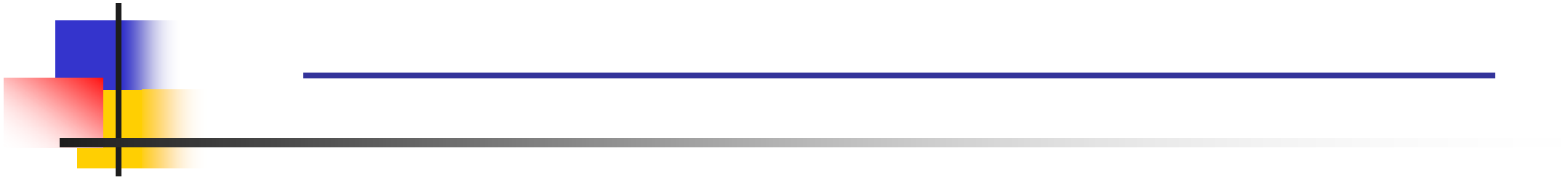
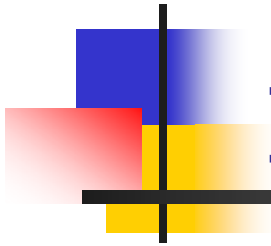
Sistemas Operativos para Sistemas Empotrados



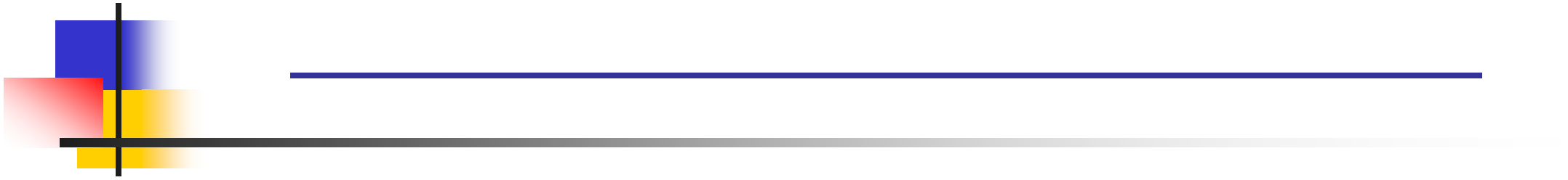
Francisco Rosales frosal@fi.upm.es



Índice de contenidos

- Visión general.
- Definición y características.
- Aspectos hardware básicos.
- Aspectos software básicos.
- ¿Cómo arranca un sistema?
- Arquitecturas de software empotrado.
- ¿Por qué elegir Linux?
- Construcción de un sistema empotrado.
- Entorno de desarrollo cruzado.
- *Kernel* de Linux, compilación.
- Sistema de ficheros raíz, generación.
- Arranque del sistema, configuración.
- Bibliografía.
- Proyecto práctico.
- Práctica ligera.

Visión general



- Análisis. ¿Cómo se comporta?
- Síntesis. ¿Cómo se construye?
- Diseño. ¿Cómo se concibe?

Sistema Empotrado - visión general

- **Sistema computador diseñado para un propósito.**

Computer system

A **computer** is a programmable machine that receives input, stores and manipulates data, and provides output in a useful format.

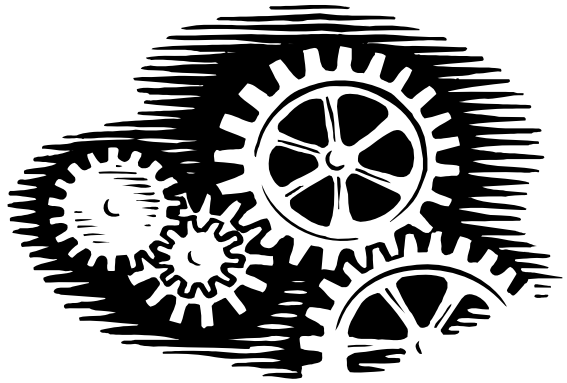
- Visión integrada, HW y SW como producto o herramienta.

- Vamos a estudiarlos desde tres puntos de vista:
  - Su comportamiento.
  - Su construcción.
  - Su concepción.

¿Cómo se comporta?

Durante su fase de producción.

- Por dentro y por fuera.
- Composición y funcionamiento interno.
- Comportamiento y requisitos externos.
- Labor de análisis.



Análisis

Un **análisis** es la descomposición de un todo en partes para poder estudiar su estructura, sistema operativo o funciones.

¿Cómo se construye?

Durante su fase de desarrollo.

- De abajo a arriba.
- Síntesis basada en componentes.
- Labor de síntesis.

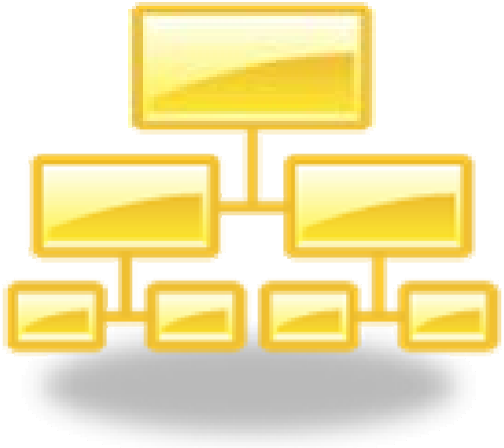
Síntesis

La **síntesis** es la "composición de un cuerpo o de un conjunto a partir de sus elementos separados en un previo proceso de análisis".

¿Cómo se concibe?

Durante su fase de diseño.

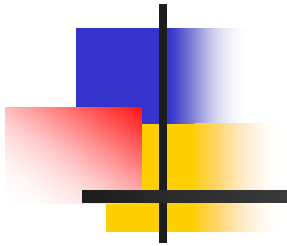
- De arriba a abajo.
- Diseño basado en restricciones.
- Labor de diseño.



Diseño

En ingeniería, se define como el proceso previo de configuración mental, "pre-figuración", en la búsqueda de una solución.

Definición y características



Sistema Empotrado - definición

Embedded system

An embedded system is a computer system designed to perform one or a few dedicated functions often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. By contrast, a general-purpose computer, such as a personal computer (PC), is designed to be flexible and to meet a wide range of end-user needs. Embedded systems control many devices in common use today.

...

Sistema Empotrado - definición

Embedded system

...

Embedded systems are controlled by one or more main processing cores that are typically either microcontrollers or digital signal processors (DSP). The key characteristic, however, is being dedicated to handle a particular task, which may require very powerful processors. For example, air traffic control systems may usefully be viewed as embedded, even though they involve mainframe computers and dedicated regional and national networks between airports and radar sites (each radar probably includes one or more embedded systems of its own).

Sistema Empotrado - definición

- sistema computador
- diseñado
  - tarea específica
    - NO propósito general
- restricciones
  - tiempo-real
- más restricciones (añadido)
  - consumo
  - ambiente de uso (vehículo)
  - específicas (medicina)
  - "no atendido"
    - *watch dog timer*
    - actualizaciones remotas
- dispositivo complejo
  - parte empotrada
  - partes mecánicas
  - control
- núcleos procesadores
  - microcontroladores, DSPs
  - muy potentes
- sistemas de varios tamaños
- grandes sistemas
  - interconexión
  - descomposición en subsistemas

Sistema Empotrado - características

Propósito específico. Interacción con dispositivos. Firmware.

Embedded system # Characteristics

...

