



# Betriebssysteme Tafelübung 0. Erste Schritte

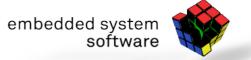
https://ess.cs.tu-dortmund.de/DE/Teaching/SS2020/BS/

#### **Horst Schirmeier**

horst.schirmeier@tu-dortmund.de https://ess.cs.tu-dortmund.de/~hsc



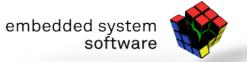




## Agenda

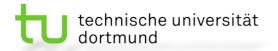
- Organisatorisches
- UNIX-Benutzerumgebung
  - Terminal, Shell
  - UNIX-Kommandos
  - GNU Compiler Collection (gcc)
- Grundlagen C-Programmierung
- Aufgabe 0: Erste Schritte in C

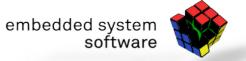




## Organisatorisches: Übungsaufgaben

- Theoriefragen und praktische Programmieraufgaben
  - Vorstellung der neuen Aufgaben in der Tafelübung
- Bearbeitung in Dreiergruppen (Ausnahme: Aufgabe A0)
  - Gruppenmitglieder sollten (aber müssen nicht) in derselben TÜ angemeldet sein
  - Hilfestellung: HelpDesk, dieses Jahr online
- Abgabe über ASSESS
- Abgabetermin abhängig von Woche der Übung:
  - eigener Übungstermin in gerader Kalenderwoche (18, 20, 22, ...):
     Donnerstag 08:00 in der Woche nach Ausgabe der Aufgabe
     Aufgabe A0: Donnerstag in KW 19, also 07.05.2020 08:00
  - eigener Übungstermin in ungerader Kalenderwoche (19, 21, 23, ...):
     Dienstag 08:00 nachdem das Folgeblatt erschienen ist Aufgabe A0: Dienstag in KW 20, also 12.05.2020 08:00





## Organisatorisches: Übungsaufgaben

- Aufgabenblätter auf der Veranstaltungswebseite https://ess.cs.tu-dortmund.de/DE/Teaching/SS2020/BS/
- Übungsvorbesprechungsfolien ebenfalls unter dieser URL
- Musterlösungen werden in der Übung präsentiert, Folien stehen bis nach den Klausuren auf der Vorlesungswebseite
- notwendig für Studienleistung: mindestens 50% der Punkte in beiden Aufgabenblöcken (je Blatt 10 Punkte + ggf. Sonderpunkte)
  - **1. Block:** A0, A1, A2  $\rightarrow$  Summe mindestens 15 Punkte
  - **2. Block:** A3, A4 → Summe mindestens 10 Punkte

Bestehen der Studienleistung ist Voraussetzung für die Klausurteilnahme!





## **UNIX-Benutzerumgebung**

- Umgebung, Terminal, Shell
- UNIX-Kommandos
- GNU Compiler Collection (gcc)





## Benutzerumgebung, Terminal

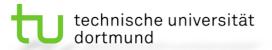
- Diese Punkte machen (u.a.) einen UNIX-Account aus:
  - Benutzername
  - Identifikation (User-ID und Group-IDs)
  - Home-Directory
  - Eingestellte Login-Shell
- Terminal
  - "Kommandozeile"
  - Früher: dedizierte Endgeräte zur Kommunikation mit Zentralrechner
  - Heute: Terminale mulation (z.B. xterm, Konsole, gnome-terminal)

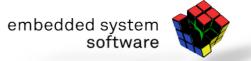




#### Terminal-Sonderzeichen

- einige Zeichen haben unter UNIX besondere Bedeutung
- Funktionen: u.a.
  - Korrektur von Tippfehlern
  - Steuerung der Bildschirm-Ausgaben
  - Einwirkung auf den Ablauf von Programmen
- Zuordnung Zeichen ⇔ Funktion leider nicht einheitlich
- Kann mit einem Kommando (stty(1)) verändert werden
- Übersicht:
- <Backspace>letztes Zeichen löschen
  - <Ctrl>-U alle Zeichen der Zeile löschen
  - <Ctrl>-C Interrupt Programm abbrechen
  - <Ctrl>-Z Stop Programm wird angehalten
  - <Ctrl>-D End Of File
  - <Ctrl>-S / <Ctrl>-Q Bildschirmausgabe anhalten/fortsetzen
  - Auf deutschen Tastaturen: <Strg> statt <Ctrl>





## **UNIX-Kommandointerpreter: Shell**

- Meist stehen verschiedene Shells zur Verfügung: sh, ksh, csh, tcsh, bash, zsh ...
- Auf GNU-Systemen gebräuchlich: bash
- Beim Öffnen eines Terminals startet die ausgewählte Login-Shell
- Wechsel der Login-Shell: chsh(1)





#### **Aufbau eines UNIX-Kommandos**

UNIX-Kommandos bestehen aus ...

