Linux Boot

EIM/INFM

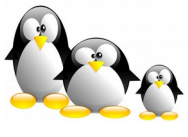
Frank Erdrich
frank.erdrich@emtrion.de

U-Boot SPL

- Vorbereiten der Hardware für das Betriebssystem
  - Taktgeneratoren für CPU und Peripherie
  - Externer Speicher
  - Laden und starten des U-Boot
- Eventuell SecureBoot

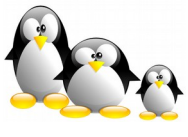
U-Boot

- Evtl. zusätzliche Hardwareinitialisierung
- Lädt den Linux-Kern von diversen Quellen
  - NAND, NOR, SD-Karte
  - Ethernet
- Kann weitere Ressourcen laden
  - DeviceTree
  - InitRAMFS



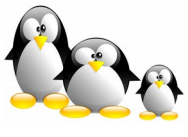
U-Boot

- Unterstützung für verschiedene Kernelformate
  - bootm -> unkomprimiert
  - bootz -> komprimiert
  - FIT-Image: Flattened Image Tree



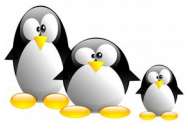
FIT-Image

- Enthält Kernel und weitere Bestandteile
- Struktur ähnlich dem Devicetree
- Integritätscheck der Bestandteile
- Mehrere verschiedene Komponenten in einem FIT-Image
  - Mehrere Kernel
  - Mehrere Devicetrees



FIT-Image

- Vereinfacht SecureBoot
  - Ganzes Image kann verschlüsselt und signiert werden anstatt der einzelnen Bestandteile
- Direkte Unterstützung für signierte Images von U-Boot



FIT-Image

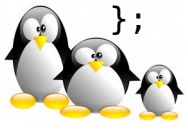
```
/dts-v1/;
/ {
     description = "First test with fit image.";
     #address-cells = <1>;

     images {
          kernel@1 {
               description = "Linux kernel";
               data = /incbin/("./arch/arm/boot/zImage");
               type = "kernel";
               arch = "arm";
               os = "linux";
               compression = "none";
               load = <0x22000000>;
               entry = <0x22000000>;
               hash@1 {
                    algo = "sha256";
               };
          };

          dtb@1 {
               description = "Devicetree blob";
               data = /incbin/("./arch/arm/boot/dts/sama5d2.dtb");
               type = "flat_dt";
               arch = "arm";
               compression = "none";
               load = <0x21000000>;
               hash@1 {
                    algo = "sha256";
               };
          };
     };

     configurations {
          default = "config@1";
          config@1 {
               description = "Testkernel";
               kernel = "kernel@1";
               fdt = "dtb@1";
          };
     };
};
```

```
mkimage -f fit_image.its linux.itb
```

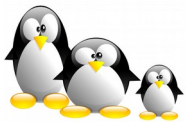


Linux laden und starten

```
$ fatload mmc 0:1 0x21000000 sama5d2.dtb
$ fatload mmc 0:1 0x22000000 zImage
$ bootz 0x22000000 - 0x21000000
```

Linux Commandline:

```
console=ttyS0,115200 earlyprintk root=/dev/mmcblk0p2 rw rootwait
```

