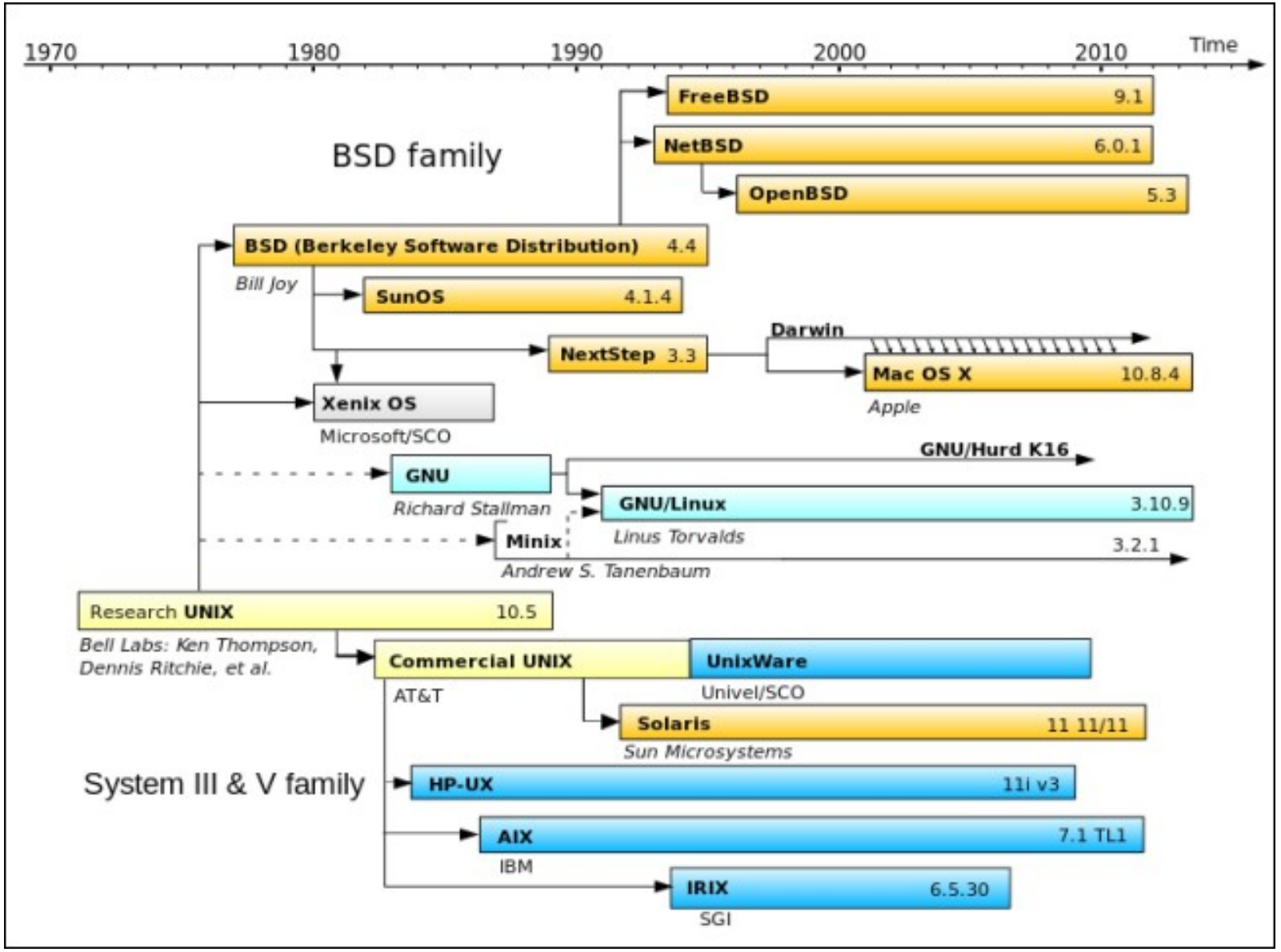
Embedded Linux introduction

Pierre Ficheux (pierre.ficheux@smile.fr)