3.1 User interface

3.2 Processors in embedded systems

- 3.2.1 Ready made computer boards
- 3.2.2 ASIC and FPGA solutions

3.3 Peripherals

3.4 Tools

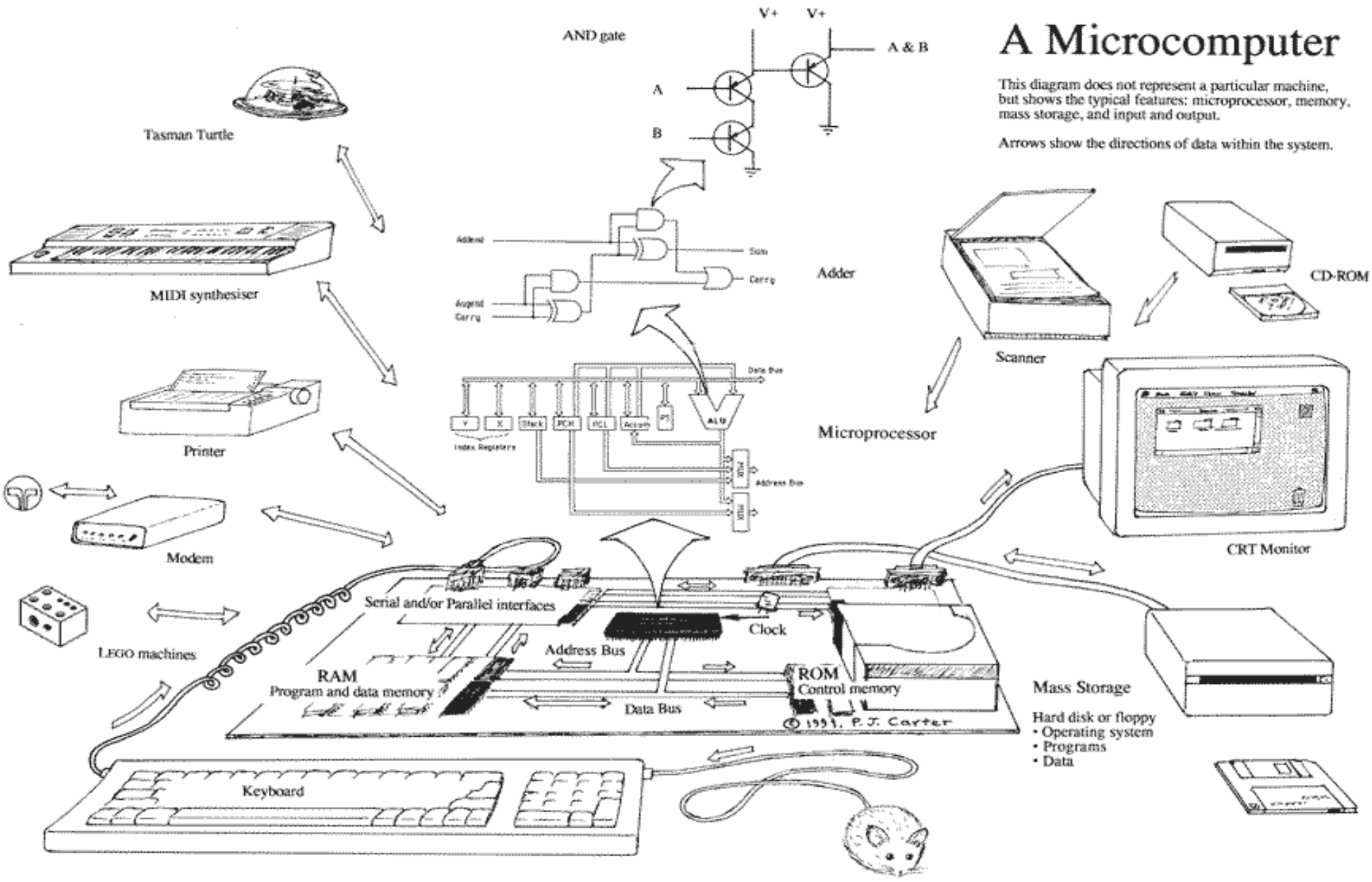
3.5 Debugging

3.6 Reliability

3.7 High vs. low volume

Aspectos hardware básicos

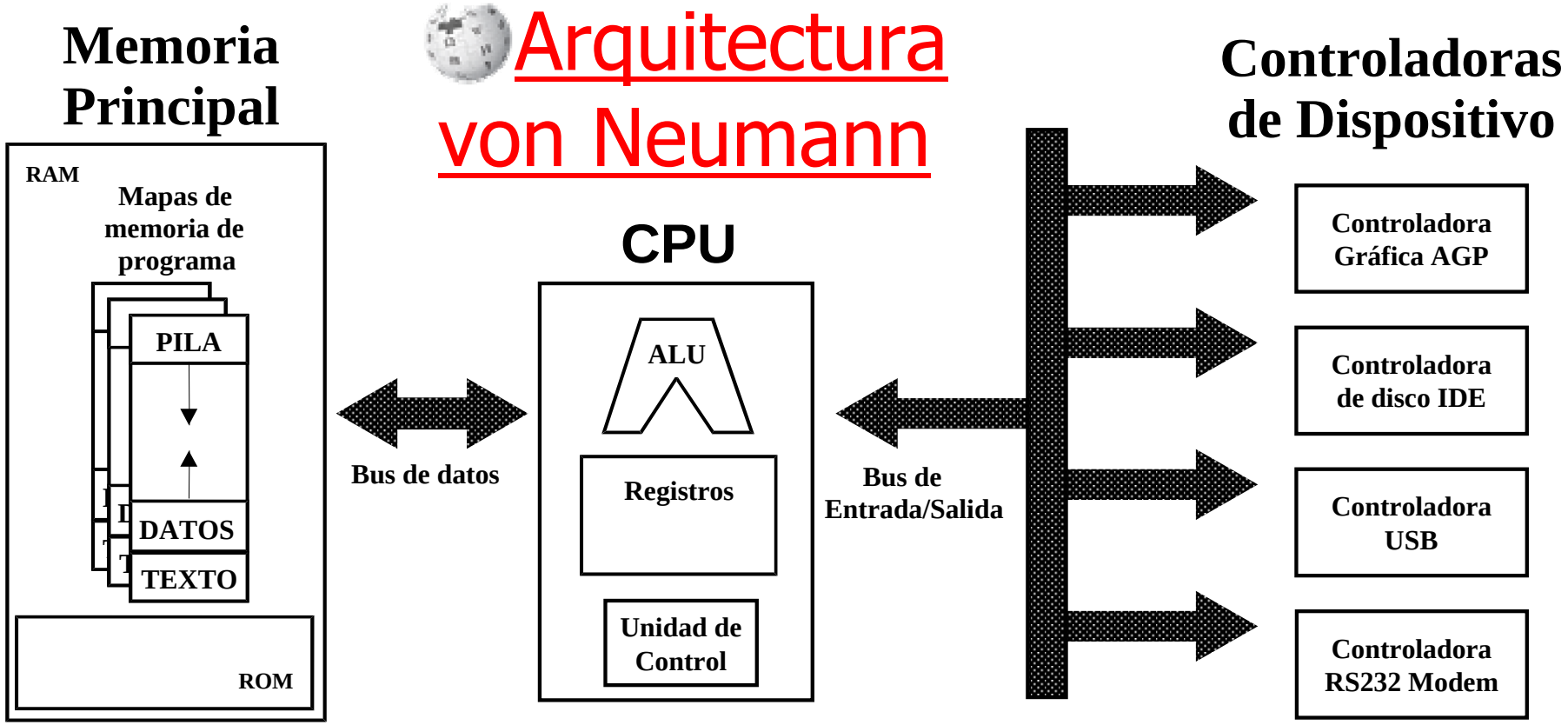
- El computador personal.
- Computador, partes hardware.
- Entender el arranque y funcionamiento del sistema.
- Catálogo de hardware.
- El procesador.
- Memory hierarchy.
- Flash memory.



