

# Ein ganzes GNU/Linux

Hans Buchmann FHNW/IME

6. Januar 2016

## Um was geht es ?

- ▶ ein GNU/Linux von Grund auf bauen
  - ▶ nicht mehr so schwer wie auch schon
- ▶ nicht völlig automatisiert
- ▶ Alternative zu **yocto** & Co.

# Ziel

## GNU/Linux auf dem BeagleBoneBlack

- ▶ command based
- ▶ Ethernet
- ▶ ssh
- ▶ sshfs
- ▶ moderne Toolchain inkl. *c++11* **C++**

# Komponenten BeagleBoneBlack und *Host*

## BeagleBoneBlack

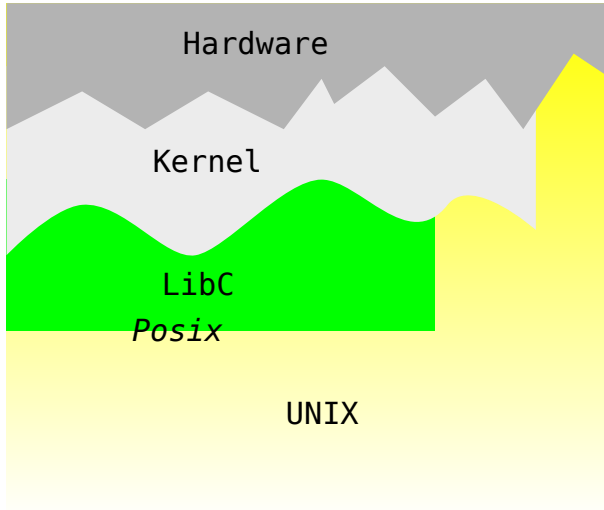
Kernel ein File

root ein Filesystem

## *Host*

Toolchain binutils, gcc

# Übersicht



# Die Komponenten für BeagleBoneBlack

Hardware **BeagleBoneBlack**

Kernel zugeschnitten auf **BeagleBoneBlack**

▶ <https://github.com/beagleboard/linux>

root das Filesystem

LibC eglibc

▶ <http://www.eglibc.org/home>

UNIX busybox

▶ <http://www.busybox.net/>

... Weitere UNIX basierte Komponenten

▶ das configure, make, make install  
Triple

# Toolchain

binutils linker & Co.

gcc compiler

- ▶ `libgcc` die Bibliothek für den Compiler

## Remark(s):

- ▶ die Toolchain muss zweimal gebaut werden
  - ▶ für den **kernel** und `libc`
  - ▶ für UNIX/**POSIX**
- ▶ das target
  - ▶ `cpu-vendor-os`

# Die Verzeichnisstruktur

```
|
├── scripts
│   └── common.h .....used in (all) scripts
├── config
│   └── Makefile
├── build ..... home of the build files
├── target-root ..... top of target root
├── tc ..... the new toolchain
├── src ..... own programs
├── work
│   └── → ../config/Makefile
```



# Build 1

## Toolchain 1

- ▶ `binutils`
- ▶ `gcc-bare`
  - ▶ nur für den *kernel*
  - ▶ das bare minimum
  - ▶ nur `C`

## Build 2

### Kernel

- ▶ kernel mit ein paar *targets*
  - ▶ `bb.org_defconfig`
  - ▶ `zImage`
  - ▶ `headers_install`
    - ▶ Interface: *kernel-libc*

## Build 3

libc

Wir brauchen eglibc

- ▶ eglibc

## Build 4

### Toolchain 2

- ▶ gcc
  - ▶ mit sysroot
  - ▶ C und C++
- ▶ Test
  - ▶ im Verzeichnis work

## Build 5

### busybox

- ▶ busybox
  - ▶ Installation auf SD-Card
  - ▶ fakeroot

# Skripts und Argumente

## initiales System

Skript	target	gebraucht für
<code>binutils.sh</code>		alles
<code>gcc-bare.sh</code>		kernel libc
<code>kernel.sh</code>	<code>bcmrpi_defconfig</code> <code>zImage</code> <code>headers_install</code>	
<code>eglibc.sh</code>		POSIX
<code>gcc.sh</code>		C/C++ POSIX
<code>busybox.sh</code>	<code>busybox</code> <code>install</code>	
<code>target-root.sh</code>		Transfer auf SD-Card

**Remark:** Alle Skripte sind **bash** Skripte

# Target

## erster Versuch

- ▶ transfer auf SD Karte
- ▶ Internet

# Skripts und Argumente

ssh

`zlib.sh`

`openssl.sh`

die kryptographischen Algorithmen

`openssh.sh`

**Remark:** `openssh.sh` hängt von `zlib.sh` und `openssl.sh` ab



## Build 6

ssh

- ▶ openssh die volle Implementation
  - ▶ zlib
  - ▶ openssl
  - ▶ openssh

sshfs

funktioniert noch nicht

- ▶ Die Bibliothek `glib`
- ▶ Ersatz
  - ▶ `sftp`

# Modules

- ▶ siehe 6-modules

# Modules

## Verzeichnisstruktur

### Host

```
BUILD_HOME
├── modules
│   ├── Makefile ..... for own modules
│   └── *.c/h ..... the own sources
└── scripts
    └── module.sh
```

### BeagleBoneBlack

```
/
└── work
```

# Herstellung Workflow

## Host

- ▶ `sh scripts/module.sh module`
- ▶ `scp, sftp, ftp`

## BeagleBoneBlack

- ▶ `insmod`
- ▶ `rmmmod`