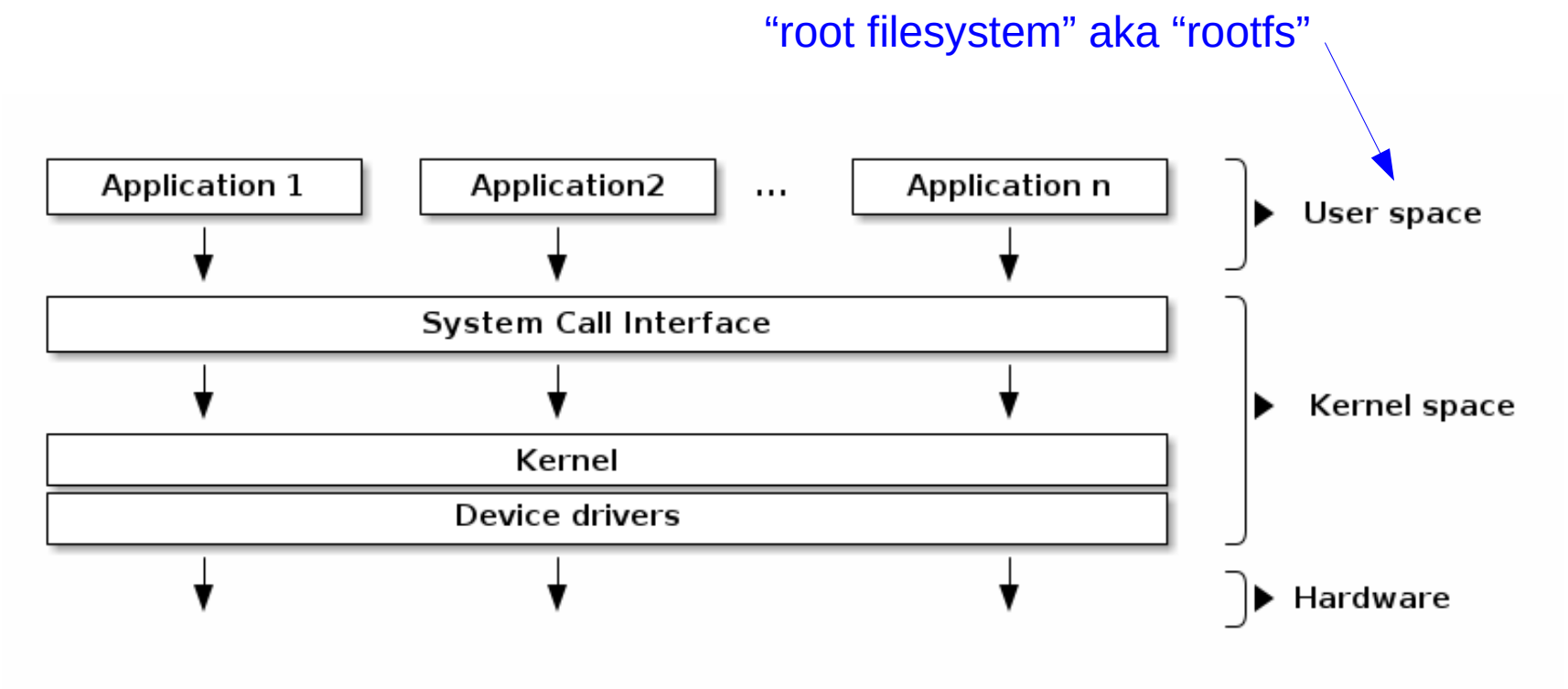
December 2023

Linux history and licensing

UNIX Family tree



UNIX/Linux architecture



Linux origins

- Linux is strongly related to GNU project (GNU is Not UNIX) by Richard Stallman (MIT, 80's)
- GNU provides an OSS environment running on proprietary UNIX
- The official name of Linux OS is *GNU/Linux* because "Linux" is only the kernel part
- No AT&T/UNIX licensing issue (finally)
- Internet has/had an very important role in the success of the Linux project
- Linus Torvalds's appearance is a bit more "comforting" than R. Stallman's one :-)

They did it !

Using Linux for embedded ?

- Pros
  - Source availability (smart for long time projects)
  - Lots of “free” tools
  - Standard compliance (POSIX, etc.)
  - Same API as “classical” Linux (desktop)
  - Hardware support (BSP = Board Support Package)
- Cons
  - High memory footprint !
  - Free but not “free of charge”
  - Lack of documentation (not true for Yocto !)
  - Licensing issues (GPL) !

Licensing

Free/OSS license

- Licensing is *VERY important* for industry !
- The license is a contract between the editor and the user
- A “free” license adds 4 fundamental “freedoms” (FSF)
  - Using software (even commercially)
  - Studying and modifying source code
  - Redistribute copies to help your neighbor
  - Modify and distribute improvements publicly

- Not to be confused with:
  - freeware/shareware (the license is not compatible with the OSS model – source code not available)
  - “public domain” (no intellectual property rights) → SQLite

Free software vs “open source”

- OSS is based on the principles of the free software
- “Split” around 1998 (Eric S. Raymond)
- OSS added 10 conditions for a compatible license
- According to Richard Stallman, the fundamental difference between the two concepts lies in their philosophy:

  “Open source is a development methodology, free software is a social movement”

GPL License

- GPL originates from the GNU project of the Free Software Foundation (Richard M. Stallman)
- GPL = **G**eneral **P**ublic **L**icense
- Based on "copyleft" (vs "copyright") and “derivative work”
- GPLv2 (1991) was the first commonly used (Linux kernel and many free tools)
- Principles :
  - The license only applies in case of redistribution
  - A GPL source code must be published (i.e. whoever receives the binary version can ask the source code)
  - No static / dynamic "link" possible between GPL code and proprietary licensed code

LGPL and “isolation”

- The GPL is complex to manage in an industrial context → LGPL
- “Linking” proprietary code with LGPL code is permitted (*Lesser*/*Library* GPL) !
- LGPL is rather “library” oriented (*Library* GPL)
- System libraries are released under LGPL (Glibc)
- In case of proprietary application it is necessary to check that no “linked” library is under GPL
- The LGPL avoids the “contamination” of a proprietary user-space program

Kernel space = GPL

- You must use the GPL in kernel space (drivers)
- If the driver does not use the GPL
  - “tainted kernel” warning message when loading
  - some API not available (such as USB - usbcore)
- In practice, there is a “tolerance” for some (big) companies
- NVIDIA finally released open source GPU Kernel module in May 2022 !
- Very few kernel developments in Android (thanks to the HAL) !

GPLv3 / LGPLv3

- New version released in 2007
- Answer to “Tivoization” (TiVO company)
- Tivoization = provide source code but does not provide any way to update the GPL/LGPL packages
- The GPLv2 simply requires the publication of sources
- The GPLv3 requires the manufacturer of a “user product” (B2C) to provide a way to update GPLv3 / LGPLv3 components !
- A “user product” is defined in §6 of GPLv3 license
- Not needed for B2B !
- Lots of projects use GPLv3 / LGPLv3 (such as Qt)
- The Linux kernel is not covered by the GPLv3 (Linus does not agree with it !)

GPL / LGPL, what to convey ?

