



Betriebssysteme

Tafelübung 0. Erste Schritte

<https://ess.cs.tu-dortmund.de/DE/Teaching/SS2017/BS/>

Olaf Spinczyk

olaf.spinczyk@tu-dortmund.de
<http://ess.cs.tu-dortmund.de/~os>



AG Eingebettete Systemsoftware
Informatik 12, TU Dortmund

Agenda

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



Organisatorisches: Übungsaufgaben

- Theoriefragen und praktische Programmieraufgaben
- Vorstellung der neuen Aufgaben in der Tafelübung
- Bearbeitung in Dreiergruppen
 - Gruppenmitglieder sollten in derselben Tü angemeldet sein
 - Hilfestellung:
 - betreute Rechnerübung!
 - INPUD-Forum (<http://inpud.cs.tu-dortmund.de/>)
- Abgabe abhängig von Woche der Übung über ASSESS:
 - Kalenderwochen 17,19,21,...: Donnerstag 10:00 bevor nächstes Blatt erscheint
 - Kalenderwochen 18,20,22,...: Dienstag 10:00 nachdem das nächste Blatt erschienen ist

3



Organisatorisches: Übungsaufgaben

- Aufgabenblätter auf der Veranstaltungswebsite
<https://ess.cs.tu-dortmund.de/DE/Teaching/SS2017/BS/>
- Übungsvorbesprechungsfolien ebenfalls unter dieser URL
- Musterlösungen werden NUR in der Übung präsentiert!
- notwendig für erfolgreiche Übungsteilnahme (Studienleistung):
mindestens 50% der Punkte in *jedem* Aufgabenblatt (**A0 inklusive!**)

Bestehen der Studienleistung ist Voraussetzung für die Klausurzulassung!

4



UNIX-Benutzerumgebung

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

5



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: Terminalemulation (z.B. xterm, Konsole, gnome-terminal)



6



Terminal-Sonderzeichen (1)

- 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

7



Terminal-Sonderzeichen (2)

- 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>

8



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)**

9



Umgebungsvariablen

- Setzen durch =
 - **BROWSER=firefox**
- Gilt nur für die aktuelle Terminal-Sitzung
- Sollten daher in Dateien definiert werden, die vom Interpreter beim Start ausgelesen werden!
- Beispieldateien: ~/.bashrc, ~/.profile, /etc/bash.bashrc
- Alternativ werden Variablen durch **export(1)** gesetzt und an aufgerufene Programme vererbt
 - **export BROWSER=firefox**
- Variablen werden durch ein \$ expandiert
- Beispiel: **echo \$BROWSER** gibt "firefox" aus
- **env** gibt alle gesetzten Umgebungsvariablen aus

10



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**“

11



Aufbau eines UNIX-Kommandos

UNIX-Kommandos bestehen aus ...

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

12



UNIX-Kommandos

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

13



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**

14



Dateisystem - Navigation

cd Verzeichnis wechseln; Terminalintrinsik

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

Ohne Argument navigiert cd in das Heimmverzeichnis des Nutzers

ls Verzeichnis auflisten; wichtige Optionen:

- l langes Ausgabeformat
- a auch mit . beginnende Dateien und Verzeichnisse werden gelistet

pwd Aktuelles Verzeichnis ausgeben

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

(**cd ~USERNAME** wechselt in das Verzeichnis des Benutzers "USERNAME")

15



Dateisystem – Manipulation

chmod Zugriffsrechte einer Datei ändern

cp Datei(en) kopieren

mv Datei(en) verlagern (oder umbenennen)

ln Datei linkern
(weiteren Verweis auf dieselbe Datei erzeugen)

ln -s symbolischen Link erzeugen

rm Datei(en) löschen

mkdir Verzeichnis erzeugen

rmdir Verzeichnis löschen (muss leer sein!)

16



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! Viele Funktionen der C-Standardbibliothek besitzen einen Handbucheintrag! (3)

