NFS Root

Introducción

Este CÓMO trata de ejecutar su sistema Slackware Linux sin ningún disco duro, o quizás con un disco duro muy pequeño, accediendo a la red para recuperar todos los archivos excepto el kernel. Si desea ir por todo, también puede arrancar con PXE el kernel, sin embargo, este CÓMO espera que tenga un lugar local para almacenar el kernel. Vamos a utilizar máquinas virtuales (VirtualBox) para **simular** un cliente sin disco. Seguiremos usando una instalación completa de Linux junto con nuestro confiable LILO para preparar y luego arrancar ese kernel.Esto puede parecer un poco inútil, ya que no hay un caso de uso para exactamente lo que estoy haciendo aquí, pero al menos debería informar sobre cómo se obtiene la raíz NFS, y algunos de los escollos y soluciones alternativas.

Ejecutando un servidor NFS

Por supuesto, necesita un servidor NFS para servir los archivos raíz. Voy a llamar a esto **slack-nfs-server** . La dirección IP es **172.17.0.80** . Hay un gran artículo sobre la configuración de un servidor NFS. A continuación, cree su servidor con Virtual Box (con redes puenteadas) y luego vuelva aquí para el siguiente paso.

El /etc/exports que estoy usando se ve así:

```
/nfs_share 172.17.0.1/24(rw,sync,no_root_squash,no_subtree_check)
```

Eso es inseguro pero lo suficientemente bueno para la configuración, podemos refinarlo más tarde. Además de esa guía vale la pena hacer:

```
chmod 777 /nfs_share
```

Para que podamos estar seguros de que podemos escribir en la raíz de nuestro recurso compartido NFS una vez que esté montado.

Creando los rootfs

Ahora cree otra máquina virtual para realizar la instalación (redes puenteadas nuevamente). Monte el disco de instalación de Slackware en esta máquina virtual. Si desea un rootfs de 32 bits, obviamente use el DVD de instalación de 32 bits, si desea un rootfs de 64 bits, use Slackware64. Si desea 32 bits (muy probablemente para un cliente sin disco) puede que necesite habilitar PAE en VirtualBox en Sistema → Procesador para que Slack32 arranque.

Suponiendo que ahora haya arrancado el instalador de Slackware con el gran kernel predeterminado, ahora puede iniciar la instalación. Inicie sesión como root y obtenga su dirección IP:

```
# dhcpcd eth0
```

Monte el nfs que creó previamente en /mnt:

```
# mount -o rw,nolock slack-nfs-server:/nfs_server /mnt
```

Si no tiene una configuración de DNS que esté bien, simplemente sustituya la dirección IP de su servidor por el servidor slack-nfs, es decir, 172.17.0.80.

Ahora necesitamos manipular algo para el instalador de Slackware. Con fdisk, cree una sola partición en su disco duro (/dev/sda1). Desafortunadamente, una partición válida en una unidad conectada al sistema es un requisito para que el instalador se ejecute. No se preocupe, solo eliminaremos toda esta máquina virtual cuando hayamos terminado, así que no hay problema.

Ahora ejecute 'setup'.

- Mapea el teclado como quieras
- No configurar ningún swap.
- En la pantalla 'Select Linux installation partition' no seleccione /dev/sda1 solo la flecha hacia abajo y seleccione '(done adding partitions, continue with setup)'.
- Instalar desde el slackware cd / dvd
- Seleccione sus paquetes como normal
- No cree un bootdisk
- No instale LILO.

Cuando termine, ahora debería encontrar una raíz fs 'instalada' en el directorio del servidor. Si planea usarlo con más de un cliente liviano, entonces sería un buen momento para realizar una copia de respaldo antes de iniciarlo.

Creando el kernel

El enorme kernel completo que viene con Slackware 14.2 está cerca de proporcionar todo lo que necesitamos, pero aún necesitamos recompilarlo. Recomendaría hacer la compilación en una máquina virtual de 32 bits si está apuntando a un cliente liviano de 32 bits, o de 64 bits si su cliente liviano es de 64 bits. Hay formas de evitar esto y compilar de forma cruzada los núcleos de 32→ 64 bits y viceversa, pero las máquinas virtuales son baratas y la vida es corta:

```
# cd /usr/src/linux
# zcat /proc/config.gz > .config
# make menuconfig
```

El orden de configuración es importante, ya que la selección de ciertas opciones hace que otras estén disponibles. Primero, necesitaremos un controlador de red compilado en el kernel para la NIC que vamos a utilizar. Para VirtualBox, la NIC predeterminada es PCnet32, y lspci probablemente le dirá la suya:

```
Device Drivers -> Network Device Support -> Ethernet driver support -> AMD
PCnet32 PCI support <*>
```

Asegúrate de que esto esté compilado en el kernel (por ejemplo, presionando 'y').

[OPCIONAL] También debemos decirle al núcleo qué dirección IP usar, que se puede configurar de forma estática, pero DHCP es mucho más fácil, por lo que generalmente querrá incluir estas opciones:

```
Networking support -> Networking options -> IP: kernel level
autoconfiguration [*]
    IP: DHCP support [*]
```

Finalmente, absolutamente necesitamos el soporte para Root FS en NFS:

```
File Systems -> Network File systems -> Root file system on NFS [*]
```

[OPCIONAL] Es bastante útil agregar una versión local a esta versión del kernel. Recomiendo hacer esto para diferenciarlo de su gran kernel estándar de Slackware y evitar sobrescribir los módulos por error. Solo podemos añadir '-nfsroot':

```
(-nfsroot) General Setup -> Local version - append to kernel release
```

Guarde la configuración y luego haga un:

```
# make bzImage
```

Mientras se ejecuta esa compilación, es hora de configurar LILO.

