

Übungen zu Systemprogrammierung 1 (SP1)

Ü1 – Einführung

Jens Schedel, Christoph Erhardt, Jürgen Kleinöder

Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität
Erlangen-Nürnberg

SS 13 – 22. bis 26. April 2013

http://www4.cs.fau.de/Lehre/SS13/V_SP1

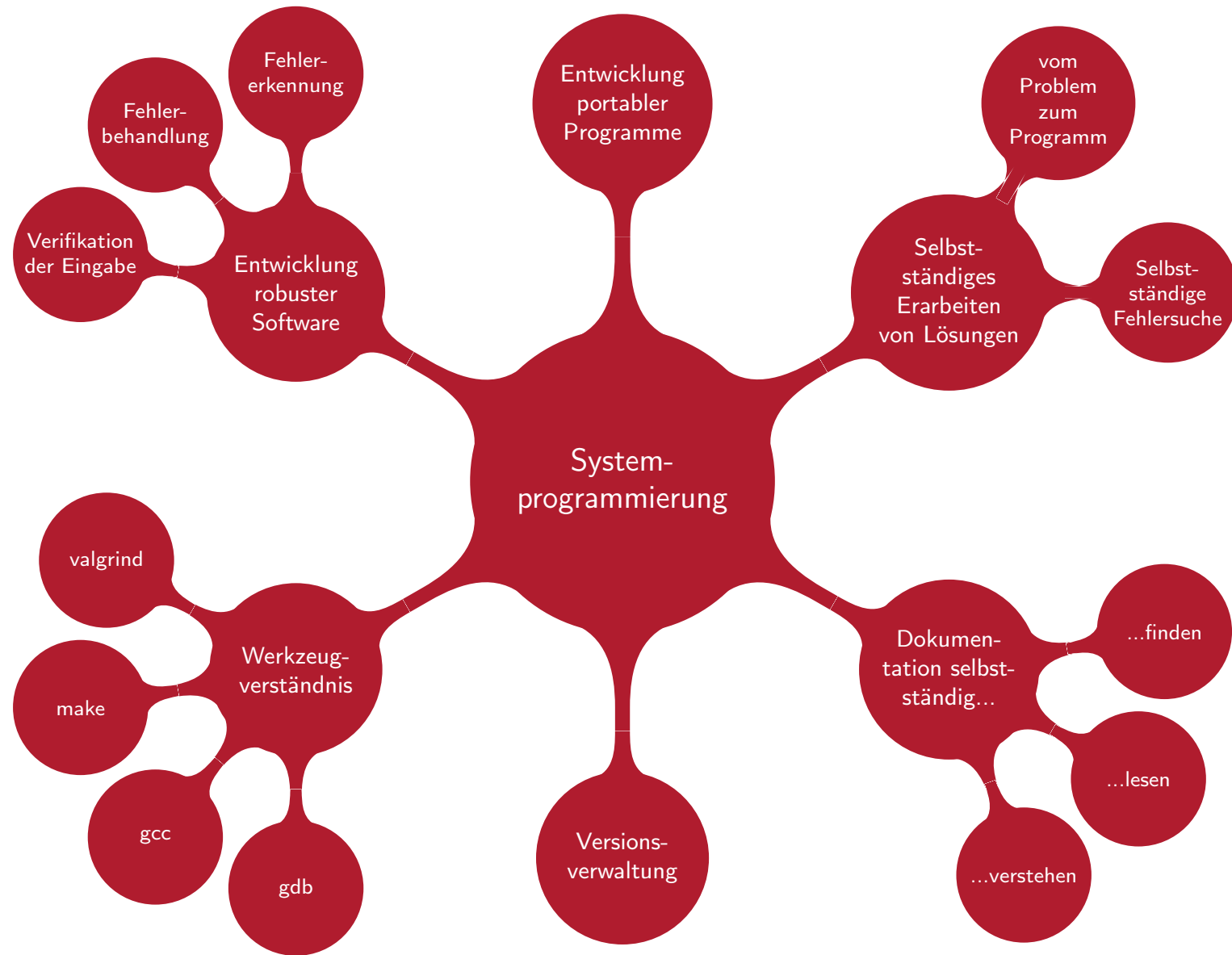


Agenda

- 1.1 Allgemeines
- 1.2 Organisatorisches
- 1.3 Linux-Kenntnisse
- 1.4 Versionsverwaltung mit SVN
- 1.5 SP-Abgabesystem
- 1.6 Gelerntes anwenden



