

C++ Workshop

1. Block, 27.04.2012

Markus Jung, Christian Käser, Robert Schneider | 27. April 2012

```
БЭИСИК  ДВК  НЧ
001
НУЖНЫ ВАМ РАСШИРЕННЫЕ ФУНКЦИИ ?1
ВЫСОКОСКОРОСТНЫЕ УСТ-СТВА?1
УСТАН-КА ВНЕШНЕЙ ФН-ЦИИ?1
03У ?48
ЖДУ
5 LET A=1.0000001
10 LET B=A
15 FOR I=1 TO 27
20 LET A=A*A
25 LET B=B-2
30 NEXT I
35 PRINT A,B,"WWW.LENINGRAD.SU/MUSEUM"
WWW.LENINGRAD.SU/MUSEUM
RUN
568044 1202420
ОСТ СТРОКЕ 35
ЖДУ
```

Quelle: Wikimedia Commons
CC-BY-SA Sergei Frolov

- 1 Organisatorisches
 - Platzzahl
 - Organisatorisches
- 2 Versionsverwaltung
 - Motivation
 - Konzepte
 - GIT
 - Weiterführende Hinweise
- 3 Entwicklungsumgebung
- 4 Praxis

- 1 Organisatorisches
 - Platzzahl
 - Organisatorisches

Warum eine Begrenzung?

- Raumgröße
- Betreuung, Gruppengröße

Warum eine Begrenzung?

- Raumgröße
- Betreuung, Gruppengröße

Platzzahl

- Max. 24 Plätze, Betreuer zusätzlich
- Warteschlange (wie bei Sprachkurs)
- Online-Teilnahme möglich
 - Die Vorträge werden (versuchsweise) als Screencast aufgezeichnet
 - Fragen können über die Mailinglist gestellt werden
 - ... oder über unseren IRC-Channel

Wie bekomme ich einen Platz?

Nach folgenden Kriterien werden die Plätze verteilt:

- ① Anwesenheit heute (27.4.) oder Entschuldigung
- ② Erforderliche Programmierkenntnisse
 - „Programmieren für Physiker“ als Referenz
 - Codebeispiel zum ersten Termin in `example_apps/easy.cpp` als Minimalanforderung
- ③ Aktive Teilnahme und Interesse (der *Willen*, teilzunehmen und das durchzuziehen)
- ④ first-come, first-served (Wer zuerst kommt, mahlt zuerst.)

Nachrücken ist möglich!

Mailing-List `cpp-workshop@lists.kit.edu`

IRC `#kit-cpp-workshop` auf `irc.euirc.net`

GitHub `www.github.com`, Organization `kit-cpp-workshop`

Namen und E-Mail-Adressen:

Christian Käser	Christian.Kaeser@student.kit.edu
Markus Jung	Markus.Jung@stud.uni-karlsruhe.de
Matthias Blaicher	matthias@blaicher.com
Robert Schneider	Robert.Schneider3@student.kit.edu
Sven Brauch	SvenBrauch@gmail.com
Oliver Schneider	mail@oli-obk.de

Termin

- jeden Freitag in der Vorlesungszeit
- 15:45-17:15
- Raum: 2-0 (Physik-Hochhaus)
- evtl. Zusatztermine nach Absprache (z.B. cmake)

Termin

- jeden Freitag in der Vorlesungszeit
- 15:45-17:15
- Raum: 2-0 (Physik-Hochhaus)
- evtl. Zusatztermine nach Absprache (z.B. cmake)

Ziele

- Erfahrungsaustausch!
- Hilfe zur Selbsthilfe / Anhaltspunkte zum Weiterlernen
- Tiefergehenderer Einblick in die Sprache
- Verschiedene Gebiete von C++ („Schweizer Taschenmesser“)
- Kooperation, Team-Programmierung
- Software-Design, programming idioms, Fehler vermeiden usw.

Themen

- Richten sich nach den Interessen der Teilnehmer!
- Grundlegende Methodiken, Werkzeuge, ...
- Objektorientierung
- Algorithmen & Datenstrukturen
- Verwendung von Bibliotheken (statisch, dynamisch gelinkt)

Themen

- Richten sich nach den Interessen der Teilnehmer!
- Grundlegende Methodiken, Werkzeuge, ...
- Objektorientierung
- Algorithmen & Datenstrukturen
- Verwendung von Bibliotheken (statisch, dynamisch gelinkt)

Aufteilung

- Theoretischer Teil, heute: Organisatorisches, git, eclipse
- Praktischer Teil: pair programming (2er-Teams) IDE + hello-world
- Übungen: im praktischen Teil, zu Hause fertig. Vorstellung einiger Lösungen in darauffolgender Woche

2 Versionsverwaltung

- Motivation
- Konzepte
- GIT
- Weiterführende Hinweise

- „Gestern ging es noch . . .“
- „Das Problem hatten wir doch schon Mal . . .“
- „Wer hat den ?&(%!\$ verzapft!?“

- „Gestern ging es noch . . .“
- „Das Problem hatten wir doch schon Mal . . .“
- „Wer hat den ?&(%!\$ verzapft!“

