# Übungen zu Systemprogrammierung 1 (SP1)

## Ü1 – Einführung

**Andreas Ziegler, Stefan Reif, Jürgen Kleinöder**

Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität
Erlangen-Nürnberg

SS 2016 – 18. bis 22. April 2016

http://www4.cs.fau.de/Lehre/SS16/V_SP1

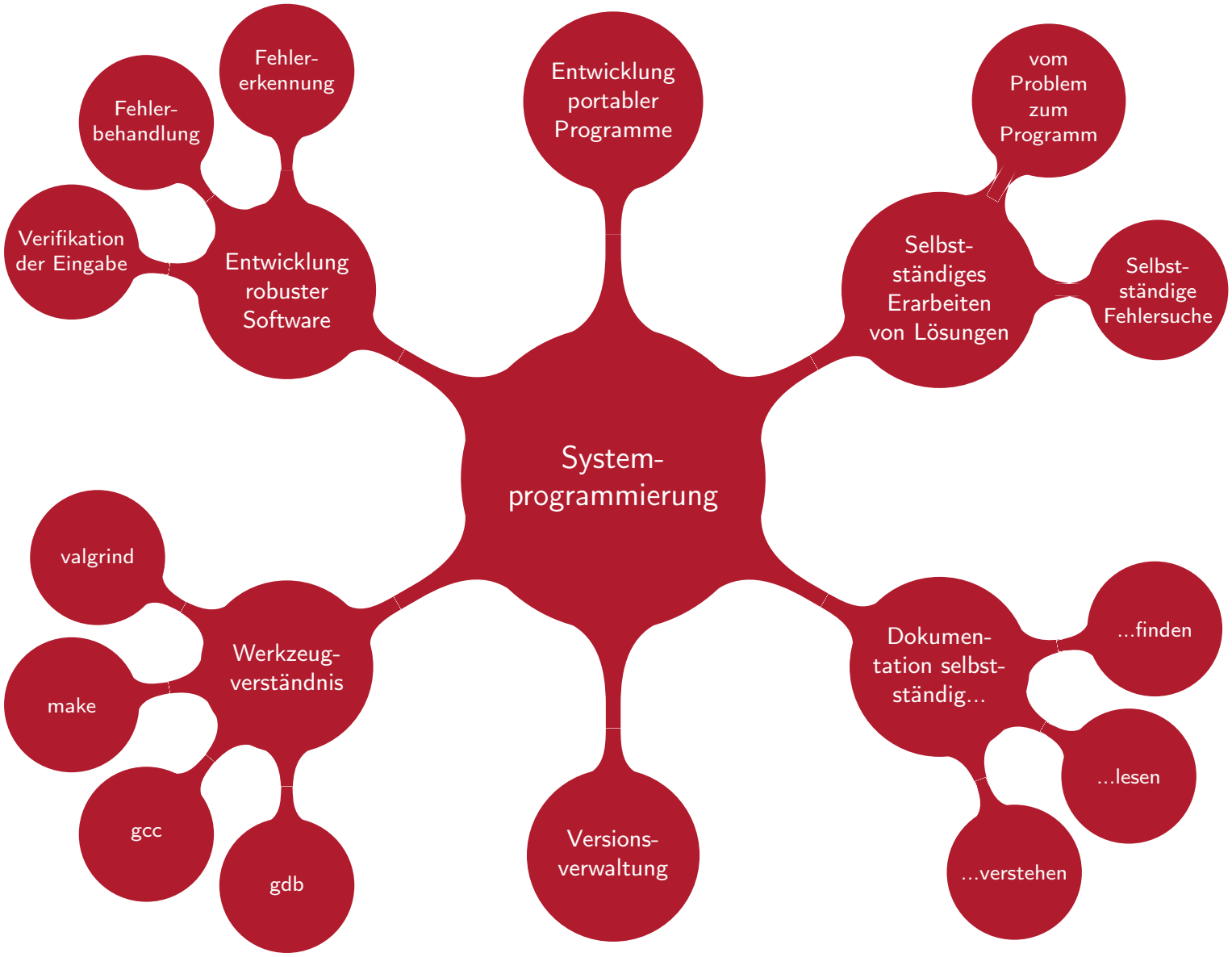


# Agenda





# Lernziele Systemprogrammierung



# Aufbau der SP-Übungen

## 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





# 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 (Notenbonus)
  - 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 (05.05., 16.05., 26.05.) und der "Berg-Dienstag" (17.05.) 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

# Kontakt zum *SP-Team*

- Forum: `http://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 Abschreibtest