http://www.saauc.org.au/members/images/diag.gif

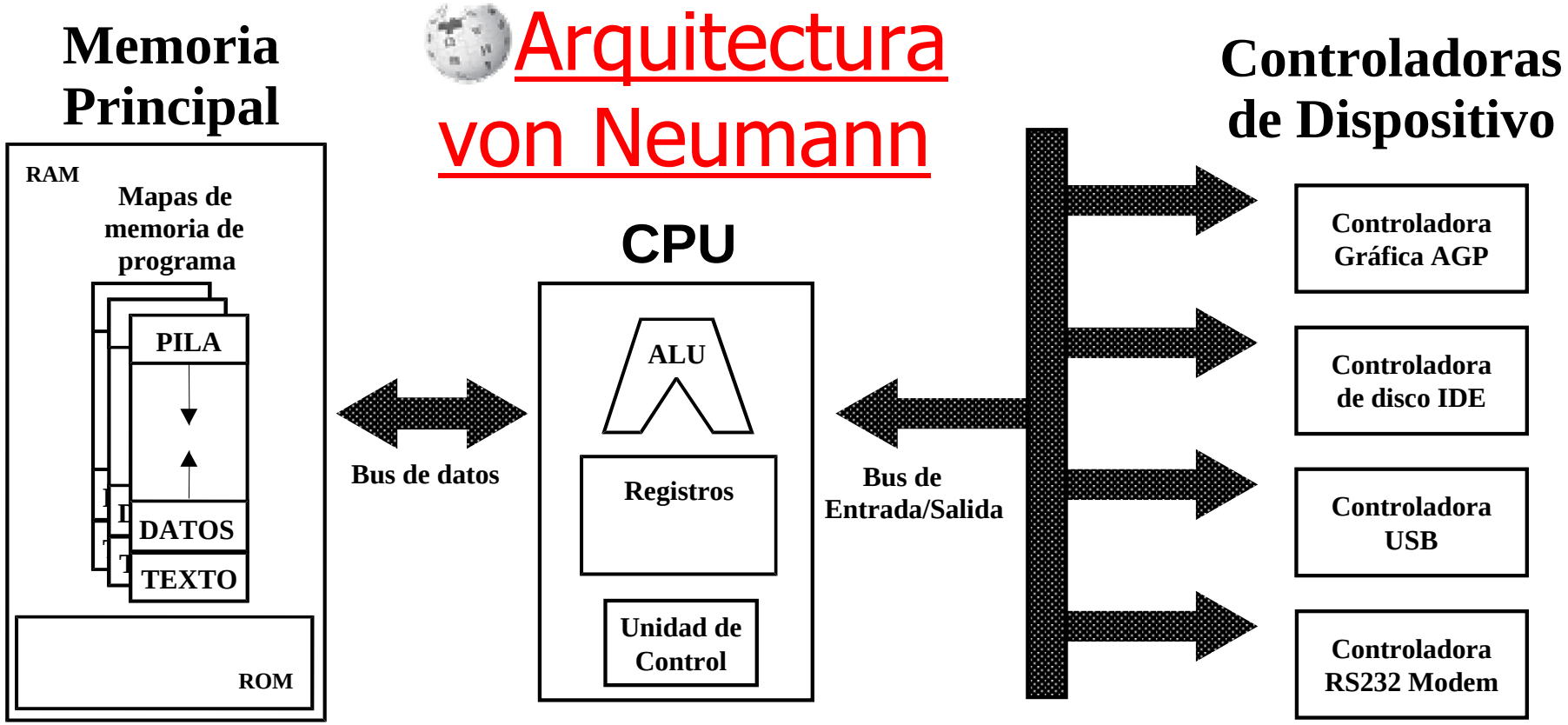
Computador, partes hardware @[DEH]

- El procesador ejecuta los programas.
- La memoria (de diferentes tipos) almacena programas y datos.
- Los dispositivos almacenan datos o los intercambian con el exterior.
- Los buses conectan una cosa con otra.
- Los periféricos son dispositivos "exteriores".



Entender el arranque y funcionamiento del sistema

- Ciclo de ejecución de la CPU.
- "Modo" de CPU (kernel/usuario).
  - Interrupción, excepción o *trap*
  - Interrupción HW y SW.
- RST: arranque ROM / *bootloader*
- Manejador, controlador, dispositivo
  - Puerto de E/S.
  - Modos de operación de la E/S.



Catálogo de hardware

Partes o componentes basados en estándares.


Ready made computer boards.

- Criterios de selección:
  - 
    Form factor
  - El Procesador
  - La Memoria
  - Dispositivos integrados
  - Posibilidades ampliación
  - Conectividad con el exterior
  - Precios

El procesador

- Tipos de procesadores:
  - PIC, *Programmable Interface Controller.*
  - µC, *Microcontroller.*
  - DSP, *Digital Signal Processor.*
  - uP, *Microprocessor.*
  - SoC, *System on Chip.*

  - ASIC, *Application-specific integrated circuit.*
  - PLD, *Programmable Logic Devices* : PAL, GAL, CPLD, FPGA

- Arquitectura:
  - La "palabra": (4,) 8, 16, 32 y 64-bits.
  - Von Neumann / Harvard.
  - Little / middle / big endianness.
  - Tipo y juego de instrucciones.
  - Unidades funcionales.

- Otras facilidades:
  - Compatibilidad
  - SW disponible
  - Depuración
  - Desarrollo

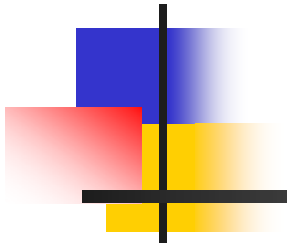
Memory hierarchy

- Interna
  - Registros
  - Caches (transparentes)
- Principal (direccionable desde la CPU)
  - RAM, *Random-access memory*, variedad.
  - ROM, *Read-only memory*, variedad.
    - Arranque: BIOS / bootloader.
- Secundaria (E/S direccionable a "bloque")
  - Discos y particiones, variedad.
  - *Filesystem*, "sistema de archivos" o formato, variedad, adaptados a diferentes necesidades y dispositivos.
- ...

Flash memory

- Muy utilizada en sistemas empotrados y móviles.
- EEPROM, tecnologías NOR y NAND.
  - *Block erasure, (16 to 512 KB) bits to 1.*
  - *Memory wear, till 10^6 P/E cycles.*
- NOR:
  - *random-access* ROM
  - XIP, *execute in place memory*
  - Uso: ROM: BIOS / bootloader
- NAND:
  - Lectura / programación de páginas (512 B a 4 KB).
  - Presentación: tal cual, DOC, USB disk
  - Uso: Disco (extraíble)
    - *Flash file system*

