

# Übungen zu Systemprogrammierung 1

Üo – Einführung

---

Sommersemester 2022

Dustin Nguyen, Jonas Rabenstein, Christian Eichler, Jürgen Kleinöder

Lehrstuhl für Informatik 4  
Friedrich-Alexander-Universität Erlangen-Nürnberg



Lehrstuhl für Verteilte Systeme  
und Betriebssysteme



FRIEDRICH-ALEXANDER  
UNIVERSITÄT  
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT



0.1 Allgemeines

0.2 Organisatorisches

0.3 Linux-Kenntnisse

0.4 Versionsverwaltung mit SVN

0.5 SP-Abgabesystem



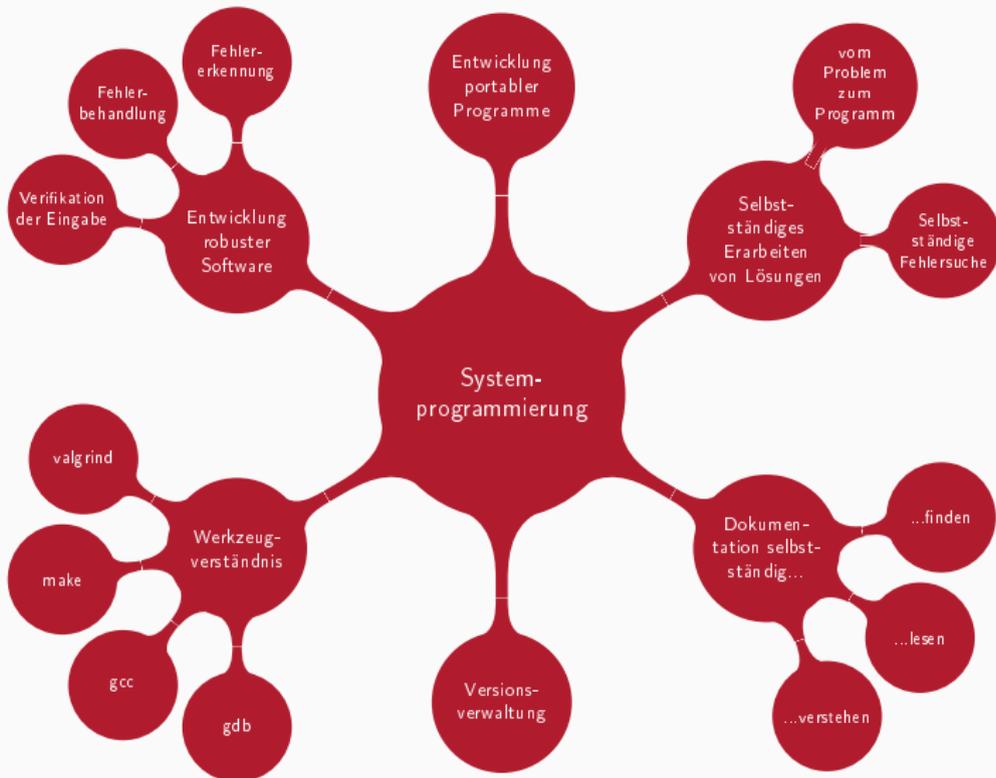
0.1 Allgemeines

0.2 Organisatorisches

0.3 Linux-Kenntnisse

0.4 Versionsverwaltung mit SVN

0.5 SP-Abgabesystem





## Tafelübungen

- Vorstellung von Betriebssystemkonzepten und Werkzeugen
- Einführung in die Verwendung der Schnittstellen
- Erarbeiten eines kleinen Programmes (Demo)
- Rechnerübungen für inhaltliche Fragen
- Besprechung der Abgaben und allgemeiner Fallstricke

## Praktischer Teil – Aufgaben

- Arbeiten mit der Betriebssystemschnittstelle
- Fehlersuche und Fehlerbehebung
- Verwenden der vorgestellten Werkzeuge
- Hilfestellung in der Rechnerübungen



