

Übungen zu Systemnahe Programmierung in C

Abschnitt 10.1: Linux Einführung

29.06.2020

Tim Rheinfels
Benedict Herzog
Bernhard Heinloth

Lehrstuhl für Informatik 4
Friedrich-Alexander-Universität Erlangen-Nürnberg



Lehrstuhl für Verteilte Systeme
und Betriebssysteme



FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

- Als die Computer noch größer waren:



Seriell

Computer

Televideo 925 (Public Domain: Wtshymanski @Wikipedia)

- Als das Internet noch langsam war:

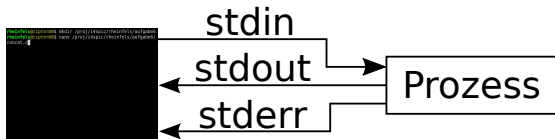


Netzwerk

Server

- Farben, Positionssprünge, etc. werden durch spezielle Zeichenfolgen ermöglicht

- Drei Standardkanäle für Ein- und Ausgaben



stdin Eingaben

stdout Ausgaben

stderr Fehlermeldungen

- Standardverhalten
 - Eingaben kommen von der Tastatur
 - Ausgaben & Fehlermeldungen erscheinen auf dem Bildschirm



- stdout in Datei schreiben

```
01 find . > ordner.txt
```

- stdout als stdin für anderer Programme

```
01 cat ordner.txt | grep tmp | wc -l
```

- Vorteil von stderr

⇒ Fehlermeldungen werden weiterhin am Terminal ausgegeben

- Übersicht

- > Standardausgabe stdout in Datei schreiben
- >> Standardausgabe stdout an exist. Datei anhängen
- 2> Fehlerausgabe stderr in Datei schreiben
- < Standardeingabe stdin aus Datei einlesen
- | Ausgabe eines Befehls als Eingabe für anderen Befehl



■ Wechseln in ein Verzeichnis mit `cd` (change directory)

```
01 cd /proj/i4spic/<login>/aufgabeX/ # absolut in Ordner
02 cd aufgabe5/                      # relativ zum aktuellen Ordner
03 cd ~                              # Benutzerverzeichnis (Home)
04 cd ..                             # übergeordnete Verzeichnis
```

■ Verzeichnisinhalt auflisten mit `ls` (list directory)

```
01 ls                                # zeige Dateien im akt. Ordner
02 ls -A                             # zeige auch versteckte Dateien
03 ls -lh                             # zeige mehr Metadaten
```



■ Datei oder Ordner kopieren mit cp (copy)

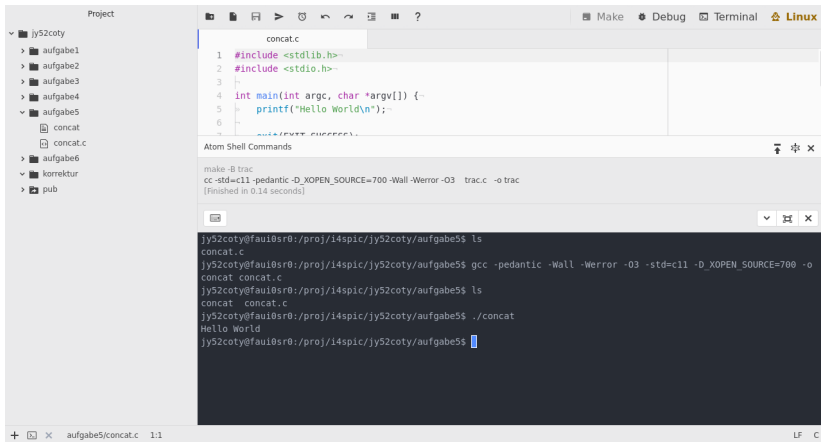
```
01 # Kopiere Datei ampel.c aus dem Home in Projektverzeichnis
02 cp ~/ampel.c /proj/i4spic/xy42abcd/aufgabe5/ampel.c
03
04 # Kopiere Ordner aufgabe5/ aus dem Home in Projektverzeichnis
05 cp -r ~/aufgabe5/ /proj/i4spic/xy42abcd/
```

■ Datei oder Ordner unwiederbringlich löschen mit rm (remove)

```
01 # Lösche Datei test1.c im aktuellen Ordner
02 rm test1.c
03
04 # Lösche Unterordner aufgabe1/ mit allen Dateien
05 rm -r aufgabe1
```



- Per Signal: CTRL-C (kann von Programmen ignoriert werden)
- Von einer anderen Konsole aus: `killall concat` beendet alle Programme mit dem Namen "concat"
- Von der selben Konsole aus:
 - CTRL-Z hält den aktuell laufenden Prozess an
 - `killall concat` beendet alle Programme namens concat
 - ⇒ Programme anderer Benutzer dürfen nicht beendet werden
 - `fg` setzt den angehaltenen Prozess fort
- Wenn nichts mehr hilft: `killall -9 concat`