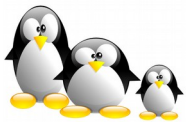


Linux startet

```
bootconsole [earlycon0] enabled
cma: Reserved 16 MiB at 0x2e800000
Memory policy: Data cache writeback
CPU: All CPU(s) started in SVC mode.
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 65024
Kernel command line: console=ttyS0,115200 earlyprintk rootfstype=ubifs ubi.mtd=0 root=ubi0:rootfs rw
rootwait
PID hash table entries: 1024 (order: 0, 4096 bytes)
Dentry cache hash table entries: 32768 (order: 5, 131072 bytes)
Inode-cache hash table entries: 16384 (order: 4, 65536 bytes)
Memory: 231356K/262144K available (7168K kernel code, 513K rwdata, 1492K rodata, 1024K init, 1142K bss,
14404K reserved, 16384K cma-reserved)
Virtual kernel memory layout:
    vector  : 0xffff0000 - 0xffff1000   (   4 kB)
    fixmap  : 0xffc00000 - 0xfff00000   (3072 kB)
    vmalloc : 0xd0800000 - 0xff800000   ( 752 MB)
    lowmem  : 0xc0000000 - 0xd0000000   ( 256 MB)
    modules : 0xbf000000 - 0xc0000000   (  16 MB)
      .text : 0xc0008000 - 0xc0800000   (8160 kB)
      .init : 0xc0a00000 - 0xc0b00000   (1024 kB)
      .data : 0xc0b00000 - 0xc0b8067c   ( 514 kB)
       .bss : 0xc0b8067c - 0xc0c9e0e8   (1143 kB)
Preemptible hierarchical RCU implementation.
  Build-time adjustment of leaf fanout to 32.
NR_IRQS:16 nr_irqs:16 16
clocksource: pit: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns: 14479245754 ns
sched_clock: 32 bits at 100 Hz, resolution 10000000ns, wraps every 21474836475000000ns
```

Exkurs: NFS Boot

- NFS -> Network File System
- Teilt Dateisystem über das Netzwerk mit anderen Systemen
- Wird im Zielsystem wie ein normales Dateisystem eingebunden

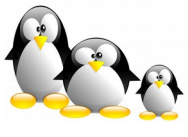


Exkurs NFS Boot

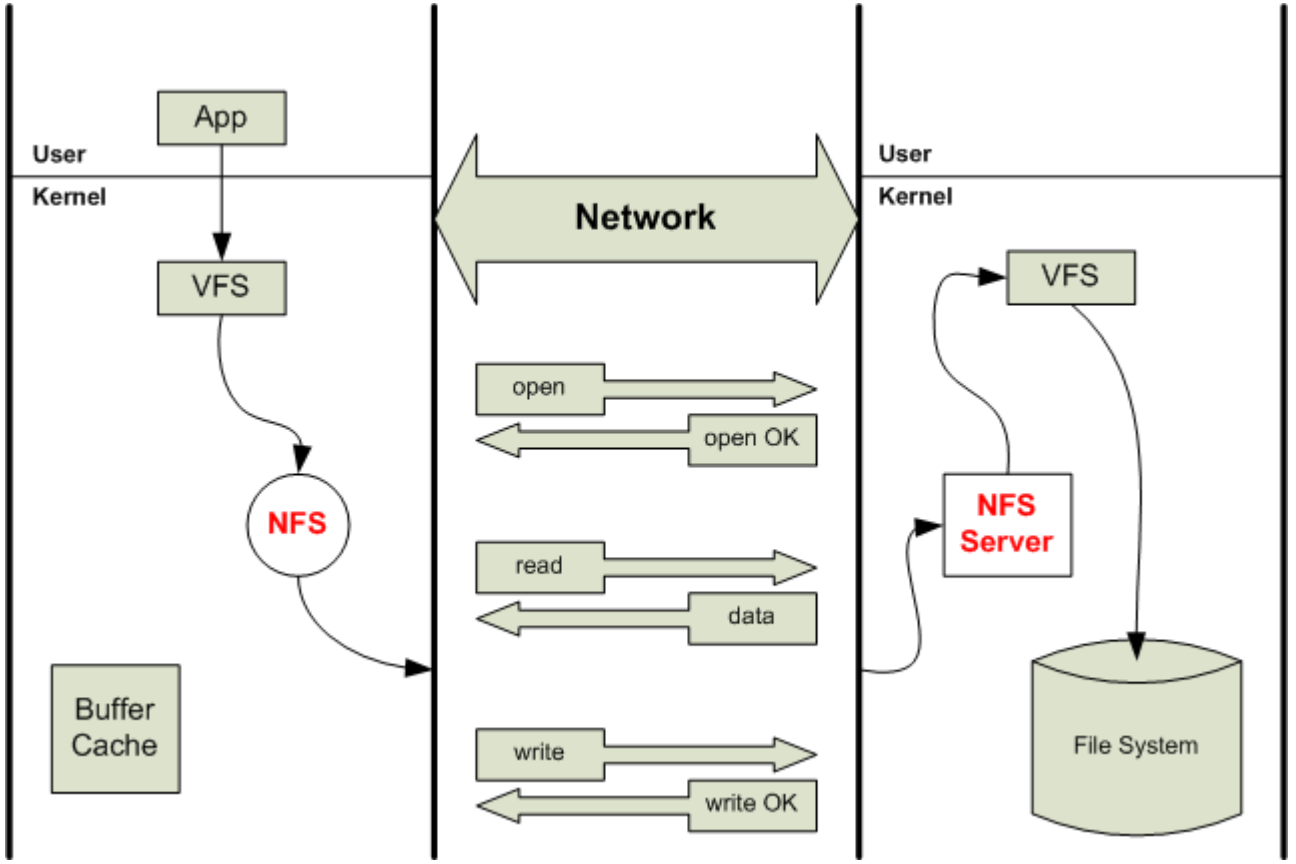
- Zu exportierendes Verzeichnis wird in /etc/exports definiert

```
/work/rootfs/sama5d2 *(rw,nohide,insecure,no_subtree_check,async,no_root_squash)
```