- For GPL / LGPL packages
  - The source code or modifications (diff)
  - Update information + tools (in case of GPL/LGPL v3 and “user product”)
- You must provide a way to display the open source licenses using a GUI on the product

“To comply with the license requirements of open source libraries, you as a developer are responsible for appropriately displaying the notices for the open source libraries that your app uses.” (Android documentation)

How to display OSS licenses in Android

Apache 2 license

- Usable in user space
- Supplied by ASF in 2004
- Close to the BSD and MIT
- The Apache License is permissive unlike copyleft licenses !
- Widely used in Android/AOSP (Google source code)
- No GPL/LGPL licensing issue with Android !

Publication of the source code (Linksys)

```
WHAT_EU
|-- gpl_DMC250,DMP100,DMRW1000.tgz
|-- gpl_DMC350.tgz
|-- GPL_PACKAGE_EU_20090317
|   |-- AX88796B.tar.gz
|   |-- busybox.tar.gz
|   |-- lib_live555.tar.gz
|   |-- lib_tag.tar.gz
|   |-- linux-2.6.tar.gz
|   |-- make.sh
|   |-- memaccess.tar.gz
|   |-- mtd-utils-1.0.0.tar.gz
|   |-- ntpclient-2007.tar.gz
|   |-- README_TOOLCHAIN.txt
|   |-- u-boot.tar.gz
|   `-- wireless_tools.29.tar.gz
|-- GPL_PACKAGE_EU_20090317.zip
|-- Offer_Source Code.doc
`-- Wireless Home Audio license notice.pdf
```

sources of GPL components

BusyBox assignment (2009)

JUDGE SCHEINDLIN

'09 CIV 10155

SOFTWARE FREEDOM LAW CENTER, INC.
Daniel B. Ravicher (DR1498)
Aaron Williamson (AW1337)
Mishi Choudhary (MC2009)
Michael A. Spiegel (MS2309)
1995 Broadway, 17th Floor
New York, NY 10023-5882
Tel.: 212-580-0800
Fax.: 212-580-0898

Attorneys for Plaintiffs *Software Freedom Conservancy, Inc.* and *Erik Andersen*

UNITED STATES DISTRICT COURT
FOR THE SOUTHERN DISTRICT OF NEW YORK

SOFTWARE FREEDOM CONSERVANCY, INC. and ERIK ANDERSEN,	
Plaintiffs,	ECF CASE
-against-	Civil Action No. CV_____
BEST BUY CO., INC., SAMSUNG ELECTRONICS AMERICA, INC., WESTINGHOUSE DIGITAL ELECTRONICS, LLC, JVC AMERICAS CORPORATION, WESTERN DIGITAL TECHNOLOGIES, INC., ROBERT BOSCH LLC, PHOEBE MICRO, INC., HUMAX USA INC., COMTREND CORPORATION, DOBBS-STANFORD CORPORATION, VERSA TECHNOLOGY INC., ZYXEL COMMUNICATIONS INC., ASTAK INC., and GCI TECHNOLOGIES CORPORATION,	**COMPLAINT**
Defendants.	

This is an action by the Software Freedom Conservancy, Inc., a 501(c)(3) not-for-profit charitable corporation organized under the laws of the State of New York, and Erik Andersen, an

Free assignment (2008 - France)

ASSIGNATION DEVANT LE TRIBUNAL DE GRANDE INSTANCE DE PARIS

L'AN DEUX MILLE HUIT et le

A LA DEMANDE DE :

1) **Monsieur HARALD WELTE**
Né le 11 février 1979, de nationalité allemande,
Demeurant 11 Glanzstrasse 12437 Berlin, Allemagne
Exerçant la profession de développeur de logiciels

2) **Monsieur Erik ANDERSEN**
Né le 4 août 1971, de nationalité américaine,
Demeurant 352 N. 525 East, Springville, UT 84663, USA
Exerçant la profession de développeur de logiciels

3) **Monsieur Rob LANDLEY**
Né le 1er février 1972, de nationalité américaine,
Demeurant 2413 Leon St #106, Austin, TX 78705, USA
Exerçant la profession de développeur de logiciels

Ayant pour avocat : **Monsieur Olivier HUGOT**
Avocat à la Cour
Association HUGOTAVOCATS
Demeurant 22, rue Saint Augustin – 75002 PARIS
Tel. : 01.44.94.83.83 Fax : 01.44.94.83.84
Toque C 2501

Elisant domicile en son cabinet

J'AI, HUISSIER DE JUSTICE SOUSSIGNE,

L'HONNEUR D'INFORMER :

FREE
Société par Actions Simplifiée, au capital de 3.036.830 € immatriculée au Registre du Commerce et des Sociétés de Paris sous le numéro B 421 938 861, dont le siège social est sis 8 rue de la Ville l'Evèque 75008 Paris, prise en la personne de son Président, Monsieur Cyril POIDATZ.

Qu'un procès lui est intenté, pour les raisons ci-après exposées, devant le Tribunal de Grande Instance de Paris sis 4 boulevard du Palais, 75001 Paris.

Tools

- Yocto and Buildroot (Linux “build systems”) include licensing features
  - Get the list of component licenses + `license.manifest` file
  - Release source code with the “archiver” class

    ```
    INHERIT += "archiver"
    ```

  - Accept “commercial” licences (using `LICENSE_FLAGS`)

    ```
    LICENSE_FLAGS_WHITELIST = "commercial"
    ```

  - Avoid using a specific license

    ```
    INCOMPATIBLE_LICENSE = "GPL-3.0 LGPL-3.0 AGPL-3.0"
    ```

- Some tools for checking licensing issues
  - Black Duck (commercial)
  - WhiteSource (commercial)
  - FOSSology (free project)