0.1 Allgemeines

**0.2 Organisatorisches**

0.3 Linux-Kenntnisse

0.4 Versionsverwaltung mit SVN

0.5 SP-Abgabesystem



- Ausgabe neuer Aufgaben in den Tafelübungen
  - Aufgabenstellung meist recht knapp
    - Nicht alles bis in letzte Detail spezifiziert
  - Gegebene Spezifikationen sind zwingend einzuhalten
- Selbstständiges Bearbeiten der Aufgaben
  - bei Problemen hilft z. B. ein Besuch in den Rechnerübungen
- Korrektur und Bewertung erfolgt durch die Tutoren
  - Korrekturen werden elektronisch (via SVN) zur Verfügung gestellt
  - eigenes Ergebnis nach Login im *WAFFEL* einsehbar
  - Korrekturrichtlinien sind auf der Webseite dokumentiert
- Übungspunkte können das Klausurergebnis verbessern (Notenbonus)
  - Abschreibetests
  - Vorstellen der eigenen Lösungen
    - **Anwesenheit in Besprechungsübungen für Bonuspunkte**



- Bearbeitungszeitraum ist angegeben in Werktagen (Mo. bis Fr.)
  - Bearbeitungszeitraum beinhaltet den Tag der Tafelübung
  - Feiertage und der „Berg-Dienstag“ (nach Pfingsten) sind nicht enthalten
  - Abgabetermin kann per Skript erfragt werden
- plant für die Bearbeitung einer Aufgabe **mindestens** 8–16 Stunden (in Worten: ein bis zwei **Tage**) ein
  - langer Bearbeitungszeitraum bietet euch Flexibilität bei der Arbeitsverteilung
  - Feedback über wirkliche Bearbeitungszeit erwünscht



- Forum: [https://studon.fau.de/crs2958245\\_join.html](https://studon.fau.de/crs2958245_join.html)
  - inhaltliche Fragen zum Stoff oder den Aufgaben
  - allgemein alles, was auch für andere Teilnehmer interessant sein könnte
- Mailingliste: [i4sp@cs.fau.de](mailto:i4sp@cs.fau.de)
  - geht an alle Tutoren
  - Angelegenheiten, die nur die eigene Person/Gruppe betreffen
- Mailingliste: [i4sp-orga@cs.fau.de](mailto:i4sp-orga@cs.fau.de)
  - geht an die SP-Organisatoren
  - Fragen zur Organisation und zum Übungsbetrieb
- Rechnerübungen (siehe Homepage)
  - Hilfe bei konkreten Problemen (z. B. Quellcode kompiliert nicht)
  - **kein** Händchenhalten, während ihr die Tastatur bedient :)
- der korrigierende Tutor
  - Fragen zur Korrektur, vergessener Gruppenbonus
  - fälschlicherweise positiver Abschreibetest



0.1 Allgemeines

0.2 Organisatorisches

**0.3 Linux-Kenntnisse**

0.4 Versionsverwaltung mit SVN

0.5 SP-Abgabesystem



- Grundkenntnisse zur Linux- und Shell-Nutzung werden voraus gesetzt.
- Lehrmaterial gibt es bei der FSI Informatik
  - `https://fsi.cs.fau.de/dw/informationen/ese/2019ws/linuxkurs`
  - Eine Aufzeichnung des Kurses ist ebenfalls verfügbar
  - Während der ersten Wochen können auch in der Rechnerübung fragen gestellt werden



- Anmeldung im CIP unter <https://remote.cip.cs.fau.de>
- Auflistung der Rechnerausstattung:  
<https://wwwcip.cs.fau.de/cipPools/roomIndex.en.html>
- Liste der SSH Host Keys  
<https://wwwcip.cs.fau.de/documentation/sshostkeys.en.html>
- Benutzung über den Webbrowser  
<https://remote.cip.cs.fau.de>



- Aufgeteilt in verschiedene *Sections*
  - 1 Kommandos
  - 2 Systemaufrufe
  - 3 Bibliotheksfunktionen
  - 5 Dateiformate (Spezielle Datenstrukturen etc.)
  - 7 verschiedenes (z. B. Terminaltreiber, IP)
- Angabe normalerweise mit *Section*: `printf(3)`
- Aufruf unter Linux:

```
$ # man [section] begriff  
$ man 3 printf
```
- Suche nach *Sections*: `man -f begriff`
- Suche nach Manual-Pages zu einem Stichwort:

```
user@host:~$ man -k stichwort
```
- **Achtung:** Manual-Pages unter Mac OS oft abweichend von Linux  
⇒ CIP ist Referenzsystem!





