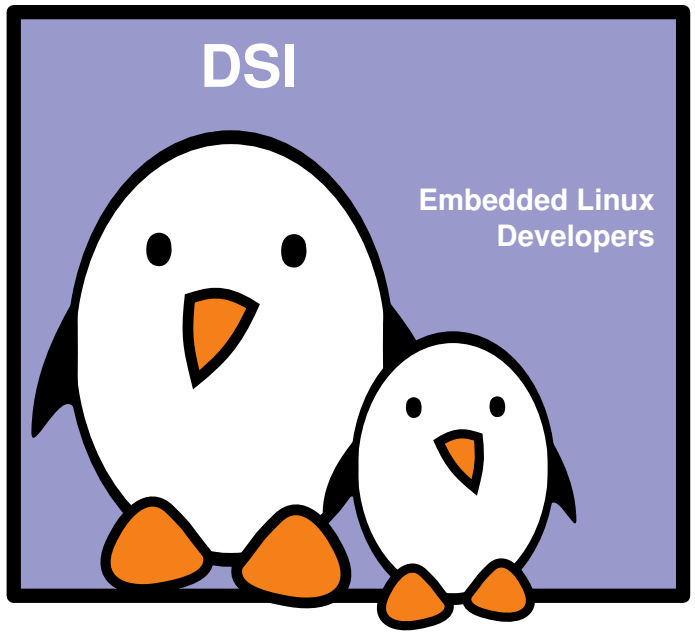
Embedded Linux system development

Free Electrons



Latest update: September 1, 2017.
Document updates and sources:
`http://free-electrons.com/doc/training/embedded-linux`
Corrections, suggestions, contributions and translations are welcome!

Derechos de copia

Hipervínculos en el documento

Hay muchos hipervínculos en el documento

- Hipervíncluos regulares:
  `http://kernel.org/`
- Enlaces a la documentación del Kernel:
  `Documentation/kmemcheck.txt`
- Enlaces a los archivos fuente y directorios del kernel:
  `drivers/input`
  `include/linux/fb.h`
- Enlaces a declaraciones, definiciones e instancias de los simbolos del kernel (funciones, tipos, datos, estructuras):
  `platform_get_irq()`
  `GFP_KERNEL`
  `struct file_operations`

Free Electrons at a glance

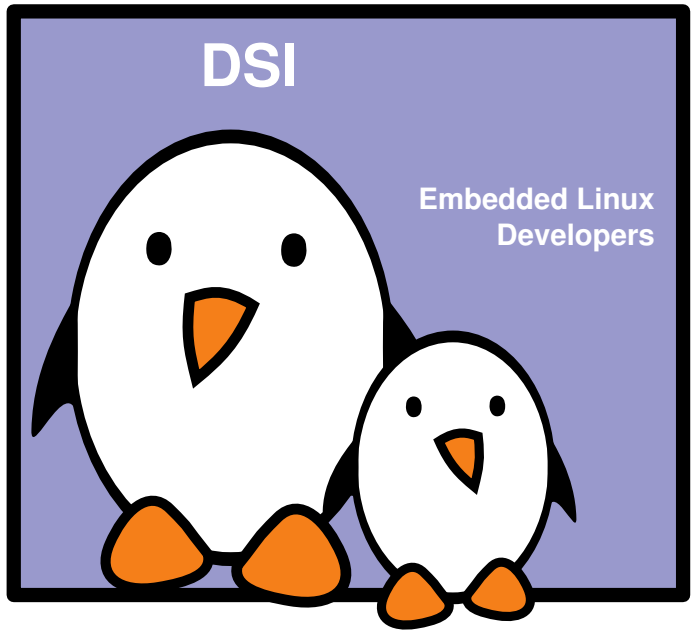
- Engineering company created in 2004
  (not a training company!)
- Locations: Orange, Toulouse, Lyon (France)
- Serving customers all around the world
  See `http://free-electrons.com/company/customers/`
- Head count: 9
  Only Free Software enthusiasts!
- Focus: Embedded Linux, Linux kernel, Android Free Software / Open Source for embedded and real-time systems.
- Activities: development, training, consulting, technical support.
- Added value: get the best of the user and development community and the resources it offers.

Free Electrons on-line resources

- All our training materials:
  `http://free-electrons.com/docs/`
- Technical blog:
  `http://free-electrons.com/blog/`
- Quarterly newsletter:
  `http://lists.free-electrons.com/mailman/listinfo/newsletter`
- News and discussions (Google +):
  `https://plus.google.com/+FreeElectronsDevelopers`
- News and discussions (LinkedIn):
  `http://linkedin.com/groups/Free-Electrons-4501089`
- Quick news (Twitter):
  `http://twitter.com/free_electrons`
- Linux Cross Reference - browse Linux kernel sources on-line:
  `http://lxr.free-electrons.com`

