UNIX use

Hans Buchmann FHNW/IME

27. September 2017

Ziel

Entwicklung von Programmen auf dem **BeagleBoneBlack**

- alles ist ein File
  - 0 – *te* Näherung
  - File: *stream of bits*
- Filesysteme
  - `mount`
  - `sshfs`
- Cross development
  - *Host* $\leftrightarrow$ **BeagleBoneBlack**

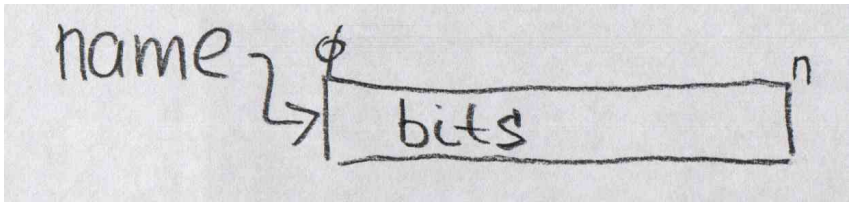
Remark: Keine Toolchain auf dem **BeagleBoneBlack**

Wichtig

- wo ist was ?
  - Verzeichnisstruktur
- wo sind wir ?
  - *Host*
    oder
  - **BeagleBoneBlack**

Alles ist ein File

stream of bits



- *name* Referenz auf die Bits (Bytes)
  - Bits(bytes) der Reihe nach
  - indexiert $0 \ldots n-1$
- File
  - Datenquelle
    - liefert Daten: Bits(Bytes)
  - Datensenke
    - absorbiert Daten: Bits(Bytes)

Ein paar Befehle

- `cat` *name*
  - *concatenate files and print on the standard output*
  - zeigt den Inhalt
- `hexdump -C` *name*
  - *display file contents in hexadecimal, decimal, octal, or ascii*
  - zeigt die Bits hexadezimal
- `dd if=... of=... count=...`
  - *convert and copy a file*
  - brauchen wir oft

Devices sind auch Files

am Beispiel SD-Karte

`/dev/mmcblk`**i** $i = 0, 1, 2...$

Remark: Name vom Betriebssystem bestimmt

Datenquelle `hexdump -C /dev/mmcblk0`

Datensenke `cp` *name* `/dev/mmcblk0`

Remark: Aufpassen

Devices sind auch Files

z.B Zufallszahlen

`/dev/random` sammelt das Rauschen: langsam

Remark: Name vom Betriebssystem bestimmt

Datenquelle `hexdump -C /dev/random`

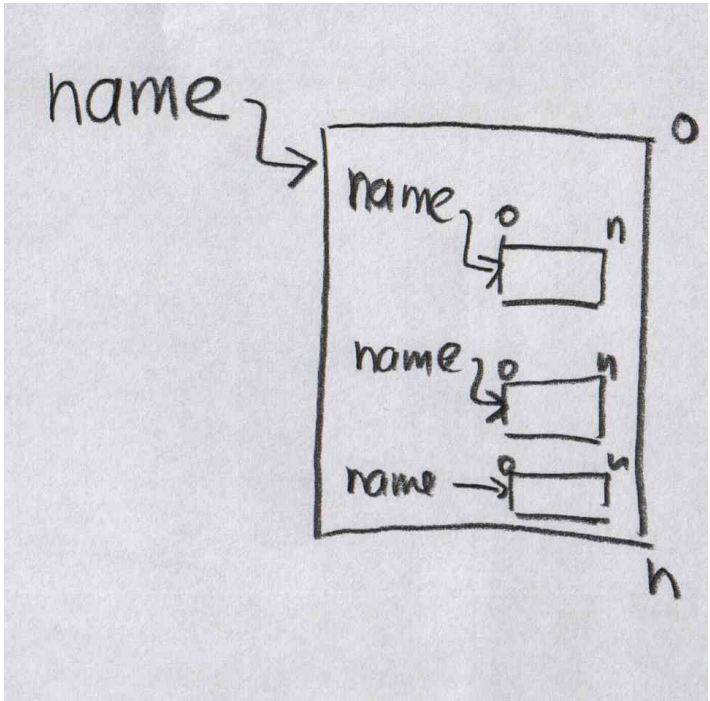
`/dev/urandom` berechnete (Pseudo) Zahlen: schnell

Datenquelle
```
play -b 16 -e signed-integer \
     -t raw -r 44000 /dev/urandom
```