0.1 Allgemeines

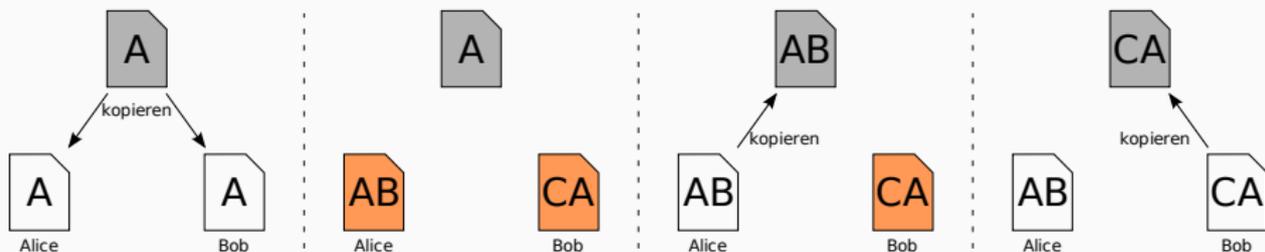
0.2 Organisatorisches

0.3 Linux-Kenntnisse

**0.4 Versionsverwaltung mit SVN**

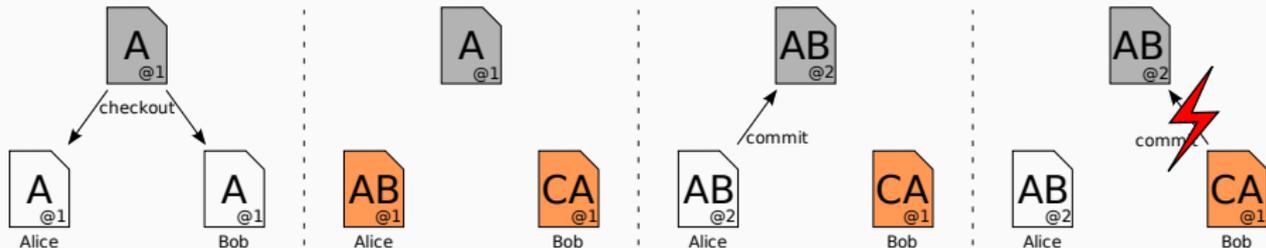
0.5 SP-Abgabesystem

## ■ Gemeinsames Bearbeiten einer Datei kann zu Problemen führen:

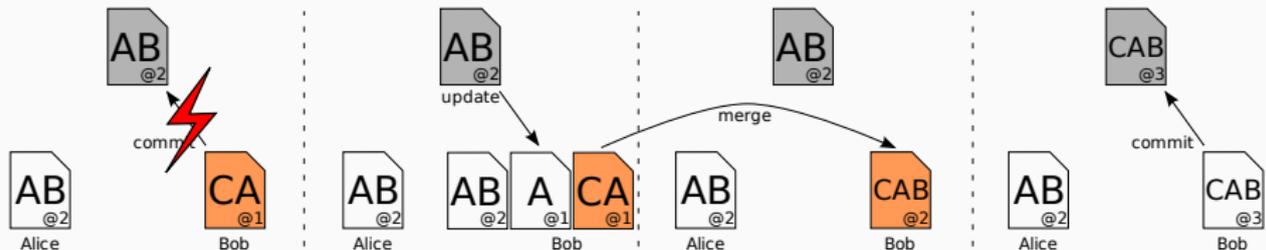


- Modifikationen werden nicht erkannt
- Änderungen von Alice gehen unbemerkt verloren

## ■ Versionsnummer zur Erkennung von Modifikationen

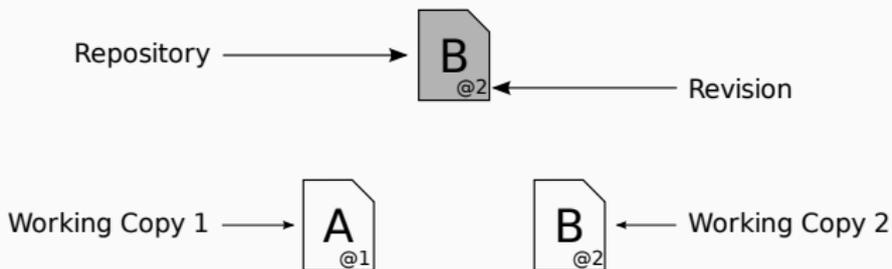


## ■ Entstandener Konflikt muss lokal gelöst werden





- SVN bietet Versionsverwaltung für Dateien und Verzeichnisse
- Speichert Zusatzinformationen zu jeder Änderung
  - Name des Ändernden
  - Zeitpunkt
  - Kommentar
- Ausführliche SVN-Dokumentation im Subversion-Buch  
<http://svnbook.red-bean.com>
- Kommando `svn`
- SP-Abgabesystem verwendet Subversion