The screenshot displays the SPiC IDE interface. On the left, a 'Project' sidebar shows a tree view with folders 'aufgabe1' through 'aufgabe6' and a file 'pub'. The main editor area shows a C file named 'concat.c' with the following code:

```
1 #include <stdlib.h>
2 #include <stdio.h>
3
4 int main(int argc, char *argv[]) {
5     printf("Hello World\n");
6 }
7
8 void *EXIT_SUCCESS;
```

Below the editor, the 'Atom Shell Commands' panel shows the command: `make -B trac cc -std=c11 -pedantic -D_XOPEN_SOURCE=700 -Wall -Werror -O3 -o concat.c -o trac` [Finished in 0.14 seconds]. At the bottom, a terminal window shows the execution of the program:

```
jy52coty@fau10sr0:/proj/i4spic/jy52coty/aufgabe5$ ls
concat.c
jy52coty@fau10sr0:/proj/i4spic/jy52coty/aufgabe5$ gcc -pedantic -Wall -Werror -O3 -std=c11 -D_XOPEN_SOURCE=700 -o
concat concat.c
jy52coty@fau10sr0:/proj/i4spic/jy52coty/aufgabe5$ ls
concat concat.c
jy52coty@fau10sr0:/proj/i4spic/jy52coty/aufgabe5$ ./concat
Hello World
jy52coty@fau10sr0:/proj/i4spic/jy52coty/aufgabe5$
```

- **Terminal:** öffnet ein Terminal und startet eine Shell
 - effiziente Interaktion mit dem System
 - optional Vollbild
- **Debug:** startet den Debugmodus
- **Make:** siehe nächste Woche



■ Programm mit dem GCC übersetzen

```
01 gcc -pedantic -Wall -Werror -O3 -std=c11 -D_XOPEN_SOURCE=700 -o  
    → concat concat.c
```

gcc ruft den Compiler auf (GNU Compiler Collection)

-pedantic aktiviert Warnungen (Abweichungen vom C-Standards)

-Wall aktiviert Warnungen (typische Fehler, z.B.: `if(x = 7)`)

-Werror wandelt Warnungen in Fehler um

-O3 aktiviert Optimierungen (Level 3)

-std=c11 setzt verwendeten Standard auf C11

-D_XOPEN_SOURCE=700

fügt POSIX Erweiterungen hinzu

-o concat legt Namen der Ausgabedatei fest (Standard: `a.out`)

concat.c ... zu kompilierende Datei(en)

■ Ausführen des Programms mit `./concat`

■ Abgaben werden von uns mit diesen Optionen getestet



- Programm mit dem GCC übersetzen
(inklusive Debugsymbole und ohne Optimierungen)

```
01 gcc -pedantic -Wall -Werror -O0 -std=c11 -D_XOPEN_SOURCE=700 -g -  
    → o concat concat.c
```

-O0 verhindert Optimierungen des Programms

-g belässt Debugsymbole in der ausführbaren Datei

⇒ ermöglicht dem Debugger Verweise zur Quelldatei herzustellen

- Hinweis: Pfeiltaste ↑ iteriert durch frühere Befehle

⇒ GCC Aufruf nur einmal tippen

- Informationen über:
 - Speicherlecks (malloc()/free())
 - Zugriffe auf nicht gültigen Speicher
- Ideal zum Lokalisieren von Segmentation Faults (SIGSEGV)
- Aufrufe:
 - `valgrind ./concat`
 - `valgrind --leak-check=full --show-reachable=yes`
 ↪ `--track-origins=yes ./concat`
- Die Ausgabe ist deutlich hilfreicher, wenn das analysierte Binary mit Debugsymbolen gebaut wird



- Das Linux-Hilfesystem
- aufgeteilt nach verschiedenen Sections
 - 1 Kommandos
 - 2 Systemaufrufe
 - 3 Bibliotheksfunktionen
 - 5 Dateiformate (spezielle Datenstrukturen, etc.)
 - 7 verschiedenes (z.B. Terminaltreiber, IP, ...)
- man-Pages werden normalerweise mit der Section zitiert:
`printf(3)`

```
01 # man [section] Begriff
02 man 3 printf
```

- Suche nach Sections: `man -f Begriff`
- Suche von man-Pages zu einem Stichwort: `man -k Stichwort`



- Abgespeckte (hübschere) Version der Manpages
- Bieten nur eine Übersicht, keine vollständige Spezifikation
- Aus der SPiC-IDE abrufbar (Hilfe-Button wenn im Linux-Modus)
- Auf der Webseite zu finden

https://www4.cs.fau.de/Lehre/SS20/V_SPIC/Linux/libc-api/

- Unsere Übersicht ersetzen die Manpages nicht
- In der Klausur: ausgedruckte Manpages!