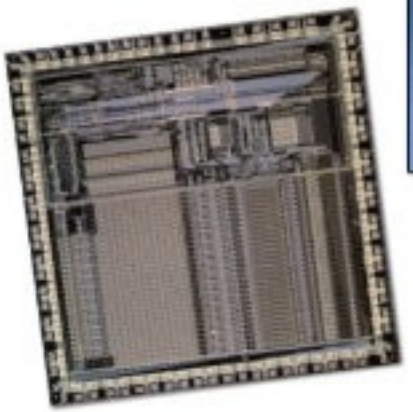
Hardware

ARM, the story

- ARM (*Acorn Risc Machine*) created in 1983 for BBC computer “Archimedes”
- Nice features, expensive, commercial failure
- ARM becomes *Advanced* Risc Machine and provides CPU design (VHDL, no HW)
- Architecture version is ARMvX
  - ARMv1 → ARM1
  - ARMv2 → ARM2
  - ARMv5 → ARM7EJ, ARM9E, ARM10E
  - ARMv6 → ARM11 (Pi 1 A+/0/B/B+)
  - ARMv7 → Cortex-A7/8/9 (32 bits, Pi 2 / Pi 3)
  - ARMv8 → Cortex-A53 (64 bits, Pi 3), -A72 (Pi 4)