- **1. Kommandoname** (der Name einer Datei, in der ein ausführbares Programm oder eine Kommandoprozedur für die Shell abgelegt ist)
  - nach dem Kommando wird automatisch in allen Verzeichnissen gesucht, die in der Umgebungs-Variable PATH gelistet sind
  - Die Pfade in PATH werden durch einen Doppelpunkt: getrennt
- daher kann man normalerweise "ls" schreiben statt "/bin/ls"

#### 2. einer Reihe von Optionen und Argumenten

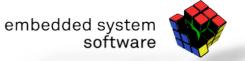
- Abtrennung Kommandoname/Optionen/Argumente durch Leerzeichen oder Tabulatoren
- Optionen sind meist einzelne Buchstaben mit einem vorangestellten "-" (Minuszeichen) (z.B. "ls -l")
- Argumente sind häufig Namen von Dateien, die von einem Kommando verarbeitet werden



#### **UNIX-Kommandos**

- man-Pages
- Dateisystem
- Benutzer
- Prozesse
- Diverse Werkzeuge
- Texteditoren





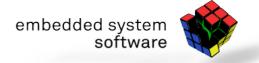
#### man-Pages

- aufgeteilt in verschiedene Sections (mehr infos: man man)
  - (1) Kommandos
  - (2) Systemaufrufe
  - (3) C-Bibliotheksfunktionen
  - (5) Dateiformate (spezielle Datenstrukturen etc.)
  - (7) Verschiedenes (z.B. IP, GPL, Zeichensätze, ...)
- man-Pages werden normalerweise mit der Section zitiert: printf(3)
  - sonst teilweise mehrdeutig (printf(1) vs. printf(3))!
- Aufruf unter Linux:

```
man [section] Begriff z.B. hsc@uran:~$ man 3 printf
```

- Suche nach Sections: man -f Begriff
- Suche nach Stichwort: man -k Stichwort





## **Dateisystem – Navigation**

**cd** Verzeichnis wechseln; im Terminal integriert

- . aktuelles Verzeichnis
- .. übergeordnetes Verzeichnis
- Verzeichnis, in dem man vor der letzten Navigation war

Ohne Argumente: navigiert cd in das Heimverzeichnis des Nutzers

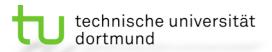
**ls** Verzeichnis auflisten; wichtige Optionen:

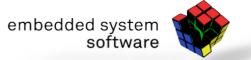
- -l langes Ausgabeformat
- -a listet auch mit "." beginnende Dateien und Verzeichnisse

pwd Aktuelles Verzeichnis ausgeben

Üblicherweise wird ~ zum Heimverzeichnis expandiert: cd ~/Downloads navigiert also z.B. zu /home/nutzer/Downloads

(cd ~nutzer wechselt in das Verzeichnis des Benutzers "nutzer")





#### **Dateisystem - Manipulation**

**chmod** Zugriffsrechte einer Datei ändern

cp Datei(en) kopieren

**mv** Datei(en) verlagern (oder umbenennen)

**In** Datei linken

(weiteren Verweis auf dieselbe Datei erzeugen)

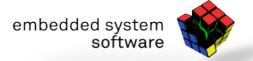
**In -s** Symbolischen Link erzeugen

rm Datei(en) löschen

mkdir Verzeichnis erzeugen

rmdir Verzeichnis löschen (muss leer sein!)





#### **Benutzer**

id, groups eigene Benutzer-ID, Gruppenzugehörigkeit

who am Rechner angemeldete Benutzer

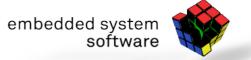
#### Zum Nachschlagen:

getuid(2) gibt die NutzerID zurück (C-Programmschnittstelle)

getgid(2) gibt die HauptgruppenID zurück (C-Programmschnittstelle)

... und weitere! Alle Funktionen der C-Standardbibliothek besitzen einen Handbucheintrag.