17



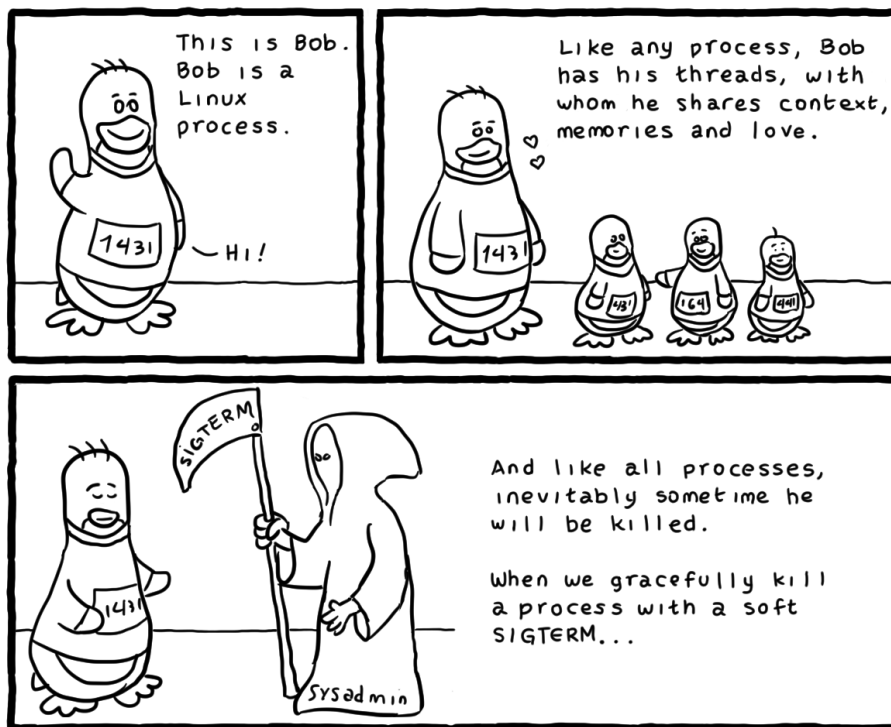
Prozesse

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)