Aspectos software básicos

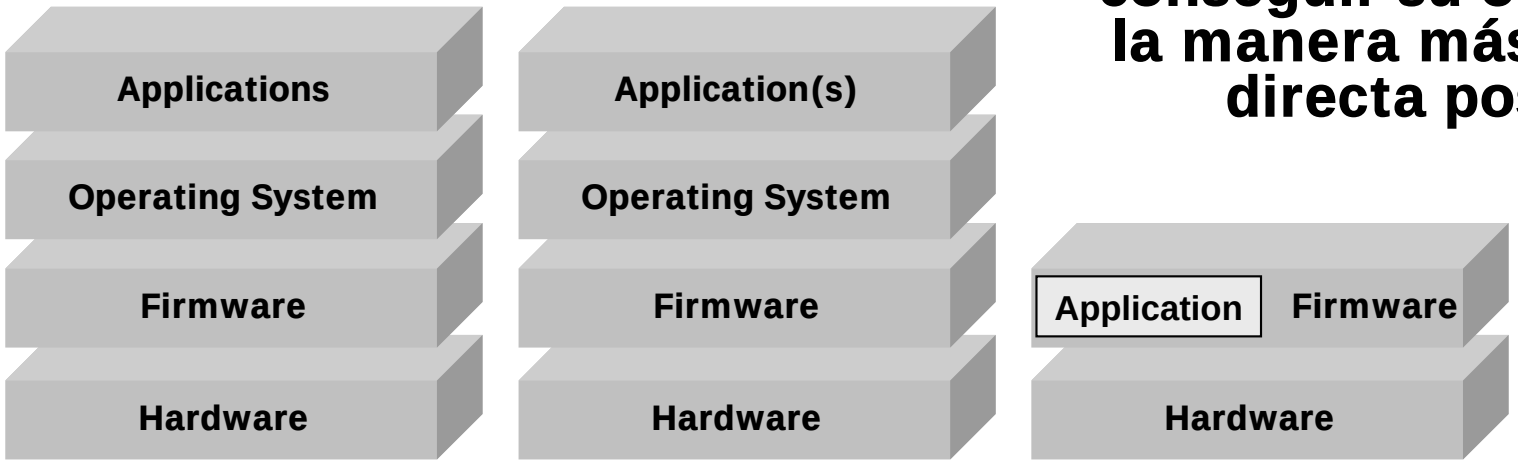


- Computador, capas software.

Computador, capas software @[DEH]

- Bajo el nivel de aplicación todo es software de sistema.
- El Sistemas Operativo:
  - Gestiona los recursos hardware adecuadamente.
  - Ofrece un conjunto de servicios que simplifican y facilitan a las aplicaciones el uso del hardware.
- Algunos sistemas empotrados sencillos prescinden de Sistema Operativo.

"Un sistema empotrado debe ser diseñado para conseguir su objetivo de la manera más simple y directa posible"

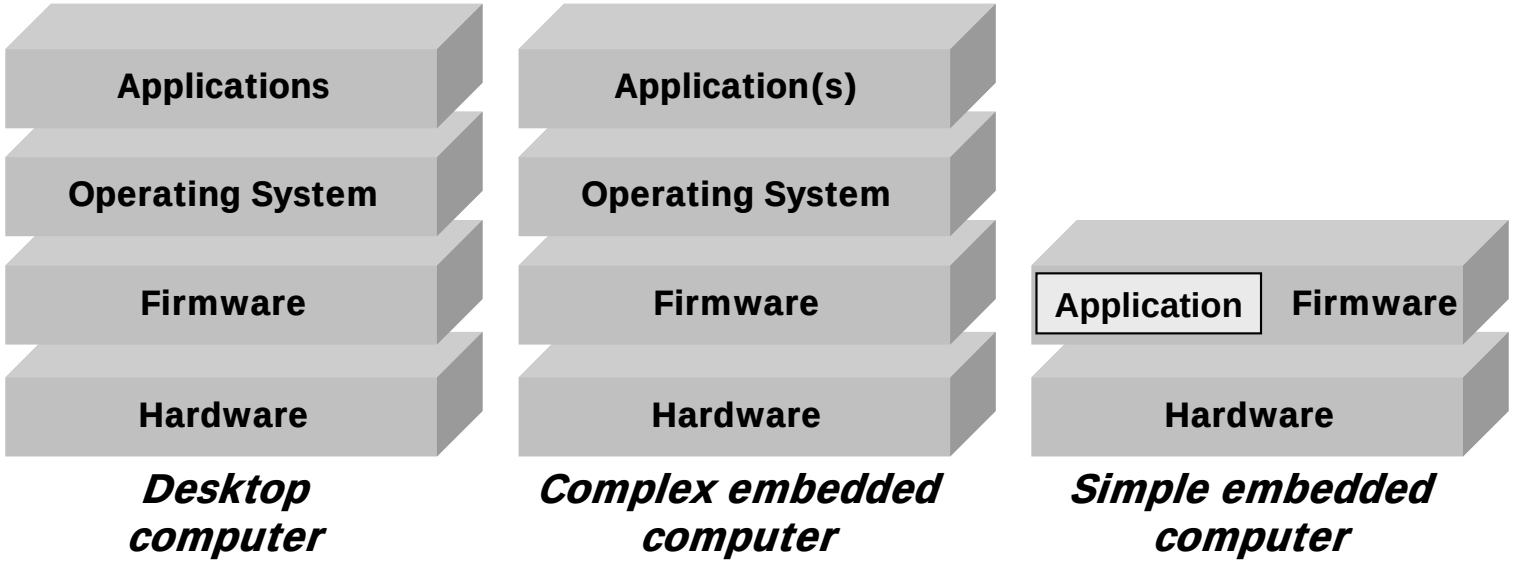


Desktop computer | *Complex embedded computer* | *Simple embedded computer*

http://my.safaribooksonline.com/0-596-00362-5/5

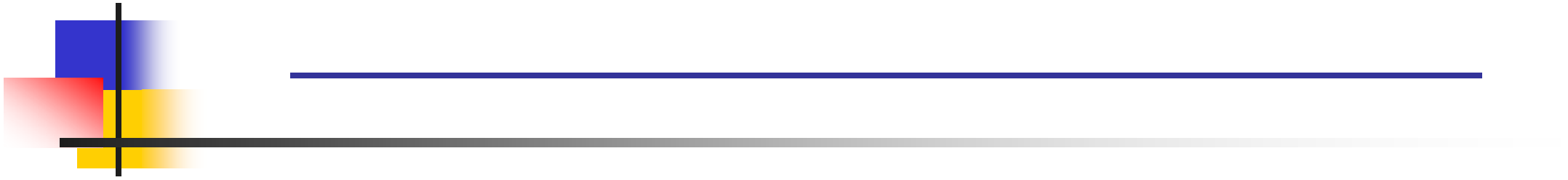
Computador, capas software @[DEH]

- El software controla la actividad del computador.
- Múltiples capas de software, interactuando sólo con las aledañas.
- El "firmware" ejecuta cuando el computador arranca.
  - Está almacenado en la ROM (o flash).
  - Inicializa el resto del hardware a un estado conocido.
  - Configura el computador para operar correctamente.
  - Contiene el *bootloader* que carga el **software de sistema** desde disco.



http://my.safaribooksonline.com/0-596-00362-5/5

¿Cómo arranca un sistema?



- *PC BIOS Boot Sequence*
- *Booting the operating system*
- *Linux startup process*

PC BIOS Boot Sequence

- A "cold boot" (the machine was cold when started).

1. Power Good signal from power supply.
2. Location FFFF0h contains a "jump" instruction to the real startup program.
3. Power-on self test (POST).
4. Executes the video card BIOS at location C000h.
5. Look for other devices' ROMs to see if any of them have BIOSes.
6. Startup screen.
7. More tests on the system.
8. A "system inventory" of what hardware is in the system.
9. Detect and configure Plug and Play devices.
10. Display a summary screen about your system's configuration.
11. Search boot sequence BIOS setting for a drive to boot from.
12. Looks for a master boot record at cylinder 0, head 0, sector 1.
13. Starts the **process of booting the operating system**.
14. If no boot device can be found, display an error and freeze up.

- A "warm boot" is when rebooted using {Ctrl}+{Alt}+{Delete}. POST is skipped and continues at step 8.

http://www.pcguide.com/ref/mbsys/bios/bootSequence-c.html

Booting the operating system

- Common primary *bootloader*
  - BIOS, OpenBIOS, etc.
- Second-stage boot loader
  - GRUB , LILO, Syslinux, etc.
  - Carga el sistema operativo y le transfiere el control.

