CIAA NXP

Alcances y limitaciones de un port basado en Buildroot/Linux

Ezequiel García

VanguardiaSur

12 de agosto de 2015

Rompiendo el hielo

Ezequiel García

- **Programador**
  - Sistemas embebidos Linux
  - Desarrollo device drivers Linux kernel
- **Contribuciones open-source**
  - Maintainer driver pxa3xx-nand (NAND)
  - Maintainer driver stk1160 (video4linux)
  - Soporte de SoCs ARM Marvell mvebu
  - Soporte de SoCs MIPS Pistachio
  - UBI block device
  - *strace* para arquitectura Nios-II
  - ... y algunas contribuciones a Buildroot y Barebox

Ezequiel García

- **Programador**
  - Sistemas embebidos Linux
  - Desarrollo device drivers Linux kernel
- **Contribuciones open-source**
  - Maintainer driver pxa3xx-nand (NAND)
  - Maintainer driver stk1160 (video4linux)
  - Soporte de SoCs ARM Marvell mvebu
  - Soporte de SoCs MIPS Pistachio
  - UBI block device
  - *strace* para arquitectura Nios-II
  - ... y algunas contribuciones a Buildroot y Barebox
  - **Responsable CIAA NXP Linux**

poll()

Levanten la mano los estudiantes

poll()

Levanten la mano los que usan Windows en forma regular

poll()

Levanten la mano los que usan Linux en forma regular

¿De qué vamos a hablar?

Para los que vieron luz y entraron

1.

Componentes del port Linux para la CIAA NXP

Para los que vieron luz y entraron

2.

Alcances.

Es decir, todas las cosas geniales que podemos hacer.

Para los que vieron luz y entraron

3.

Limitaciones.

Es decir, todas las cosas geniales que **NO** podemos hacer.

Empezamos con algunas definiciones

Linux

Todos sabemos exactamente qué es Linux...

Linux

Todos sabemos exactamente qué es Linux...**¿o no?**

Ecosistema Embedded Linux

Todos los sistemas embebidos basados en **Linux** tienen más o menos los mismos componentes fundamentales

- **Bootloader**
- **Linux Kernel**
- **Filesystem**

Ecosistema Embedded Linux

Todos los sistemas embebidos basados en **Linux** tienen más o menos los mismos componentes fundamentales

- **Bootloader**
- **Linux Kernel**
- **Filesystem**
  ¿Eso es todo? ¿No nos falta algo?

Ecosistema Embedded Linux

Todos los sistemas embebidos basados en **Linux** tienen más o menos los mismos componentes fundamentales

- **Bootloader**
- **Linux Kernel**
- **Filesystem**
  ¿Eso es todo? ¿No nos falta algo?
- **Toolchain**

Ecosistema Embedded Linux

Todos los sistemas embebidos basados en **Linux** tienen más o menos los mismos componentes fundamentales

- **Bootloader**
- **Linux Kernel**
- **Filesystem**
  ¿Eso es todo? ¿No nos falta algo?
- **Toolchain**

Cada uno de estos componentes es desarrollado por un grupo de personas diferentes, y en forma más o menos independiente.

¿Qué hace cada componente?

Bootloader

El objetivo del bootloader es simple: encontrar y cargar un kernel.

Bootloader

El objetivo del bootloader es simple: encontrar y cargar un kernel. Una vez que se carga el kernel, el bootloader **no permanece residente en memoria**.

Kernel

El objetivo del kernel es bastante menos simple: manejar el hardware y administrar el acceso a los recursos de CPU, memoria y periféricos.

Kernel

El objetivo del kernel es bastante menos simple: manejar el hardware y administrar el acceso a los recursos de CPU, memoria y periféricos.
Permanece residente en memoria durante toda la vida del sistema.

Filesystem

El Filesystem tiene todos los programas y aplicaciones de usuario necesarios para que el sistema haga algo útil.

`/lib/modules/x.y.z/`
`/usr/bin/uptime`
`/usr/bin/find`
`/bin/cat`
`/bin/cp`
`/bin/sh`
`/etc/init.d/...`

Filesystem

Vemos que un filesystem se compone de quichicientos paquetes diferentes. Necesitamos una herramienta para:

- seleccionar
- configurar
- compilar
- instalar

Filesystem

Busybox: La navaja suiza de los embebidos Linux.
Provee más de 300 herramientas típicas de Unix,
en un único binario. `https://en.wikipedia.org/wiki/BusyBox`

¿De dónde obtenemos cada componente?