18



SIGTERM vs. SIGKILL

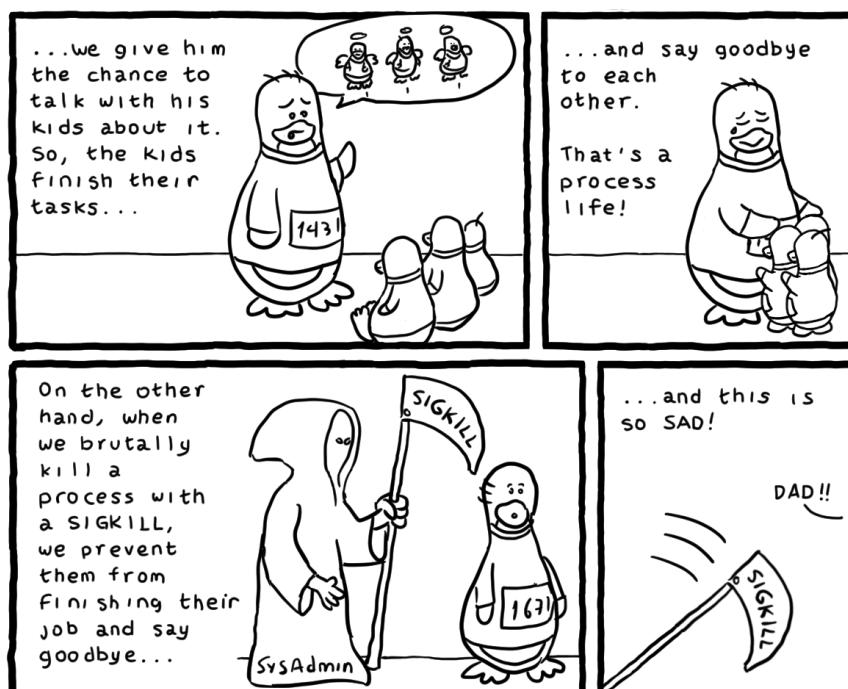


Quelle: Daniel Stori (<http://turnoff.us/geek/dont-sigkill/>) CC BY-NC-ND 4.0

19



SIGTERM vs. SIGKILL

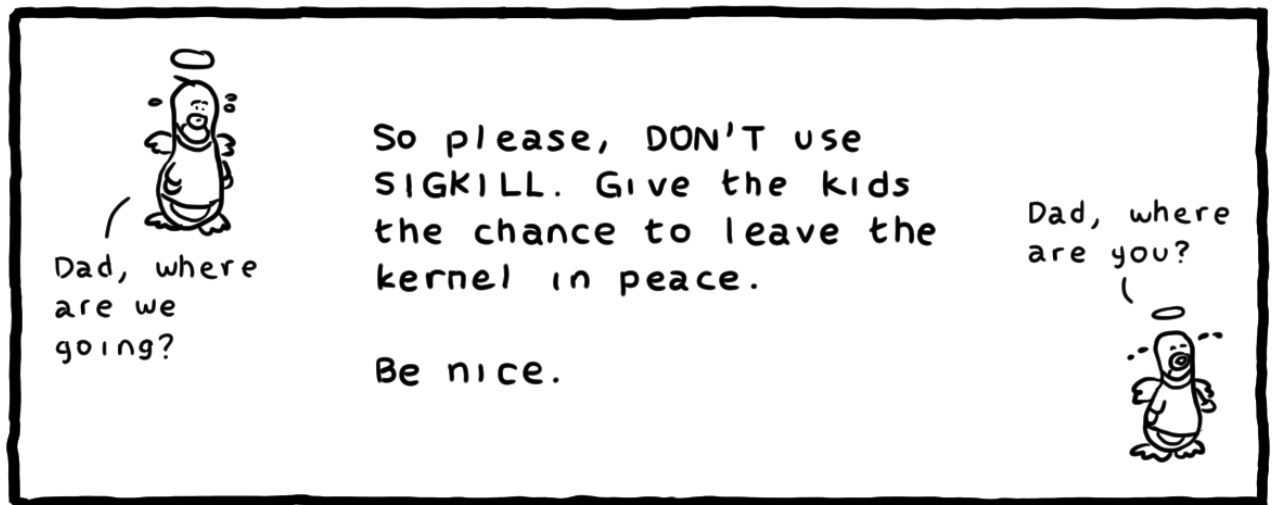


Quelle: Daniel Stori (<http://turnoff.us/geek/dont-sigkill/>) CC BY-NC-ND 4.0

20



SIGTERM vs. SIGKILL



Daniel Stori {turnoff.us}

Quelle: Daniel Stori (<http://turnoff.us/geek/dont-sigkill/>) CC BY-NC-ND 4.0

21



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 |

22



Texteditoren

- Geschmackssache – aber einen solltet ihr beherrschen!
- Klassiker mit Nerdfaktor: **vim**, **emacs**
(mit graphischem Frontend: **xemacs**, **gvim**)
- Minimalisten: **pico**, **nano**
- weitere mit X-Frontend: **kate**, **gedit**, **geany**, **Eclipse**, **QtCreator**, ...
- zum Programmieren nicht geeignet:
Office-Programme (**MS Word**, **LibreOffice Writer**, ...)

23



GNU Compiler Collection

- ursprünglich: GNU C Compiler
- mittlerweile: Sammlung von verschiedenen Compilern
(u.a. C, C++, Java, Objective-C, Fortran 95, ...)
- viele versch. Zielplattformen (x86, AMD64, ARM, IA-64 ...)
- 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):
-Werror, **-ansi**, **-Wpedantic**, **-D_POSIX_SOURCE**
- Warnungen sind grundsätzlich ernstzunehmen und zu beseitigen, daher möglichst immer mit **-Werror** übersetzen!

24



Übung macht den Meister!

- mit UNIX-Umgebung experimentieren
 - in der Rechnerübung,
 - in der Linux-VM von der BS-Website, oder
 - in einer eigenen Linux-Installation

25



Grundlagen C-Programmierung

- → Foliensatz C-Einführung (bis Folie 23)

26



Aufgabe 0: Erste Schritte in C

- Programm:

```
/* sum_n.c: Addiert alle Zahlen von 1 bis n */
#include <stdio.h>
int sum_n(int n) {
    int i = 1, res = 0;
    while (i <= n) res += i++;
    return res;
}
int main() {
    printf("%d\n", sum_(5));
    return 0;
}
```

- compilieren/linken:

```
hsc@uran:~/bs/a0$ gcc -Wall -o sum_n sum_n.c
sum_n.c: In function 'main':
sum_n.c:9: warning: implicit declaration of function 'sum_'
/tmp/ccKCnWMh.o: In function `main':
sum_n.c:(.text+0x3e): undefined reference to `sum_'
collect2: ld returned 1 exit status
hsc@uran:~/bs/a0$
```

- Da haben wir uns wohl vertippt ...

27



Aufgabe 0: Erste Schritte in C

- Programm:

```
/* sum_n.c: Addiert alle Zahlen von 1 bis n */
#include <stdio.h>
int sum_n(int n) {
    int i = 1, res = 0;
    while (i <= n) res += i++;
    return res;
}
int main() {
    printf("%d\n", sum_n(5));
    return 0;
}
```

- compilieren/linken:

```
hsc@uran:~/bs/a0$ gcc -Wall -o sum_n sum_n.c
hsc@uran:~/bs/a0$ ls
sum_n      sum_n.c
```

- ausführen:

```
hsc@uran:~/bs/a0$ ./sum_n
15
hsc@uran:~/bs/a0$
```

28



Speicheradressen von Variablen ausgeben

- Möglich durch den Adressoperator &
- printf() Formatzeichen ist %p

```
#include <stdio.h>
int main(void) {
    int x;
    printf("Die Variable x ist an Adresse %p.\n", (void *)&x);
    return 0;
}
```