  - Sistemas empotrados tienen que arrancar sin dilación.
  - Se evita la carga ejecutando directamente desde ROM.

- Network booting

Linux startup process

Boot loader phase

Kernel phase

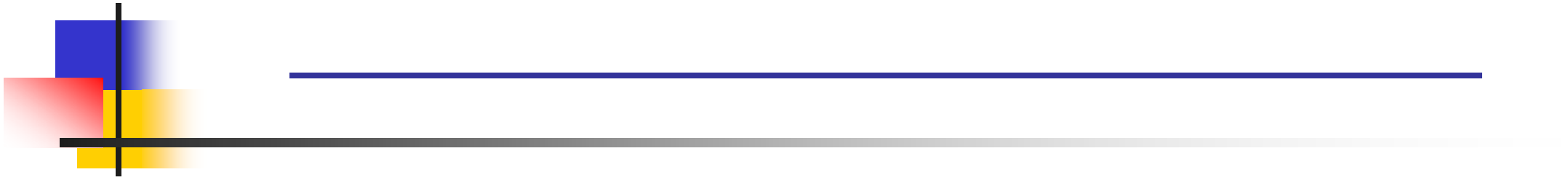
- Decompresses the kernel image.
- Initializes computer's memory, processors, I/O subsystems, and storage devices.
- Mounts the root filesystem read-only.
- First process (PID==1) executes `/sbin/init`.

Init process (SysV init style)

- Keeps everything up and running.
- Follows `/etc/inittab` instructions.
- The runlevel tells what `/etc/rc` scripts to run.

Systemd (alternativa más moderna)

Arquitecturas de software empotrado



- *Embedded software architectures*
- Sistemas Operativos Empotrados
- Real-Time Operating System checklist

Embedded software architectures

- *Simple control loop: loop + functions*
- *Interrupt controlled system: event handlers* + loop
- *Cooperative multitasking: tasks + idle routine (yield)*
- *Preemptive multitasking or multi-threading: RTOS*
- *Microkernels and exokernels.*
- *Monolithic kernels: e.g. Embedded Linux or Windows CE*
- *Exotic custom operating systems: add-hoc*
- *Additional software components:*
  upper-layer components: networking, storage, etc.

Sistemas Operativos Empotrados

Embedded operating system

An **embedded operating system** is an operating system for embedded computer systems. These operating systems are designed to be very compact and efficient, forsaking many functions that non-embedded computer operating systems provide, and which may not be used by the specialized applications they run. They are frequently also real-time operating systems.

- Diseñado compacto y eficiente, con lo imprescindible.

Sistemas Operativos Empotrados

List of real-time operating systems

...

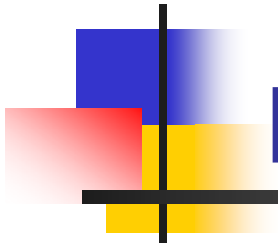
- Comparativa con . . . . ~183 RTOSes

List of operating systems # Embedded

...

9.1 Personal digital assistants (PDAs)
9.2 Digital media players
9.3 Robots
9.4 Smartphones
9.5 Routers
9.6 Other embedded

- En total. . . . . ~90
- *Embedded Linux* . . . . . ~20 distribuciones
- Windows CE . . . . . 5 versiones



Real-Time Operating System checklist

Muy buen artículo!, con el RTOS *checklist* y una comparativa de ~65 RTOSes según ~30 características!!

Original de 1999

- *Selecting a Real-Time Operating System*
  http://leadingedgeinnovations.biz/EmbeddedSystemProgramming/files/99/9903/f-hawley.pdf

Fusilado en 2002

- *Embedded Systems Design*
  http://books.google.com/books?id=sSoPLxAQrtkC&pg=PA33

Real-Time Operating System checklist

☑ **Language / Microprocessor Support** The first step in finding an OS for your project is to look at those vendors supporting the language and microprocessor you'll be using.
☑ **Tool Compatibility** Make sure your OS works with your ICE, compiler, assembler, linker, and source code debuggers.
☑ **Services** OSes provide a variety of services. Make sure your OS supports the services (queues, times, semaphores) you expect to use in your design.
☑ **Footprint** OSes are often scalable, including only those services you end up needing in your applications. Based on what services you'll need, and the number of tasks, semaphores, and everything else you expect to use, make sure your OS will work in the RAM and ROM you have.
☑ **Performance** Can your OS meet your performance requirements? Make sure you understand benchmarks vendors give you and how they apply to the hardware you will really be using.
☑ **Software Components** Are required components (protocol stacks, communications services, real-time databases, Web services, virtual machines, graphics libraries, and so on) available for your OS? How much effort will it be to integrate them?
☑ **Device Drivers** If you're using common hardware, are device drivers available for your OS?

http://leadingedgeinnovations.biz/EmbeddedSystemProgramming/files/99/9903/f-hawley.pdf

Real-Time Operating System checklist

☑ **Debugging Tools** OS vendors may have debugging tools that help find defects that are harder to find with source-level debuggers (such as deadlocks, forgotten semaphore puts, and so on).

☑ **Standards Compatibility** Are there safety or compatibility standards your application demands? Make sure your OS complies.

☑ **Technical Support** Phone support is typically covered for a limited time after your purchase or on a year-to-year basis through support. Sometimes applications engineers are available. Additionally, some vendors provide training and consulting.

☑ **Source vs. Object Code** With some OSes you get the source code to the operating system when you buy a license. In other cases, you get only object code or linkable libraries.

☑ **Licensing** Make sure you understand how the OS vendor licenses their OS. With some vendors, run-time licenses are required for each board shipped and development tool licenses are required for each developer.

☑ **Reputation** Make sure you're dealing with someone you'll be happy with.

☑ **Other Services** Real-time operating systems provide developers a full complement of features: several types of semaphores (counting, mutual exclusion), times, mailboxes, queues, buffer managers, memory system managers, events, and more.

http://leadingedgeinnovations.biz/EmbeddedSystemProgramming/files/99/9903/f-hawley.pdf

¿Por qué elegir Linux?

- Razones para escoger Linux
- Linux en sistemas empotrados
- Dispositivos basados en Linux
- Linux empotrado
- Linux empotrado # diferencias

Razones para escoger Linux @ [BELS]

- Calidad y fiabilidad del código. (Modelo desarrollo SW abierto)
  - Modular y estructurado.
  - Legible.
  - Extensible.
  - Configurable.
  - Predecible.
  - Recuperación frente errores.
  - Longevo.
- Disponibilidad del código.
- Soporte del hardware.
- Protocolos de comunicación y estándares software.
- Herramientas disponibles.
- La Comunidad.
- Licencias.
- Independencia de vendedor.
- Coste.

Linux en sistemas empotrados

Linux

...

Embedded devices

Due to its low cost and ease of modification, an embedded Linux is often used in embedded systems. Linux has become... a major competitor of Symbian OS which is used in the majority of smartphones—16.7% of smartphones sold worldwide during 2006 were using Linux—and it is an alternative to the proprietary Windows CE and Palm OS operating systems on mobile devices.

> ...the dominant mobile operating system for smartphones, running on 79.3% of units sold worldwide during the second quarter of 2013.

Dispositivos basados en Linux

Linux-powered devices

Linux-based devices or **Linux devices** are computer appliances that are powered by the Linux operating system kernel. They are often minimalistic and purpose-built, thus may be environmentally friendly and create less electronic waste per unit.