Información sobre el curso

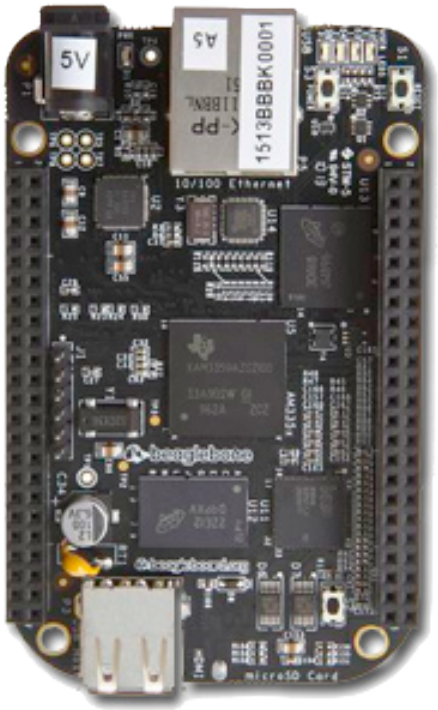
Free Electrons



Hardware utilizado en esta sesión de entrenamiento

BeagleBone Black de CircuitCo

- Texas Instruments AM335x (ARM Cortex-A8)
- CPU potente, con aceleración 3D, procesardores adicionales de tiempo real (PRUs) y muchos periféricos.
- 512 MB de RAM
- 2 GB de almacenamiento eMMC en la placa (4 GB en la Rev C)
- puertos USB host y USB device
- conector microSD
- puerto HDMI
- 2 x 46 pins peines, con acceso a varios buses de expansión (I2C, SPI, UART y más)
- Un inmenso número de placas de expansión, llamadas *capes*. Ver `http://beagleboardtoys.com/`.



Participe!

Durante las lecturas...

- No vacile en hacer preguntas. Otras personas en la audiencia pueden tener preguntas similares también.
- Esto ayuda al entrenador a detectar cualquier explicación que no haya sido clara o no haya sido profundizada.
- No vacile en compartir su experiencia, por ejemplo, comparando Linux/Android con otros sistemas operativos que se utilicen en su compañía.
- Su punto de vista es muy valioso, porque puede ser similar al de sus colegas y diferente al del entrenador.
- Su participación puede hacer nuestra sesión más interactiva y que los temas sean más fáciles de aprender.

Directivas de los laboratorios prácticos

Durante los laboratorios prácticos...

- Abra la copia electrónica del material de lectura y úsela en todo el laboratorio práctico para encontrar las diapositivas que necesite nuevamente.
- No copie y pegue desde las diapositivas PDF.
  Las diapositivas contienen caracteres UTF-8 que se ven igual a los caracteres ASCII, pero no van a se interpretados por los shells o los compiladores.

Coopere!

Como en la comunidad de Software Libre y Código abierto, la cooperación durante los laboratorios prácticos es valiosa en está sesión de entrenamiento:

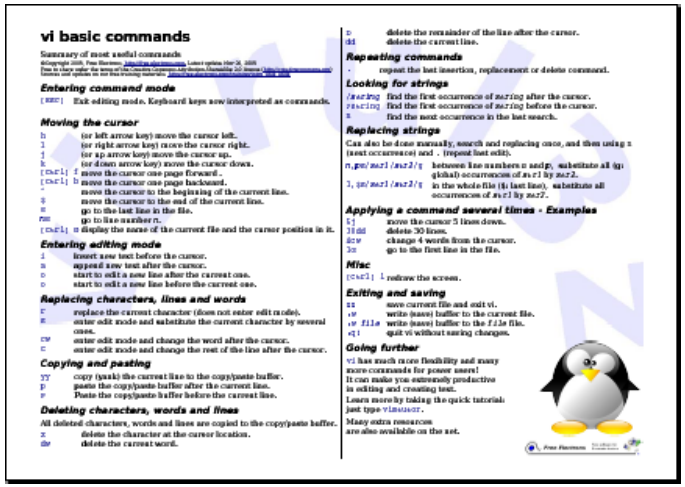
- Si ud completa los laboratorios antes que otras personas, no vacile en ayudar a otras personas e investigar problemas que enfrenten. Mientras más rápido progresemos como grupo, más tiempo vamos a tener para explorar temas adicionales.
- Explique lo que entendió a otros participantes cuando sea necesario. También ayuda a consolidar sus conocimientos.
- No dude en reportar posibles errores a su instructor.
- No dude en buscar también soluciones en Internet.

Ayuda memoria de comandos

- Esta ayuda memoria proporciona ejemplos de necesidades típicas (buscar archivos, extraer un archivo tar ...)
- Nos ahorra 1 día de formación en línea de comandos UNIX / Linux.
- Nuestro mejor consejo: en el shell de línea de comandos, siempre presione la tecla `Tab` para completar los nombres de los comandos y las rutas de archivo. Esto evita el 95% de errores de escritura.
- Obtenga una copia electrónica en `http://free-electrons.com/doc/training/embedded-`

Comandos básicos del vi

- El editor `vi` es muy útil para hacer cambios rapidos en archivos en un destino embebido
- Aunque no es muy fácil de usar al principio, `vi` es muy potente y sus 15 comandos principales son fáciles de aprender y son suficientes para el 99% de las necesidades de todos!
- Obtenga una copia electrónica en `http://free-electrons.com/doc/training/embedded-linux/vi_memento.pdf`
- También puede realizar el tutorial rápido ejecutando `vimtutor`. Esta es una inversión digna!



vi basic commands

Summary of most useful commands

Entering command mode

[ESC] Exit editing mode. Keyboard keys now interpreted as commands.