- Exportierte Verzeichnisse anzeigen:
  `$ sudo showmount -e`
- Neue Dateien können direkt auf dem Host in das Verzeichnis kopiert werden und sind sofort auf dem Client sichtbar



NFS



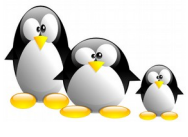
Quelle: http://www.read.seas.harvard.edu/~kohler/class/05f-osp/notes/fig18-01.png

NFS Boot - U-Boot

- Boards sind dafür vorbereitet

```
$ setenv nfsroot /work/rootfs/sama5d2
$ setenv serverip 192.168.8.1
$ saveenv
$ run net_boot
```

- nfsroot: Pfad zum Verzeichnis auf dem Host
- serverip: IP-Adresse des Hosts



Linux startet

```
bootconsole [earlycon0] enabled
cma: Reserved 16 MiB at 0x2e800000
Memory policy: Data cache writeback
CPU: All CPU(s) started in SVC mode.
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 65024
Kernel command line: console=ttyS0,115200 earlyprintk rootfstype=ubifs ubi.mtd=0 root=ubi0:rootfs rw
rootwait
PID hash table entries: 1024 (order: 0, 4096 bytes)
Dentry cache hash table entries: 32768 (order: 5, 131072 bytes)
Inode-cache hash table entries: 16384 (order: 4, 65536 bytes)
Memory: 231356K/262144K available (7168K kernel code, 513K rwdata, 1492K rodata, 1024K init, 1142K bss,
14404K reserved, 16384K cma-reserved)
Virtual kernel memory layout:
    vector  : 0xffff0000 - 0xffff1000   (   4 kB)
    fixmap  : 0xffc00000 - 0xfff00000   (3072 kB)
    vmalloc : 0xd0800000 - 0xff800000   ( 752 MB)
    lowmem  : 0xc0000000 - 0xd0000000   ( 256 MB)
    modules : 0xbf000000 - 0xc0000000   (  16 MB)
      .text : 0xc0008000 - 0xc0800000   (8160 kB)
      .init : 0xc0a00000 - 0xc0b00000   (1024 kB)
      .data : 0xc0b00000 - 0xc0b8067c   ( 514 kB)
       .bss : 0xc0b8067c - 0xc0c9e0e8   (1143 kB)
Preemptible hierarchical RCU implementation.
  Build-time adjustment of leaf fanout to 32.
NR_IRQS:16 nr_irqs:16 16
clocksource: pit: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns: 14479245754 ns
sched_clock: 32 bits at 100 Hz, resolution 10000000ns, wraps every 21474836475000000ns
```

Exkurs Devicetree

- Wie weiß der Linux Kernel, welche Hardware auf dem System vorhanden ist?

Exkurs Devicetree

- X86: BIOS / UEFI
- ACPI:
  Advanced Configuration and Power Interface
- Und eingebettete Systeme ohne BIOS?