Linux empotrado

Embedded Linux

Embedded Linux is the use of Linux in embedded computer systems such as mobile phones, personal digital assistants, media players, set-top boxes, and other consumer electronics devices, networking equipment, machine control, industrial automation, navigation equipment and medical instruments. According to survey conducted by Venture Development Corporation, Linux was used by 18% of embedded engineers.

Linux empotrado # diferencias

Differences from other Linux operating systems

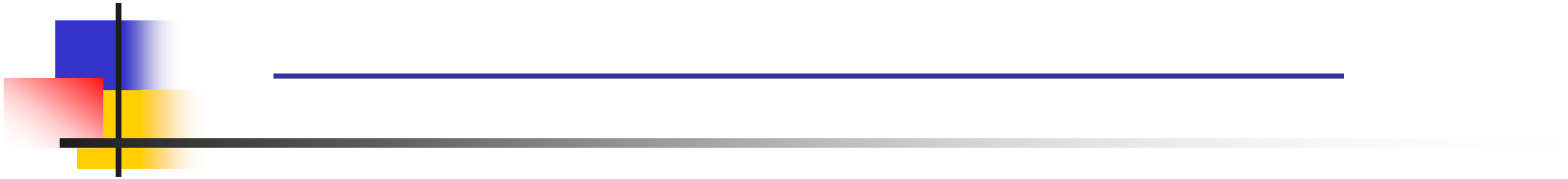
Unlike desktop and server versions of Linux, embedded versions of Linux are designed for devices with relatively limited resources, such as cell phones and set-top boxes. Due to concerns such as cost and size, embedded devices usually have much less RAM and secondary storage than desktop computers, and are likely to use flash memory instead of a hard drive. Since embedded devices serve specific rather than general purposes, developers optimize their embedded Linux distributions to target specific hardware configurations and usage situations. These optimizations can include reducing the number of device drivers and software applications, and modifying the Linux kernel to be a real-time operating system.

Instead of a full suite of desktop software applications, embedded Linux systems often use a small set of free software utilities such as busybox, and replace the glibc C standard library with a more compact alternative such as dietlibc, uClibc, or Newlib.

Ejemplos de dispositivos basados en Linux

- La grabadora de vídeo digital **TiVo**.
- **Android**, uno de los sistemas operativos para smartphones más populares.
- El hotspot de **AT&T MiFi**, para que varias personas compartan salida a internet.
- El *Large Hadron Collider*, **LHC**.
- Neveras con pantalla, como algunas de Electrolux.
- Las televisiones **Sony Bravia HDTV**.
- **Chumby**, un despertador con multitud de funciones y más de 1000 aplicaciones.
- Los GPS de **TomTom**, y de otras marcas.
- **Kindle**: el superventas lector de ebooks de Amazon.
- Los coches que se conducen solos o coches autónomos.

Construcción de un sistema empotrado

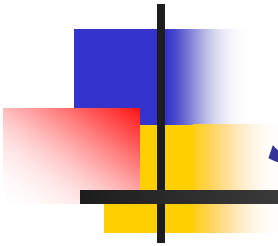


- *System building: goal and solutions*
- *System building, manually*
- *System building tools, compared*
- buildroot: making Embedded Linux easy
- Otras herramientas útiles
- *Commercial toolsets - Summary*

System building: goal and solutions

- Goal
  - Integrate all the software components, both third-party and in-house, into a working root filesystem
  - It involves the download, extraction, configuration, compilation and installation of all components, and possibly fixing issues and adapting configuration files
- Several solutions
  - Manually
  - System building tools
  - Distributions or readymade filesystem

http://free-electrons.com/doc/embedded_linux_sysdev.pdf#72



System building, manually

Ejemplos, de menos a más complejo:

- *Simple embedded Linux system*

  http://www.linuxfordevices.com/c/a/Linux-For-Devices-Articles/Tutorial-A-simple-embedded-Linux-system/

- *Building an embedded Linux system emulator*

  http://www.linuxfordevices.com/c/a/Linux-For-Devices-Articles/Getting-started-with-an-embedded-Linux-system-emulator/

- *How To Build a Minimal Linux System from Source Code*

  http://users.cecs.anu.edu.au/~okeefe/p2b/buildMin/buildMin.html

- *Linux From Scratch*

  http://www.linuxfromscratch.org/lfs/

System building tools, compared

Tool name	License	Small systems	Glibc / uClibc	Reproducibility / Leverage	Popularity	Actively maintained
Buildroot	Free	Yes	G+U	Good	Good	Yes
Scratchbox	Free	Yes	G+U	Poor	Low	Yes
OpenEmbedded	Free	No	G+U	Good	Very good	Definitely
LTIB	Free	Yes	G+U	Good	Low	Yes
PTXdist	Free	Yes	G	Good	Low	Yes
Gentoo embedded	Free	No?	G(U?)	?	Low?	Yes
Firmware Linux	Free	Yes	?	Poor	Low	Rob never sleeps
Vendor tools	Closed	?	G+U	Good	N/A	Yes
Home made	Closed	?	?	?	N/A	?

http://elinux.org/images/6/6c/Building-tools-bof.pdf#11

buildroot: making Embedded Linux easy

The major buildroot features are:

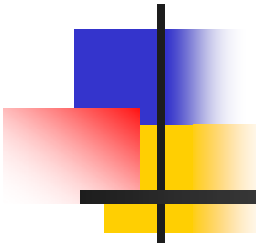
- **Handle everything**:
  - cross-compiling toolchain
  - bootloader compilation
  - kernel image compilation
  - root filesystem generation
- **Very easy** to set up:
  - menuconfig configuration interface
- **Several hundreds of packages.**
- **Multiple filesystem types** for the root filesystem.
- Can use glibc, eglibc or uClibc cross-compilation toolchain.
- **Simple structure,** easy to understand and extend.

http://buildroot.uclibc.org

Otras herramientas útiles

- buildroot, *making Embedded Linux easy*
  http://buildroot.uclibc.org
- busybox, *The Swiss Army Knife of Embedded Linux*
  http://www.busybox.net
- qemu, *open source machine emulator*
  http://www.qemu.org
  http://alien.slackbook.org/dokuwiki/doku.php?id=slackware:qemu
- dropbear, *ssh server and client*
  http://matt.ucc.asn.au/dropbear/dropbear.html

- Otras:
  - minicom
  - fakeroot

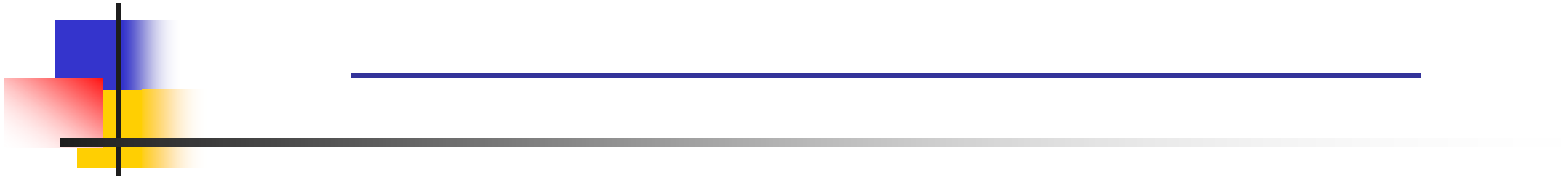


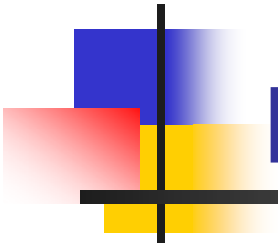
Commercial toolsets - Summary

- Major vendors: MontaVista, Wind River, TimeSys
  - Involved in Linux development.
- Smaller vendors: Koan, Sysgo, Denx...
  - Trying to differentiate their products.
- Community based companies: Denx, CodeSourcery
  - Contribute to community tools.
  - Mainly offer support and development.
- Traditional RTOS vendors: LynuxWorks
  - Forced to support Linux to retain market share.
  - Strong real-time expertise but lack community involvement.