# Agenda





# 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`



# 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`



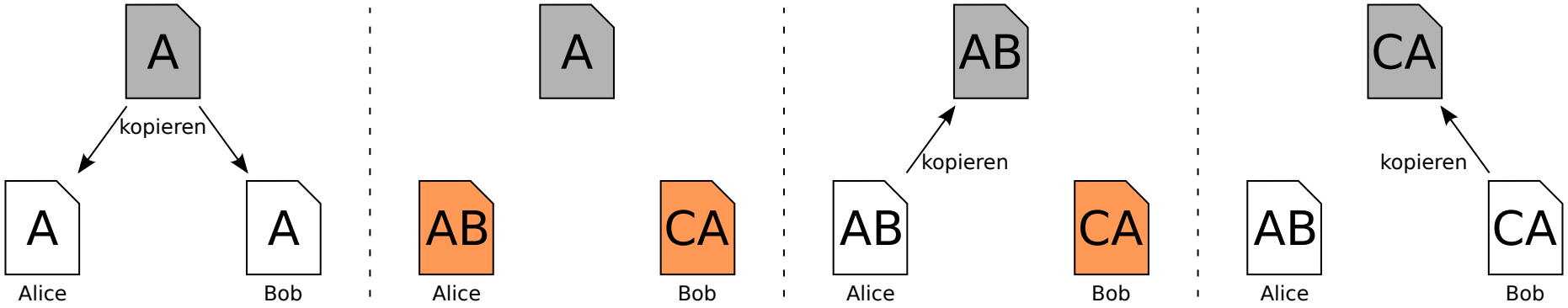
# Agenda





# Warum Versionsverwaltung?

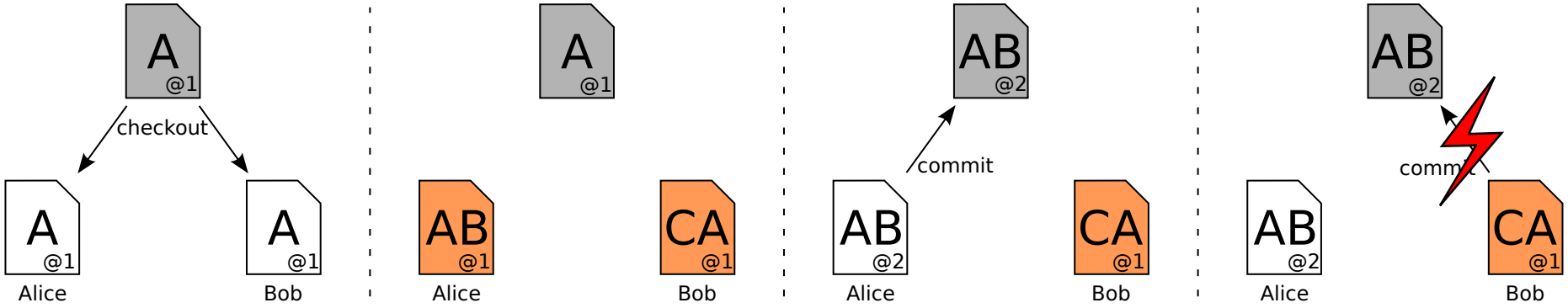
- Gemeinsames Bearbeiten einer Datei kann zu Problemen führen:



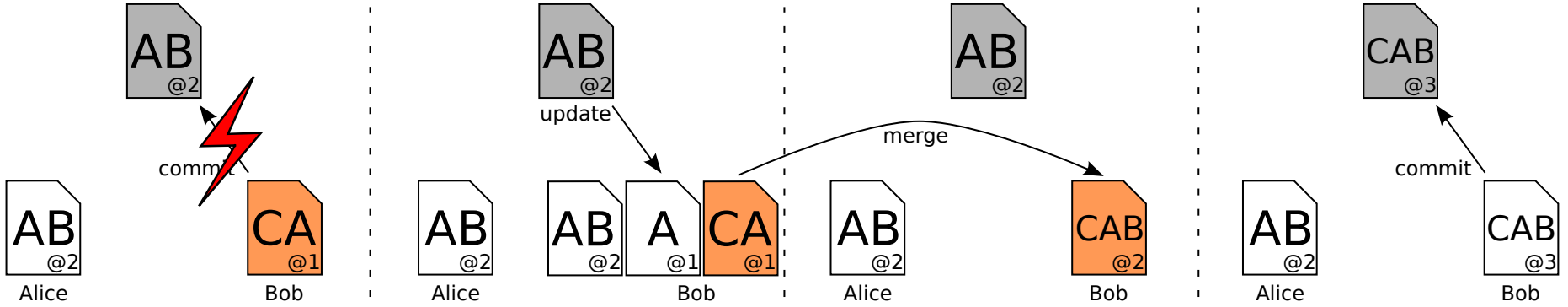

  - Modifikationen werden nicht erkannt
  - Änderungen von Alice gehen unbemerkt verloren