Ecosistema Embedded Linux

Cada componente que usamos en un proyecto o producto tiene una fuente u origen.

- **Bootloader**
  - Oficial o upstream
  - Community fork
  - Vendor fork

Ecosistema Embedded Linux

Cada componente que usamos en un proyecto o producto tiene una fuente u origen.

- **Bootloader**
  - Oficial o upstream
  - Community fork
  - Vendor fork
- **Linux Kernel**
  - Oficial o upstream
  - Community fork
  - Vendor fork

Ecosistema Embedded Linux

Cada componente que usamos en un proyecto o producto tiene una fuente u origen.

- **Bootloader**
  - Oficial o upstream
  - Community fork
  - Vendor fork
- **Linux Kernel**
  - Oficial o upstream
  - Community fork
  - Vendor fork
- **Filesystem**
  - Buildroot
  - Yocto
  - Debian, Fedora, etc.

Ecosistema Embedded Linux

Cada componente que usamos en un proyecto o producto tiene una fuente u origen.

- **Bootloader**
  - Oficial o upstream
  - Community fork
  - Vendor fork
- **Linux Kernel**
  - Oficial o upstream
  - Community fork
  - Vendor fork
- **Filesystem**
  - Buildroot
  - Yocto
  - Debian, Fedora, etc.
  - **Hecho a mano** (aunque espero que nadie lo haga)

Las elecciones para nuestra querida
CIAA NXP

Ecosistema Embedded Linux

Cada componente que usamos en un proyecto o producto tiene una fuente u origen.

- **Bootloader**
  - Oficial o upstream
  - Community fork
  - Vendor fork ⟵
- **Linux Kernel**
  - Oficial o upstream ⟵
  - Community fork
  - Vendor fork
- **Filesystem**
  - Buildroot ⟵
  - Yocto
  - Debian, Fedora, etc.
  - Hecho a mano

U-Boot

Usamos un "vendor fork"que ofrece una empresa llamada *Emcraft*.
El repositorio está disponible en github:
`https://github.com/EmcraftSystems/u-boot`

Linux

Usamos Linux upstream, con algunos parches encima.

Buildroot

Usamos Buildroot upstream, con algunos parches encima.

Toolchain

Usamos una toolchain especial para cortex-M, OSELAS de Pengutronix.
`http://www.pengutronix.de/oselas/toolchain/`

uCLinux

uCLinux nació como un fork del kernel. Breve historia:

- 1991: Nace Linux.
- 1998: Release de un kernel basado en Linux 2.0.33, para Motorola DragonBall.
- 2002: Se integra al Linux oficial en la versión v2.5.46.

uCLinux

uCLinux nació como un fork del kernel. Breve historia:

- 1991: Nace Linux.
- 1998: Release de un kernel basado en Linux 2.0.33, para Motorola DragonBall.
- 2002: Se integra al Linux oficial en la versión v2.5.46.
- 2015: ¡Hace más de una década que Linux soporta plataformas sin MMU!

uCLinux

Como vemos esto pasó hace mucho tiempo.

uCLinux

Como vemos esto pasó hace mucho tiempo.
Esto pasó hace tanto tiempo, ¡que no se había inventado **git**!

uCLinux

Actualmente, uCLinux es una distribución de paquetes fuente, orientada a sistemas sin MMU.
El proyecto no tiene demasiada popularidad, aunque sigue activo.

Repasando

Ecosistema Embedded Linux

La CIAA NXP tiene los mismos componentes fundamentales que cualquier otro sistema embebido.

- **Bootloader**
- **Linux Kernel**
- **Filesystem**

Una ventaja a medias

Usar el mismo código base que se usa en el resto de los sistemas embebidos, significa que la cantidad de usuarios y desarrolladores es enorme.

Una ventaja a medias

Usar el mismo código base que se usa en el resto de los sistemas embebidos, significa que la cantidad de usuarios y desarrolladores es enorme.
Usar el mismo código base que se usa en el resto de los sistemas embebidos, **no** necesariamente implica que esté muy **testeado**.

Ecosistema Embedded Linux

Linux v4.3 soportará varios periféricos de la CIAA:

- GPIO
- UART
- Ethernet
- USB
- RTC
- SPI/SSP
- SPIFI
- ...

