Übungen zu Systemprogrammierung 1 (SP1)

VL 2 – Speicherverwaltung

Jens Schedel, Christoph Erhardt, Jürgen Kleinöder

Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität
Erlangen-Nürnberg

WS 2012/13 – 29. Oktober bis 02. November 2012

http://www4.cs.fau.de/Lehre/WS12/V_SP1

Agenda

Versionierungsschema

- Subversion nummeriert fortlaufend ab Revision 0 (1,2,3...)
- spezielle Revisionsschlüsselwörter
  - `HEAD`: aktuelle Version des Repositieries (neueste Version)
  - `BASE`: Revision eines Eintrags (Datei oder Verzeichnis) der Arbeitskopie
  - `COMMITTED`: Letzte Änderungsrevision eines Eintrags – meist älter als `BASE`
  - `PREV`: `COMMITTED - 1`
- Revision zu einem bestimmten Zeitpunkt
  - `{“2012-10-09 08:07“}`

Basisoperationen II

- `diff`: Änderungen der Arbeitskopie anzeigen

```
> svn status
M hallo
> svn diff
Index: hallo
===================================================================
--- hallo (revision 23)
+++ hallo (working copy)
@@ -0,0 +1 @@
+welt
```

- `revert`: Änderungen an der Arbeitskopie zurücksetzen

```
> svn revert hallo
Reverted 'hallo'
> svn status
>
```


Basisoperationen II

- `list/ls`: Dateien/Verzeichnisse im Repository anzeigen

```
> svn ls
branches/
trunk/
```

- `log`: Historie anzeigen

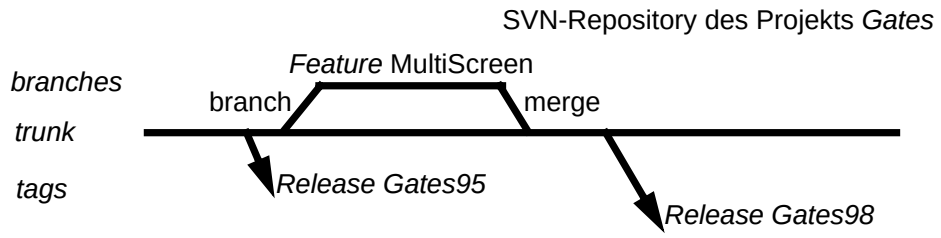
```
> svn log
------------------------------------------------------------------------
r1 | www-data | 2010-04-20 15:03:14 +0200 (Tue, 20 Apr 2010) | 1 line

init repository
------------------------------------------------------------------------
```

- `move/mv`: Datei umbenennen oder verschieben
- `copy/cp`: Datei/Teilbaum kopieren

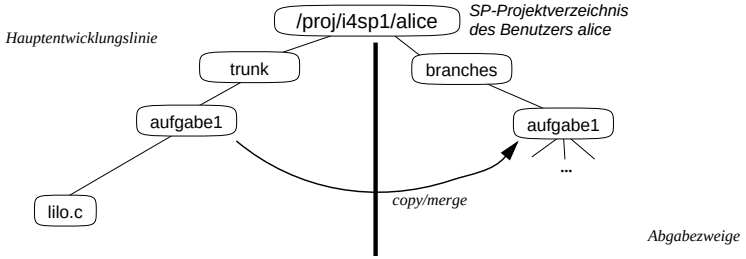
```
> svn cp aufgabe2 contest
> # aufgabe2 wurde in contest kopiert
```


Konventionelles Repository-Layout



- Unterteilung des Wurzelverzeichnisses
  - Hauptentwicklungslinie: *trunk*
  - Verzeichnis mit Entwicklungszweigen: *branches*
    - Größere Features können entkoppelt in einem eigenen Zweig (*branch*) entwickelt und nach Fertigstellung wieder in die Hauptlinie eingebracht (*merge*) werden
  - Eingefrorene Versionen: *tags*
    - Besondere Versionen können benamt (*getaggt*) werden (z.B. Release)

Funktionsweise der SP-Abgabe



- Erzeugung eines Abgabezweig für jede Aufgabe in *branches*
  - unterhalb von *branches* nichts von Hand editieren oder eingechecken
- Abgabe unter folgender URL einsehbar:

`https://www4.cs.fau.de/i4sp/ws12/sp1/<login>/branches/<aufgabe>/`

  - Dort werden die Dateien angezeigt, die zur Bewertung herangezogen werden

Agenda

Verkettete Liste

- Wann setzt man eine verkettete Liste ein?
- Anforderungsanalyse für verkettete Liste
  - Wieviele Listenelemente gibt es maximal?
  - Welche Lebensdauer muss ein Listenelement besitzen?
  - In welchem Kontext muss ein Listenelement sichtbar sein?
- Wir brauchen einen Mechanismus, mit dem Listenelemente
  - in a-priori nicht bekannter Anzahl
  - zur Laufzeit des Programmes erzeugt und zerstört werden können

Dynamische Speicherverwaltung

- Rückblick: Aufgabe 10.1 (Menschenkette) in AuD SS 2012

```
WaitingHuman somebody = new WaitingHuman("Mrs. Somebody");
WaitingHuman nobody = new WaitingHuman("Mr. Nobody");

somebody.add(nobody);
```