# Warum Versionsverwaltung?

- Versionsnummer zur Erkennung von Modifikationen




- Entstandener Konflikt muss lokal gelöst werden

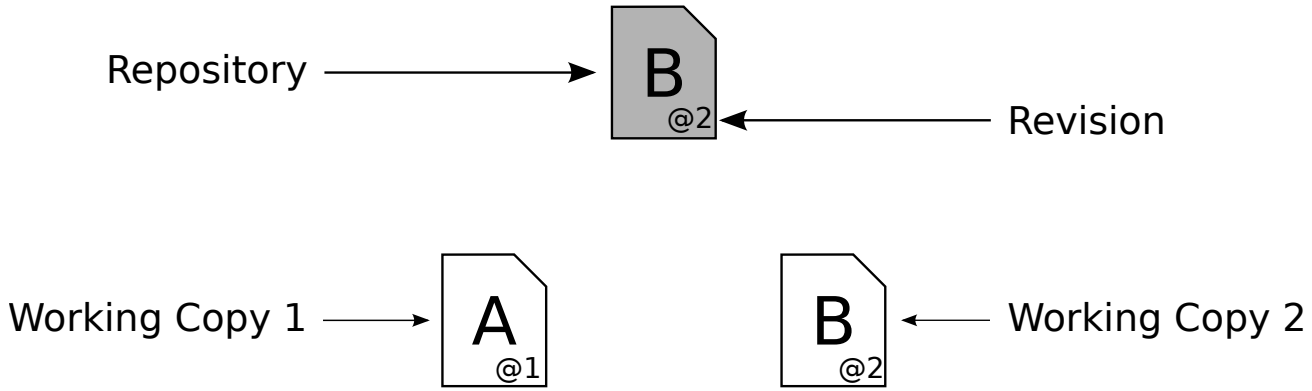


# Das Versionsverwaltungssystem Subversion (SVN)

- 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`
- Grafische Frontends
  - Tortoise SVN (Windows)
  - SCPlugin (Mac OS X)
- SP-Abgabesystem verwendet Subversion

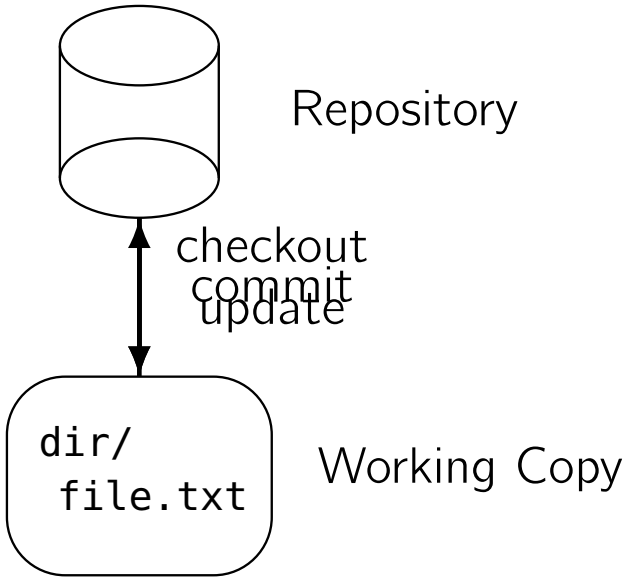


# Terminologie




- 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)

# Basisoperationen




- `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

# Exkurs: Kommentar bei commit

- 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

```
> 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

- `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



# Basisoperationen 3

- `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.
    [...]
```



# Agenda





# SP-Abgabesystem

- Für jeden Teilnehmer wird folgendes bereitgestellt:
  - ein Repository `https://www4.cs.fau.de/i4sp/ss16/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

