Introducción a NanoBSD

Resumen

Este documento proporciona información sobre las herramientas NanoBSD, las cuales pueden ser usadas para crear imágenes de FreeBSD para aplicaciones embebidas, adecuadas para su uso en una memoria USB, tarjeta de memoria, u otro medio de almacenamiento.

Tabla de contenidos

1. Introducción a NanoBSD 1
2. NanoBSD paso a paso 1

1. Introducción a NanoBSD

NanoBSD es una herramienta desarrollada por Poul-Henning Kamp <phk@FreeBSD.org> y mantenida ahora por Warner Losh <imp@FreeBSD.org>. La herramienta crea una imagen para aplicaciones embebidas, adecuada para su uso en una memoria USB, tarjeta de memoria, u otro medio de almacenamiento.

Puede usarse para crear imágenes de instalación especializadas, diseñadas para instalar y mantener sistemas comúnmente llamados "aparatos de computación" (computer appliances). Los aparatos de computación incluyen el hardware y software en el mismo producto, lo cual significa que todas las aplicaciones vienen preinstaladas. El aparato se conecta a una red existente y puede comenzar a funcionar (casi) de inmediato.

Las funcionalidades de NanoBSD incluyen:

- Los ports y paquetes funcionan de la misma manera que en FreeBSD — Cada aplicación puede ser instalada y usada en una imagen de NanoBSD, de la misma manera que en FreeBSD.
- No se pierde funcionalidad - Si es posible hacer algo con FreeBSD, es posible hacer lo mismo con NanoBSD, excepto si las funciones específicas fueron explícitamente eliminadas de la imagen de NanoBSD cuando fue creada.
- El sistema es de sólo lectura en tiempo de ejecución, por lo tanto es seguro quitar el cable de alimentación. No hay necesidad de correr fsck(8) después de un apagado abrupto del sistema.
- Es fácil de crear y personalizar — Haciendo uso de un único script y solamente un archivo de configuración, es posible crear imágenes reducidas y personalizadas para satisfacer un conjunto arbitrario de requisitos.

2. NanoBSD paso a paso

2.1. El Diseño de NanoBSD

Una vez que la imagen está presente en el medio, es posible arrancar NanoBSD. El medio de almacenamiento está dividido por defecto en tres partes:

- Dos particiones de imagen: `code#1` y `code#2`.
- La partición del archivo de configuración, que se puede montar bajo el directorio /cfg en tiempo de ejecución.

Estas particiones normalmente se montan como solo lectura.

Los directorios /etc y /var son discos malloc (malloc disks) md(4).

La partición del archivo de configuración reside en el directorio /cfg. Contiene archivos para el directorio /etc y se monta brevemente como una partición de solo lectura tras el arranque del sistema; por lo tanto, es necesaria para copiar archivos modificados desde /etc de vuelta hacia el directorio /cfg si se espera que los cambios persistan después de reiniciar el sistema.

Ejemplo 1. Haciendo Cambios Persistentes en /etc/resolv.conf

```
# vi /etc/resolv.conf
[...]
# mount /cfg
# cp /etc/resolv.conf /cfg
# umount /cfg
```



La partición que contiene /cfg debería ser montada solo en el arranque y cuando se sustituyan las directivas de los archivos de configuración.

Mantener /cfg montado todo el tiempo no es una buena idea, especialmente si el sistema NanoBSD se ejecuta en un medio de almacenamiento masivo que puede verse afectado negativamente por un alto número de escrituras en la partición (como cuando el sincronizador del sistema de archivos vuelca los datos hacia los discos del sistema).

2.2. Compilar una Imagen de NanoBSD

Una imagen de NanoBSD se construye usando el sencillo script nanobsd.sh que puede encontrarse en el directorio /usr/src/tools/tools/nanobsd. Este script crea una imagen que se puede copiar al medio de almacenamiento utilizando la herramienta dd(1).

Los comandos necesarios para construir una imagen NanoBSD son:

```
# cd /usr/src/tools/tools/nanobsd ①
# sh nanobsd.sh ②
# cd /usr/obj/nanobsd.full ③
```

```
# dd if=_.disk.full of=/dev/da0 bs=64k ④
```

① Cambia el directorio actual por el directorio base del script de construcción de NanoBSD.

② Comienza el proceso de compilación.

③ Cambia el directorio actual por el lugar donde se encuentran las imágenes compiladas.

④ Instala NanoBSD en el medio de almacenamiento.

2.2.1. Opciones al Construir una Imagen de NanoBSD

Cuando se construye una imagen de NanoBSD, se pueden pasar varias opciones a nanobsd.sh en la línea de comando. Estas opciones pueden tener un impacto significativo en el proceso de construcción.

Algunas opciones tienen propósitos de verbosidad:

- `-h`: imprime un resumen de la página de ayuda.
- `-q`: hace que la salida sea más silenciosa.
- `-v`: hace que la salida sea más verbosa

Otras opciones se pueden usar para restringir el proceso de construcción. A veces no es necesario reconstruir todo desde las fuentes, especialmente si ya se ha construido una imagen y sólo se han hecho pequeños cambios.

- `-k`: no construir el kernel
- `-w`: no construir world
- `-b`: no construir ni el kernel ni world
- `-i`: no construir una imagen de disco. Como no se creará un fichero, no será posible copiarlo con dd(1) al medio de almacenamiento.
- `-f`: no construir una imagen de disco de la primera partición (útil para actualizaciones)
- `-n`: añade `-DNO_CLEAN` a `buildworld`, `buildkernel`. También se guardan todos los ficheros que se hayan construido en una ejecución previa.