Acorn Archimedes

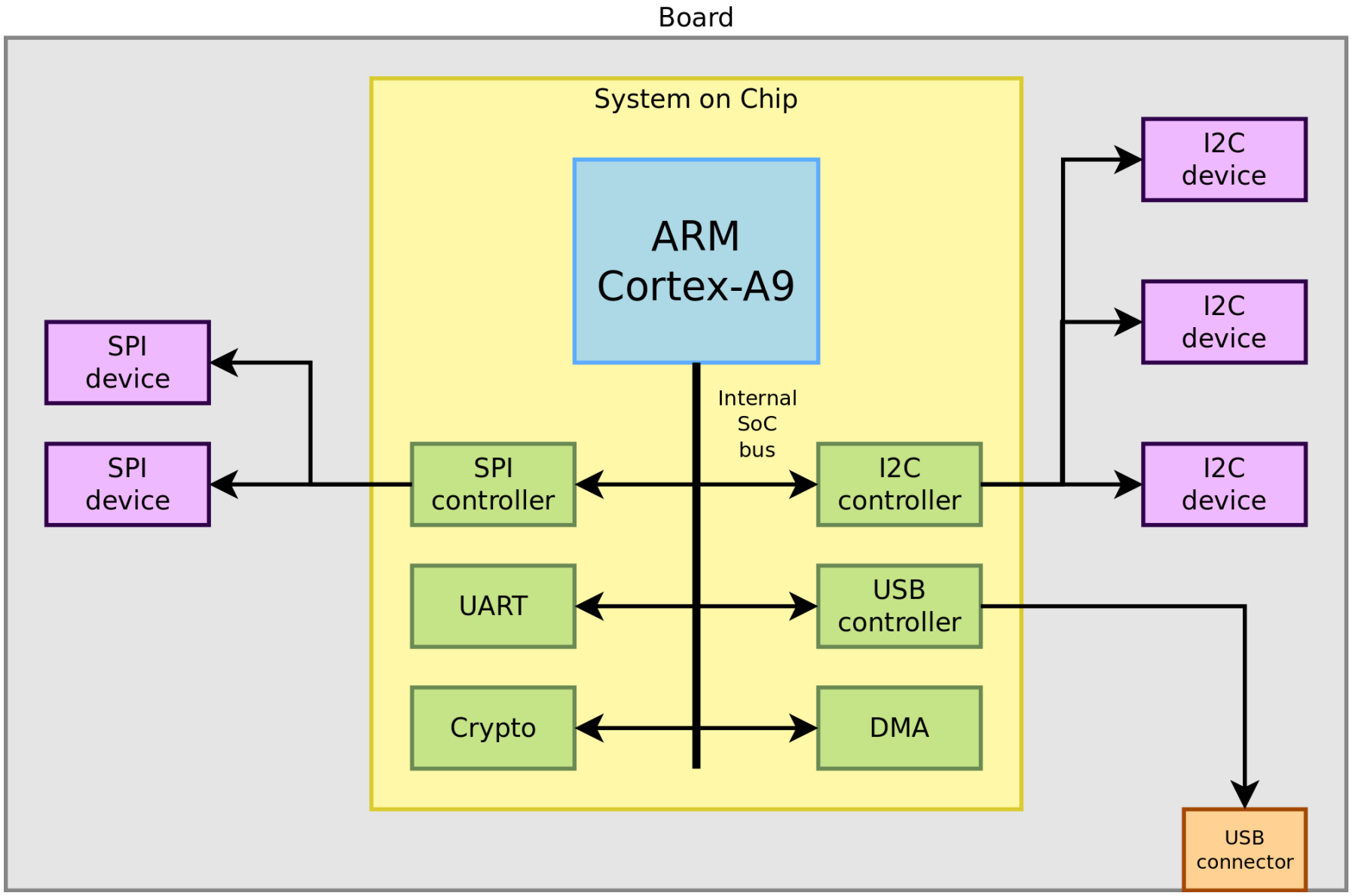
ARM



ACORN

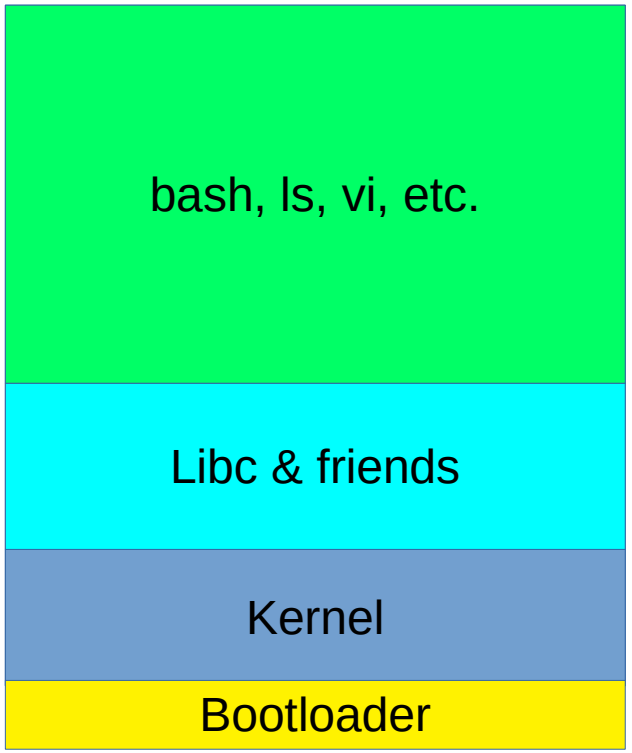
ARM SoC / SoM (Toradex)

ARM SoC / SoM

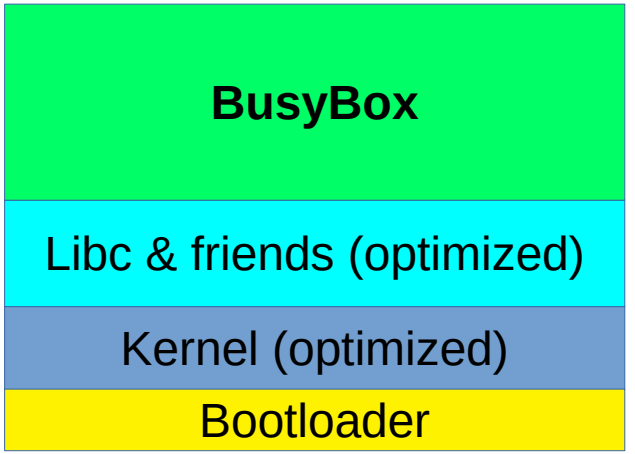

Board
System on Chip
ARM
Cortex-A9
Internal
SoC
bus
SPI
device
SPI
device
SPI
controller
UART
Crypto
I2C
controller
USB
controller
DMA
I2C
device
I2C
device
I2C
device
USB
connector

GNU/Linux architecture

GNU/Linux or Embedded Linux ?



GNU/Linux



Embedded Linux

+ = root-fs

Starting the Linux OS in 4 steps

- Starting the bootloader which loads the (static) Linux kernel
  - `vmlinuz` (for a standard x86 distributions)
  - `zImage/Image/bzImage` (for a compiled kernel)
  - A specific name such as `kernel[7/8].img` (Raspberry Pi OS)
- The kernel initializes the hardware devices
- The kernel mounts the root-filesystem and runs the “init” process
- Finally, the user space services are started by “systemd” or “SysvInit”

The bootloader

- On x86, the BIOS starts the GRUB bootloader
- For other architectures (such as ARM), the most popular bootloaders are:
  - U-Boot
  - Barebox
- In some cases, the bootloader is loaded by a dedicated firmware (SPL = Secondary Program Loader)

The “init” process

- The "father" of the system processes (PID = 1)
- Use “SysvInit” or “systemd”
- SysvInit is based on simple scripts (original UNIX approach)
- SysvInit is replaced by “systemd” for recent distributions (lots of services to start and manage)

```
$ ls -l /sbin/init
lrwxrwxrwx 1 root root 20 avril 28 15:03 /sbin/init ->
/lib/systemd/systemd
```

- “systemd” is available for Embedded Linux !

Filesystem Hierarchy Standard (FHS)

`/bin`	User commands
`/sbin`	System commands (“root” only)
`/lib`	Libraries and kernel modules
`/etc`	Editable Text File (configuration files)
`/dev`	Device nodes (for driver access)
`/var`	System logs
`/usr`	Same as “/” for bin, sbin, lib
`/home`	Home directories for users
`/opt`	For external (binary) programs
`/boot`	Linux kernel, device tree and additional files
`/proc`	Processes status and more (kernel pseudo filesystem)
`/sys`	Hardware bus status (PCI, USB) and more (kernel pseudo filesystem)

Systemd

From SysvInit to systemd

- SysvInit was the standard way to start a UNIX system
- SysvInit is based on `/etc/rc?.d` directory (? = S or 0 to 6)
- S / 0 to 6 is called the “runlevel”
  - A `Sxx` scripts starts a service (xx = 0 to 99)
  - A `Kxx` scripts stops it
- `Sxx`/`Kxx` are symlinks to `/etc/init.d/<service>`
- Start/stop a service with `service <name> stop/start`
- Pros
  - A standard since SVR4 (portability)
  - Simple approach (shell based)
- Cons
  - Can’t handle large number of services
  - Very long boot procedure (sequential approach)!

Systemd introduction

- Ubuntu released a replacement for SysvInit in 2006 (Upstart)
- Upstart was not that fair (issues and limitations)
- Systemd is a set of utilities to start a Linux system (not UNIX !)
  - Startup and shutdown (with parallelization capabilities !)
  - Services
- Developed by Red Hat
- The `systemctl` command is the most used
- Systemd is viewed by many people as being excessively complex
- Complexity is seen by some people as being against UNIX philosophy