Aufgaben

- Archivierung der Entwicklungsgeschichte
 - „Zeitmaschine“
- Koordination paralleler Entwicklungsvorgänge

Version/Revision/Commit

Eine Momentaufnahme der verwalteten Dateien, verknüpft mit Metadaten wie Autor, Datum und einer Beschreibung

Versionsgeschichte

Stellt den Bezug zwischen einzelnen Versionen her

Repository (kurz: Repo)

Speichert und verwaltet die Versionsgeschichte sowie die zugehörigen Versionen

Arbeitskopie

Ein Abbild einer Version im Dateisystem, an dem auch Änderungen vorgenommen werden können

Log - /mnt/vgluks/data/cuviso/Whiteboard/cdev-usbxx/Firmware/src

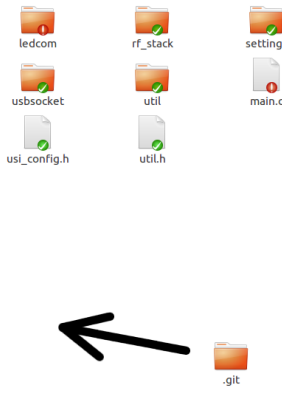
Revisions Table Showing Revisions: 3a3164b to 2b9b0e4

Grä	Revision	Autor	Datum	Logmeldung
	3a3164b	Markus Jung	Mi 02:22	fixes garbage sent to host because of wrong rf-stack signal
	7518cb1	Markus Jung	Apr 11 2012	Einführung USB-Socket im Testbetrieb
	4181c3c	Markus Jung	Apr 10 2012	Kleine Code-/Stilkorrekturen am fifo
	a728397	Markus Jung	Apr 05 2012	Ersetzt alten CircularBuffer durch neuen FiFo
	884faf3	Markus Jung	Apr 02 2012	Datenverarbeitung zum PC verlagert Integration von clib-cd
	0ee2010	Markus Jung	Mär 30 2012	settings.c: Auf RF_CLOCK_CONFIG umgestellt
	2212d07	Markus Jung	Mär 30 2012	Formatierung + Schwellenwertanpassung von 4 auf 15
	fcd52e9	Markus Jung	Mär 30 2012	Fix für verdrehte Tasterreihenfolge wieder entfernt (wurde be
	e0af28a	Markus Jung	Mär 30 2012	Umbau auf Maus-Reports Button-Reihenfolge softwareseitig
	ee3c866	Markus Jung	Mär 30 2012	Handling der Funk und USB-Inputs korrigiert (IN_RANGE_FLAG

Affected File(s) (double-click to compare with base) Message

Aktion	Pfad	Message
+23/-21	Firmware/src/main.c	Formatierung + Schwellenwertanpassung von 4 auf 15
+3/-3	Firmware/src/settings.c	

Zurück Vor Limit: 100 Schließen





Dezentral

- Jeder Nutzer hat ein (lokales) Repository
- Erfordert keinen kontinuierlichen Zugriff auf den Server

Branch

- Softwareentwicklung ist selten linear („Entwicklungsweig“)
- Versionsverwaltungen bilden das durch „branches“ ab

Merge

- Führt Entwicklungsweige zusammen
- Konkurrierende Änderungen können Konfliktlösung erfordern

Tag

Markiert Versionen zum schnellen Zugriff

github

- Hosting für GIT-Repositories
- Für Open-Source-Entwicklung kostenlos
- „social coding“

Fork

- Eine Abspaltung eines GIT-Repositories
 - Jeder `clone` eines Repositories ist eigentlich ein Fork
- Eigene Entwicklung finden im eigenen Fork statt

Pull-Request

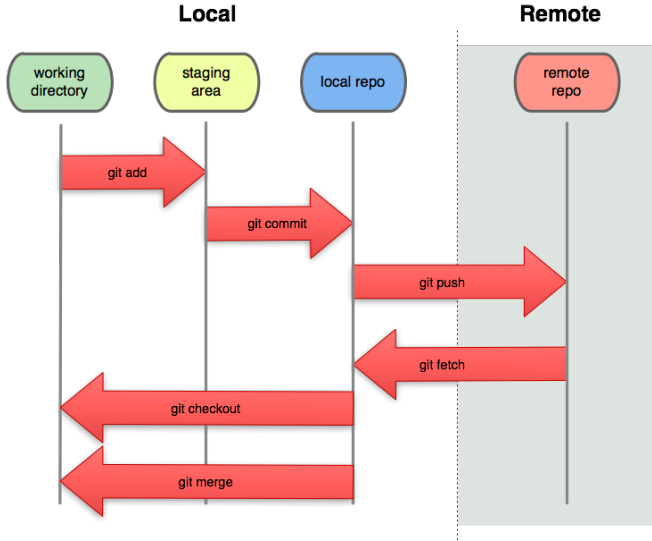
- Aufforderung an den Besitzer eines externen Repositories, Änderungen zu übernehmen

Wie nutzen wir GitHub?