Moving the cursor

h (or left arrow key) move the cursor left.
l (or right arrow key) move the cursor right.
k (or up arrow key) move the cursor up.
j (or down arrow key) move the cursor down.
[Ctrl] f move the cursor one page forward.
[Ctrl] b move the cursor one page backward.
0 move the cursor to the beginning of the current line.
$ move the cursor to the end of the current line.
G go to the last line in the file.
nnn go to line number nn.
[Ctrl] g display the name of the current file and the cursor position in it.

Entering editing mode

i insert new text before the cursor.
a append new text after the cursor.
o start to edit a new line after the current one.
O start to edit a new line before the current one.

Replacing characters, lines and words

r replace the current character (does not enter edit mode).
R enter edit mode and substitute the current character by several ones.
cw enter edit mode and change the word after the cursor.
C enter edit mode and change the rest of the line after the cursor.

Copying and pasting

yy copy (yank) the current line to the copy/paste buffer.
p paste the copy/paste buffer after the current line.
P Paste the copy/paste buffer before the current line.

Deleting characters, words and lines

All deleted characters, words and lines are copied to the copy/paste buffer.
x delete the character at the cursor location.
dw delete the current word.
D delete the remainder of the line after the cursor.
dd delete the current line.

Repeating commands

. repeat the last insertion, replacement or delete command.

Looking for strings

/string find the first occurrence of string after the cursor.
?string find the first occurrence of string before the cursor.
n find the next occurrence in the last search.

Replacing strings

Can also be done manually, search and replacing once, and then using n (next occurrence) and . (repeat last edit).

n,m s/mar1/mar2/g between line numbers n and m, substitute all (g) globall occurrences of mar1 by mar2.
1, $ s/mar1/mar2/g in the whole file ($ last line), substitute all occurrences of mar1 by mar2.

Applying a command several times - Examples

5j move the cursor 5 lines down.
3dd delete 30 lines.
dcw change 4 words from the cursor.
3G go to the first line in the file.

Misc

[Ctrl] l redraw the screen.

Exiting and saving

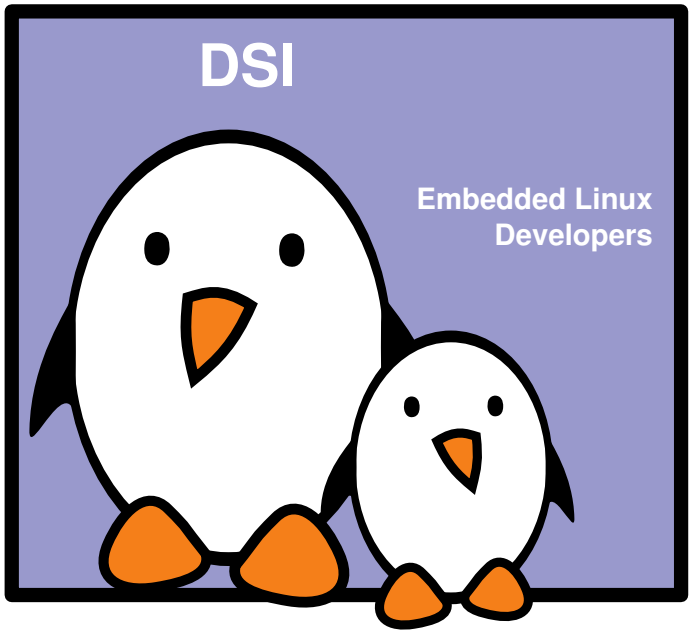
ZZ save current file and exit vi.
:w write (save) buffer to the current file.
:w file write (save) buffer to the file file.
:q! quit vi without saving changes.

Going further

vi has much more flexibility and many more commands for power users! It can make you extremely productive in editing and creating text.
Learn more by taking the quick tutorials just type vimtutor.
Many extra resources are also available on the net.

Introducción a Linux Embebido

Free Electrons



Nacimiento del software libre

- 1983, Richard Stallman, **proyecto GNU** y el concepto de **software libre**. Comienza el desarrollo de *gcc*, *gdb*, *glibc* y otras herramientas importantes
- 1991, Linus Torvalds, **proyecto Linux kernel**, un nucleo de sistema operativo similar a Unix. Junto con el software GNU y otros componentes de código abierto: forman un sistema operativo completo GNU/Linux
- 1995, Linux es más popular en sistemas servidor
- 2000, Linux es más popular en **sistemas embebidos**
- 2008, Linux es más popular en dispositivos móviles
- 2010, Linux es más popular en teléfonos

¿Software libre?

- Un programa es considerado **libre** cuando su licencia ofrece a todos sus usuarios las siguientes **cuatro** libertades
  - Libertad de ejecutar el Sofrware para cualquier propósito
  - Libertad de estudiar el Software y modificarlo
  - Liberatd de redistribuir copias
  - Libertad de distribuir copias de versiones modificadas
- Estas libertades estan concedidas para uso tanto comercial como no-comercial
- Implican la disponibilidad del código fuente, el Software puede ser modificado y distribuido a los clientes
- **Una opción interesante para los sistemas embebidos!**

¿Qué es Linux embebido?