Lernziele Systemprogrammierung



Tafelübungen und Besprechungen

- Vorstellung von Betriebssystemkonzepten und Werkzeugen
- Einführung in die Verwendung der Schnittstellen
- Erarbeiten eines kleinen Programmes (Aktive Mitarbeit!)
- Besprechung der Abgaben und allgemeiner Fallstricke

Praktischer Teil – Aufgaben

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



Agenda

- 1.1 Allgemeines
- 1.2 Organisatorisches
- 1.3 Linux-Kenntnisse
- 1.4 Versionsverwaltung mit SVN
- 1.5 SP-Abgabesystem
- 1.6 Gelerntes anwenden



Tafelübungen und Besprechungen

- Üblicherweise abwechselnd Tafelübung und Besprechung
 - Ausnahme: Besprechungen zu den Aufgaben 1 und 2
- Nächste Übungswoche beginnt am Donnerstag

	Mon, 15.04	Tue, 16.04	Wed, 17.04	Thu, 18.04	Fri, 19.04	Sat, 20.04	Sun, 21.04	Mon, 22.04	Tue, 23.04	Wed, 24.04	Thu, 25.04	Fri, 26.04	Sat, 27.04
Vorlesung		V1		V2					V3		V4		
Übung								U1	U1	U1	U1	U1	
Mo								A1	A1	A1	A1	A1	
Di									A1	A1	A1	A1	
Mi										A1	A1	A1	
Do											A1	A1	
Fr												A1	
	Mon, 28.04	Tue, 29.04	Wed, 30.04	Thu, 01.05	Fri, 02.05	Sat, 03.05	Sun, 04.05	Mon, 06.05	Tue, 07.05	Wed, 08.05	Thu, 09.05	Fri, 10.05	Sat, 11.05
Vorlesung				V5									
Übung				U2	U2			U2	U2	U2			
Mo	A1							A2	A2	A2		A2	
Di	A1	A1							A2	A2		A2	
Mi	A1	A1		A1						A2		A2	
Do	A1	A1		A1	A1			A2	A2	A2		A2	
				A2	A2								
Fr	A1	A1		A1	A1			A1	A2	A2		A2	
					A2			A2					

Quelle: http://www4.cs.fau.de/Lehre/SS13/V_SP1/semesterplan.shtml



Praktischer Teil – Aufgaben

- Ausgabe neuer Aufgaben in den Tafelübungen
 - Aufgabenstellung meist recht knapp
 - Nicht alles bis in letzte Detail spezifiziert
 - Gegebene Spezifikationen sind dennoch zwingend einzuhalten
- Selbstständiges Bearbeiten der Aufgaben (vorzugsweise im CIP)
 - bei Problemen hilft z.B. ein Besuch in den Rechnerübungen
- Korrektur und Bewertung erfolgt durch den jeweiligen Tafelübungsleiter
 - korrigierte Ausdrucke werden in den Besprechungen ausgegeben
 - teilweise auch elektronisch zur Verfügung gestellt
 - eigenes Ergebnis nach Login im *WAFFEL* einsehbar
- Übungspunkte können das Klausurergebnis verbessern (Bonusnote)
 - Abschreibtests
 - Vorstellen der eigenen Lösungen



