

DESARROLLO DE LA FONERA 1 Y FONERA 2

1. Conociendo la arquitectura.

Conocimientos iniciales.

Sin necesidad de investigación, sospechábamos que el procesador era de bajo consumo de arquitectura RISC. Lo más cercano y usual en este tipo de dispositivos de bajo consumo es un procesador de tipo ARM. En telefonía móvil es lo único que se utiliza (exceptuando algunos modelos con gpu específica de tipo POWERVR-variante del ARM).

1.1 Especificaciones fonera 2.0

Actividad	Fonera 2.0	Fonera 2.0n
Alcance Wifi	20-50m	40-200m
WiFi a WiFi	1MB/s	4MB/s
WiFi a Internet	800-900KB/s	8MB/s
WiFi a HDD*	1MB/s	8MB/s
Descargas torrent	hasta 300 KB/s	hasta 800 KB/s
Samba por WiFi	1MB/s	8MB/s
Descarga y subida simultánea	NO	SI

Componente	Fonera 2.0	Fonera 2.0n
CPU	Atheros 180MHz	Ralink 300MHz
RAM	32MB	64MB
Flash	8MB	8MB
Puertos Ethernet	1+1	1+4
USB	NEC	Ralink
WiFi	Atheros 802.11g	Ralink 802.11n

Comprobamos que el chip Atheros no es ARM para nuestra sorpresa, es una variante de tipo MIPS e investigando la tendencia en routers es utilizar MIPS es sus CPU mono o doble-núcleo.

2. Desarrollo para la misma

2.1 Problemas:

- MIPS no es x86-x64
- Compiladores distintos.
- No posee compilador propio (sí editor)
- Escritura o acceso al router debido a que tiene una lista de comandos limitado.
- El montaje no ha podido realizarse en un primer momento ya que intentamos montarlo de varias formas y no conseguimos nada.

Los tres primeros problemas sencillos son de "fácil" solución, pasa por recompilar gcc con las flags para procesadores MIPS o utilizar uno ya compilado y preparado que está incluido en el SDK para la fonera, para posteriormente subir el binario al dispositivo.

2.2 Compilación en atheros

El SDK no se compila, sino que lo bajamos ya precompilado, es decir, con los ejecutables incluidos. De esta forma nos quitamos los problemas con Ubuntu u otra distro Linux.

Empezamos:

a) Descargamos esto: <http://downloads.openwrt.org/kamikaze/7.09/atheros-2.6/OpenWrt-SDK-atheros-2.6-for-Linux-i686.tar.bz2>

b) Descomprimos con: ``tar xjvf OpenWrt-SDK-atheros-2.6-for-Linux-i686.tar.bz2``
Sobra decir que se necesitan permisos en el directorio.

c) Ya está instalado. Ahora a compilar:
``cd OpenWrt-SDK-atheros-2.6-for-Linux-i686/staging_dir_mips/bin``
`./mips-linux-uclibc-gcc ejemplo.c -o ejemplo`

**Si queremos evitarnos el acceder todo el rato al mismo directorio exportamos el PATH (o editamos el .bashrc para que sea permanente) de forma:
`export PATH=$PATH:<ruta de los binarios (donde se encuentre el mips-linux-gcc)>`

3. Compilando paquetes de software

Antes de continuar debemos tener el build-essentials instalado así como las dependencias de netcat.

Paquete necesario de netcat:

Antes de todo deberemos descargar el paquete de netcat en:
<http://netcat.sourceforge.net/download.php> en el enlace:
<http://garr.dl.sourceforge.net/sourceforge/netcat/netcat-0.7.1.tar.gz> que ocupa 389 kb.

3.1. Estructura de las fuentes de los src

a) Descomprimos con: ``tar xvzf netcat-0.7.1.tar.gz`` y nos creará un nuevo directorio con el nombre de este paquete.

b) Nos desplazaremos dentro del directorio y dentro de este veremos el fichero ./configure, que es un programa script que sirve para crear un makefile que contenga información sobre el compilador, el target, máquinas destino y/o opciones de instalación, etc.

c) Para hacer posible una correcta compilación de netcat para la estructura MIPS.

