

Herramientas de software para el diseño de sistemas embebidos

Lección 2

Prof.Ing. Jeferson González G.

CE-5303 Introducción a los Sistemas Embebidos

Área de Ingeniería en Computadores

Instituto Tecnológico de Costa Rica

1 Introducción

2 Desarrollo de Software para Sistemas Embebidos

- Metodologías de Desarrollo
- Cross-Toolchain
- Kit de Desarrollo de Software - SDK
- Construcción de SW para Sistemas Embebidos
- Simuladores

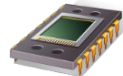
Desarrollo de SW para Sistemas Embebidos

El desarrollo de sistemas empotrados al igual que cualquier desarrollo de software necesitan compiladores, enlazadores, ambientes de desarrollo integrado (IDEs), entre otros.

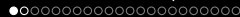
- Diferentes metodologías



Estación de trabajo
x86



Sistema empotrado
ARM



Desarrollo Cruzado

Herramientas disponibles para desarrollo son distintas según la metodología

- Existen herramientas de desarrollo de plataforma cruzada (cross-platform development tools) o de herramientas de desarrollo cruzado (cross-development tools)

Toolchain

Conjunto de programas informáticos (herramientas) que **se usan para crear un determinado producto** (normalmente otro programa o sistema informático).

- **Uso en cadena:** La salida de un programa es la entrada del siguiente.
- En estaciones de trabajo, bajo GNU/Linux, se utiliza toolchain **nativo** - típicamente para x86.
- Un toolchain muy simple incluiría un compilador y un enlazador (*linker*)

Cross-toolchain

- Para sistemas empotrados no es usual ni práctico utilizar toolchains nativos. - **¿Por qué?**

Cross-toolchain

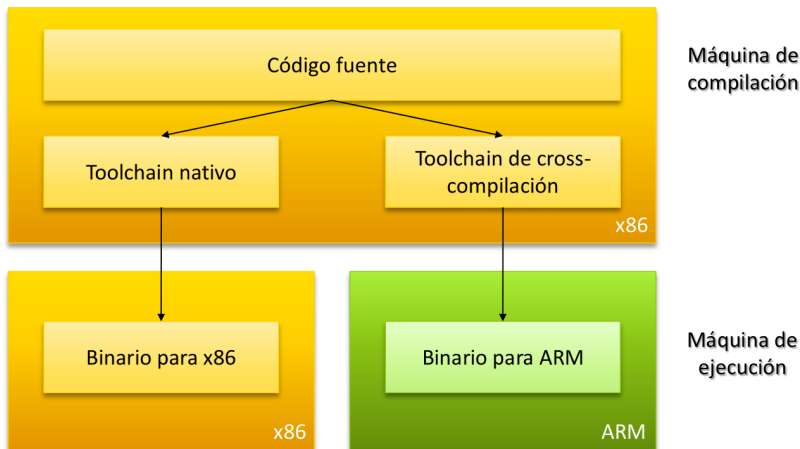
- Para sistemas empuotrados no es usual ni práctico utilizar toolchains nativos. - **¿Por qué?**
- El sistema embebido está restringido por aspectos de:
 - Memoria (RAM, almacenamiento)
 - Periféricos (Video, teclado, mouse, etc)
 - Dependencia de la arquitectura

Cross-toolchain

- Para sistemas empotrados no es usual ni práctico utilizar toolchains nativos. - **¿Por qué?**
- El sistema embebido está restringido por aspectos de:
 - Memoria (RAM, almacenamiento)
 - Periféricos (Video, teclado, mouse, etc)
 - Dependencia de la arquitectura

Solución: Uso de toolchain cruzado (cross-toolchain)

Estructura toolchain



GNU make

Herramienta de generación y/o automatización de código

- comando **make** determina que parte del código debe ser compilado
- Se realiza por medio de archivos **Makefiles**

Estructura de un Makefile

```
objetivo: prerequisites  
<tab><tab> comando(s)
```

Ejemplo Makefile sencillo

```
PHONY: limpiar
```

```
#Variables de aplicación
```

```
APP=holamundo
```

```
SRC=$(APP).c
```

```
OBJ=$(APP).o
```

```
#Variables de compilador
```

```
CC=gcc
```

```
CFLAGS = -c -Wall
```

```
LFLAGS = -Wall
```

```
$(APP): $(OBJ)
```

```
$(CC) $(CFLAGS) $(OBJ) -o $(APP)
```

```
$(OBJ): $(SRC)
```

```
$(CC) $(LFLAGS) $(SRC)
```

```
limpiar|:
```

```
rm $(APP) $(OBJ)
```

GNU Binutils

Herramientas de programación para la manipulación de código objeto.