Linux embebido es el uso del **kernel de Linux** y varios componentes **open-source** en sistemas embebidos

Ventajas de Linux y open-source para sistemas embebidos

Reutilización de componentes

- La principal ventaja de Linux y open-source en sistemas embebidos es la **habilidad** de reutilizar componentes
- El ecosistema de open-source ya provee de varios componentes para características estandares, desde soporte de Hardware hasta protocolos, pasando por multimedia, gráficos, bibliotecas criptográficas, etc.
- Tan pronto como un dispositivo Hardware, o un protocolo, o una característica se torna conocida, existen varias chances de tener un componente open-source que lo soporte.
- Permite diseñar de forma rápida y desarrollar productos complejos, basados en componentes existentes.
- No es necesario redesarrollar otro kernel de sistema operativo, una pila TCP/IP, una pila USB u otra biblioteca gráfica.
- **Permite enfocarce en el valor agregado del producto.**

Bajo costo

- El Software libre puede ser duplicado en la cantidad de dispositivos que se quiera, libre de cargos.
- Si su sistema embebido utiliza solo Software libre, se puede reducir los costos de licencias de Software a cero. Incluso, las herramients de desarrollo son libres, a menos que se elija una versión de Linux embebido comercial.
- **Permite tener un mayor presupuesto para el Hardware o para incrementar las habilidades y conocimiento de la compañía**

Control total

- Con código abierto, disponemos del código fuente de todos los componentes del sistema
- Permite modificaciones ilimitadas, cambios, ajustes, depuración, optimización, por un período de tiempo ilimitado
- Sin una dependencia "bloqueante" de un vendedor externo
  - Componentes que no sean de código abierto se deben evitar cuando el sistema se diseña y desarrolla
- **Permite tener un control total sobre el Software que forma parte del sistema**

Calidad

- Varios componentes de código abierto son ampliamente utilizados en millones de sistemas
- Generalmente son de mayor calidad que los desarrollos in-house o incluso de los vendedores propietarios
- Por supuesto, no todos los componentes de código abierto son de buena calidad, pero los más ampliamente utilizados los son.
- **Permite diseñar su sistema con componentes fundacionales de alta calidad**

Facilita la prueba de nuevas caracteristicas

- Dado la disponibilidad del código abierto, es simple obtener una copia del Software para evaluarlo
- Permite de forma simple estudiar las opciones mientras se está decidiendo
- Mucho más simple que la compra y procesos de demostración necesitan con la mayoría de los productos propietarios
- **Permiten explorar de forma simple nuevas posibilidades y soluciones**

Soporte de la comunidad

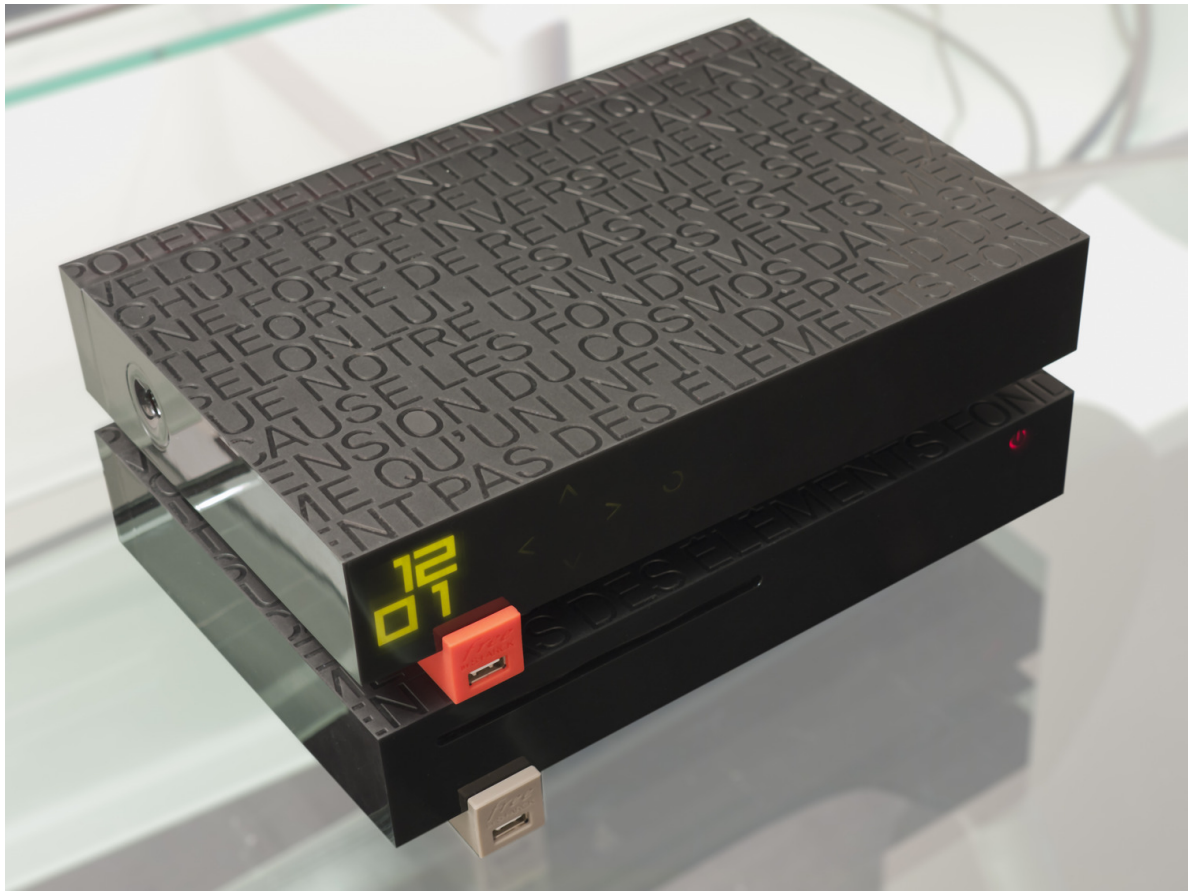
- Los componentes de Software de código abierto son desarrollados por comunidades de desarrolladores y usuarios
- Esta comunidad puede proveer soporte de alta calidad: se puede contactar directamente a los desarrolladores principales del componente que se está usando. La probabilidad de obtener una respuesta no depende de la compañía para la que trabajemos.
- En general mejor que el soporte tradicional, pero es necesario entender como funciona la comunidad para hacer un uso correcto de las posibilidades de soporte
- **Permite acelerar la resolución de problemas cuando se esté desarrollando el sistema**

