Práctica III

Linux Kernel

Implementación de Sistemas Operativos II

Maestría en Sistemas Embebidos

Año 2022

Autor

Mg. Ing. Gonzalo Sanchez

Tabla de contenido

Registro de cambios

Revisión	Cambios realizados	Fecha
1.0	Creación del documento	14/09/2021
1.1	Agregado de sección Puesta a punto de hardware Agregado de sección Cargar e iniciar el Kernel mediante Uboot	15/09/21
1.2	Actualización de fechas y version de Kernel	24/09/22

Kernel de Linux

Una vez que se tiene a disposición un bootloader compilado y funcionando en la SBC seleccionada, el paso siguiente para construir un sistema funcional es compilar el kernel de linux para poder cargarlo en la mencionada SBC.

Obtención de fuentes, configuración y compilación

Existen dos maneras de trabajar con las fuentes del kernel de Linux, la primera es descargando el árbol completo de fuentes, cuyo administrador es el propio Linus Torvalds. La segunda es trabajar simplemente con la rama de versiones estables. Para hacer un despliegue de productos, siempre es recomendable trabajar con una versión estable, por lo que si se tiene el árbol completo, además se debe agregar la rama de versiones estables.

Dado que en esta materia no es necesario hacer cambios en las fuentes del kernel, ni tampoco seguir el desarrollo o los *release candidates*, se recomienda trabajar directamente con la rama de versiones estables.

En esta práctica, dentro del directorio **ISO_II** se debe crear un directorio llamado **linux_kernel**. Luego de crearlo, se debe ingresar al mismo.

```
$ mkdir -p -v $HOME/ISO_II/linux_kernel

$ cd $HOME/ISO_II/linux_kernel
```

Una vez dentro del directorio de trabajo, es necesario clonar la rama de versiones estables del kernel de Linux.

```
$ git clone
git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-
stable.git
```

Habiendo clonado el repositorio, se debe ingresar al directorio generado el cual tendrá el nombre **linux-stable**. Si se desea que el repositorio se clone directamente en el

directorio actual, en este caso **linux_kernel**, recuerde agregar un punto al final del comando, como argumento separado, indicando donde se deben copiar los contenidos del repositorio indicado. Luego de clonar el repositorio, es necesario efectuar un checkout a la versión estable **5.10**. Esto se hace de la misma forma que en práctica anterior, eligiendo el tag correspondiente.

```
$ git checkout linux-5.10.y
```

Es de notar que la última parte del tag es la letra "*y*". Esto se debe a que la versión estable recibe actualizaciones de seguridad por algunos años, por lo que se incrementa el número de versión en este último campo. Es bueno siempre hacer un *pull* (fetch + merge) para tener disponible la última versión actualizada.

Como en el caso de Uboot, para poder compilar los archivos fuentes, se debe primero definir dos variables de entorno, las cuales son **CROSS_COMPILE** y **ARCH**. Recuerde que *estas variables de entorno solo son válidas para este proyecto, dado que están definidas en su makefile*, pueden no existir en otros proyectos o bien tener nombres o significados diferentes.

Estas variables de entorno se definen antes de invocar a make, y lo recomendable es definirlas dentro del bash que se está utilizando mediante el comando *export*.

```
$ export CROSS_COMPILE=arm-linux-
```

```
$ export ARCH=arm
```

Es necesario aclarar que el prefijo del compilador debe contener el guión medio final dado que lo único que se le adosa es el *gcc* al final para invocar al compilador. Asimismo, el PATH en donde está almacenado este cross compilador debe ser conocido para el sistema, caso contrario el comando make finalizará con un error por no encontrar el compilador.

Antes de compilar el proyecto, debe ser configurado para el SoC en cuestión. De la misma manera que Uboot, el kernel de linux posee configuraciones mínimas por defecto

para algunos hardwares, los que se dice están mantenidos en el mainline. Para poder ver todos los soportados se debe navegar hasta la carpeta **/arch/arm**. En el caso de la BeagleBone Black, la configuración mínima se da para el SoC, por lo que el archivo es **omap2plus_defconfig**.

```
$ cd
```

Para poder navegar en esta configuración y ver detalles con la posibilidad de modificarla, es posible ejecutar *make menuconfig*. Se invita al lector a investigar las configuraciones que se hacen sobre la compilación y modificar alguna si lo cree conveniente.

Finalmente, para compilar el proyecto se ejecuta simplemente *make*. En el directorio de trabajo se generan dos archivos que son necesarios, **zImage** y **am335x-boneblack.dtb.** El archivo *zImage* es el propio Kernel compilado y comprimido, mientras que el archivo *am335x-boneblack.dtb* es el device tree blob, necesario para que el kernel pueda "ver" cierto tipo de hardware que de otra forma no puede ser descubierto.