Praktischer Teil – Bearbeitung der Aufgaben

- einzeln oder in Zweier-Teams je nach Aufgabe
 - bei Teamarbeit müssen beide Partner in der **gleichen** Tafelübung sein
- Bearbeitungszeitraum ist angegeben in Werktagen (bei uns: Montag bis Freitag)
 - Bearbeitungszeitraum beinhaltet den Tag der Tafelübung
 - Feiertage (01.05., 09.05., 20.05, und 30.05.) und der Berg-Dienstag (21.05.) 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://fsi.cs.fau.de/forum/18>
 - inhaltliche Fragen zum Stoff oder den Aufgaben
 - allgemein alles, was auch für andere Teilnehmer interessant sein könnte
- Mailingliste: i4sp@cs.fau.de
 - geht an alle Übungsleiter
 - Angelegenheiten, die nur die eigene Person/Gruppe betreffen
- Rechnerübungen
 - Hilfe bei konkreten Problemen (z.B. Quellcode kompiliert nicht)
 - **kein** Händchenhalten während ihr die Tastatur bedient :)
 - angebotene Termine siehe Homepage
- der eigene Übungsleiter
 - Fragen zur Korrektur
 - fälschlicherweise positiver Abschreibetest



Agenda

- 1.1 Allgemeines
- 1.2 Organisatorisches
- 1.3 Linux-Kenntnisse
- 1.4 Versionsverwaltung mit SVN
- 1.5 SP-Abgabesystem
- 1.6 Gelerntes anwenden



Voraussetzungen

- UNIX-Grundkenntnisse werden vorausgesetzt
 - Übungsleiter sind in den Rechnerübungen bei Bedarf behilflich
- Zur Auffrischung: UNIX-Einführung der FSI
<http://fsi.cs.fau.de/vorkurs>

Linux-Install-Party der FSI

- am Montag, den 29.04.2013, um 16:00 Uhr
- weitere Informationen unter
http://fsi.cs.fau.de/dw/fsi/aktionen/linuxinstall_ss13



Dokumentation aus 1. Hand: Manual-Pages

- 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: `man -k stichwort`



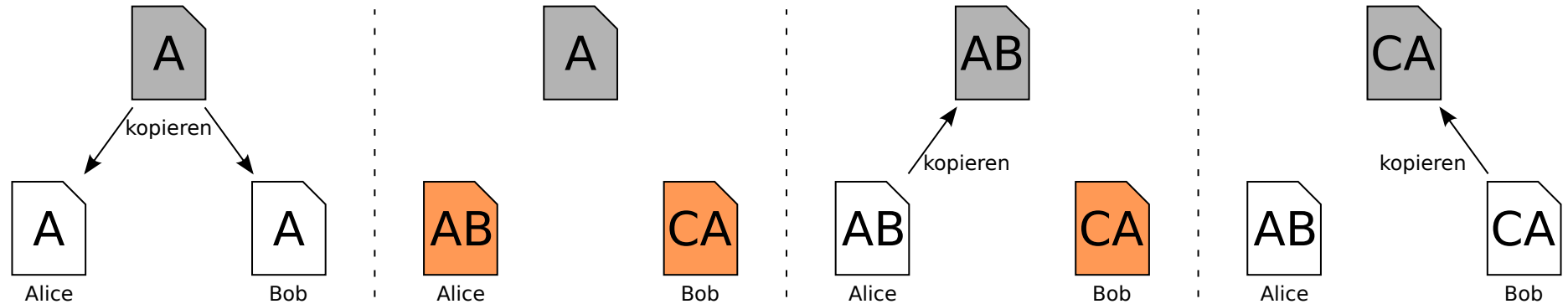
Agenda

- 1.1 Allgemeines
- 1.2 Organisatorisches
- 1.3 Linux-Kenntnisse
- 1.4 Versionsverwaltung mit SVN
- 1.5 SP-Abgabesystem
- 1.6 Gelerntes anwenden



Warum Versionsverwaltung?