Formando parte de la comunidad

- La posibilidad de formar parte de la comunidad de desarrollo de algunos de los componentes utilizados en sistemas embebidos: reporte de fallas, prueba de nuevas versiones y características, parches que corrigen errores o agregan nuevas características, etc.
- La mayoría del tiempo, los componentes de código abierto no son el núcleo de valor del producto: es interes de todos contribuir al mismo
- Para los *ingenieros*: una forma muy **motivante** de ser reconocido fuera de la compañía, comunicarse con otros en el mismo campo, **la oportunidad de nuevas posibilidades**, etc.
- Para los *gerentes*: **factor de motivación** para los ingenieros, permite a la compañía ser **reconocida** en la comunidad de código abierto y por lo tanto obtener soporte de forma más simple y ser **más atractivo** para los desarrolladores de código abierto

Algunos ejemplos de sistemas embebidos ejecutando Linux

Ruters personales



Televisión

Terminales de punto de venta

Maquinas de corte Laser

Maquina de Viticultura

Hardware para sistemas embebidos en Linux

Procesador y arquitectura (1)

- El kernel de Linux y la mayoría de otros componentes dependientes de la arquitectura soportan un amplio rango de arquitecturas de 32 y 64 bits
  - x86 y x86-64, como se encuentran en plataformas PC, pero también sistemas embebidos (multimedia, industrial)
  - ARM, con miles de diferentes SoC (multimedia, industrial)
  - PowerPC (principalmente aplicaciones en tiempo real e industriales)
  - MIPS (principalmente para aplicaciones de red)
  - SuperH (principalmente aplicaciones multimedia)
  - Blackfin (arquitectura DSP)
  - Microblaze (soft-core para FGPA de Xilinx)
  - Coldfire, SCore, Tile, Xtensa, Cris, FRV, AVR32, M32R

Procesador y arquitectura (2)

- Ambas arquitecturas, MMU y no-MMU son soportadas, aunque la arquitectura no-MMU tiene algunas limitaciones.
- Linux no está diseñado para microcontroladores pequeños.
- Salvo el juego de herraminetas, el cargador de inicio y el kernel, todos los otros componentes generalmente son **independientes de la arquitectura**

RAM y almacenamiento

- **RAM**: un sistema básico con Linux puede funcionar con 8 MB de RAM, pero un sistema más realista usualmente requiere 32 MB de RAM. Depende del tipo y tamaño de la aplicación.
- **Almacenamiento**: un sistema básico con Linux puede trabajar con 4 MB de almacenamiento, pero usualmente se requiere una mayor cantidad.
  - El almacenamiento en Flash está soportado, tanto flash NAND como NOR, con sistemas de archivos específicos
  - Almacenamiento en bloque incluyendo tarjetas SD/MMC y eMMC son soportadas
- Es preferible no tener muchas restricciones en la cantidad de RAM/almacenamiento: tener una cierta flexibilidad en este nivel nos permite reutilizar tantas aplicaciones como sea posible.

Comunicación

- El kernel de Linux tiene soporte para varios protocolos de comunicaciones comunes
  - I2C
  - SPI
  - CAN
  - 1-wire
  - SDIO
  - USB
- Y también soporte para redes extensivo
  - Ethernet, Wifi, Bluetooth, CAN, etc.
  - IPv4, IPv6, TCP, UDP, SCTP, DCCP, etc.
  - Firewalling, ruteo avanzado, multicast

Tipos de Hardware y plataformas 1/2

- **Plataformas de evaluación** del proveedor de SoC. Usualmente costosas, pero con muchos perifericos incluidos. Generalmente no recomendable para productos reales.
- **Componente en Módulo**, una pequeña placa solamente con CPU/RAM/flash y algunos otros componentes, con conectores para acceder a todos los periféricos. Puede ser utilizado para construir productos finales en pequeñas o medianas cantidades.
- **Plataformas de desarrollo comunitario**, una nueva tendencia para hacer un SoC particular popular y facilmente disponible. Estas están listas para utilizar y a un bajo costo, pero generalmente tienen menos periféricos que las plataformas de evaluación. En algunos casos pueden ser utilizadas para productos reales.

