt.

(a) Legen Sie auf Ihrer Festplatte einen Ordner für dieses Praktikum an. Öffnen Sie die Kommandozeile und wechseln Sie in diesen Ordner. Erzeugen Sie in diesem Ordner ein leeres *Git-Repository*. Wählen Sie für das Repository einen geeigneten Namen.

(b) Erstellen Sie mit Ihrem Texteditor im Hauptverzeichnis Ihres Repositories eine Datei `main.c` mit folgendem Inhalt:

```c
int main() {
  return 42;
}
```

(c) Legen Sie im Hauptverzeichnis Ihres Repositories das sogenannte *Build-Directory* an, das die erzeugte Konsolenanwendung enthalten wird. Typischerweise wird hierfür der Name `build` verwendet.

(d) Erzeugen Sie Ihre Konsolenanwendung aus oben angelegter Datei `main.c`, indem Sie in der Kommandozeile im Hauptverzeichnis Ihres Repositories folgenden Befehl eingeben:

```
$ gcc -o build/main main.c
```

Dieser Befehl hat die Form

```
$ gcc -o OUTPUT_FILE INPUT_FILE
```

`INPUT_FILE` gibt die zu übersetzende Quellcodedatei an, `OUTPUT_FILE` die zu erzeugende, ausführbare Datei – die Konsolenanwendung. Achten Sie darauf, dass Sie die richtigen Verzeichnisse verwenden.

(e) Welchen Rückgabewert erwarten Sie, wenn Sie Ihre Konsolenanwendung ausführen?

(f) Führen Sie Ihre Konsolenanwendung aus. Überprüfen Sie, ob der Rückgabewert korrekt ist. Entspricht der Rückgabewert nicht Ihren Erwartungen, korrigieren Sie mögliche Fehler.

(g) Funktioniert Ihre Konsolenanwendung fehlerfrei, merken Sie die Änderungen an Ihrem Projekt – also die Datei `main.c` – zur Ablage im Repository vor (`add`).

(h) Überprüfen Sie nun die Änderungen im einzelnen (`diff --cached`). Vermeiden Sie sogenannte *White-space-Only* Änderungen, also solche, die nur aus Leerzeichen und Tabulatoren bestehen. Oft werden derartige Änderungen rot hervorgehoben, um dem Benutzer das Auffinden zu erleichtern. Sollten Sie hier weitere Änderungen vornehmen, merken Sie diese ebenfalls zur Ablage im Repository vor.

(i) Überprüfen Sie den Zustand (`status`) Ihres Repositories – genauer gesagt der *Arbeitskopie*. Ist der Zustand korrekt, legen Sie die vorgemerkten Änderungen im Repository ab (`commit`). Vergeben Sie eine aussagekräftige *Commit-Message*.

(j) Sehen Sie sich die Änderungshistorie (`log`) an.

(k) Stellen Sie Ihr Repository den übrigen Gruppenmitgliedern zur Verfügung.

3. Hello other number

In dieser Aufgabe ändern Sie den Rückgabewert und legen diese Änderung im Repository ab.

Arbeiten Sie mit dem Ergebnis der vorherigen Aufgabe weiter. Öffnen Sie dazu eine Kommandozeile und wechseln Sie in Ihr Projektverzeichnis – nicht das Build-Directory.

(a) Überlegen Sie sich selbst einen Rückgabewert zwischen 0 und 255. Ändern Sie die Datei `main.c` entsprechend und erzeugen Sie Ihre Konsolenanwendung erneut.

(b) Führen Sie Ihre Konsolenanwendung aus, überprüfen Sie den Rückgabewert und beheben Sie ggf. auftretende Fehler.

(c) Sehen Sie sich den Zustand der Arbeitskopie Ihres Repositories an. Sehen Sie sich die Änderungen (`diff`) in `main.c` an. Entspricht die angezeigte Änderung Ihren Erwartungen, merken Sie die Änderungen an `main.c` zur Ablage im Repository vor.

(d) Sehen Sie sich den veränderten Zustand Ihres Repositories an. Legen Sie die vorgemerkten Änderungen tatsächlich im Repository ab. Vergeben Sie eine aussagekräftige Commit-Message. Überprüfen Sie mit ob die Änderungen abgelegt sind. Sehen Sie sich den veränderten Zustand Ihres Repositories an.

V. Aufgaben – Teil 3

4. Hello World

In dieser Aufgabe nutzen Sie die Standardausgabe, um Text aus Ihrem Programm auf der Konsole auszugeben. Arbeiten Sie dazu mit dem Ergebnis der vorherigen Aufgabe weiter.

(a) Lesen Sie die Abschnitte 1 und 2 des C-Tutorials von Brian W. Kernighan[1], einem der beiden Autoren des ursprünglichen C-Handbuchs „The C Programming Language“ von 1978. Stören Sie sich nicht daran, dass dort erwartet wird „that you have programmed in some language before“. Diese Erfahrung werden Sie sich in Vorlesung und Praktikum erarbeiten.

[1] https://www.bell-labs.com/usr/dmr/www/ctut.pdf

(b) Implementieren Sie das Hello-World-Programm aus dem C-Tutorial. Fügen Sie dabei in Ihrem Programm eine neue erste Zeile ein:

```
#include <stdio.h>
```

Diese ist im Tutorial nicht abgedruckt.

(c) Erzeugen Sie Ihre Konsolenanwendung mit dem *GCC*. Beheben Sie ggf. auftretende Fehler.

(d) Welche Ausgabe erwarten Sie, wenn Sie Ihre Konsolenanwendung ausführen?

(e) Führen Sie die Konsolenanwendung aus und überprüfen Sie die Ausgabe. Beheben Sie ggf. auftretende Fehler.

(f) Legen Sie Ihre Änderungen im Repository ab.

5. Hello Module

In dieser Aufgabe erstellen Sie mehrere Konsolenanwendungen, die jeweils ein Modulkürzel aus Ihrem Studiengang ausgeben. Diese Aufgabe ist bewusst eine Fingerübung, um das bisher Gelernte zu trainieren. Je besser Sie Ihre Werkzeuge beherrschen, desto leichter fällt Ihnen der „mechanische" Teil des Programmierens und Sie können sich auf die Problemlösung konzentrieren.

(a) Laden Sie sich von der Webseite Ihres Fachbereichs das Modulhandbuch Ihres Studiengangs herunter. Beginnen Sie beim ersten Modul.

(b) Legen Sie auf Ihrer Festplatte im Ordner für dieses Praktikum einen neuen Unterordner mit Namen `hello_module` an. Erzeugen Sie in diesem Ordner ein leeres Git-Repository.

(c) Legen Sie mit Ihrem Texteditor in diesem Ordner eine Datei namens `MODULE.c` an, wobei `MODULE` für das Modulkürzel steht. Implementieren Sie in dieser Datei eine Konsolenanwendung, die lediglich das entsprechende Modulkürzel ausgibt.

(d) Erzeugen Sie die Konsolenanwendung namens `MODULE`, wobei `MODULE` für das Modulkürzel steht. Führen Sie die Konsolenanwendung aus. Beheben Sie jeweils ggf. auftretende Fehler.

(e) Legen Sie die Änderung im Repository ab.

(f) Wiederholen Sie die Schritte (c) bis (e) mindestens für zehn Module *ohne* dabei von dem Code abzuschreiben, den Sie in den vorherigen Durchläufen erzeugt haben.