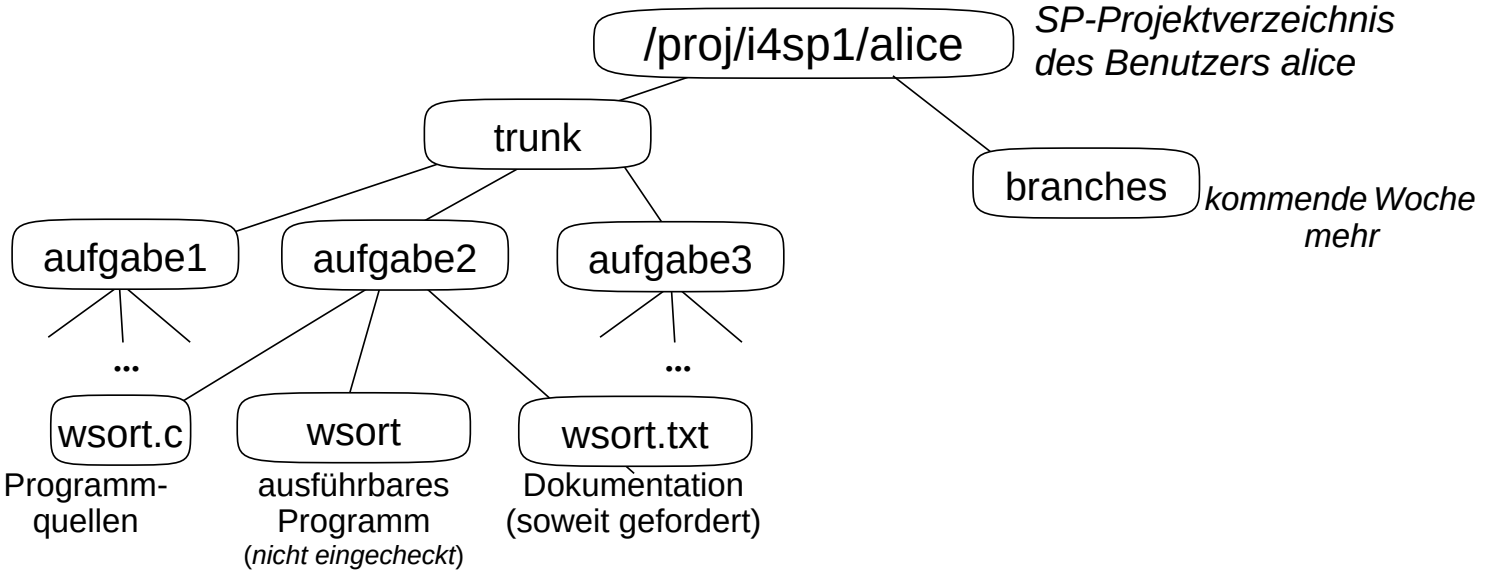
## SVN-Passwort

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

```
> /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 28.04.2016
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

# Bereitgestellte Dateien

- 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

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



# Agenda





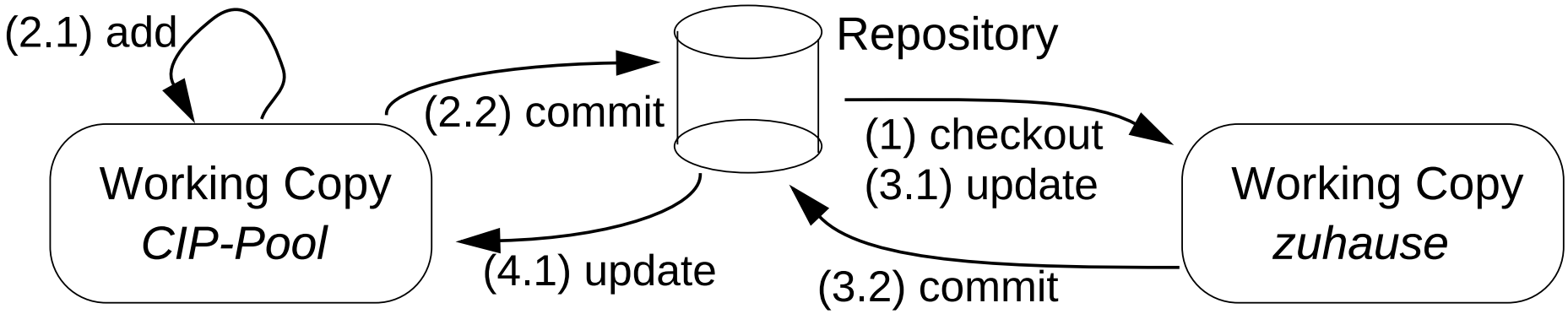
# Aktive Mitarbeit erforderlich!

## „Aufgabenstellung“

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



# Arbeiten zuhause




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)

# Beispiel-Workflow für Aufgabe 1

```
alice@faui06a[~] cd /proj/i4sp1/alice/trunk
alice@faui06a[trunk] mkdir aufgabe1
alice@faui06a[trunk] cd aufgabe1
alice@faui06a[aufgabe1] nano lilo.c
...
alice@faui06a[aufgabe1] cd ..
alice@faui06a[trunk] svn add aufgabe1
A aufgabe1
A aufgabe1/lilo.c
alice@faui06a[trunk] svn commit
...
Committed revision 2.
alice@faui06a[trunk] vim aufgabe1/lilo.c
...
alice@faui06a[trunk] svn commit -m 'Bugfix in printf'
...
Committed revision 3.
alice@faui06a[trunk] /proj/i4sp1/bin/submit aufgabe1
...
# Aufgabe 1 ist jetzt abgegeben
```