Puesta a punto de hardware

Configuración de la conexión de red

Para que la carga del kernel y del correspondiente Device Tree sea más sencilla, es conveniente configurar un servidor NFS para pasar estos archivos por red a la SBC. Esta comunicación y transferencia de archivos se hace mediante el protocolo TFTP, y facilita también el montaje de un FileSystem, que es un elemento necesario para cualquier sistema funcional.

Lo primero que se debe configurar para que esta comunicación funcione son las direcciones IP tanto del servidor como de la SBC. Esta configuración se efectúa en Uboot mediante comandos específicos para que la SBC sepa dónde tiene que solicitar los archivos y eventualmente un FileSystem.

```
=> setenv ipaddr 192.168.0.100

=> setenv serverip 192.168.0.1
```

Las direcciones mostradas son a modo de ejemplo, es claro que ambas direcciones deben estar dentro del mismo segmento de red o subred para que la comunicación sea posible. Además, estas direcciones son del tipo estáticas, con lo que la resolución DHCP debe estar deshabilitada para la interfaz a utilizar, o bien puede configurar un static lease por medio de su servidor DHCP para que la SCB tenga siempre el mismo IP.

Instalación y configuración de servidor TFTP

Una vez que se configura la conexión de red, lo siguiente es instalar el servidor TFTP que se utiliza para enviar y recibir archivos desde la SBC. Es necesario instalar el paquete **tftpd-hpa** para esto.

```
$ sudo apt install tftpd-hpa
```

Al instalar este paquete, se crea un directorio con una ruta específica, el cual es el que termina sirviendose en la dirección IP especificada. Este directorio se especifica en el archivo */etc/default/tftpd-hpa*, mediante la variable **TFTP_DIRECTORY**.

Lo usual es que el directorio que se comparte mediante el servidor sea **/var/lib/tftpboot/** pero se recomienda chequear la variable de configuración anterior para no tener errores de comunicación. También puede definir su propio directorio como compartido, entendiendo que el mencionado directorio compartido necesita acceso root para escritura. En el caso que se quiera hacer un cambio al archivo */etc/default/tftpd-hpa*, es necesario que se reinicie el servicio asociado al servidor, mediante el comando siguiente.

```
$ sudo /etc/init.d/tftpd-hpa restart
```

Para corroborar que la conexión TFTP funciona de manera correcta, se coloca en el directorio compartido un archivo de texto plano simple, el cual se puede cargar desde Uboot. Mediante la consola de Uboot, ejecute:

```
=> tftp 0x81000000 archivo.txt
```

Este comando hace que se descargue el archivo mencionado desde la estación de trabajo a la dirección de memoria 0x81000000 de la SBC. Si la transacción fue exitosa, entonces el comando memory-dump en la dirección de RAM mencionada tiene como salida el contenido del archivo de texto.

```
=> md 0x81000000
```

No hay diferencia alguna entre un archivo de texto y el kernel de linux, por lo tanto la configuración de la red está lista para cargar el kernel y el device tree correspondiente.

Cargar e iniciar el Kernel mediante Uboot

Ya configurado todo el sistema, para poder cargar el kernel en RAM de la SBC, es necesario copiar los archivos **zImage** y **am335x-boneblack.dtb** en el directorio compartido mediante el servidor TFTP, y luego utilizando comandos en la consola de Uboot, cargar los mismos a direcciones específicas de RAM.

En realidad, las direcciones de RAM podrían diferir, siempre y cuando se tenga cuidado de estar en la región de direcciones de RAM válidas.

Los próximos comandos deben ser ejecutados en Uboot, los cuales cargan el kernel comprimido y el device tree en distintas direcciones de RAM y luego se indica que se debe iniciar el kernel y el device tree cargados sin utilizar **initramfs**.

```
=> tftp 0x81000000 zImage

=> tftp 0x82000000 am335x-boneblack.dtb

=> bootz 0x81000000 - 0x82000000
```

Si no se tiene salida de consola mostrando que el kernel está iniciando, se debe reiniciar el sistema y ejecutar el siguiente comando antes de tratar de bootear nuevamente:

```
=> setenv bootargs console=ttyS0,115200n8
```

El comando mostrado configura la salida de consola al dispositivo /ttyS0 a 115.200 baudios sin paridad y un ancho de 8 bits.

NOTA: *El kernel intentara iniciar, pero al no encontrar un filesystem, mostrará un mensaje de kernel panic y detendrá su ejecución.*