SysvInit to systemd migration

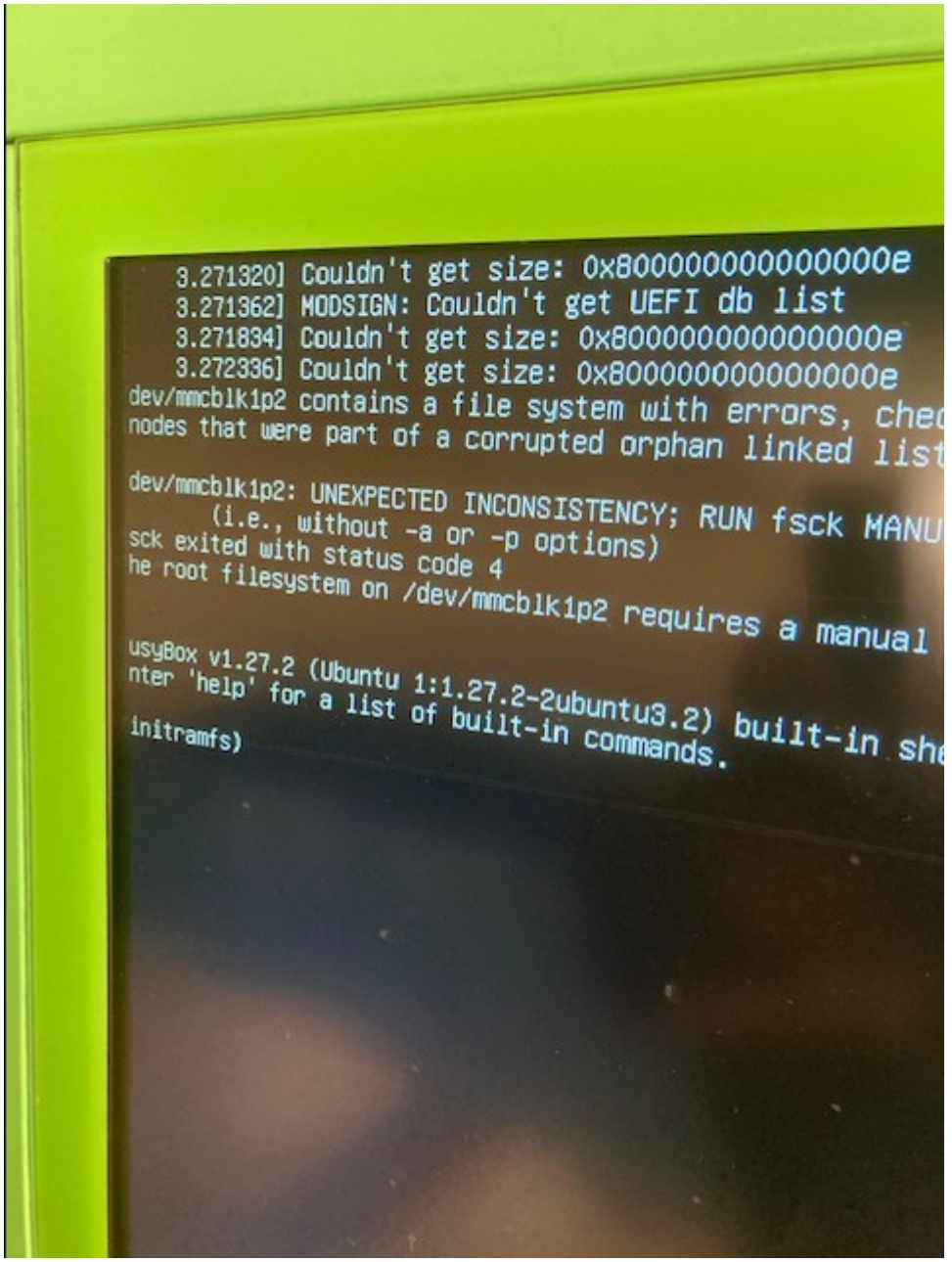
- Some services (or packages) are still not ported to systemd
- Several distributions (including Ubuntu 22) use both systemd and SysvInit scripts !!

Building an Embedded Linux distribution

Using a “standard” distribution ?

- Debian, Ubuntu, etc. are usable for a POC
  - Package management
  - Handling security issues
  - Native compilation
  - Supported by HW providers (Adafruit, NVIDIA, etc.)
- Real embedded development should use a build system such as Yocto !

Using Ubuntu for embedded ?



How to build an EL distribution

- Using a “build system” (Buildroot, Yocto, OpenWrt)
- Commercial tools (Wind River, MontaVista, etc.) → €€€
- Building the distribution “from scratch”
  - Close to LFS (Linux From Scratch)
  - Main issues = cross compilation (?), dependencies, update, etc.

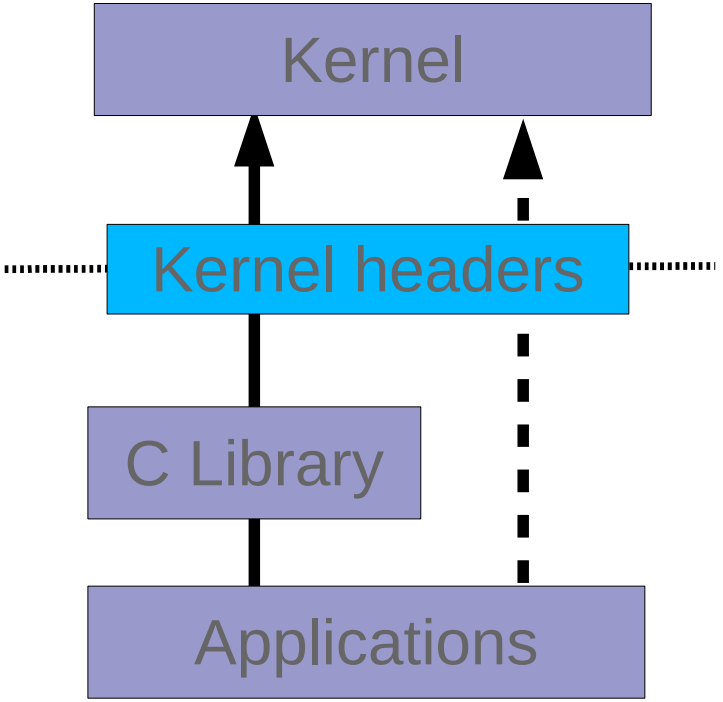
→ everything is provided by the BSP (Board Support Package from the HW maker)