- Gemeinsames Bearbeiten einer Datei kann zu Problemen führen:

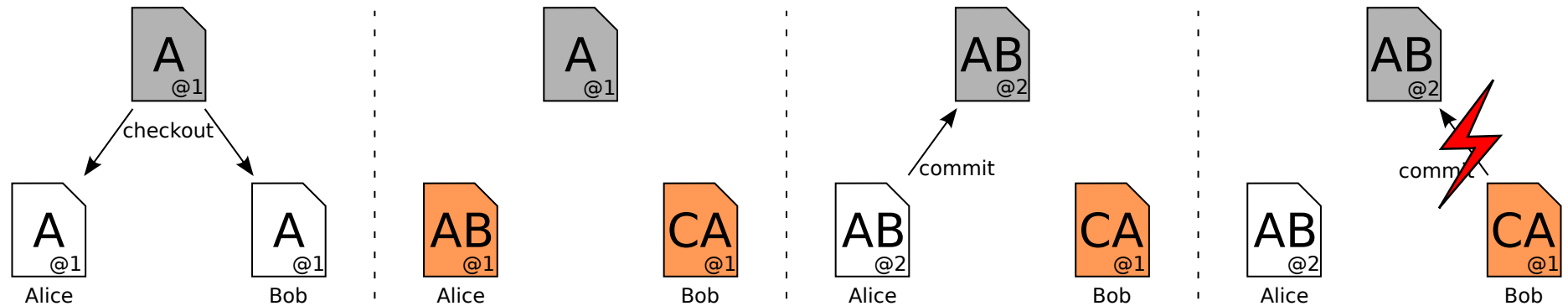


- Modifikationen werden nicht erkannt
- Änderungen von Alice gehen unbemerkt verloren

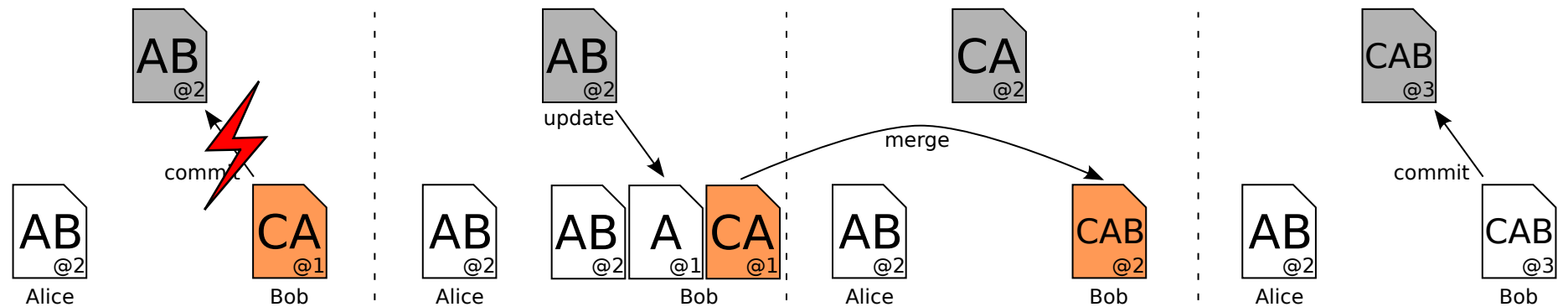


Warum Versionsverwaltung?

■ Versionsnummer zur Erkennung von Modifikationen



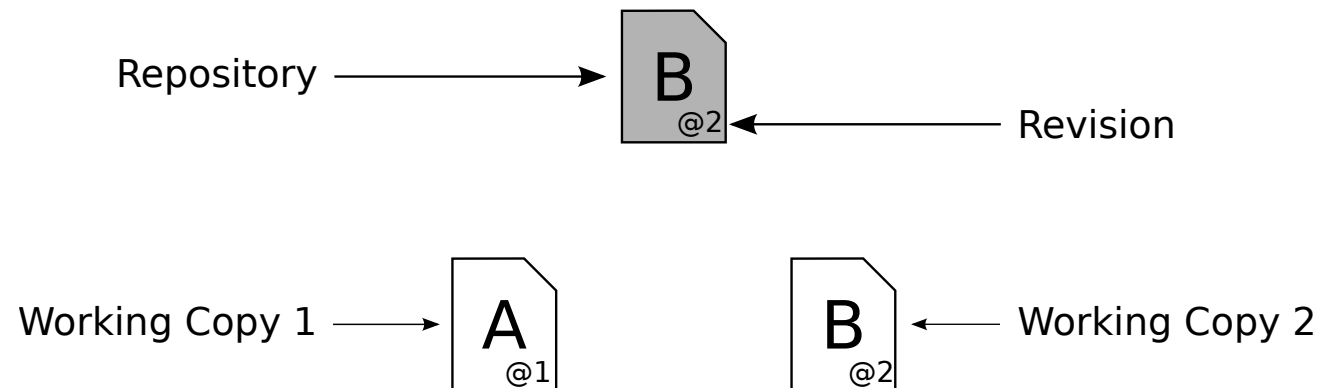
■ Entstandener Konflikt muss lokal gelöst werden



