

Vertiefung Systemtechnik Eingebettete Systeme

Prof. Dr. rer. nat. Peter Gerwinski

19. Oktober 2017

Vertiefung Systemtechnik – Eingebettete Systeme

<https://gitlab.cvh-server.de/pgerwinski/es.git>

1 Einführung

- 1.1 Was sind eingebettete Systeme?
- 1.2 Vertiefung Systemtechnik
- 1.3 In dieser Lehrveranstaltung

2 Einführung in Unix

- 2.1 Grundkonzepte
- 2.2 Die Kommandozeile: Grundlagen
- 2.3 Dateisysteme
- 2.4 Ein- und Ausgabeströme
- 2.5 Pipes
- 2.6 Verzweigungen und Schleifen

3 TCP/IP in der Praxis

...



Änderungen
vorbehalten

Was sind eingebettete Systeme?

Der Ausdruck **eingebettetes System** (...) bezeichnet einen elektronischen Rechner oder auch Computer, der in einen technischen Kontext eingebunden (eingebettet) ist.

https://de.wikipedia.org/wiki/Eingebettetes_System

- keine Aussage über die Größe
- in der Praxis: so klein wie möglich, z. B.:
 - mehrere Rechnerschränke
 - Industrie-PC
 - Einplatinencomputer
 - Mikro-Controller
- Bedienung über spezielle Ein-/Ausgabegeräte (z. B. Flugsimulator, Telefon)
- Wartung über speziellen Administrator-Zugang
 - Bus-Schnittstelle (RS-232, CAN-BUS)
 - Netzwerk (TCP/IP, Ethernet oder WLAN)
- Programmierung oft außerhalb des Systems (Cross-Entwicklungswerkzeuge)

... für Fortgeschrittene
↓

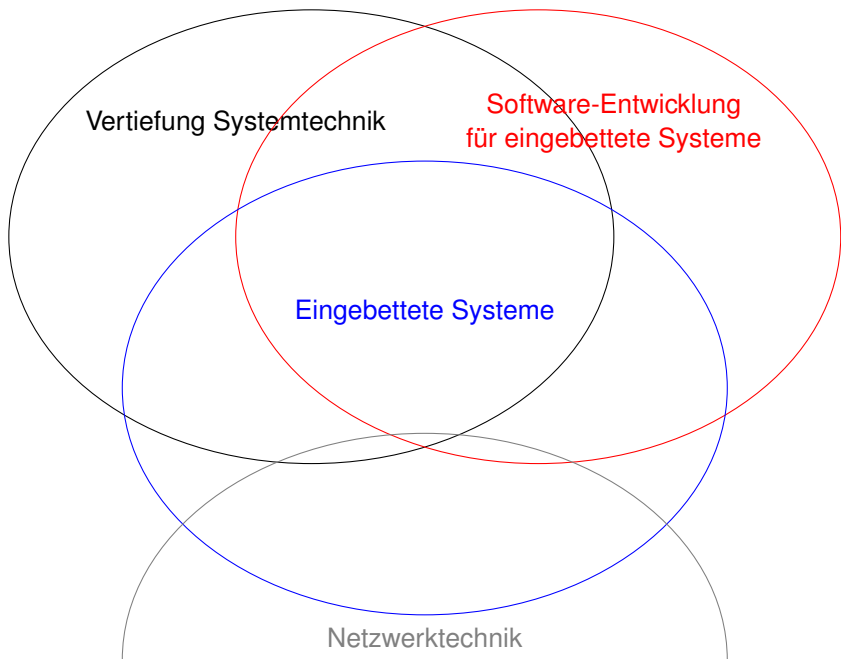
Vertiefung Systemtechnik

↑

Echtzeitsysteme

↗
Thema dieser
Lehrveranstaltung:
Echtzeit

↖
Geräte aller Art



In dieser Lehrveranstaltung

- Einführung in Unix
- TCP/IP und Bus-Systeme in der Praxis
- Web-Interfaces
- C-Programmierung für Fortgeschrittene
- Theorie und Praxis der Echtzeitprogrammierung
- Prüfungsleistung: Projektaufgabe
Eingebettetes System bzw. Echtzeit-System eigener Wahl
zum Laufen bringen

→ **Projektaufgabe überlegen**

Weitere Ideen:

- Exkursion: TFC – Simulatoren und Technik GmbH,
Velbert-Nierenhof
- Einführung in die GUI-Programmierung



Änderungen
vorbehalten

2 Einführung in Unix

2.1 Grundkonzepte

- 1965** Vorgänger: Multics (Multiplexed Information and Computing Service)
„überladen“
- 1970** Unix: Einfachheit als Grundkonzept
- 1972** Umstellung auf neu entwickelte Programmiersprache C
- 1975** AT&T: Unix inkl. Quelltext für Universitäten
- 1977** Berkeley Software Distribution (BSD)
- 1983** GNU-Projekt
- 1987** Minix
- 1991** Linux
- 1993** FreeBSD, NetBSD
- 1994** OpenBSD
- 2000** Darwin (Mac OS X, BSD-basiert)
- 2008** Android (Linux-basiert)