- **Repository: zentrales Archiv aller Versionen**
  - Zugriff erfolgt beispielsweise per Internet
- **Revision (Versionsnummer)**
  - Fortlaufend ab Revision 0
- **Working Copy (Arbeitskopie)**
  - lokale Kopie einer bestimmten Version des Repositories
  - kann versionierte und unversionierte Dateien und Verzeichnisse enthalten
  - es kann mehrere Arbeitskopien zu einem Repository geben (z. B. CIP und daheim)

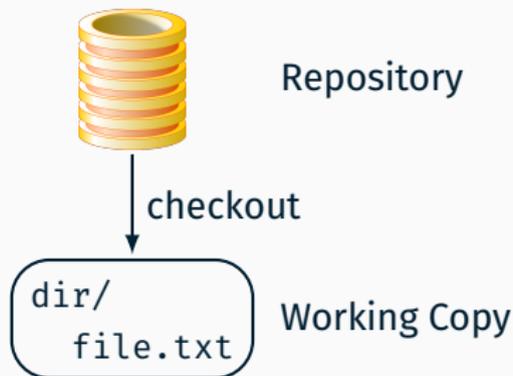


Repository

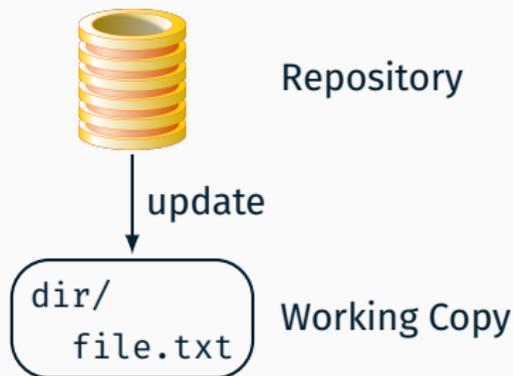


Working Copy

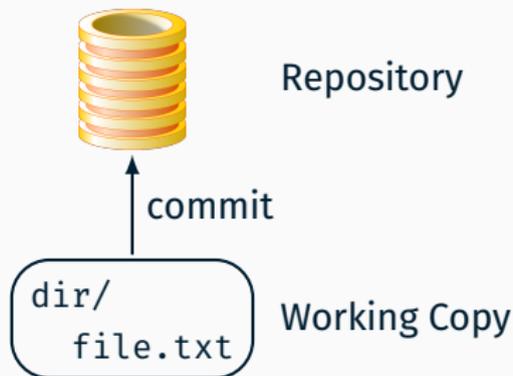
- `svn checkout/co`: Anlegen einer neuen Arbeitskopie
- `svn update/up`: Neuste Revision aus dem Repository holen
  - Bezieht sich auf aktuelles Verzeichnis und alle enthaltenen Verzeichnisse
- `svn commit/ci`: Einbringen einer neuen Version in das Repository



- `svn checkout/co`: Anlegen einer neuen Arbeitskopie
- `svn update/up`: Neuste Revision aus dem Repository holen
  - Bezieht sich auf aktuelles Verzeichnis und alle enthaltenen Verzeichnisse
- `svn commit/ci`: Einbringen einer neuen Version in das Repository



- `svn checkout/co`: Anlegen einer neuen Arbeitskopie
- `svn update/up`: Neuste Revision aus dem Repository holen
  - Bezieht sich auf aktuelles Verzeichnis und alle enthaltenen Verzeichnisse
- `svn commit/ci`: Einbringen einer neuen Version in das Repository



- `svn checkout/co`: Anlegen einer neuen Arbeitskopie
- `svn update/up`: Neuste Revision aus dem Repository holen
  - Bezieht sich auf aktuelles Verzeichnis und alle enthaltenen Verzeichnisse
- `svn commit/ci`: Einbringen einer neuen Version in das Repository