- In Java: Neues Listenelement wird mit Hilfe von `new` instanziiert
  - Reservieren eines Speicherbereiches für das Objekt
  - Initialisieren des Objektes durch Ausführen des Konstruktors
- In C: Anlegen eines Listenelementes mittels `malloc(3)`

```
struct listelement *newElement;
newElement = malloc( sizeof(struct listelement) );
if( newElement == NULL ) {
    // Fehlerbehandlung
}
```

  - Zurückgegebener Speicher ist undefinierten/zufälligen Wert
  - Inititialisierung muss per Hand erfolgen

Dynamische Speicherverwaltung

- Explizite Initialisierung mit definiertem Wert: `memset(3)`

```
memset(newElement, 0, sizeof(struct listelement));
```

- Mit 0 vorinitialisierter Speicher kann mit `calloc(3)` angefordert werden

```
struct listelement *newElement;
newElement = calloc( 1, sizeof(struct listelement) );
if ( newElement == NULL ) { /* Fehler */ }
```

- Im Gegensatz zu Java gibt es in C keinen Garbage-Collection-Mechanismus
  - Speicherbereich muss von Hand mittels `free(3)` freigegeben werden
  - Nur Speicher, der mit einer der alloc-Funktionen angefordert wurde, darf mit `free(3)` freigegeben werden!
  - Zugriff auf freigegebenen Speicherbereich ist undefiniert

Agenda

Compiler und Optionen

- Übersetzen einer Quelldatei mit `gcc`

```
> gcc -o test test.c
```

  - Zur Erinnerung: Starten der ausführbaren Datei `test` mit `./test`
- Verhalten des `gcc` kann durch Optionen beeinflusst werden
  - `-g`: Erzeugt Debug-Symbole in der ausführbaren Datei
  - `-c`: Übersetzt Quellcode in Maschinencode, erzeugt aber kein ausführbares Programm
  - `-m32`: Erzeugt ausführbare Dateien für 32-Bit Systeme
  - `-Wall`: aktiviert weitere Warnungen, die auf mögliche Programmierfehler hinweisen
  - `-Werror`: `gcc` behandelt Warnungen wie Fehler

Gängige Compiler-Warnungen

- *implicit declaration of function 'printf'*
  - bei Bibliotheksfunktionen fehlt entsprechendes `#include`
    - entsprechende Manual-Page gibt Auskunft über den Namen der nötigen Headerdateien

```
> man 3 printf
SYNOPSIS
        #include <stdio.h>

        int printf(const char *format, ...);
```

  - bei einer eigenen Funktionen fehlt die Forward-Deklaration
- *control reaches end of non-void function*
  - in der Funktion, die einen Wert zurückliefern soll, fehlt an einem Austrittspfad eine passende `return`-Anweisung

Agenda

Portable Programme

- Entwicklung portabler Programme durch Verwendung definierter Schnittstellen

ANSI-C99

- Normierung des Sprachumfangs der Programmiersprache C
- Standard-Bibliotheksfunktionen z.B. `printf`, `malloc`

Single UNIX Specification V3 (SUSv3)

- Standardisierung der Betriebssystemschnittstelle
- SUSv3 wird von verschiedenen Betriebssystemen implementiert
  - SUN Solaris, HP/UX, AIX
  - Linux
  - Mac OS X (Darwin)

Agenda

Anforderungen an abgegeben Lösungen

- C-Sprachumfang konform zu ANSI-C99
- Betriebssystemschnittstelle konform zu SUSv3
- **warnungs-** und **fehlerfrei** im CIP-Pool mit folgenden `gcc`-Optionen übersetzen

```
-m32 -std=c99 -pedantic -D_XOPEN_SOURCE=600 -Wall -Werror
```

  - `-std=c99 -pedantic` erlauben nur ANSI-C99-konformen C-Quellcode
  - `-D_XOPEN_SOURCE=600` erlaubt nur SUSv3-konforme Betriebssystemaufrufe
- einzelne Aufgaben können hiervon abweichen, dies wird in der Aufgabenstellung entsprechend vermerkt

Agenda

Einfach verkettete FIFO-Liste

- Zielsetzungen
  - Kennenlernen der Umgebung und Entwicklungswerkzeuge
  - Dynamische Speicherverwaltung und Umgang mit Zeigern
  - Verwendung des Abgabesystems
- Strukturdefinition

```
struct listelement {
    int value;
    struct listelement *next;
};
typedef struct listelement listelement; // optional
```


Schnittstelle

- Nur folgende Funktionen zu implementieren
  - `insertElement`: Fügt einen neuen, nicht-negativen Wert in die Liste ein, wenn dieser noch nicht vorhanden ist. Tritt ein Fehler auf, wird -1 zurückgegeben. Ansonsten wird der eingefügte Wert zurückgegeben.
  - `removeElement`: Entfernt den ältesten Wert in der Liste und gibt diesen zurück. Ist die Liste leer, wird -1 zurückgeliefert.
- Keine Listen-Funktionalität in der `main()`-Funktion
  - Allerdings: Erweitern der `main()` zum Testen erlaubt und **erwünscht**
- Sollte bei der Ausführung einer verwendeten Funktion (z.B. `malloc(3)`) ein Fehler auftreten, sind keine Fehlermeldungen auszugeben.

Agenda

Aktive Mitarbeit erforderlich!

„Aufgabenstellung“

- Programm schreiben, welches „Hallo Welt!“ ausgibt
- Sieb des Eratosthenes implementieren
  - Erstes Argument gibt an, bis zu welcher Zahl geprüft werden soll
- Programme übersetzen