Incluye: (entre otros)

- as - ensamblador
- ld - enlazador
- ar - crea, modifica y extrae desde archivos
- nm - lista de símbolos en archivos objeto
- ranlib - genera índice para archivos
- strings - lista las cadenas de caracteres imprimibles
- strip - elimina símbolos de archivos objeto
- gprof - perfilador

GNU Compiler Colection - GCC

- Conjunto de compiladores creados por el proyecto GNU, disponible bajo la licencia GPL
- Lenguajes soportados: Ada, ANSI C, C++, Fortran, Java y Objective-C/C++.
- Arquitecturas Soportadas: ARM, AVR, Blackfin, MIPS, Motorola, PowerPC, SPARC, SuperH, x86.

<https://gcc.gnu.org/install/specific.html>

Biblioteca de C - Clib

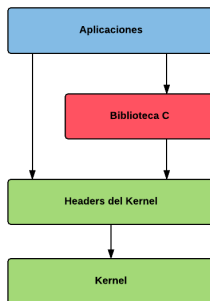
Funciona como interface entre las aplicaciones y el kernel.
Provee una API en C bien conocida para el fácil desarrollo de aplicaciones

- Diferentes bibliotecas: glibc, uClibc, eglibc, etc.
- La elección de la biblioteca se hace en el momento de generación del toolchain.

Header del Kernel

Biblioteca de C y programas compilados requieren interactuar con el kernel

- Llamadas al sistemas
- Estructuras de datos
- Definiciones de constantes



Construcción de Toolchain

Tres máquinas deben ser distinguidas cuando se discute la creación de un toolchain.

- ❶ La máquina de construcción, donde se construirá el toolchain.
- ❷ La máquina host, donde se ejecutará el toolchain.
- ❸ La máquina objetivo, donde los binarios creados por el toolchain serán ejecutados.

Pasos para la construcción del toolchain

- Extraer e instalar los headers del kernel
- Extraer, configurar, compilar e instalar los binutils
- Extraer, configurar y compilar una primera versión de GCC que genere binarios para la máquina objetivo. Será empleado para cross-compilar la biblioteca de C.
- Extraer, configurar, compilar la biblioteca de C.
- Reconfigurar y compilar la versión final del cross-compilador GCC.

Contrucción del Toolchain

Construir el toolchain *en casa*: proceso manual, óptimo.

- Obtener toolchain **precompilado**: Solución más común en desarrolladores. Simple. Se elige según:
 - CPU
 - Endianness
 - Tipo Clib
- Herramientas **automatizadas** a base de scripts, Makefiles, etc. Ej: Crosstool, OpenEmbedded, Yocto Project.

SDK

Conjunto de herramientas de desarrollo de software que le permite al programador crear aplicaciones para un sistema concreto, por ejemplo ciertos paquetes de software, frameworks, plataformas de hardware, computadoras, videoconsolas, sistemas operativos, etc.

- RidgeRun SDK: <https://www.ridgerun.com/www/index.php/download-center.html>
- Kinect SDK
- Jetpack SDK (Nvidia Jetson TX1/TX2)
- Android SDK

Estructura de un SDK

Directorio	Contenido
<i>bootloader</i>	El bootloader para el sistema empujado de interés
<i>build-tools</i>	Los paquetes y directorios necesitados para construir el toolchain
<i>debug</i>	Herramientas de depuración y paquetes asociados
<i>doc</i>	Documentación del proyecto
<i>Images</i>	Imágenes binarias del bootloader, kernel, filesystem, listas para utilizar en el hardware
<i>kernel</i>	Diferentes versiones del kernel para el hardware de interés
<i>project</i>	Archivos de configuración y preferencias del sistema
<i>roofs</i>	Sistema de archivos
<i>sysapp</i>	Aplicaciones requeridas para su sistema empujado
<i>tmp</i>	Directorio temporal para experimentos
<i>tools</i>	Toolchain y Clib

Construcción de SW para Sistemas Embebidos

Una vez construido el toolchain, se debe construir el resto de elementos de software del sistema:

- Aplicaciones y bibliotecas
- Scripts
- Kernel
- Sistemas de Archivos

Cross-compilación

Una vez que el componente es elegido, debe ser compilado para el sistema empotrado.