- Beim Aufruf von `svn commit` öffnet sich ein Editor zum Eingeben des commit-Kommentars
  - Im CIP wird standardmäßig der Editor `nano` verwendet
  - Anderer Editor kann über die Umgebungsvariable `EDITOR` eingestellt werden

```
user@host:~$ export EDITOR=nano
```

- Umgebungsvariable ist nur in dieser Shell-Sitzung gültig
- Durch Eintragen des Kommandos in die Konfigurationsdatei der eigenen Shell (z. B. `.bashrc`) wird der Standardeditor für jede neue Shell geändert

- Übergabe des Kommentars als Argument von `svn commit`  

```
user@host:~$ svn commit -m "Mein Kommentar"
```



- `svn add`: Dateien unter Versionskontrolle stellen
  - Bei einer leeren Arbeitskopie müssen entsprechende Dateien oder Verzeichnisse erst eingefügt werden
- `svn del/remove/rm`: Dateien lokal löschen und nicht länger unter Versionskontrolle halten
- `svn status/st`: Änderungen der Arbeitskopie anzeigen

```
$ svn status
A  aufgabe1/lilo.txt
M  aufgabe1/lilo.c
?  aufgabe1/lilo
!  aufgabe1/lilo.o
```

A Datei wurde unter Versionskontrolle gestellt

M Dateiinhalt wurde verändert

? Datei steht nicht unter Versionskontrolle

! Datei steht unter Versionskontrolle, ist aber nicht mehr in der Arbeitskopie vorhanden



- `svn help <command>`: Integrierte Hilfe zu den Kommandos

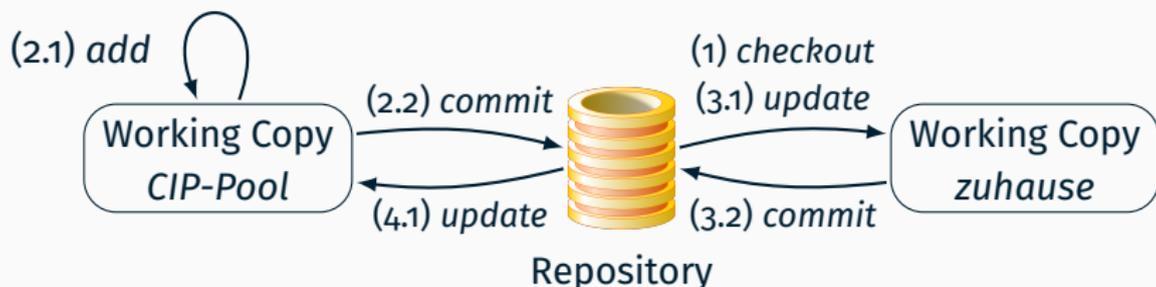
Beispiel:

```
$ svn help delete
```

```
delete (del, remove, rm): Remove files and directories from  
                           version control.
```

```
usage: 1. delete PATH...  
       2. delete URL...
```

1. Each item specified by a PATH is scheduled for deletion upon the next commit. Files, and directories that have not been committed, are immediately removed from the working copy unless the `--keep-local` option is given. PATHs that are, or contain, unversioned or modified items will not be removed unless the `--force` or `--keep-local` option is given.  
[...]



- 1 Zusätzliche Arbeitskopie(n) erstellen (checkout, einmalig)
- 2 Start der Arbeit an einer Aufgabe im CIP-Pool
  - angelegte Dateien und Verzeichnisse unter Versionskontrolle stellen (*add*)
  - Zwischenstand ins Repository einchecken (*commit*)
- 3 Arbeit zuhause fortsetzen
  - Arbeitskopie zunächst auf den aktuellen Stand bringen (*update*)
  - Zwischenstand ins Repository einchecken (*commit*)
- 4 Arbeit im CIP-Pool fortsetzen
  - Arbeitskopie zunächst auf den aktuellen Stand bringen (*update*)