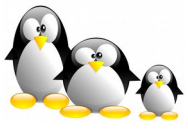
Exkurs Devicetree

- https://www.devicetree.org
- Beschreibt die vorhandene Hardware
- Oder auch nicht...!!!
- Erinnerung: FIT-Images werden ebenfalls mit dieser Struktur beschrieben

Exkurs Devicetree

```
/dts-v1/;

/ {
    node1 {
        a-string-property = "A string";
        a-string-list-property = "first string", "second string";
        // hex is implied in byte arrays. no '0x' prefix is required
        a-byte-data-property = [01 23 34 56];
        child-node1 {
            first-child-property;
            second-child-property = <1>;
            a-string-property = "Hello, world";
        };
        child-node2 {
        };
    };
    node2 {
        an-empty-property;
        a-cell-property = <1 2 3 4>; /* each number (cell) is a uint32 */
        child-node1 {
        };
    };
};
```



Exkurs Devicetree

- Im Devicetree ist nun definiert, welche Hardware/Peripherie zur Verfügung steht.
- Nur, woher weiß der Kernel, welche Treiber er für die einzelnen Peripheriegeräte laden muss?

- Siehe Beispieldevicetree...

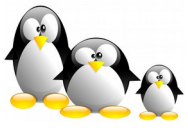
Exkurs Devicetree

- Compatible String
- Beispiel: drivers/tty/serial/imx.c

```
static const struct of_device_id imx_uart_dt_ids[] = {
    { .compatible = "fsl,imx6q-uart", .data = &imx_uart_devdata[IMX6Q_UART], },
    { .compatible = "fsl,imx53-uart", .data = &imx_uart_devdata[IMX53_UART], },
    { .compatible = "fsl,imx1-uart", .data = &imx_uart_devdata[IMX1_UART], },
    { .compatible = "fsl,imx21-uart", .data = &imx_uart_devdata[IMX21_UART], },
    { /* sentinel */ }
};
MODULE_DEVICE_TABLE(of, imx_uart_dt_ids);
```

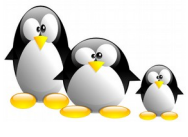
Linux Boot II

- Kernel startet den Init-Prozess
- Zu finden unter /sbin/init
- Hat PID 1 (Process ID)
- Startet alles weitere im System
  - Konsole
  - Dateisystemmounts
  - Dienste



Linux Boot II

- Unterstützen verschiedene Runlevel
  - 0: Shutdown
  - S: Single-User (Minimalsystem)
  - 1: meist gleichzusetzen mit S
  - 2: Multiuser ohne Netzwerk
  - 3: Multiuser mit Netzwerk, ohne GUI
  - 5: Runlevel 3 mit GUI
  - 6: Reboot



Linux Boot II

- Gesteuert durch /etc/inittab (SysV)
  - Definiert default-runlevel
  - Definiert für jeden Runlevel die zu startenden Dienste

```
is:3:initdefault:
p3:s1234:powerfail:/usr/sbin/shutdown -y -i5 -g0 >/dev/msglog 2<>/dev/...
sS:s:wait:/sbin/rcS              >/dev/msglog 2<>/dev/msglog </dev/console
s0:0:wait:/sbin/rc0              >/dev/msglog 2<>/dev/msglog </dev/console
s1:1:respawn:/sbin/rc1           >/dev/msglog 2<>/dev/msglog </dev/console
s2:23:wait:/sbin/rc2             >/dev/msglog 2<>/dev/msglog </dev/console
s3:3:wait:/sbin/rc3              >/dev/msglog 2<>/dev/msglog </dev/console
s5:5:wait:/sbin/rc5              >/dev/msglog 2<>/dev/msglog </dev/console
```

Linux Boot II

- Kann durch eigene Scripte erweitert werden
  - /etc/rc.d/rcx.d (x={0, 1, 2, 3, 4, 5})