2 Einführung in Unix

2.1 Grundkonzepte

Unix und C: Einfachheit als Grundkonzept

- Vermeiden von Ausnahmen
- Baukastensystem

C: Hauptprogramm
= „normale“ Funktion

```
int main (int argc, char **argv)
{
    printf ("Hello, _world!\n");
    return 0;
}
```

Unix: übergeordnetes Verzeichnis = „normales“ Verzeichnis

```
cassini/home/peter/foo> ls -la
insgesamt 24
drwxr-xr-x  2 peter peter  4096 Okt  6 13:30 .
drwxr-xr-x 172 peter peter 20480 Okt  6 13:30 ..
cassini/home/peter/foo> cd ..
cassini/home/peter>
```


2 Einführung in Unix

2.1 Grundkonzepte

Unix und C: Einfachheit als Grundkonzept

- Vermeiden von Ausnahmen
- Baukastensystem

C: Bibliotheken

- Grundfunktionen: `libc`
- 3d-Grafik – Kernfunktion: `libGL`
- 3d-Grafik – Utilities: `libGLU`
- 3d-Grafik – Utility Toolkit: `libglut`
- ...

Unix: Programme arbeiten zusammen

```
cassini/home/peter/bo> find . -name "*klausur*.tex" \  
| grep -v "fig" | less  
./2013ss/net/script/slides/net-probeklausur-20120712.tex  
./2013ss/net/20130924.0/net-klausur-20130924.tex  
./2011ws/rarch/20120322.0/rarch-klausur-20120322.tex  
...  
./2012ss/hs/20130318.0/hs-klausur-20130318.tex  
./2012ss/hs/20120715.0/hs-probeklausur-20120715.tex  
./2012ss/hs/20120720.0/hs-klausur-20120720.tex
```

2.2 Die Kommandozeile: Grundlagen

- Programm aufrufen: Namen eingeben, z. B.: `ls`
- Optionen: `ls -l`
- Lange Optionen (GNU-Konvention): `ls --help`
- Text schreiben: `echo "Hello, world!"`
- (String-)Variable setzen: `FOO=bar`
- Variable abrufen: `echo $FOO`

```
cassini/home/peter/bo> FOO=ls
cassini/home/peter/bo> echo $FOO
ls
cassini/home/peter/bo> $FOO
2011ws  2012ws  2013ws  doc          misc  projekte
2012ss  2013ss  briefe  material    orga
cassini/home/peter/bo>
```

2.2 Die Kommandozeile: Grundlagen

- Befehl zurückholen: Pfeiltasten \uparrow , \downarrow
- Befehl bearbeiten: Pfeiltasten \leftarrow , \rightarrow usw.
- Befehl vervollständigen: TAB
- Befehl rückwärts suchen: Ctrl+R
- Bildschirm löschen: Ctrl+L
- Befehl abbrechen: Ctrl+C

- Hilfe-Option: `ls --help`
- Unix-Handbuch – *manual*: `man ls`
(Beenden mit `q`)

2.2 Die Kommandozeile: Grundlagen

- Verzeichnisse für Programme: `echo $PATH`
- Programm in explizitem Verzeichnis aufrufen: `/bin/ls -l`
- Programm im aktuellen Verzeichnis aufrufen: `./hello`

MS-DOS: Ausführbare Programme werden gefunden,
wenn sie im `PATH` stehen
oder sich im aktuellen Verzeichnis befinden.

Unix: Ausführbare Programme werden gefunden,
wenn sie im `PATH` stehen.

—> Vermeiden von Ausnahmen

Das aktuelle Verzeichnis (`.`) *kann* im `PATH` stehen,
muß dies aber nicht –
und sollte es aus Sicherheitsgründen auch nicht.