Remark: der Befehl `play` hat viele Optionen

Filesystem

Files für Files

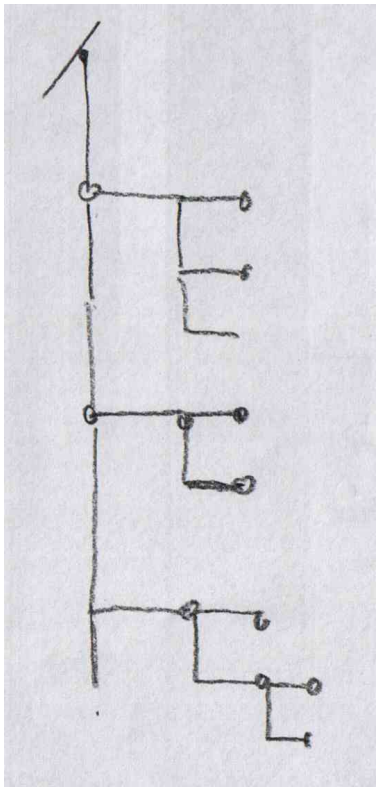


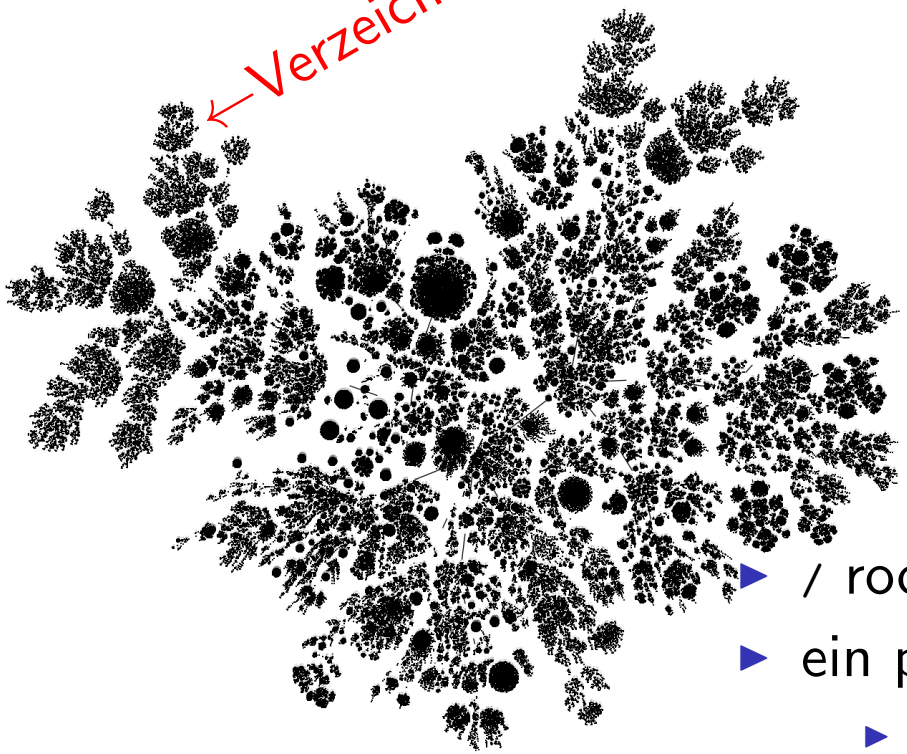
- File der weitere Files enthält
- Verschiedene Filesysteme
  - vfat Microsoft
  - ext4 UNIX
  - ... noch viele andere

```
cat /proc/filesystems
```