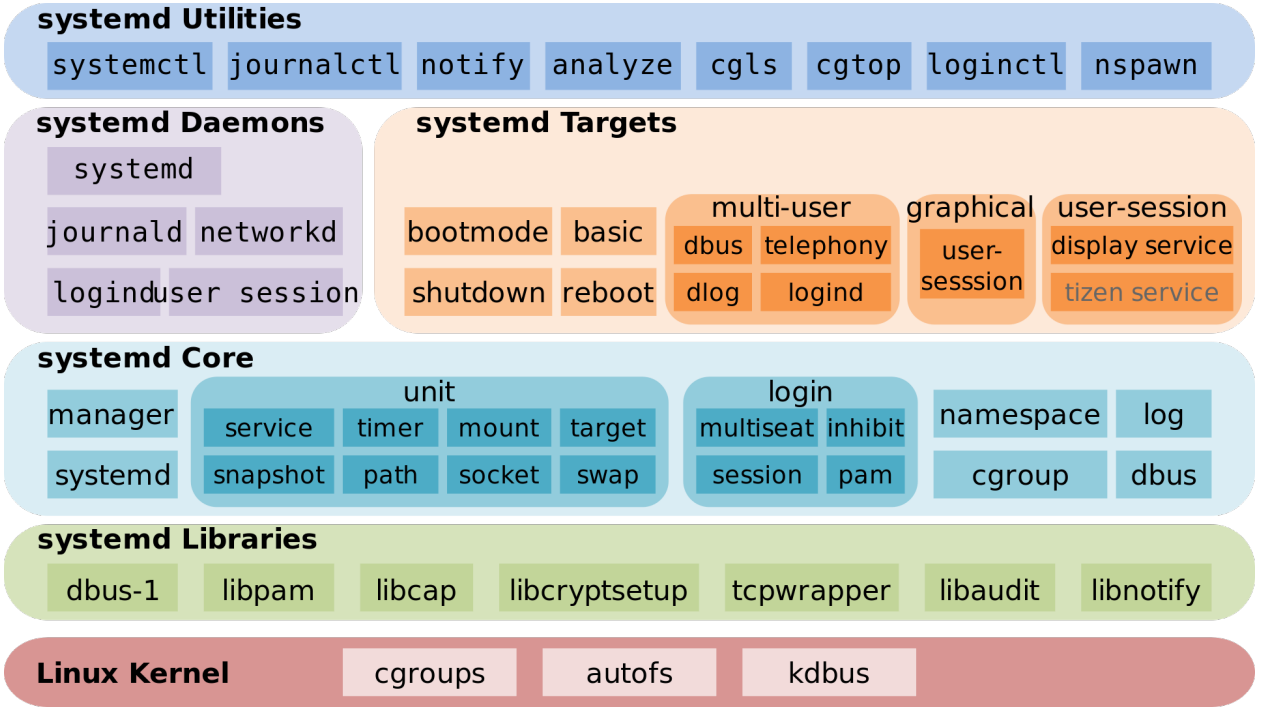
```
lrwxrwxrwx.  1 root root        10 23. Jul 2018  rc0.d -> rc.d/rc0.d
lrwxrwxrwx.  1 root root        10 23. Jul 2018  rc1.d -> rc.d/rc1.d
lrwxrwxrwx.  1 root root        10 23. Jul 2018  rc2.d -> rc.d/rc2.d
lrwxrwxrwx.  1 root root        10 23. Jul 2018  rc3.d -> rc.d/rc3.d
lrwxrwxrwx.  1 root root        10 23. Jul 2018  rc4.d -> rc.d/rc4.d
lrwxrwxrwx.  1 root root        10 23. Jul 2018  rc5.d -> rc.d/rc5.d
lrwxrwxrwx.  1 root root        10 23. Jul 2018  rc6.d -> rc.d/rc6.d
```

Linux Boot II

- Diverse INIT-Systeme
  - Systemd
  - SysV
  - Upstart

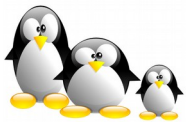
SystemD

- Entwickelt von Lennart Poettering und Kay Sievert
- 2010 erschienen



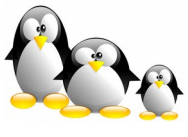
SystemD

- Ersetzt nach und nach SysV
- Parallele Ausführung des Starts
  - Baut komplexen Abhängigkeitsbaum auf
- Abwärtskompatibel zu SysV
  - Setzt die Scripte in Systemd-Services um



SystemD

- Stark kritisiert
  - Unix-Philosophie: ein Programm soll genau ein Problem lösen
- Systemd:
  - Systemstart
  - Logging
  - ...
- Binäre Logdateien
- Ist nur auf Linux nutzbar



SystemD

- Stark kritisiert
  - Binäre Logdateien
  - Ist nur auf Linux nutzbar
  - Ignorierte Bugreports
  - ...