2.3 Dateisysteme

- Dateien listen: `ls`
langes Listenformat: `ls -l`
rückwärts nach Zeit sortiert: `ls -lrt`
- Datei ausgeben: `cat hello.c`
- Datei anzeigen: `less hello.c`

2.3 Dateisysteme

- Arbeitsverzeichnis anzeigen: `pwd`
- Arbeitsverzeichnis wechseln: `cd script`
(*kein* Programm, sondern Shell-Befehl)
- übergeordnetes Verzeichnis: `cd ..`
- eigenes *Home*-Verzeichnis: `cd`
- Wurzelverzeichnis: `cd /`
- wieder zurück: `cd -`

```
cassini/home/peter/bo/2013ss/net/script> cd /usr/bin
cassini/usr/bin> cd ../lib
cassini/usr/lib> cd
cassini/home/peter>
```

2.3 Dateisysteme

- Dateien kopieren (*copy*): `cp`
- Dateien verschieben/umbenennen (*move*): `mv`
- Dateien löschen (*remove*): `rm`

```
cassini/home/peter> cp -p foo/test.txt
cp: missing destination file operand after `foo/test.txt'
Try `cp --help' for more information.
cassini/home/peter> cp -p foo/test.txt .
cassini/home/peter> mv test.txt bla.txt
cassini/home/peter> cat bla.txt
Dies ist ein Test.
cassini/home/peter> rm bla.txt
cassini/home/peter>
```

Aktuelles Verzeichnis: `.`

2.3 Dateisysteme

- `grep`: Dateien durchsuchen

```
cassini/.../ainf/20131031.0> grep printf *.c  
philosophy.c:  printf ("The answer is %d.\n", answer ());
```


2.3 Dateisysteme

- Datenträger in Verzeichnis *einhängen*: `mount`

```
cassini/home/peter> ls /media/sd-card/  
cassini/home/peter> mount /media/sd-card  
cassini/home/peter> ls /media/sd-card/  
DCIM  NIKON001.DSC  
cassini/home/peter> umount /media/sd-card  
cassini/home/peter> ls /media/sd-card/  
cassini/home/peter>
```

2.3 Dateisysteme

- *Zugriffsrechte*

```
phoenix/home/peter/bo/2013ws/ainf/20131031.0> ls -l
...
-rw-r--r-- 1 peter peter    1539 Nov 29  2012 orbit-x1.c
```

2.3 Dateisysteme

- *Zugriffsrechte*

```
phoenix/home/peter/bo/2013ws/ainf/20131031.0> ls -l
...
-rw-r--r-- 1 peter peter    1539 Nov 29  2012 orbit-x1.c
```




Benutzer (u – *user*) darf lesen und schreiben

2.3 Dateisysteme

- *Zugriffsrechte*

```
phoenix/home/peter/bo/2013ws/ainf/20131031.0> ls -l  
...  
-rw-r--r-- 1 peter peter    1539 Nov 29  2012 orbit-x1.c
```



Gruppe (g – *group*) darf lesen

2.3 Dateisysteme

- *Zugriffsrechte*

```
phoenix/home/peter/bo/2013ws/ainf/20131031.0> ls -l
...
-rw-r--r-- 1 peter peter    1539 Nov 29  2012 orbit-x1.c
```



alle anderen (o – *other*) dürfen lesen

2.3 Dateisysteme

- *Zugriffsrechte*

```
phoenix/home/peter/bo/2013ws/ainf/20131031.0> ls -l
...
-rw-r--r-- 1 peter peter    1539 Nov 29  2012 orbit-x1.c
```

- Zugriffsrechte ändern:

`chmod o-r orbit-1x.c` – Lesezugriff entziehen

`chmod g+w orbit-1x.c` – Schreibzugriff gewähren

`chmod 640 orbit-1x.c` – auf `-rw-r-----` setzen

2.3 Dateisysteme

- *Zugriffsrechte*


```
phoenix/home/peter/bo/2013ws/ainf/20131031.0> ls -l
...
-rw-r--r-- 1 peter peter    1539 Nov 29  2012 orbit-x1.c
```

- Zugriffsrechte ändern:

`chmod o-r orbit-1x.c` – Lesezugriff entziehen

`chmod g+w orbit-1x.c` – Schreibzugriff gewähren

`chmod 640 orbit-1x.c` – auf `-rw-r-----` setzen


The diagram shows three red curly braces under the octal permissions 6, 4, and 0. The first brace under '6' spans the first two characters 'rw' of the permission string. The second brace under '4' spans the third character 'r'. The third brace under '0' spans the last five characters '-----'.

2.3 Dateisysteme

- *ausführbare* Dateien

```
cassini/home/peter/bo/2013ws/systech/20131008.0> cat test
ls -l systech-20131008.*
cassini/home/peter/bo/2013ws/systech/20131008.0> chmod +x test
cassini/home/peter/bo/2013ws/systech/20131008.0> ls -l test
-rwxr-xr-x 1 peter peter  25 Okt 6 16:45 test
cassini/home/peter/bo/2013ws/systech/20131008.0> ./test
-rw-r--r-- 1 peter peter 4120 Okt 6 16:44 systech-20131008.aux
...
```

- ausführbare Textdateien: *Skripte*

hier: ausführbare Textdatei mit Shell-Befehlen
(ohne spezielle Kennung): Shell-Skript

Kennung: 1. Zeile enthält `#!` und den Interpreter,
z. B. `#!/bin/bash`

2.3 Dateisysteme

- *Symbolische Verknüpfungen – symbolic links*

Verweis auf die eigentliche Datei

→ Wenn man die Datei löscht, zeigt der Link ins Leere.

