Minimales UNIX

Hans Buchmann FHNW/IME

17. Dezember 2014

Um was geht es ?
Minimales UNIX

- *minimal*: ein einziger Prozess
- *busybox* ein kleines UNIX
- ein kleines UNIX aus einem grossen UNIX

Remark: Grosse UNIX gibt es viele
- kleine weniger

Single Process

ein einfaches `init`

```
#include <unistd.h>

int main(int argc,char** args)
{
  static const char MSG[]="Hello World from '"
                          __FILE__ "'\t'" __DATE__ "'\n";
  write(STDOUT_FILENO,MSG,sizeof(MSG));
  while(1){}
}
```

Was ist zu tun ?

- `minimal-1` herstellen
  - siehe `devel`
  - `-static` ohne dynamische Bibliotheken
- Kleines `root-fs` mit nur einem File
  - `minimal-1`

  Remark: versuche *extended* Partitionen

  auf SD-Karte
- *uboot*
  - `setenv bootargs '...'`
- starten

Remark: ein sehr kleines aber vollwertiges UNIX

busybox

ist ein kleines UNIX

- `https://github.com/raspberrypi/target_fs`
- Herstellung
  - Siehe Minimal 3

initrd
Initiale RAM Disk

- enthält ein kleines UNIX
- entsteht aus einem grossen UNIX
- spezielles Fileformat: cpio

Aufgabe

grosses UNIX → `initrd` → kleines UNIX

- erzeuge `initrd`
  - mit `mkinitcpio`
- erzeuge *target-root* aus `initrd`
  - mit `cpio`
- erzeuge *partition* auf *SD-Card*
- setze `bootargs`
  - im `u-boot`

cpio

eine Anwendung von *pipes*

create

```
find . | cpio --create | gzip --stdout -> file
```

- find erzeugt Fileliste
- cpio alle Files in einen *stream*
- gzip Komprimation in *archive*

extract

```
gzip -d -c path-to-archive | cpio --extract
```

- gzip Dekomprimiert archive in *stream*
- cpio erzeugt files aus *stream*