Práctica IV

BusyBox

Implementación de Sistemas Operativos II

Maestría en Sistemas Embebidos

Año 2021

Autor

Mg. Ing. Gonzalo Sanchez

Tabla de contenido

Registro de cambios

Revisión	Cambios realizados	Fecha
1.0	Creación del documento	27/09/2021
1.1	Corrección de etiquetas en sección “Configuración de servidor NFS”	06/10/2021

BusyBox

Para que el sistema tenga funcionalidad, hay que agregar programas que corran sobre el sistema operativo, y aquí es donde BusyBox interviene. Provee al constructor del sistema un abanico altamente configurable de aplicaciones para el usuario final, algunas necesarias para la configuración del sistema, otras para agregar utilidades.

Obtención de fuentes, configuración y compilación

Como en todos los casos anteriores, la obtención de las fuentes de busybox se hará desde la página oficial, mediante un repositorio git.

El acceso remoto y anónimo está dado por medio del repositorio oficial, pero el mismo al ser anónimo no permite que se hagan aportes al proyecto.

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

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

```
$ cd $HOME/ISO_II/busybox
```

Una vez dentro del directorio de trabajo, es necesario clonar el repositorio correspondiente:

```
$ git clone git://busybox.net/busybox.git .
```

Recuerde que es posible agregar la ruta donde clonar las fuentes como argumento separado luego de la dirección del repositorio. Un punto al final del comando indica que se deben copiar los contenidos del repositorio en la ruta actual. Luego de clonar el repositorio, es necesario efectuar un checkout a la última versión estable, en este caso es la rama **1_33_stable**.

```
$ git checkout 1_33_stable
```

Si desea puede hacer un pull sobre la rama, aunque esto no es necesario si acaba de clonar el repositorio (git clone siempre clona el estado más reciente del repositorio).

Como en los casos anteriores, 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.

A diferencia de las prácticas anteriores, busybox ya tiene una configuración inicial, lo que hace innecesaria la carga de una plantilla de configuraciones por defecto.

Para poder navegar la 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.

La compilación se puede hacer tanto de manera estática (generando un binario estilo stand-alone) o utilizando bibliotecas compartidas, enlazadas de manera dinámica. Se invita al lector a buscar la opción de compilación de manera estática siendo la más sencilla.

Instalación de busybox en directorio a servir

Habiendo compilado BusyBox, es necesario crear un directorio el cual servirá de root filesystem, donde se indica a BusyBox que se instale. Este directorio se denominará **nfsroot**.

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

Para poder hacer el install correctamente de BusyBox, es necesario configurar el *make install behavior*. Esta opción está dentro del menú *Settings* y se debe seleccionar la ruta absoluta como destino de todos los archivos.

Una vez hecho esto, se puede correr el comando *make install* y se copiaran todos los archivos correspondientes al directorio indicado.

```
$ make install
```

Inicio de sistema: Tiny-System

Configuración del servidor NFS

La modificación de cualquier archivo dentro del filesystem será más sencilla utilizando un servidor NFS y montando mencionado filesystem en la SBC.

Lo primero e indispensable para que el servidor NFS funcione es instalar el paquete **nfs-kernel-server**:

```
$ sudo apt install nfs-kernel-server
```

Luego de la instalación, es necesario configurar la ruta que se está sirviendo como filesystem mediante el archivo **/etc/exports**. Para poder modificar este archivo, se debe acceder con permisos *root*. Se debe agregar en el archivo una línea similar a la que se muestra a continuación:

```
/home/<user>/ISO_II/nfsroot
<IP_SBC>(rw,no_root_squash,no_subtree_check)
```

NOTA: *El campo <user> debe ser modificado con el nombre de usuario que está utilizando. El campo <IP_SBC> debe ser modificado con el IP de la propia SBC. Las opciones que van entre paréntesis deben estar a continuación de la dirección de IP sin ningún espacio. Todo debe estar en una única línea (i.e. sin saltos de línea). El directorio* ***nfsroot*** *debe existir, por lo que si no existe debe ser creado mediante mkdir.*

Para verificar que lo configurado es correcto, se hace un reinicio del servicio y se verifican errores.

```
$ sudo /etc/init.d/nfs-kernel-server restart
```

Configuración de argumentos de kernel - bootargs

Funcionando el servidor NFS, en la consola de Uboot es necesario agregar algunas opciones a la variable de ambiente **bootargs**.

```
=> setenv bootargs root=/dev/nfs rw ip=<IP_SBC>
console=ttyS0,115200n8
nfsroot=<IP_SERVER>:/home/<USER>/ISO_II/nfsroot,nfsvers=3
```

Las nuevas opciones indican al kernel que se utilizara un filesystem del tipo NFS (el kernel debe estar configurado con la opción de soporte habilitada), que será de lectura/escritura, la dirección IP de la SBC, y la ruta de donde se obtiene el root filesystem, que es el directorio servido por el servidor NFS.

NOTA: *La última opción de la ruta nfsroot es la versión de NFS a utilizar, y dependerá de la versión instalada en la workstation y de las versiones soportadas por la SBC (según el kernel). Si la opción* ***nfsvers=3*** *no funciona, es necesario intentar con el valor "2" o "4".*

Iniciar el sistema

Ya configurado todo el sistema, con el servidor NFS funcionando y las configuraciones de red necesarias, se carga e inicia el kernel como en la práctica anterior. Gracias a que BusyBox provee algunos de los directorios necesarios definidos para el root filesystem,

el error de montaje del mismo será distinto, muy probablemente indicando que no puede ser hallado el directorio “xxxxx”. Si esto no ocurre tiene un sistema semi-completo. No es funcional porque faltan algunos directorios donde se montan los pseudo-filesystems.

Se puede verificar el funcionamiento del comando **ps**, observando la salida en consola. El error mostrado es de fácil solución, se debe crear el directorio ***/proc***.

Para que el sistema sea mínimo funcional es necesaria la creación de algunos directorios más, en particular el directorio ***/etc***.

Una vez creado ***/etc***, el programa ***/sbin/init*** busca dentro del directorio el archivo **inittab**. Se puede obtener un archivo ejemplo de inittab (llamado de la misma manera) en el directorio examples de BusyBox (i.e. *$HOME/ISO_II/busybox/examples*).

También es necesario un archivo script **rcS** dentro del directorio ***/etc/init.d***, pero a este momento ninguno existe, por lo que hay que crearlos. El archivo **rcS** debe contener al menos dos líneas mínimamente y estas corresponden a:

1. Montaje del filesystem /proc.
2. Montaje del filesystem /sys.

Ambas tareas se dejan al lector para investigar y aplicar.