Tipos de Hardware y plataformas 1/2

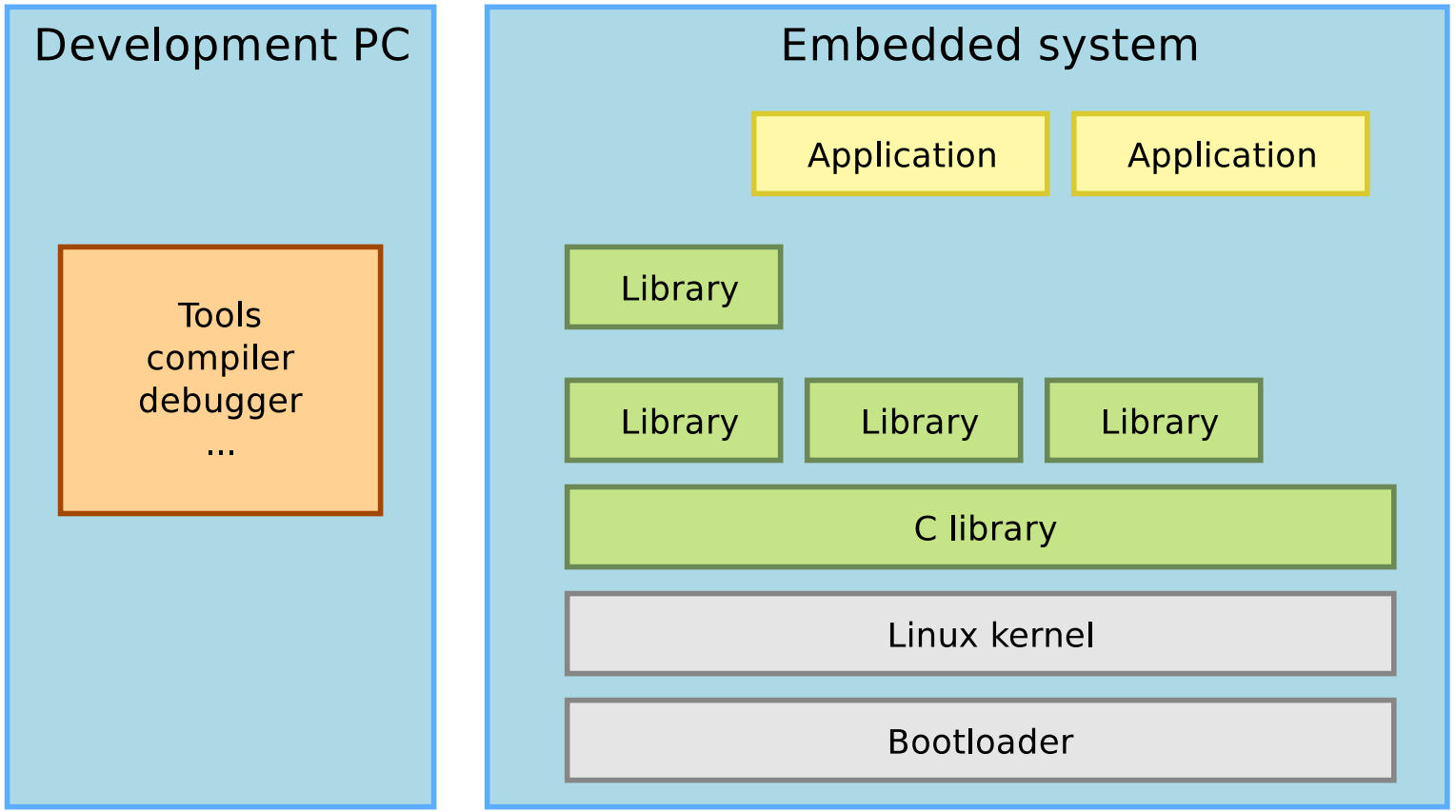
- **Plataforma personalizada**. Los esquemáticos para placas de evaluación o plataformas de desarrollo están disponibles más comunmente de forma libre, haciendo mas simple el desarrollo de plataformas personalizadas.

Criterios para la elección del Hardware

- Aseguresé que el Hardware que planea utilizar está actualmente soportado por el kernel de Linux, y posee un cargador de inicio de cógio abierto, especialmente para el SoC destino.
- Teniendo soporte en las versiones oficiales de los proyectos (kernel, cargador de inicio): la calidad es mejor, y nuevas versiones están disponibles.
- Algunos proveedores de SoC y/o proveedores de placas no contribuyen sus cambios hacia la linea principal del kernel de Linux. Recomiende que lo hagan, o utilice otro producto de ser posible. Una buena métrica es ver las diferencias entre su kernel y el oficial.
- **Entre un hardware correctamente soportado en el kernel oficial de Linux y un hardware mal soportado, existiran enormes diferencias en tiempo de desarrollo y costo.**

Arquitectura de sistemas embebidos en Linux

Arquitectura global



Componentes Software

- Juego de herramientas de compilación cruzada
  - Compilador que corre en la máquina de desarrollo, pero genera código para el destino
- Cargador de inicio
  - Ejecutado por el Hardware, responsable de la inicialización básica, cargao y ejecuta el kernel
- Kernel de Linux
  - Contiene los proceso y el manejo de memoria, red, controladores de dispositivos y provee servicios para la aplicación en el espacio de usuario
- Biblioteca C
  - La interfaz entre el kernel y las aplicaciones en el espacio de usuario
- Bibliotecas y aplicaciones
  - De terceros (Third-party or in-house)