Verknüpfung anlegen: `ln -s datei link`

(Richtung: wie bei `cp`)

Beispiel: `ln -s opengel-magic-double.c opengl-magic.c`

- *Harte Verknüpfungen – hard links*

Dieselben Daten auf dem Datenträger

sind unter mehreren Namen verfügbar.

→ Wenn man einen löscht, sind die Daten noch da.

(aus Anwendersicht eher selten, daher hier nicht ausführlich)

2.3 Dateisysteme

- **find**: Dateien anhand ihrer Eigenschaften suchen

```
$ find . -name "*orbit-x*"
./20131031.0/orbit-x.c
./20131031.0/orbit-x1.c
./20131031.0/orbit-x
./20131107.0/orbit-x.c
./20131107.0/orbit-x1.c
./20131107.0/orbit-x
$ find . -name "*orbit-x*" -perm /u+x
./20131031.0/orbit-x
./20131107.0/orbit-x
$ find . -name "*orbit-x*" -perm /u+x -exec ls -l {} \;
-rwxr-xr-x 1 peter peter 15831 Okt 31 13:19 ./20131031.0/orbit-x
-rwxr-xr-x 1 peter peter 15831 Okt 31 13:19 ./20131107.0/orbit-x
```

2.4 Ein- und Ausgabeströme

- Standard-Ausgabe in Datei umleiten

```
$ echo "Dies ist ein Test." > test.txt
```

```
$ cat test.txt
```

```
Dies ist ein Test.
```

- Standard-Ausgabe an Datei anhängen

```
$ echo "Dies ist noch ein Test." >> test.txt
```

```
$ cat test.txt
```

```
Dies ist ein Test.
```

```
Dies ist noch ein Test.
```

2.4 Ein- und Ausgabeströme

- Fehler-Ausgabe in Datei umleiten

```
$ cat gibtsnicht.txt > fehler.txt
cat: gibtsnicht.txt: No such file or directory
$ cat fehler.txt
$ cat gibtsnicht.txt 2> fehler.txt
$ cat fehler.txt
cat: gibtsnicht.txt: No such file or directory
```

2.4 Ein- und Ausgabeströme

- Standard-Eingabe aus Datei lesen

```
$ bc
bc 1.06.95
Copyright [...] 2006 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type 'warranty'.
2 + 2
4
$ echo "2 + 2" > test.bc
$ bc < test.bc
4
```

2.5 Pipes

Standard-Ausgabe von Programm A
wird zu Standard-Eingabe von Programm B

```
$ echo "2 + 2" | bc  
4
```

—→ sehr mächtiger „Baukasten“

2.5 Pipes

- `sed`: *stream editor*

Suchen und Ersetzen (und noch viel mehr)

```
$ echo "Schlimmer geht nimmer." | sed -e 's/nim/im/g'  
Schlimmer geht immer.
```

2.5 Pipes

- **grep**: Standard-Eingabe durchsuchen

```
$ ls | grep slides
```

```
pgslides.sty
```

```
$ ls *.pdf | grep -v fig
```

```
logo-hochschule-bochum-cvh-text.pdf
```

```
logo-hochschule-bochum.pdf
```

```
NPN_transistor_basic_operation.pdf
```

```
rtech-20131002.pdf
```

```
$ ls -l $(ls *.pdf | grep -v fig)
```

```
-rw-r--r-- 1 peter peter 14488 Sep  2 21:02 logo-hochschule-bochum-cvh-text.pdf
```

```
-rw-r--r-- 1 peter peter 31581 Dez 26 2011 logo-hochschule-bochum.pdf
```

```
-rw-r--r-- 1 peter peter 8538 Okt  2 2012 NPN_transistor_basic_operation.pdf
```

```
-rw-r--r-- 1 peter peter 6753149 Okt  1 22:59 rtech-20131002.pdf
```


2.6 Verzweigungen und Schleifen

```
$ if grep Pipes test.txt; then echo "gefunden"; \  
    else echo "nicht gefunden"; fi  
nicht gefunden  
$ for x in foo bar baz; do echo $x; done  
foo  
bar  
baz
```

Vertiefung Systemtechnik – Eingebettete Systeme

<https://gitlab.cvh-server.de/pgerwinski/es.git>

1 Einführung

2 Einführung in Unix

2.1 Grundkonzepte

2.2 Die Kommandozeile: Grundlagen

2.3 Dateisysteme

2.4 Ein- und Ausgabeströme

2.5 Pipes

2.6 Verzweigungen und Schleifen

3 TCP/IP in der Praxis

...



Änderungen
vorbehalten