Cooking Embedded Linux like a “chef”

- Cross toolchain (x86 → ARM)
- Bootloader (U-Boot, Barebox)
- Linux kernel
- root-filesystem (based on BusyBox ?), including system files and libraries (Glibc and friends)

Cross toolchain

- The cross toolchain includes :
  - GCC (or LLVM ?)
  - Binutils (`as`, `ld`, `readelf`, etc.)
  - Libc (Glibc, uClibc, musl, etc.)
- It depends on several components
  - Kernel (system calls)
  - Libc
  - Host PC environment
- Finally should include
  - GDB
  - Adapted/ported libraries (Qt, etc.)
- Build a toolchain (Yocto, BR) or get a binary one !



Cross toolchain (kernel dependencies)

- Lots of definitions in kernel sources (NOT libc)
  - System calls definition in `unistd*.h`

```
#define __NR_exit 1
#define __NR_fork 2
```

  - Constants in `fnctl.h`

```
#define O_RDONLY        00000000
#define O_WRONLY        00000001
#define O_RDWR          00000002
```

  - Structures (`struct stat` in `stat.h`)
- The cross toolchain is created during the first build for Yocto and BR

Binary toolchain

- Main providers
  - Linaro (***)
  - Arm (***)
  - SIEMENS (Sourcery CodeBench)
  - Distribution package (Ubuntu)
- Pros
  - Very easy and fast to install
  - Some free support by community
  - Source code available
- Cons
  - Fixed configuration (libc, kernel version)

Building a toolchain

- Build systems (today's choice !)
  - Yocto
  - Buildroot
- Dedicated tools (deprecated ?)
  - crosstool-NG → Yann Morin (Orange labs)

Cross toolchain ABI

- ARM used ABI (Application Binary Interface)
  - EABI → Embedded ABI
  - EABIHF → hard floating point (FPU)
- Cross toolchain generates EABI or EABIHF
- The command name is :

  `<arch>-[<vendor>]-[<os->][<abi>]-<command>`

`arm-none-linux-gnueabi-gcc` → Linux/EABI

`arm-linux-gnueabihf-gcc` → Linux/EABIHF (Linaro / Arm)

`arm-poky-linux-gnueabi-gcc` → Linux/Yocto

`aarch64-linux-gnu-gcc` → Linux/arm64 (Linaro / Arm)

`arm-none-eabi-gcc` → arm “bare metal”

Using a binary toolchain (Linaro ?)

- Installing (extracting archive)

```
$ tar xf gcc-linaro-<version>-x86_64_arm-linux-gnueabihf.tar.xz
```

- Set environment variables (use a script !)

```
$ export PATH=$PATH:<install-dir>/bin
$ export ARCH=arm
$ export CROSS_COMPILE=arm-linux-gnueabihf-
```

Needed for (kernel) cross compilation

- Testing the toolchain

```
$ source ~/bin/set_env_linaro-<version>.sh
$ arm-linux-gnueabihf-gcc -v
$ arm-linux-gnueabihf-gcc -o hello hello.c
```

- Same procedure for all toolchains (including Yocto) !

Cross-compiling + installing the kernel

- Get the source code from the HW provider (GitHub ?)
- Set `ARCH` and `CROSS_COMPILE`
- Load the configuration (aka “defconfig”)

```
$ cd <kernel-src-dir>
$ make bcm2835_defconfig # config file for Pi 3
```

- Compile the kernel (static + modules + device tree)

```
$ make -j <N>
```

- Static kernel image is in `arch/<arch>/boot`
- Install static kernel + modules (?)

```
$ make install INSTALL_PATH=<board-boot-path>
$ make modules_install INSTALL_MOD_PATH=<board-rootfs-path>
```

N = number of cores → use `nproc`

BusyBox

- Aka the “swiss army knife” for Embedded Linux
- Created by Bruce Perens en 1996 as a Debian tool (Linux distribution on a floppy disk !)
- Used for Linux Router Project in 1999
- Maintained by Erik Andersen then Denys Vlasenko (2006)
- ONE executable (`busybox`) and several symlinks (`sh`, `ls`, `cp`, etc.)
- More than 300 commands in less than 1 Mb

```
$ ls -l sh cp busybox
-rwsr-xr-x 1 root root 966016 23 janv. 14:08 busybox
lrwxrwxrwx 1 root root      7 23 janv. 14:13 cp -> busybox
lrwxrwxrwx 1 root root      7 23 janv. 14:13 sh -> busybox
...
```

Why BusyBox ?

- GNU/Linux based on “coreutils” and “bash” pkg (big !)

```
$ ls -l /bin/bash /bin/ls
-rwxr-xr-x 1 root root 873516 22 juin   2011 /bin/bash
-rwxr-xr-x 1 root root 118808  3 nov.   2010 /bin/ls
```

- BusyBox is a replacement for most of Linux commands

```
$ ls -l busybox
-rwxrwxr-x 1 pierre pierre 966016 déc.  25 22:08 busybox
```

- 95 % of EL distributions are (were?) based on BusyBox
- Simple and portable
- GPL v2 licensing
- Toybox is a replacement used by Android (BSD license)
- Same compilation procedure as kernel (defconfig) !

Compiling + installing BusyBox

- Extract the source code (get it from the BusyBox project ?)
- Set `CROSS_COMPILE`
- Load the configuration (defconfig)

```
$ cd <busybox-src-dir>
$ make defconfig
```

- Compile + install BusyBox

```
$ make -j <N>
$ make CONFIG_PREFIX=<board-rootfs-path> install
```

Introducing build systems

What is a “build system” ?