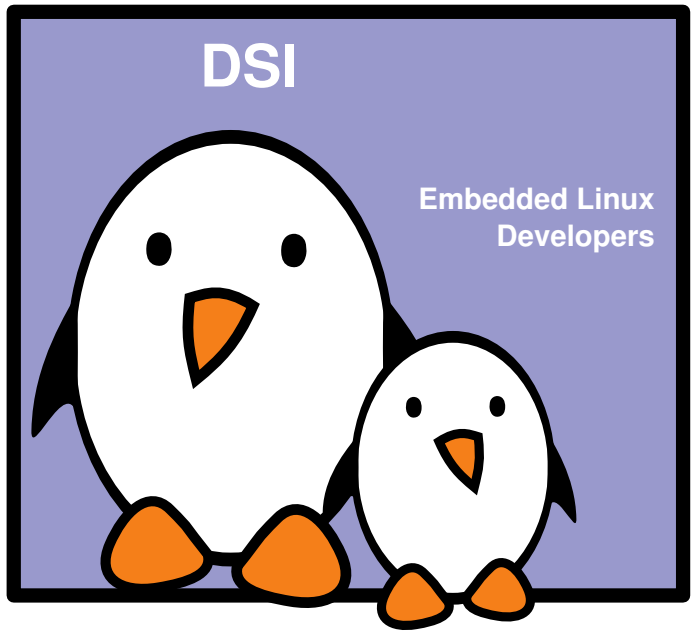
Trabajo sobre Linux embebido

Varias tareas de distinto tipo se necesitan al momento de desplegar un Linux embebido en un producto:

- **Board Support Package development (BSP)**
  - Un BSP contiene un bootloader y un kernel con los device drivers adecuados para el Hardware destino
  - Es el propósito del entrenamiento en *desarrollo del Kernel*
- **Integración del sistema**
  - Integrar todos los componentes, bootloader, kernel, y bibliotecas de terceros y aplicaciones y aplicaciones in-house en un sistema funcional
  - Es el propósito de *este* entrenamiento
- **Desarrollo de aplicaciones**
  - Aplicaciones Linux normales, pero utilizando bibliotecas especialmente elegidas

Embedded Linux development environment

Free Electrons

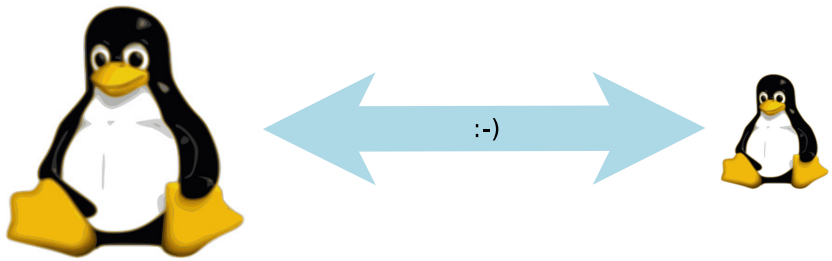


Soluciones de Linux Embebidas

- Dos formas de cambiar a Linux embebido
  - Utilice **soluciones proporcionadas y apoyadas por vendedores** como MontaVista, Wind River o TimeSys. Estas soluciones vienen con sus propias herramientas de desarrollo y entorno. Utilizan una mezcla de componentes de código abierto y herramientas propietarias.
  - Utilice **soluciones comunitarias**. Están completamente abiertas, soportado por la comunidad.
- En estas sesiones de formación, no promovemos un Proveedor y, por lo tanto, utilizamos soluciones comunitarias
  - Sin embargo, conociendo los conceptos, cambiar a otro proveedor es simple

SO para el desarrollo de Linux

- Recomendamos encarecidamente el uso de Linux como sistema para incorporar a los desarrolladores de Linux, por múltiples razones.
- Todas las herramientas de la comunidad están desarrolladas y diseñadas para funcionar Linux. Intentar utilizarlos en otros sistemas operativos (Windows, Mac OS X) conducirá a problemas, y su uso en estos sistemas generalmente no son apoyados por los desarrolladores de la comunidad.
- Como Linux también se ejecuta en el dispositivo embebido, todo el conocimiento ganado por el uso de Linux en el escritorio se aplicará de manera directa al dispositivo embebido.



Distribución de Linux de escritorio

- **Cualquier distribución de Linux de escritorio suficientemente reciente** se puede utilizar para la estación de trabajo de desarrollo
  - Ubuntu, Debian, Fedora, openSUSE, Red Hat, etc.
- Hemos elegido Ubuntu, ya que es una distribución de Linux de escritorio **ampliamente utilizado y fácil de usar**
- La configuración de Ubuntu en las computadoras portátiles de formación ha quedado intacta después del proceso de instalación normal. Aprender Linux embebido también consiste en aprender las herramientas necesarias en la estación de trabajo de desarrollo!