Configurando LILO

Llamemos al kernel /boot/vmlinuz-nfsroot. Agregue una sección al archivo lilo.conf:

```
image=/boot/vmlinuz-nfsroot
  label = nfs
  read-only
  append= "root=/dev/nfs ip=dhcp nfsroot=172.17.0.80:/nfs_share,v3 rw"
```

Si no desea utilizar dhcp, ahora tendrá que leer Documentation/filesystems/nfs/nfsroot.txt en las fuentes del kernel para descubrir las muchas opciones que puede incluir para ip= distinto de 'dhcp'.

Obviamente, mantenga su kernel de Linux predeterminado en otra sección image = para que pueda cambiar entre el arranque de nfsroot y el kernel normal para jugar con estas cosas.

No puede especificar una entrada normal de root= en esta sección porque LILO no reconoce /dev/nfs para root (el dispositivo no existe realmente para LILO). Entonces, en lugar de eso, simplemente especifíquelo en la línea append= que LILO no intenta interpretar, y LILO incluirá esta imagen nfsroot adicional sin error.

La v3 parece ser realmente importante para hacer que suceda algo en el arranque. Si eso no está establecido, ninguna comunocación ocurrirá.

El 'rw' también es importante. Previene el fsck de la raíz fs. porque la raíz es NFS y no se puede verificar. Slackware no arranca correctamente si damos 'ro'. En lugar de usar 'rw', opcionalmente podría hackear fsck de los scripts de inicio de Slackware en su raíz NFS, sin embargo, usar 'rw' es más rápido (aunque más sucio).

Con la compilación del kernel terminada, copie el kernel en el directorio /boot y renómbrelo:

```
cp /usr/src/linux/arch/x86/boot/bzImage /boot/vmlinuz-nfsroot
```

Puede crearse en otro lugar que no sea arch/x86 dependiendo de su arquitectura, por ejemplo. x64, arm.

No se olvide de ejecutar LILO:

```
# lilo
```

Primer arranque

Lo anterior es suficiente para obtener un sistema Slackware de arranque, o debería serlo. Hay algunos pasos adicionales que quizás desee hacer ahora.

Módulos

Ninguno de los módulos ha sido instalado, vamos a agregarlos. Al apagar el sistema nfsroot y volver a arrancar en la máquina virtual de compilación del kernel de Slackware, ahora puede compilar los módulos que faltan. Primero montará los rootfs, tal como lo hicimos desde la máquina virtual del instalador:

```
mount -o rw,nolock slack-nfs-server:/nfs_share /mnt/tmp
```

Luego pude compilar e instalar los módulos:

```
# cd /usr/src/linux
# make modules
# make modules_install INSTALL_MOD_PATH=/mnt/tmp
```

Para el último comando, evite agregar una barra diagonal final a /mnt/tmp, e intente no olvidar la INSTALL_MOD_PATH, de lo contrario, puede que solo haya sobrescrito los módulos de su sistema. Si le hubiera dado a u kernel un sufijo local (por ejemplo, -nfsroot), habría sido protegido contra eso.

Swap en NFS

Puede crear un archivo swap en su recurso compartido NFS en algún lugar como este:

```
# dd if=/dev/zero of=/nfs_share/swapfile bs=1024 count=64k
```

Luego formatearlo para swap:

```
# mkswap /nfs_share/swapfile
```

Luego, en el cliente asocie un dispositivo de bucle invertido con el archivo:

```
# losetup /dev/loop0 /swapfile
```

Entonces comience a usar el dispositivo de bucle invertido para swap:

```
# swapon /dev/loop0
```

Obviamente, debe agregar los dos últimos comandos a /etc/rc.d/rc.local u otro script de inicio para ejecutar en cada arranque.

Bloqueo de / etc / exports

Suponiendo que su cliente liviano se conecte desde una dirección predecible, ahora que ha instalado los módulos, finalmente puede bloquear el acceso solo al cliente liviano (/etc/exports en el servidor):

```
/nfs_share 172.17.0.81/32(rw,sync,no_root_squash,no_subtree_check)
```

Y es de suponer que no queremos que todos y cada uno utilicen nuestro directorio rootfs recién preparado, así que bájelo y califíquelo por dirección IP (en el servidor):

```
# cd /
# mv nfs_share 172.17.0.81
# mkdir nfs_share
# mv 172.17.0.81 nfs_share
```

Ahora en la máquina cliente, configure LILO para que nfsroot solicite el recurso compartido nfs basado en la dirección IP del cliente con '% s':

```
image=/boot/vmlinuz-nfsroot
  label = nfs
  read-only
  append= "root=/dev/nfs ip=dhcp nfsroot=172.17.0.80:/nfs_share/%s,v3 rw"
```

La raíz NFS nunca se considerará segura, pero al menos esto hace que la contaminación cruzada de nfsroots sea menos probable.

Tenga en cuenta que estoy usando dhcp en el ejemplo anterior, pero agregué una entrada a /etc/dnsmasq.conf en mi enrutador que asigna la dirección MAC del cliente liviano a la dirección IP 172.17.0.81 para que el cliente siempre obtenga esa dirección.

Fuentes

- Escrito originalmente por User bifferos
- Traducido por: Victor 2019/02/22 21:41 (UTC)

howtos, nfs, author bifferos