#### **Prozesse**

**ps** Prozessliste ausgeben

**-u x** Prozesse des Benutzers x

**-ef** alle Prozesse (-e), ausführliches Format (-f)

**top -o cpu** Prozessliste, sortiert nach aktueller Aktivität

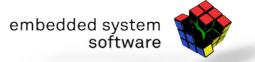
kill <pid>Prozess "abschießen" (Prozess kann aber dennoch

geordnet terminieren oder sogar ignorieren)

**kill -9** <*pid*> Prozess "gnadenlos abschießen" (Prozess kann nicht

mehr hinter sich aufräumen oder ignorieren)





## diverse Werkzeuge

cat Dateien hintereinander ausgeben

more, less Dateien bildschirmweise ausgeben

head, tail Anfang/Ende einer Datei ausgeben (10 Zeilen)

cal Kalender im Terminal anzeigen

wc Zeilen, Wörter und Zeichen zählen

grep, fgrep, egrep Nach bestimmten Mustern o. Wörtern suchen

find Dateibaum durchlaufen

sed Stream-Editor, z.B. zum Suchen/Ersetzen

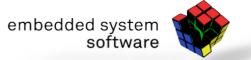
**tr** Zeichen aufeinander abbilden oder löschen

date Datum auf diverse Art und Weise ausgeben

**cut** Einzelne Felder aus Zeilen ausschneiden

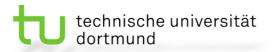
**sort** Sortieren





#### **Texteditoren**

- Geschmackssache aber einen solltet ihr beherrschen!
- Klassiker mit Nerdfaktor: vim/gvim, emacs, Xemacs
- Minimalisten: pico, nano
- Weitere mit X-Frontend: kate, gedit, geany, Eclipse, QTCreator, ...
- zum Programmieren nicht geeignet: Office-Programme (MS Word, LibreOffice Writer, ...)



#### **GNU Compiler Collection**

- Ursprünglich: GNU C Compiler
- Mittlerweile: Sammlung von verschiedenen Compilern (u.a. C, C++, Java, Objective-C, Fortran 95, ...)
- viele verschiedene Zielplattformen (x86, AMD64, ARM, ...)
- C-Compiler: gcc
- Compilieren und Linken eines C-Programms:
  - -Wall alle Warnungen ausgeben
  - -o <Ziel> Name für ausführbare Datei
- Weitere nützliche Parameter (siehe man-Page):
   -std=c11 -Werror -ansi -Wpedantic -D\_POSIX\_SOURCE
- Warnungen sind grundsätzlich ernstzunehmen und zu beseitigen, daher möglichst immer mit -Werror übersetzen.



## Übung macht den Meister!

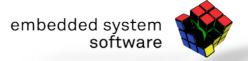
- Mit UNIX-Umgebung experimentieren
  - <del>in der Rechnerübung,</del>
  - in der Linux-VM von der BS-Website,
  - auf den Linux-Servern der IRB,
  - im Windows Subsystem for Linux (WSL) oder
  - in einer eigenen Linux-Installation
- Anleitung für die Entwicklungsumgebung auf der Veranstaltungswebseite



# Grundlagen C-Programmierung

• → Foliensatz C-Einführung (bis Folie 23)





#### Aufgabe 0: Erste Schritte in C

Programm:

```
/* test.c: Überprüft ob übergebene Zahl gerade ist */
#include <stdio.h>
int is_even(int x) {
   if (x % 2 == 0) return 1;
   else return 0;
}
int main(void) {
   printf("%d\n", is_evne(4));
   return 0;
}
```

Compilieren/Linken:

Da haben wir uns wohl vertippt ...



## Aufgabe 0: Erste Schritte in C

Programm:

```
/* test.c: Überprüft ob übergebene Zahl gerade ist */
#include <stdio.h>
int is_even(int x) {
   if (x % 2 == 0) return 1;
   else return 0;
}
int main(void) {
   printf("%d\n", is_even(4));
   return 0;
}
```

Compilieren/Linken:

```
$ gcc -Wall -o test test.c
$ ls
test test.c
```

Ausführen:

```
$ ./test
1
```