Ecosistema Embedded Linux

Y la lista sigue:

- ...
- DMA
- I2C
- Watchdog
- PWM basado en SCT

Para más detalles consultar:
`https://github.com/manabian/linux-lpc/wiki/Status`

Requisitos mínimos

Requisitos mínimos

¿Qué se necesita para que funcione Linux?

Requisitos mínimos

¿Qué se necesita para que funcione Linux?

- **CPU**

Requisitos mínimos

¿Qué se necesita para que funcione Linux?

- **CPU**
- **RAM**

Requisitos mínimos

¿Qué se necesita para que funcione Linux?

- **CPU**
- **RAM**
- **Algunos periféricos básicos**

RAM

Necesitamos 4 MiB como mínimo.

RAM

Necesitamos 4 MiB como mínimo.
Necesitamos 8 o 16 MiB para hacer algo útil.

MMU

No hace falta.

MMU

No hace falta.
¿Magia negra?

MMU

No hace falta.
~~¿Magia negra?~~ **Portabilidad**

MMU

¿Qué es la MMU?.

- Conversión entre direcciones virtuales y direcciones físicas. Esto permite que cada proceso corra en su propio espacio de direcciones, en forma aislada al resto del sistema.
- Protección ante accesos a direcciones no autorizadas.

Portabilidad (la magia negra)

```
/* Un poco simplificado. */
#ifdef CONFIG_MMU
unsigned long copy_from_user(...);
unsigned long copy_to_user(...);
unsigned long clear_user(...);
#else
#define copy_from_user(to,from,n)      (memcpy(to, from, n), 0)
#define copy_to_user(to,from,n)        (memcpy(to, from, n), 0)
#define clear_user(addr,n)             (memset(addr, 0, n), 0)
#endif
```

Portabilidad (la magia negra)

```
/* En arch/arm/mm/Makefile */
ifneq ($(CONFIG_MMU),y)
obj-y                                   += nommu.o
endif

/* En arch/arm/mm/nommu.c */
void *__arm_ioremap(phys_addr_t phys_addr, size_t
    size, unsigned int mtype)
{
        return phys_addr;
}
```

Limitaciones de Linux en la CIAA NXP

Limitaciones: MMU

No hay MMU.

Limitaciones: Stack fijo

La MMU es necesaria para la implementación de un stack variable.
Sin MMU, los procesos no tienen stack variable, sino estático.

Limitaciones: threads

¿Tenemos alguna implementación de POSIX threads?

Limitaciones: threads

¿Tenemos alguna implementación de POSIX threads?
No en forma oficial, pero hay algunas implementaciones extra-oficiales (out-of-tree).

Limitaciones: fork()

Sin MMU no podemos implementar fork().

Limitaciones: fork()

Sin MMU no podemos implementar fork().
Para ejecutar procesos, podemos reemplazar fork() por vfork().

Limitaciones: referencias y bibliografías

Para más detalles acerca de estas limitaciones, hay extensa bibliografía al respecto.

- http://www.linuxjournal.com/article/7221
- http://free-electrons.com/doc/uclinux_introduction.pdf
- http://events.linuxfoundation.org/sites/events/files/slides/optimize-uclinux.pdf
- http://events.linuxfoundation.org/sites/events/files/slides/uClinux%20ELC_43_small.pdf
- http://electronicdesign.com/embedded/practical-advice-running-uclinux-cortex-m3m4

Conclusión

Conclusión

1.

Componentes del port Linux para la CIAA NXP.

Se usan los mismos que en cualquier sistema embebido Linux

(con una configuración especial)

Conclusión

2.

Alcances.

Disponemos de una cantidad importante de features, drivers y utilidades.

Conclusión

2.

Alcances.

Disponemos de una cantidad importante de features, drivers y utilidades.

Tenemos soporte para casi todos los bloques de hardware del MCU NXP LPC4337.

Conclusión

2.

Alcances.

Disponemos de una cantidad importante de features, drivers y utilidades.

Tenemos soporte para casi todos los bloques de hardware del MCU NXP LPC4337.

No tenemos que aprender nuevas APIs, sino que podemos usar las mismas que en cualquier otro Linux.

Conclusión

3.

Limitaciones.

Es decir, todas las cosas geniales que **NO** podemos hacer.

- **No hay MMU**
- **Los procesos tienen un stack fijo**
- **No hay fork. Para lanzar un proceso se usa vfork + exec**
- **No hay pthreads**

¿Preguntas?

Ahora o en cualquier momento :-)

ciaa-linux@googlegroups.com

ezequiel@vanguardiasur.com.ar