Derzeit (pair programming) ist der Workflow:

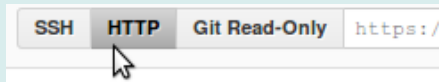
- 1 Wir hosten die Materialien zu einem Workshop in einem *repo*.
- 2 Du forkst das repo in deinen GitHub-Account.
- 3 Dann clonest du dir *deinen Fork* auf deinen Rechner (`git clone`).
- 4 Jetzt kannst du mit deiner lokalen Kopie arbeiten, z.B. die Aufgaben lösen.
- 5 Nutze Git auch lokal zur Versionsverwaltung! Committe häufig!
- 6 Veröffentliche deine Änderungen mit `git push` (damit landen sie in deinem Fork bei GitHub).
- 7 Erstelle nach dem Lösen *aller* Aufgaben auf GitHub einen *pull request* und nenne deinen Betreuer im Text.
- 8 Ein Betreuer wird über deine Lösung schauen, diese kommentieren *und* den *pull request ablehnen*.

GIT Workflow



Repository erzeugen

- Leeres neues Repository: `git init [<Ordnername>]`
- Kopie eines existierenden Repositories:
`git clone <URL> [<Ordnername>]`
 - Es gibt verschiedene Protokolle zum Zugriff auf ein Repository
 - SSH ist üblich und bequem, erfordert aber einen SSH-Key
 - Wir empfehlen daher (bis auf weiteres) die Nutzung von HTTPS



Änderungen vornehmen

- Neue/geänderte Datei für die nächste Revision vormerken:
`git add <Datei/Ordner>`
- Datei aus der Versionskontrolle entfernen:
`git rm [--cached] [-r] <Datei/Ordner>`
- Vorgemerkte Änderungen verwerfen: `git reset <Datei/Ordner>`
- Änderungen übernehmen (neue Revision erzeugen):
`git commit [-m "commit message"]`
Commit-Message-Styleguide:
 - Eine Zeile Kurzzusammenfassung
 - Danach ggf. eine Leerzeile und weiterer Text
 - Zeilenlänge ≤ 72 Zeichen

Austausch mit anderen Repositories

- Änderungen an externes Repository übertragen:
`git push [--all] [--tags] [<Repository>] [<Branch>]`
- Änderungen aus externem Repository beziehen:
`git fetch [<Repository>]`
- Änderungen aus externem Repository beziehen und anwenden:
`git pull [<Repository>]`
- Verwaltung externer Repositories: `git remote`

Verzweigungen und ähnlicher Wildwuchs

- Aktuelle Revision/Branch der Arbeitskopie wechseln:
`git checkout [-b] [<Branch>]`
- Verwaltung von Branches: `git branch`
- Zweig mit eigenem zusammenführen: `git merge <Branch>`
- Tagging: `git tag <Tag> [<Revision>]`

Verschiedenes

- Status der Arbeitskopie: `git status`
- History/Protokoll: `git log`
- Name setzen: `git config user.name <Name>`
- E-Mail setzen: `git config user.email <EMail>`

- `man git beziehungsweise git help`
- <http://help.github.com/> - GitHub-Hilfe
- <http://www.youtube.com/watch?v=4XpnKHJAok8>
Google Talk "Linus Torvalds on Git"
- <http://git-scm.com/> - Offizielle Website
- <https://github.com/Gazler/githug> - Interaktives Git Tutorial
- <http://gitref.org/> - Git Referenz
- <http://progit.org/book/> - The Pro Git Book
- <http://gitready.com/> - Git tips and tricks

3 Entwicklungsumgebung

IDE

Integrated Development Environment

IDE

Integrated Development Environment

Sammlung *und* Integration von Werkzeugen

- intelligenter Texteditor
- grafischer Debugger
- Autovervollständigung
- einfache Anbindung an den Compiler
- Hilfefunktion
- meist erweiterbar (z.B. git-Integration)

- grafisch** IDEs sind üblicherweise grafisch, man kann daher viele Dinge über GUIs konfigurieren und anstoßen
- integriert** es werden zahlreiche Werkzeuge miteinander vereint, z.B. ein Text-Editor mit einem Debugger
- strukturiert** in der IDE wird zumeist auch die Struktur eines Projektes abgebildet (Ressourcen, source code usw.)
- editor** die IDE kennt das gesamte Projekt und kann daher die Informationen aus allen Dateien nutzen (autocompletion, definition lookup etc.)
- code analysis** durch ihre umfassendes Wissen kann die IDE auch den code analysieren, etwa um die Struktur von Klassen darstellen zu können
- refactoring** das Umstrukturieren von code ist in gewissen Grenzen ebenfalls automatisiert möglich

- Code::Blocks
- *Eclipse*
- GNU Emacs
- KDevelop
- NetBeans
- Qt Creator
- proprietär: Visual Studio (Microsoft / Windows)
- proprietär: Xcode (Apple / OS X, iOS)
- ...

4 Praxis

- Aufgabe 1: Hello World
- Aufgabe 2: Fibonacci-Zahlen
- Bonusaufgabe: Project Euler

URL zum Workshop:

<https://github.com/kitt-cpp-workshop/workshop-ss12-01>

Die Aufgabenbeschreibungen und weiterführende Hinweise enthält die Datei `README.md`