

Eingebettete Systeme

Prof. Dr. rer. nat. Peter Gerwinski

18. Oktober 2018

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)

In dieser Lehrveranstaltung

- Einführung in Unix
- TCP/IP und Bus-Systeme in der Praxis
- C-Programmierung für Fortgeschrittene
- Einführung in die Echtzeitprogrammierung
- Prüfungsleistung: Projektaufgabe
Eingebettetes System eigener Wahl zum Laufen bringen

→ **Projektaufgabe überlegen**

Weitere Ideen:

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



Änderungen
vorbehalten

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

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

z. B.: `printf()` = „normale“ Funktion
aus eine Bibliothek (`libc`)

Unix: Programme arbeiten zusammen

```
cassini/home/peter/bo> find . -name "*klausur*.tex" \  
| xargs grep -l "PBM-Datei"  
./2014ws/ainf/20150130.0/ainf-klausur-20150130.tex  
./2016ws/hp/20170920.0/klausur.tex  
./2016ws/hp/20170206.0/klausur.tex  
./2011ws/rarch/20120322.0/rarch-klausur-20120322.tex  
./2012ws/klausuren-gerwinski/rarch-klausur-20120322.tex  
./2013ws/ainf/20140918.0/ainf-klausur-20140918.tex  
./2017ws/hp/20180213.k1/klausur.tex  
./2017ws/hp/20180205/klausur.tex  
./2015ws/ainf/20160913/ainf-klausur-20160913.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.

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

- Datenträger in Verzeichnis *einhängen*: `mount`

```
cassini/home/peter> ls /media/usb1/  
cassini/home/peter> mount /media/usb1  
cassini/home/peter> ls /media/usb1/  
es-20181018.pdf  hello.c  hexapode  KIS-Bericht.pdf  
cassini/home/peter> umount /media/usb1  
cassini/home/peter> ls /media/usb1/  
cassini/home/peter>
```

2.3 Dateisysteme

- *Zugriffsrechte*

```
cassini/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*

```
cassini/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*

```
cassini/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*

```
cassini/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*

```
cassini/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*


```
cassini/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


6 4 0

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
...
```

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*

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

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.

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