http://free-electrons.com/doc/embedded_linux_sysdev.pdf#109

Entorno de desarrollo cruzado





Entorno de desarrollo cruzado

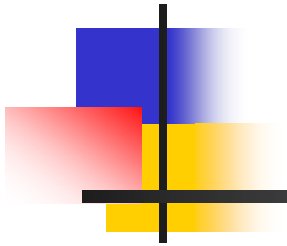
Definiciones

- *Host* sistema donde se trabaja desarrollando código.
- *Target* sistema donde ejecutará finalmente el código generado.
- *Toolchain* conjunto de herramientas de desarrollo.
  - Compilador, montador, depurador, *binutils*, etc.
  - GNU *toolchain*
  - Configuración de un *target* : **arquitectura, *kernel*** y **biblioteca**.
  - *cpu-vendor-kernel-os-tool* Ej: powerpc-linux-uclibc-gcc
- Bibliotecas:
  - glibc es demasiado compleja y grande.
  - Alternativas: uClibc, newlib, dietlibc.
  - Compilación dinámica vs. estática.

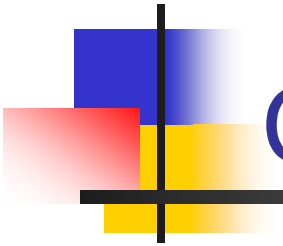
Entorno de desarrollo cruzado

- Uso, ejemplo práctico:
  - *Embedded Programming with the GNU Toolchain*
    http://www.bravegnu.org/gnu-eprog
  - Desde los fuentes:
    ```
    powerpc-linux-gcc <params>
    make CC=powerpc-linux-gcc <goals>
    ```
  - Con autoconf:
    ```
    ./configure --host=powerpc-linux --prefix=<rootfs>
    ```
  - El *kernel*:
    ```
    make ARCH=powerpc CROSS_COMPILE=powerpc-linux- <goals>
    ```
- *Anatomy of a Program in Memory*
  http://duartes.org/gustavo/blog/post/anatomy-of-a-program-in-memory
- Alternativas para **depuración** cruzada:
  - Tarjeta de desarrollo
  - *Target* virtual
  - *Remote debugging with gdb*
  - JTAG, *Joint Test Action Group*

Kernel de Linux, compilación



- Compilación del *kernel*
- El *kernel* de Linux



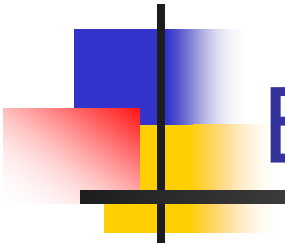
Compilación del *kernel*

- Selección de la versión.
- Descarga de los fuentes.
  - *The Linux Kernel Archives*
    http://www.kernel.org
- Documentación:
  - Subdirectorio: `linux-source-2.6.XX/Documentation/`
- Configuración, compilación e instalación (en local):

```
make menuconfig install modules modules-install
```

- Módulo vs. *driver*
- Compilación cruzada:

```
make CROSS_COMPILE=powerpc-linux- ARCH=powerpc ...
```



El *kernel* de Linux

Exploración de los fuentes del núcleo.

- *KernelHacking*

  http://kernelnewbies.org/KernelHacking

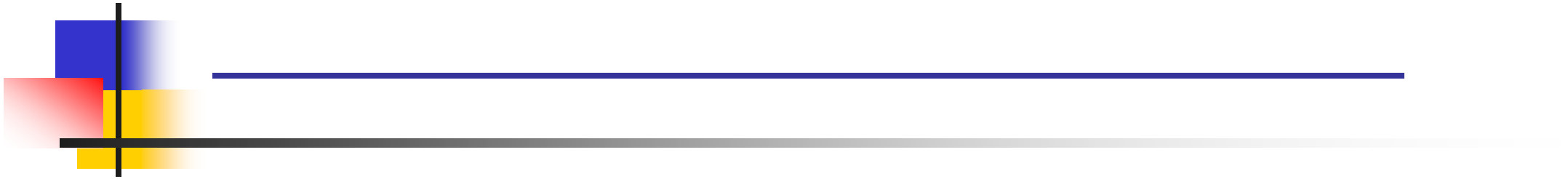
- *Linux Cross Reference*

  http://lxr.linux.no/#linux+v2.6.21.7/

- *Interactive map of Linux kernel*

  http://www.makelinux.net/kernel_map

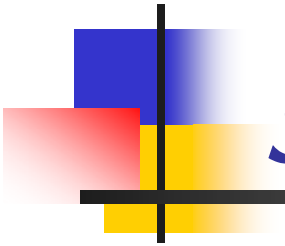
Sistema de ficheros raíz, generación



- Sistemas de ficheros raíz
- *System foundations*

Sistemas de ficheros raíz

- Tipos de *filesystem* para empotrados (para flash):
  - Abstracción mtdblock y
    - Cualquier SF de escritura directa: ext2.
    - Mejor, jffs2 de escritura incremental.
  - Mejor aún, abstracción mtd y UBIFS (Nokia).
  - Initramfs (mejor que un ramdisk)
    http://www.kernel.org/doc/Documentation/filesystems/ramfs_rootfs_initramfs.txt
- Contenido mínimo y estándares:
  - *Filesystem Hierarchy Standard*
  - *Linux Standard Base*
- Licencias y módulos vs. *drivers.*

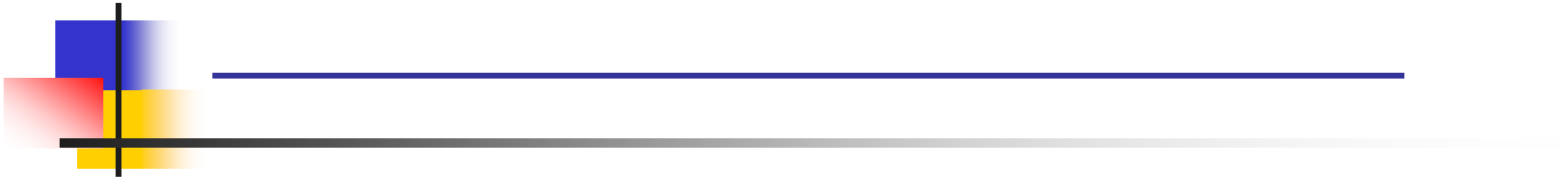


System foundations

- A basic root file system needs at least
  - A traditional directory hierarchy: `/bin`, `/etc`, `/lib`, `/root`, `/usr/bin`, `/usr/lib`, `/usr/share`, `/usr/sbin`, `/var`, `/sbin`
  - A set of basic utilities: the init program, a shell and other traditional Unix command line tools. This is usually provided by Busybox.
  - The C library and related ones (thread, math, etc.) installed in `/lib`.
  - A few configuration files: `/etc/inittab`, initial scripts `/etc/init.d`
- On top of this foundation common to most embedded Linux system, we can add third-party or in-house components.

http://free-electrons.com/doc/embedded_linux_sysdev.pdf#75

Arranque del sistema, configuración



- Aspectos configurables y parámetros de arranque
- Aspectos de alto nivel

Aspectos configurables y parámetros de arranque

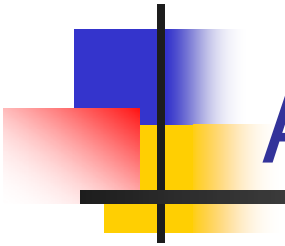
¿Cómo arranca Linux?