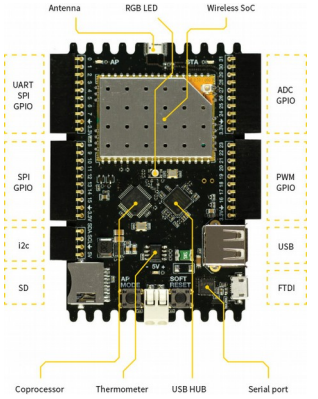
- A build system creates an Linux image (distribution) from the sources
- A “cross compilation framework”
- NOT a distribution, BUT a tool to create one
- Does not provide sources but “recipes” (how to build packages from source code)
- Creates cross toolchain (most of the time)
- Creates image components
  - bootloader (u-boot.bin)
  - Linux kernel (bzImage, zImage, Image) + `*.ko` + DTB
  - Root-filesystem images (tar, ubifs, jffs2, ext2/3/4, etc.)

Benefits for the industry

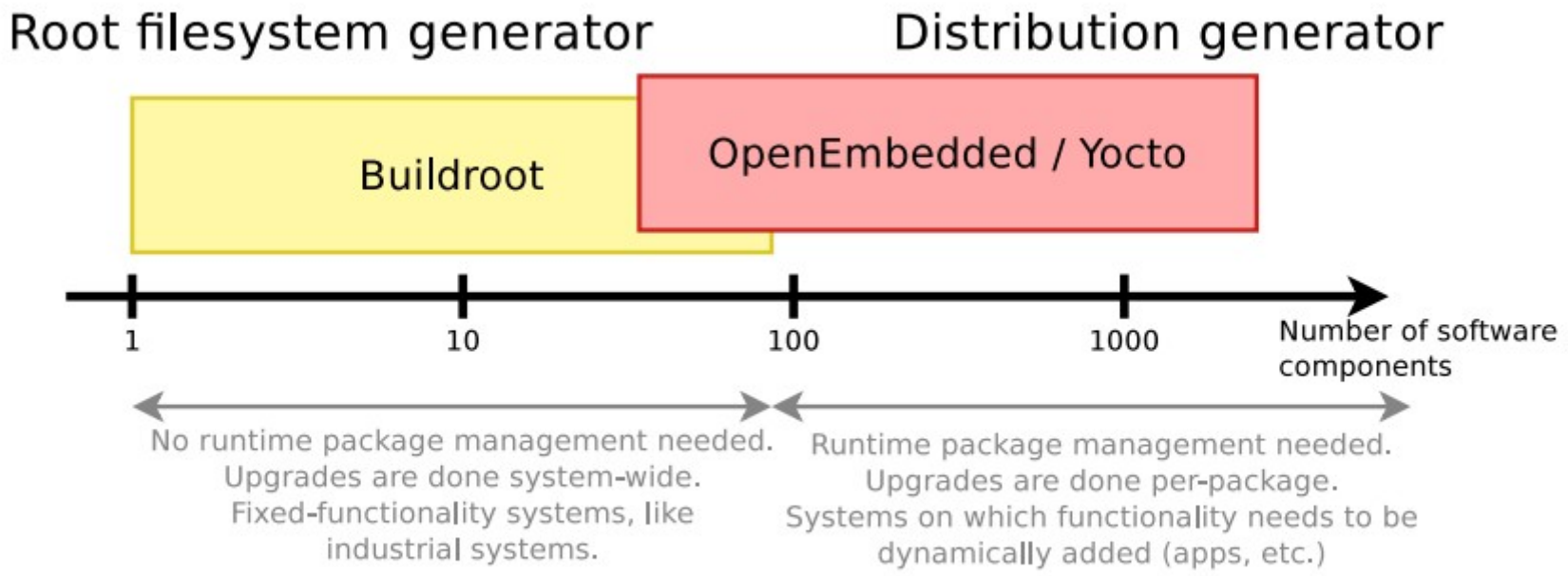
- Several tasks for an industrial project
  - Creating a BSP
  - Creating a SDK (for developers)
  - Developing your application(s) (your job !)
  - System integration and maintenance
- A build system is not a development tool but it creates one (SDK)
- A build system is an integration tool, software should be “smart enough” to be integrated
- A build system helps you for tasks “outside your job” :-)

Main build systems

- Yocto/OpenEmbedded
  - Written in Python (BitBake)
  - Very powerful but needs training
  - Mostly text mode (poor GUI = Toaster)
- Buildroot
  - Based on standard GNU-Make
  - Was a tool for uClibc developers
  - Independent project since 2009
  - GUI for configuration but no packages
- OpenWrt
  - Close to Buildroot
  - Handle binary packages



Buildroot or Yocto ?



(c) Thomas Petazzoni

Bibliography

- Free licenses expertise company http://inno3.fr
- http://systematic-paris-region.org/fr/livrets-bleus/fondamentaux-juridiques
- https://www.apache.org/licenses/LICENSE-2.0
- https://www.gnu.org/licenses/gpl-3.0.html
- https://tldrlegal.com/license/gnu-lesser-general-public-license-v3-(lgpl-3)
- Linux Torvalds about GPLv3 https://lkml.org/lkml/2006/9/25/161
- Qt and LGPLv3 https://youtu.be/bwTlCBbB3RY ***
- Licenses in Qt https://doc.qt.io/qt-5/licenses-used-in-qt.html
- Linus Torvalds Aalto conference https://www.youtube.com/watch?v=MShbP3OpASA ***
- https://developer.nvidia.com/blog/nvidia-releases-open-source-gpu-kernel-modules
- https://www.phoronix.com/news/Linux-6.6-Illicit-NVIDIA-Change