Vereichnisstruktur

Hierarchie

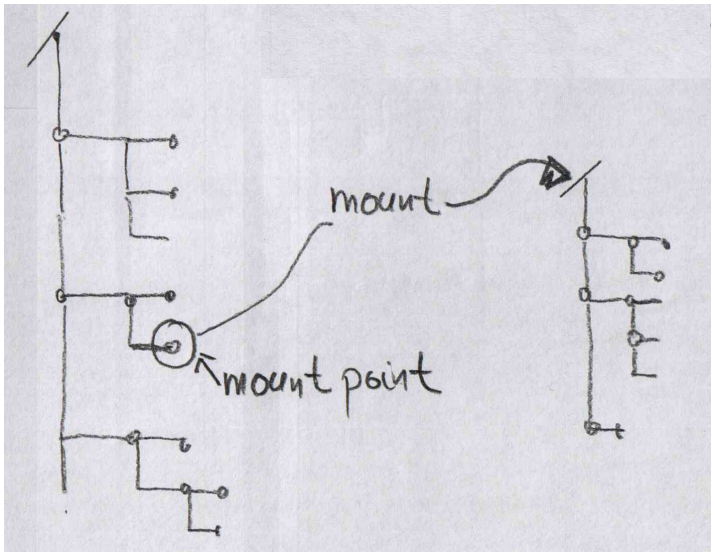




- / root
- ein paar Befehle
  - `ls, tree, cd`

`mount` *fileSystem mountPoint*

Verbindet Filesysteme

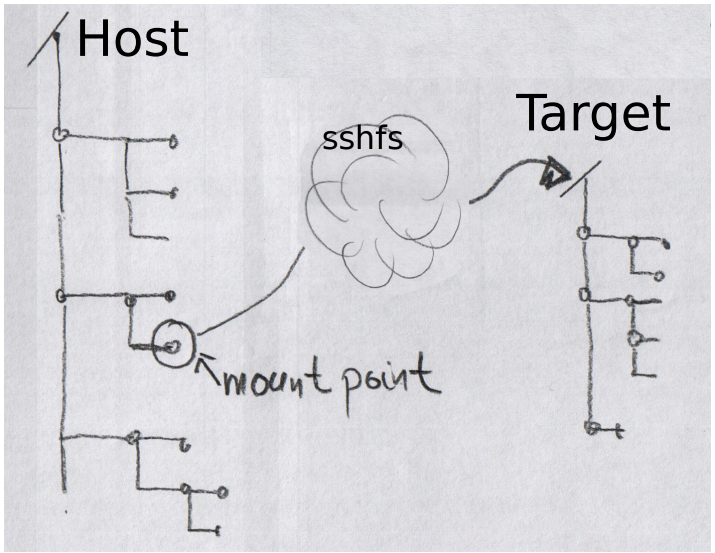


- `mount` */dev/mmcblk0p1 mountPoint*
- `umount` *mountPoint*

Remark: *mountPoint* sieht wie ein normales Verzeichnis aus

`sshfs user@target:`*path mountPoint*

via `ssh`



- `sshfs user@target:`*path mountPoint*
  - braucht `ssh`
- `fusermount -u` *mountPoint*
  - umount

Remark: *mountPoint* Sieht wie ein normales Verzeichnis aus

Crossdevelopment

- Development
  - Wo sind die Files
- CrossDevelopment
  - Wo sind die Files

Development
noch nicht Cross

- Source file `src/hello-world-c.c`
  - C
  - POSIX

  unabhängig von Platform
- Object file (Maschinencode) `hello-world-c.o`
  - erzeugt mit: `gcc -c ../src/hello-world-c.c -o hello-world-c.o`
  - Maschinencode:
    - `file hello-world-c.o`
    - `objdump -d hello-world-c.o`
- Executable file `hello-world-c`
  - `gcc hello-world-c.o -o hello-world-c`
  - Maschinencode:
    - `file hello-world-c`
    - `objdump -d hello-world-c`

Was es braucht ?

Files

- Source file
  - wo ist der *include file* `stdio.h`
- Object File
  - `nm hello-world-c.o`
  - wo ist `puts`
- Executable
  - `nm hello-world-c`
  - `ldd hello-world-c`
  - wo sind die Bibliotheken

Wo sind die Files ?

irgendwo in einem Unterverzeichnis von /

- Include Files `gcc -c -v ../src/hello-world-c.c -o hello-world-c.o`
  - `stdio.h` ?
  - `stddef.h` ?
- Bibliotheken
  - z.B. `libc.so`

Development
der Normalfall

- Host==Target
- root Host==root Target

CrossDevelopment

- immer noch POSIX
- Host!=Target

Verzeichnisstruktur

Host

```
2-unix-use ..... somewhere on the host
├── target-root ..... different possibilities
├── config
│   └── Makefile  for making BeagleBoneBlack  executables
├── src ..... c,c++
├── tc ..... toolchain
└── work  ..... connected with BeagleBoneBlack current dir
```

CrossDevelopment

Target *Host*:

- Makefile
  - `PREFFIX=`
- `target-root`
  - `ln -s / target-root`

Remark: unüblich !

CrossDevelopment

Target **BeagleBoneBlack**

- Makefile
  - `PREFFIX=../tc/bin/arm-linux-gnueabihf-`
- `target-root`
  - `sshfs debian@192.168.7.2:/ target-root`

Verzeichnisstruktur

BeagleBoneBlack

The big Picture

- Source File: `hello-world-c.c`
- falls es nicht klapt ?
  - wo ist der File ?

Development

`hello-world-c.c`

Host	Target	OS	Toolchain	Verbindung	Bemerkunger
BBB	**BBB**	Debian	mitgeliefert		
Host	**BBB**	Debian	`gcc-7.2.0-arm-64bit.tar.bz2`	sshfs	
Host	**BBB**	minimal	`gcc-7.2.0-arm-64bit.tar.gz`	SD-Card	später
Host	**BBB**	minimal	`gcc-7.2.0-arm-64bit.tar.gz`	curlftpfs	später