Usuarios root y no root de Linux

- Linux es un sistema operativo multiusuario
  - El **usuario root es el administrador**, y puede realizar operaciones privilegiadas como: montar sistemas de archivos, configurar la red, crear archivos de dispositivos, cambiar la configuración del sistema, instalar o eliminar software
  - Todos los **otros usuarios no tienen privilegios**, y no pueden realizar estas operaciones a nivel de administrador
- En un sistema Ubuntu, no es posible iniciar sesión como `root`, sólo como usuario normal.
- El sistema se ha configurado para que la primer cuenta de usuario creada pueda ejecutar operaciones privilegiadas a través de un programa llamado `sudo`.
  - Ejemplo: `sudo mount /dev/sda2 /mnt/disk`

Paquetes de Software

- El mecanismo de distribución para el software en GNU / Linux es diferente de la de Windows
- Las distribuciones de Linux proporcionan una forma central y coherente de instalar, actualizar y eliminar aplicaciones y bibliotecas: **paquetes**
- Packages contains the application or library files, and associated meta-information, such as the version and the dependencies
  - `.deb` en Debian y Ubuntu, `.rpm` en Red Hat, Fedora, openSUSE
- Los paquetes se almacenan en **repositorios**, usualmente en servidores HTTP o FTP
- Sólo debe utilizar paquetes de repositorios oficiales para su distribución, a menos que sea estrictamente necesario.

Gestión de paquetes de software (1)

Instrucciones para los sistemas GNU / Linux basados en Debian (Debian, Ubuntu...)

- Los repositorios de paquetes se especifican en `/etc/apt/sources.list`
- Para actualizar la lista de paquetes del repositorio: `sudo apt-get update`
- Para encontrar el nombre de un paquete a instalar, lo mejor es usar el motor de búsqueda en `http://packages.debian.org` o en `http://packages.ubuntu.com`. También puede usar: `apt-cache search <keyword>`

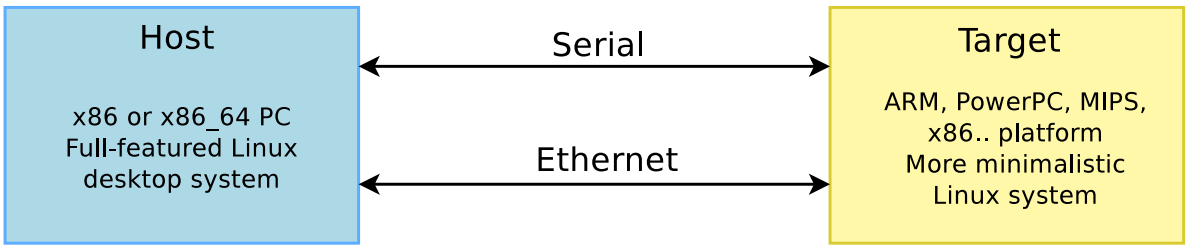
Gestión de paquetes de Software (2)

- Para instalar un paquete dado:
  `sudo apt-get install <package>`
- Para eliminar un parquete dado:
  `sudo apt-get remove <package>`
- Para instalar todas las actualizaciones de paquetes disponibles:
  `sudo apt-get dist-upgrade`
- Para obtener información acerca de un paquete:
  `apt-cache show <package>`
- Interfaces gráficas
  - Para GNOME Synaptic
  - Para KDE KPackageKit

Más detalles sobre la gestión de paquetes:
`http://www.debian.org/doc/manuals/apt-howto/`

Host vs. destino

- Al hacer el desarrollo embebido, siempre hay una división entre
  - El *host*, la estación de trabajo de desarrollo, que es típicamente una PC potente
  - El *destino*, que es el sistema embebido bajo desarrollo
- Están conectados por diversos medios: casi siempre una línea serie para fines de depuración, con frecuencia una conexión Ethernet, a veces una interfaz JTAG para la depuración a bajo nivel



Programa de comunicación por línea serie

- Una herramienta esencial para el desarrollo embebido es un programa de comunicación por línea serie, como HyperTerminal en Windows
- Hay múltiples opciones disponibles en Linux: Minicom, Picocom, Gtkterm, Putty, etc.
- En estos sesiones de entrenamiento, recomendamos utilizar el más simple de ellos: `picocom`
  - Se instala con `sudo apt-get install picocom`
  - Se ejecuta con
    `picocom -b BAUD_RATE /dev/SERIAL_DEVICE`
  - Se termina con `Control-A Control-X`
- Donde `SERIAL_DEVICE` generalmente es
  - `ttyUSBx` para conversores USB a serie
  - `ttySx` para puertos series reales

Consejos sobre la línea de comandos

- El uso de la línea de comandos es obligatorio para muchas operaciones necesarias para el desarrollo de Linux
- Es una manera muy poderosa de interactuar con el sistema, con la cual usted puede ahorrar mucho tiempo.
- Algunos consejos útiles
  - Puede utilizar varias pestañas en el Gnome Terminal
  - Recuerde que puede utilizar rutas de acceso relativas (por ejemplo: `../../ linux`) además de rutas absolutas (por ejemplo: `/home/user`)
  - En un shell, pulse `[Control] [r]` y, a continuación, una palabra clave, esto buscará a través del historial de comandos. Presione `[Control] [r]` de nuevo para buscar hacia atrás en la historia
  - Puede copiar y pegar rutas directamente desde el el terminal con arrastrar y soltar.

Preparar el entorno del laboratorio

- Descargue el archivo del laboratorio
- Aplique permisos correctos