Das Versionsverwaltungssystem Subversion (SVN)

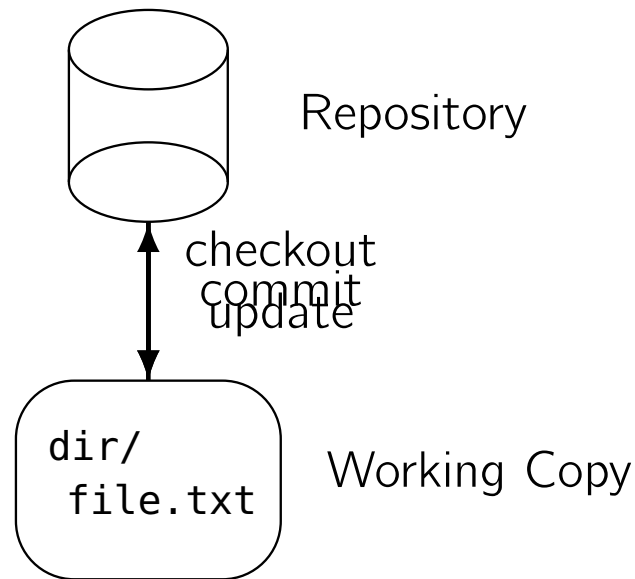
- SVN bietet Versionsverwaltung für Dateien und Verzeichnisse
- Speichert Zusatzinformationen zu jeder Änderung
 - Name des Ändernden
 - Zeitpunkt
 - Kommentar
- Ausführliche SVN-Dokumentation im Subversion-Buch
<http://svnbook.red-bean.com>
- Kommando `svn`
- Graphische Frontends
 - TortoiseSVN (Windows)
 - SCPlugin (Mac OS X)
- 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/daheim)





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



Exkurs: Kommentar bei commit

- Beim Aufruf von `svn commit` öffnet sich ein Editor zum Eingeben des commit-Kommentars

- Im CIP wird standardmäßig der Editor `joe` verwendet

- Zum Speichern und Verlassen `Strg-k x` drücken
- Hilfemenü öffnet sich mit `Strg-k h`

- Anderer Editor kann über die Umgebungsvariable `EDITOR` eingestellt werden

```
> 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`