Linux startup process

- *From Power Up To Bash Prompt*
  http://users.cecs.anu.edu.au/~okeefe/p2b/p2b-HOWTO.pdf
- *What happens before the login prompt*
  http://oldfield.wattle.id.au/luv/boot.html

¿Qué se puede parametrizar en el arranque?

- *The Linux BootPrompt-HowTo*
  http://www.faqs.org/docs/Linux-HOWTO/BootPrompt-HOWTO.html
- *Linux kernel parameters*
  http://www.cyberciti.biz/howto/question/static/linux_kernel_parameters.php



Aspectos de alto nivel

- *Linux Hotplugging*

  http://linux-hotplug.sourceforge.net/

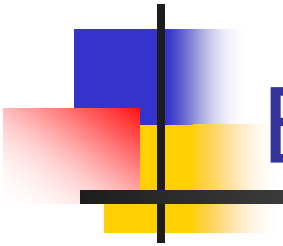
- *Udev: Device Management In Modern Linux System*

  http://www.linux.com/news/hardware/peripherals/180950-udev

- *The Unix Toolbox, useful for IT and advanced users*

  http://cb.vu/unixtoolbox.xhtml

Bibliografía



Bibliografía

[SRTOS] Selecting a Real-Time Operating System.
Greg Hawley

[ESD] Embedded Systems Design.
Arnold Berger
CMP Books

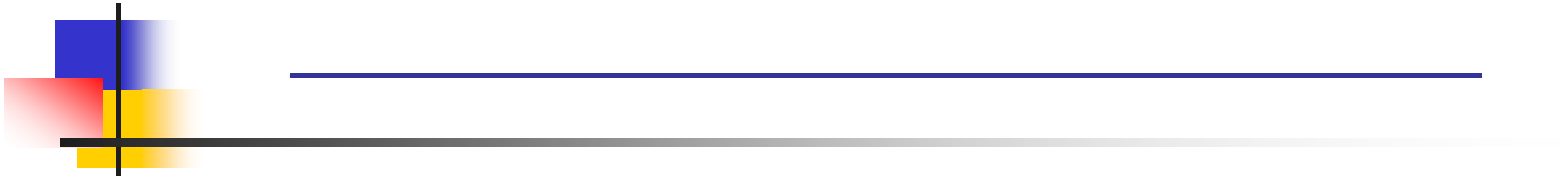
[BELS] Building Embedded Linux Systems. 2nd Edition.
Karim Yaghmour; Jon Masters; Gilad Ben-Yossef; Philippe Gerum
O'REILLY

[DEH] Designing Embedded Hardware.
John Catsoulis.
O'REILLY

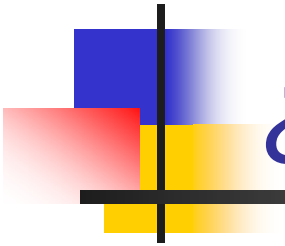
[LFD] http://www.linuxfordevices.com

[WPD] http://www.wikipedia.org

Proyecto práctico



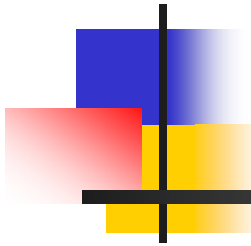
- ¿Qué objetivo perseguimos?
- Terminología y herramientas
- El target
- Desarrollos sobre el target
- Enlaces a los guiones a seguir



¿Qué objetivo perseguimos?

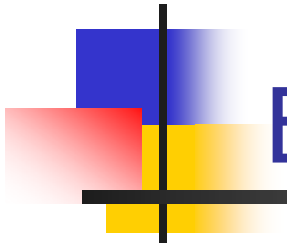
- Que el alumno obtenga la experiencia de haber elaborado un **Sistema Empotrado** completo.
  - Deberá seguir un guión dado, saliéndose lo mínimo.
  - Deberá indagar para entender los pasos que está dando.
  - Deberá contestar a ciertas preguntas para demostrarlo.

- Esta experiencia se basará en:
  - El Sistema Operativo Linux.
  - Herramientas como: buildroot y busybox.
  - Sobre la plataforma hardware dada concreta.

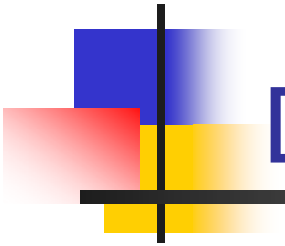


Terminología y herramientas

- En clase se irá explicando la terminología y se irán presentando las herramientas utilizadas para poder entender todo el proceso de desarrollo de un Sistema Linux Empotrado.
  - qemu
  - busybox
  - buildroot
  - toolchain
  - minicom
  - fakeroot
  - etc. etc.



El target

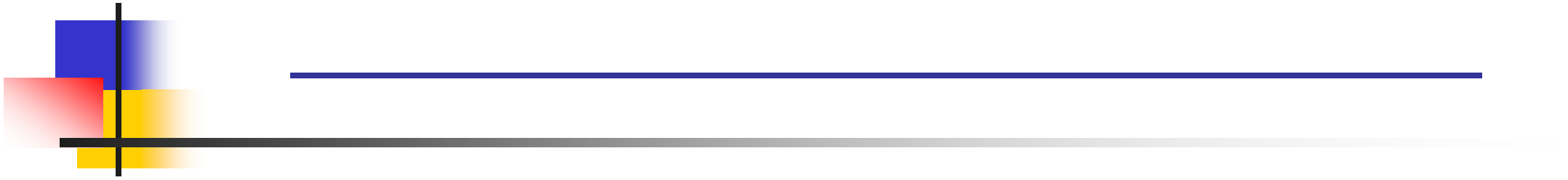


Desarrollos sobre el target

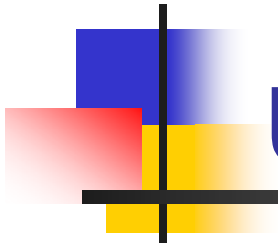
- Automatización del desarrollo con buildroot
  - Ficheros de configuración dados.
- Hito: Estampación del *kernel* ad-hoc.
  - Por línea serie, minicom.
- Hito: Arranque por NFS.
  - Parametrización del arranque de *kernel*.
- Hito: Arranque de sistema de ficheros ad-hoc.
  - CargaLan, programa de aplicación ad-hoc.

- Guión de actividades con preguntas para contestar.
  - Disponible en la página de la asignatura.
- Presentación y defensa previa cita con el profesor.

Práctica ligera



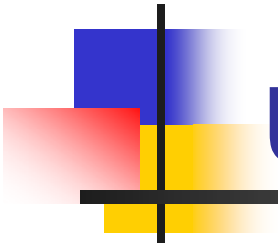
- Un sistema Linux básico
- Un arranque real de un PC



Un sistema Linux básico

- Características:
  - Propio kernel
  - Arranque de qemu-x86 (virtual)
  - initramfs
  - busybox

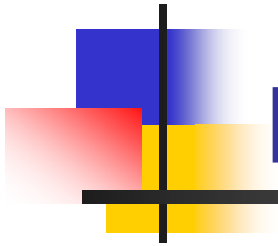
- Guión de actividades con preguntas para contestar.
  - Disponible en la página de la asignatura.



Un arranque real de un PC

- Características:
  - Propio kernel
  - Arranque de propio PC!!
  - initramfs
  - busybox
  - syslinux, *bootloader* para *pendrive*

- Guión de actividades con preguntas para contestar.
  - Disponible en la página de la asignatura.



Enlaces a los guiones a seguir

- Guiones de actividad y otra documentación, disponibles en la página Web de la asignatura:

"Sistemas empotrados, ubicuos y móviles"

http://datsi.fi.upm.es/docencia/SEUM

FIN