- Exportaremos el bash con ``vim .bashrc``
- Introducimos en la última línea:

```
export PATH=<ruta de los binarios (donde se encuentre el mips-linux-gcc)>/  
export CC=mips-linux-uclibc-gcc-4.1.2
```

Guardamos los cambios y nos salimos.

d) Vamos al directorio donde tenemos descomprimido el paquete. Ejecutamos:
`./configure --host=mips-linux --prefix=</ruta donde quieras los binarios>`

*** Parámetros que influyen en el configure:

Si no es la primera vez que compilas pasamos de estos puntos.

--host: Establece la máquina destino.

--prefix: establece el destino del paquete de instalación (binarios) por defecto /usr/bin

--includedir: Incluye las cabeceras o librerías a parte de las propias del sistema que esta compilando (/usr/include).

--mandir: Fija la ruta de los archivos man.

--build o --target: También fijamos máquina destino.

Cada configure puede tener sus propias opciones.

3.2. Variables que influyen en la compilación

La variable CC se iguala al compilador de C.

CFLAGS: Opciones de compilación LDFLAGS: Opciones de linkado CPPFLAGS relativo al preprocesador de C.

Una vez configurado ya hemos generado el makefile, tan solo nos queda ejecutar *make && make install*

Si todo a ido bien y tenemos permisos en el directorio destino y para ejecutar make tras unos segundos accedemos a la carpeta que hemos nombrado en la opción --prefix y tendremos los binarios, manuales, etc. para nuestra máquina destino.

3.3. Subiendo el archivo

Necesitamos que el servidor de SSH este funcionando en el router y tener un cliente SSH en nuestro equipo. Se pueden utilizar otros métodos como wget o ftp desde el router, obviamos los métodos de conexión porque no es el tema.

Utilizamos el scp de la forma:

``scp <fichero> root@dd-wrt:/tmp`` que es la ruta donde tenemos permisos de escritura dentro del router, como ya se sabe esto está escrito de forma volátil en RAMdisk, por lo que con un reinicio se perdería. Para hacerlo permanente habría que escribirlo en el firmware.

4. Acceso al dispositivo

Solamente queda subir el ejecutable a la fonera. Es necesario tener instalado ssh sino no hacemos nada ya que el automontaje por alguna razón no funciona correctamente.

scp root@192.168.1.1:/tmp /<ruta al fichero>/programa

Passwd: *****

5. Ejecución del programa en el router.

a) Accedemos por ssh de la forma habitual: ssh root@192.168.1.1.

b) Nos dirigimos a tmp que es justo la carpeta anterior al directorio actual: cd ..

c) Ejecución: ./netcat.

Para la fonera 2.0

Para la fonera 2.0 el proceso es exactamente idéntico a excepción del paquete a descargar “*ftp://toolchains.x86.debian.sp1.tar.bz2*” y se descomprime de la misma forma. Se actualiza la ruta del PATH añadiendo la ruta del toolchains.x86 y la variable CC a mipsel-linux-uclibc-gcc.

- Exportaremos el bash con `vim .bashrc`
- Introducimos en la última línea:

```
export PATH=<ruta de los binarios (donde se encuentre el mipsel-linux-gcc)>/  
export CC=mipsel-linux-uclibc-gcc-4.1.0
```

El resto del proceso es idéntico.

FAQ

1. ¿Por qué no instalar los SDK?

Porque lleva el mismo trabajo al tener que actualizar la variable entorno CC.

2. ¿Por qué mi programa compilado para la fonera 1 no funciona para la fonera 2?

Porque la arquitectura del Atheros de la fonera 1 es BIGENDIAN y la del Ralink de la fonera 2 no.

3. El make no genera los ejecutables ¿Qué puedo hacer?

Asegurate de tener todas las dependencias además del build-essentials.

Asegurate que al ejecutar el configure revisa las variables de entorno y la opción host.

Por último prueba a incluir librerías con la opción --includedir a los propios includes de los SDK.

4. No puedo acceder al router por el comando ssh

Prueba alternativas como instalar un servidor apache en la máquina local y utilizar el comando wget desde el router.

5. Me da error al copiar el ejecutable en el router

Asegurate de tener permisos en la carpeta y que el sistema no sea de sólo lectura.

6. He hecho todo lo anterior y no he podido compilar

Definitivamente no todas las fuentes son compilables para todas las máquinas. Depende de la parte escrita en ensamblador propio. Hay rutinas no portadas, por lo tanto, si un programa depende de ese tipo de código sólo podrá ser ejecutada por su arquitectura.