```
> svn commit -m "Ich schreibe lieber gleich in die Befehlszeile  
und nicht in den Editor"
```



Basisoperationen 2

- **add:** Dateien unter Versionskontrolle stellen
 - Bei einer leeren Arbeitskopie müssen entsprechende Dateien oder Verzeichnisse erst eingefügt werden
- **del/remove/rm:** Dateien lokal löschen und nicht länger unter Versionskontrolle halten
- **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



Agenda

- 1.1 Allgemeines
- 1.2 Organisatorisches
- 1.3 Linux-Kenntnisse
- 1.4 Versionsverwaltung mit SVN
- 1.5 SP-Abgabesystem**
- 1.6 Gelerntes anwenden



- Für jeden Teilnehmer wird folgendes bereitgestellt:
 - ein Repository `https://www4.cs.fau.de/i4sp/ss13/sp1/<login>`
 - ein Projektverzeichnis `/proj/i4sp1/<login>` mit Arbeitskopie
- Die Erzeugung erfolgt in der Nacht nach der *WAFFEL*-Anmeldung

SVN-Passwort

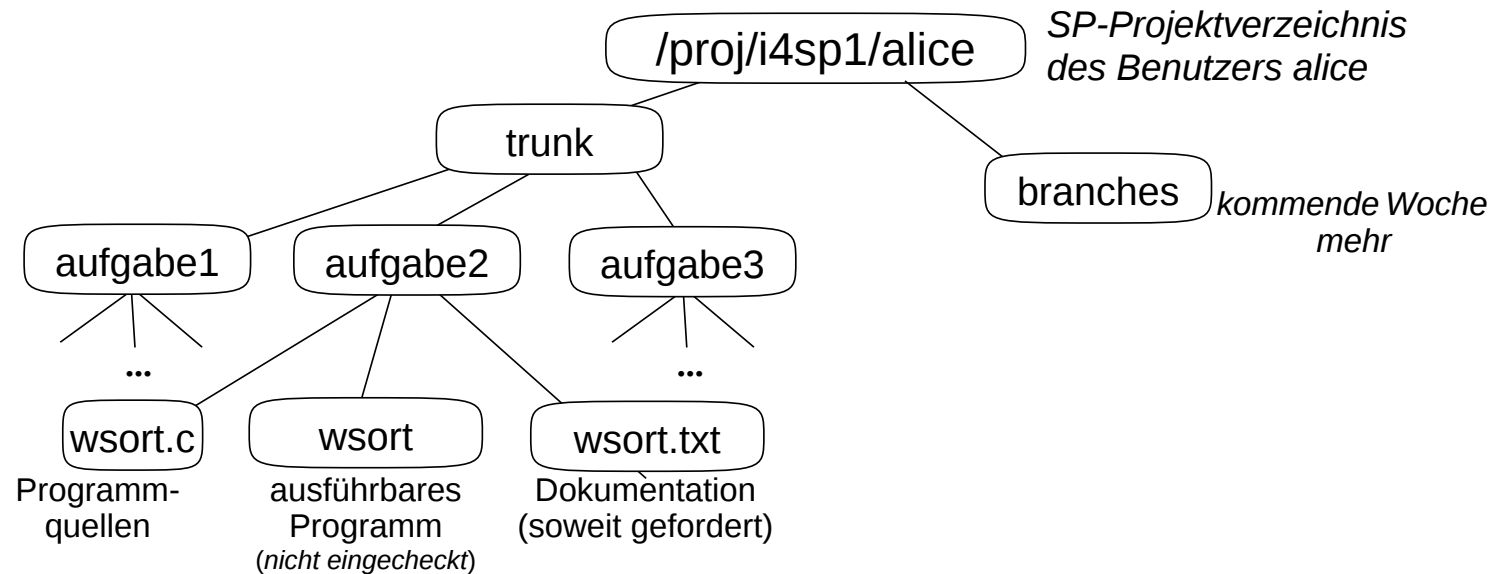
- Zum Zugriff auf das Repository muss ein Subversion-Passwort gesetzt werden

```
> /proj/i4sp1/bin/change-password
```

 - Das Passwort wird innerhalb der nächsten Stunde aktiv



Aufbau des SP-Repositories



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



Abgabe einer Aufgabe

- Zur Abgabe folgendes Skript aufrufen

```
> /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 29.04.2013  
um 17:30:00 Uhr
```

- Abgaben nach dem Abgabezeitpunkt sind möglich

- bei Vorliegen eines triftigen Grundes
- Wertung nur nach expliziter Rücksprache mit dem Übungsleiter
- ansonsten wird letzte rechtzeitige Abgabe gewertet



- Für einige Aufgaben stellen wir verschiedene Dateien zur Verfügung
 - Programmgerüste
 - Beispielergebnisse
 - 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

```
> /proj/i4sp1/bin/copy-public-files-for aufgabe1
```



Agenda

- 1.1 Allgemeines
- 1.2 Organisatorisches
- 1.3 Linux-Kenntnisse
- 1.4 Versionsverwaltung mit SVN
- 1.5 SP-Abgabesystem
- 1.6 Gelerntes anwenden



„Aufgabenstellung“

- Öffentliche Dateien für Aufgabe 1 ins Projektverzeichnis kopieren
- Vorgabe der Aufgabe 1 abgeben
 - Erforderliche Dateien: `lilo.c`