0.1 Allgemeines

0.2 Organisatorisches

0.3 Linux-Kenntnisse

0.4 Versionsverwaltung mit SVN

**0.5 SP-Abgabesystem**



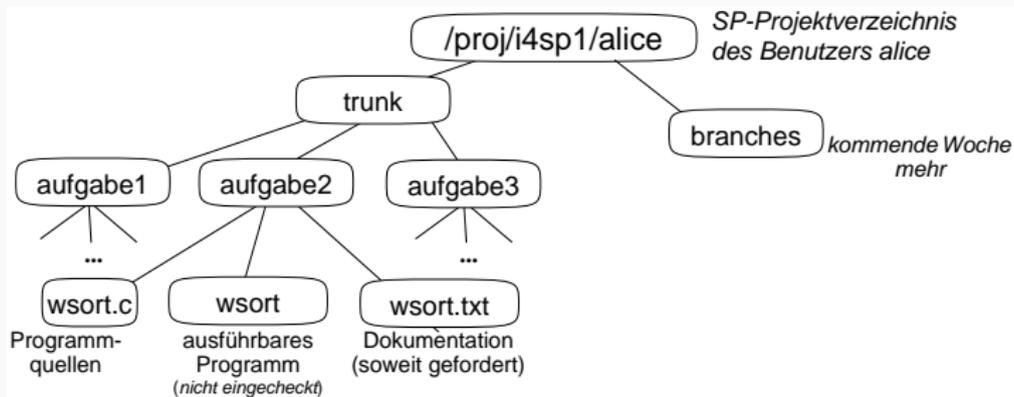
- Für jeden Teilnehmer wird folgendes bereitgestellt:
  - ein Repository `https://i4sp.cs.fau.de/ss22/sp1/<login>`
  - ein Projektverzeichnis `/proj/i4sp1/<login>` mit Arbeitskopie
    - Hinweis: Falls der Ordner `/proj/i4sp1/` nicht unter `/proj/` erscheint: **trotzdem** manuell hineinwechseln (`cd /proj/i4sp1/<login>`), das Verzeichnis wird dann automatisch eingebunden!
- Die Erzeugung erfolgt in der Nacht nach der *WAFFEL*-Anmeldung

## SVN-Passwort

Zum Zugriff auf das Repository muss ein Subversion-Passwort gesetzt werden:

```
user@host:~$ /proj/i4sp1/bin/change-password
```

→ Das Passwort wird **innerhalb der nächsten 10 Minuten** aktiv



- `trunk` enthält ein Unterverzeichnis `aufgabeX` für jede Aufgabe
- unterhalb von `branches` **nichts** editieren oder von Hand ändern



- Zur Abgabe folgendes Skript aufrufen

```
user@host:~$ /proj/i4sp1/bin/submit aufgabe1
```

- dieses gibt die aktuellste Version der Lösung zu Aufgabe 1 ab
- mehrmalige Abgabe ist möglich
  - durch erneuten Aufruf des *submit*-Skripts
  - gewertet wird die letzte rechtzeitige Abgabe
- **Eigener** Abgabetermin kann per Skript erfragt werden

```
$ /proj/i4sp1/bin/get-deadline aufgabe1
```

```
Dein Abgabezeitpunkt fuer die Aufgabe 1: lilo ist 01.01.1970  
um 17:30:00 Uhr
```



- Für einige Aufgaben stellen wir verschiedene Dateien zur Verfügung
  - Programmgerüste
  - Beispieleingaben
  - Verzeichnisbäume zum Ausprobieren des Programms
- Die Dateien befinden sich in `/proj/i4sp1/pub/aufgabe<number>`
- Manchmal ist es notwendig nur einige der öffentlichen Dateien ins eigene Projektverzeichnis zu kopieren.  
Hierzu kann das Skript `copy-public-files-for` verwendet werden:

```
user@host:~$  
/proj/i4sp1/bin/copy-public-files-for aufgabe1
```



```
alice@fau06a[~] cd /proj/i4sp1/alice/trunk
alice@fau06a[trunk] mkdir aufgabe1
alice@fau06a[trunk] cd aufgabe1
alice@fau06a[aufgabe1] nano lilo.c
...
alice@fau06a[aufgabe1] cd ..
alice@fau06a[trunk] svn add aufgabe1
A aufgabe1
A aufgabe1/lilo.c
alice@fau06a[trunk] svn commit
...
Committed revision 2.
alice@fau06a[trunk] vim aufgabe1/lilo.c
...
alice@fau06a[trunk] svn commit -m 'Bugfix in printf'
...
Committed revision 3.
alice@fau06a[trunk] /proj/i4sp1/bin/submit aufgabe1
...
# Aufgabe 1 ist jetzt abgegeben
```