- Herramientas de compilación son distintas a las del sistema (compilador, ensamblador, enlazador, etc).
- Los archivos no son instalados en /usr, /usr/lib sino que en un directorio diferente (incluso virtual/imagen).

Cross-compilación: Ejemplo

```
IDIR=/home/jgonzalez/simplescalar/sslittle-na-sstrix/bin
```

```
CC = $(IDIR)/gcc
```

```
CFLAGS = -O -I/home/jgonzalez/simplescalar/sslittle-na-sstrix/include/  
          -L/home/jgonzalez/simplescalar/sslittle-na-sstrix/bin/
```

```
RANLIB= $(IDIR)/ranlib
```

```
MAKELIB=$(IDIR)/ar rcv
```

```
pisa:
```

```
    -mkdir lib
```

```
    cd src ; $(MAKE) -f Makefile $@
```

```
    cd src-glu ; $(MAKE) -f Makefile $@
```

```
    cd src-aux ; $(MAKE) -f Makefile $@
```

```
    cd demos_pisa; $(MAKE) -f Makefile $@
```

```
|
```

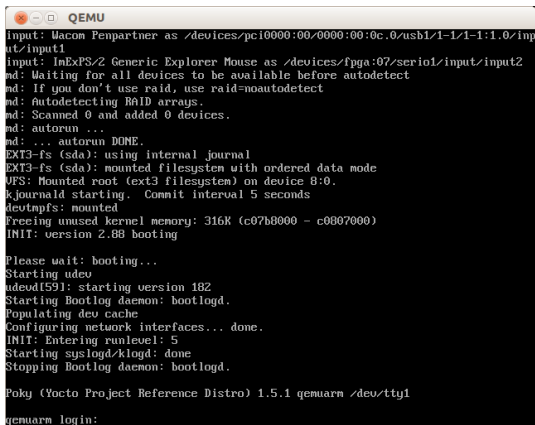
Simulador

Es un programa que se ejecuta en una estación de trabajo (host), el cual simula la funcionalidad y el set de instrucciones de un procesador de un sistema emputrado.



Simulador: QEMU

Simula una amplia variedad de procesadores y periféricos: x86, x86-64, PowerPC, ARM, SPARC, MIPS, M68k, etc.



```
QEMU
input: Wacom Penpartner as /devices/pci0000:00/0000:00:0c.0/usb1/1-1/1-1:1.0/inp
ut/input1
input: InExPS/2 Generic Explorer Mouse as /devices/fpga:07/serio1/input/input2
md: Waiting for all devices to be available before autodetect
md: If you don't use raid, use raid=noautodetect
md: Autodetecting RAID arrays.
md: Scanned 0 and added 0 devices.
md: autorun ...
md: ... autorun DONE.
EXT3-fs (sda): using internal journal
EXT3-fs (sda): mounted filesystem with ordered data mode
UFS: Mounted root (ext3 filesystem) on device 8:0.
kjournal starting. Commit interval 5 seconds
devtmpfs: mounted
Freeing unused kernel memory: 316K (c07b8000 - c0807000)
INIT: version 2.88 booting

Please wait: booting...
Starting udev
udevd[591]: starting version 182
Starting Bootlog daemon: bootlogd.
Populating dev cache
Configuring network interfaces... done.
INIT: Entering runlevel: 5
Starting syslogd/klogd: done
Stopping Bootlog daemon: bootlogd.

Poky (Yocto Project Reference Distro) 1.5.1 qemuarm /dev/tty1
qemuarm login:
```

Referencias



Wayne Wolf (2010)

High Performance Embedded Computing



Miguel Angel Aguilar U. (2009)

Material de clase: Introducción a los Sistemas Embebidos.



María Haydeé Rodríguez B. (2014)

Material de clase: Sistemas Empotrados



GNU Compiler Collection

<https://gcc.gnu.org/>