Se puede usar un fichero de configuración para ajustar cuantos elementos como se quiera. Cárgalo con `-c`

Las últimas opciones son:

- `-K`: no instalar un kernel. Una imagen de disco sin un kernel no será capaz de realizar una secuencia normal de arranque.

2.2.2. El Proceso Completo de Creación de Imágenes

El proceso completo de creación de imágenes tiene muchos pasos. Los pasos exactos dependerán de las opciones escogidas cuando se inicia el script. Asumiendo que el script se ejecuta sin opciones particulares, esto es lo que pasará.

1. `run_early_customize`: comandos que se definen en un fichero de configuración que se ha proporcionado.
2. `clean_build`: Simplemente limpia el entorno de construcción mediante el borrado de ficheros construidos previamente.
3. `make_conf_build`: Ensambla make.conf a partir de las variables `CONF_WORLD` y `CONF_BUILD`.
4. `build_world`: Construye world.
5. `build_kernel`: Construye los ficheros del kernel.
6. `clean_world`: Limpia el directorio de destino.
7. `make_conf_install`: Ensambla make.conf a partir de las variables `CONF_WORLD` y `CONF_INSTALL`.
8. `install_world`: Instala todos los ficheros construidos durante la fase `buildworld`.
9. `install_etc`: Instala los ficheros necesarios en el directorio /etc basándose en el comando `make distribution`.
10. `setup_nanobsd_etc`: la primera configuración específica de NanoBSD tiene lugar en esta fase. Se crea /etc/diskless y se define el sistema de ficheros raíz como sólo lectura.
11. `install_kernel`: se instalan los ficheros del kernel y los módulos.
12. `run_customize`: se llama a todas las rutinas personalizadas definidas por el usuario.
13. `setup_nanobsd`: se configura un esquema especial de directorios de configuración. /usr/local/etc se mueve a /etc/local y se crea un enlace simbólico de vuelta desde /etc/local a /usr/local/etc.
14. `prune_usr`: se eliminan los directorios vacíos de /usr.
15. `run_late_customize`: en este punto se ejecutan los últimos scripts personalizados.
16. `fixup_before_diskimage`: Lista todos los ficheros instalados en un metalog.
17. `create_diskimage`: crea la imagen de disco real basada en los parámetros proporcionados sobre las geometrías del disco.
18. `last_orders`: no hace nada de momento.

2.3. Personalizar una Imagen de NanoBSD

Esta es probablemente la característica más importante y más interesante de NanoBSD. Es también donde pasarás la mayor parte del tiempo cuando desarrolles con NanoBSD.

Invocar el siguiente comando forzará a nanobsd.sh a leer su configuración desde el archivo myconf.nano situado en el directorio actual:

```
# sh nanobsd.sh -c myconf.nano
```

La personalización puede realizarse de dos formas:

- Opciones de configuración
- Funciones personalizadas

2.3.1. Opciones de Configuración

Con las opciones de ajuste, es posible configurar opciones que se pasan tanto a la fase `buildworld` como a la fase `installworld` del proceso de construcción de NanoBSD, así como opciones internas pasadas al proceso principal de construcción de NanoBSD. Mediante estas opciones es posible hacer más pequeño el sistema de forma que quepa en tan solo 64 MB. Puedes utilizar las opciones de configuración para hacer FreeBSD incluso más pequeño hasta que consista sólo en el kernel y en dos o tres ficheros en espacio de usuario.

El archivo de configuración consiste en opciones de configuración que sobrescriben los valores por defecto. Las directivas más importantes son:

- `NANO_NAME` - Nombre de la construcción (utilizado para los nombres de los directorios de construcción).
- `NANO_SRC` - Ruta al árbol de fuentes utilizado para construir la imagen.
- `NANO_KERNEL` - Nombre del fichero de configuración del kernel utilizado para construir el kernel.
- `CONF_BUILD` - Opciones pasadas a la fase de construcción `buildworld`.
- `CONF_INSTALL` - Opciones pasadas a la fase de construcción `installworld`.
- `CONF_WORLD` - Opciones pasadas a las fases de construcción `buildworld` e `installworld`.
- `FlashDevice` - Define el tipo de medio a utilizar. Consulta FlashDevice.sub para más detalles.

Hay muchas más opciones de configuración que podrían ser relevantes dependiendo del tipo de NanoBSD que se desea.

Personalización General

Hay tres etapas, por diseño, en las cuales es posible hacer cambios que afectan al proceso de construcción, simplemente estableciendo una variable en el fichero de configuración proporcionado:

- `run_early_customize`: antes de que se haga cualquier otra cosa.
- `run_customize`: después de que se hayan dispuesto todos los ficheros estándar
- `run_late_customize`: al final del proceso, justo antes de que se cree la imagen definitiva de NanoBSD.

Para personalizar una imagen de NanoBSD, en cualquiera de estos pasos, es mejor añadir un valor específico a la variable que corresponda.

La variable `NANO_EARLY_CUSTOMIZE` se utiliza en el primer paso del proceso de construcción. En este momento no hay un ejemplo de lo que se puede hacer con esta variable, pero esto podría cambiar en el futuro.

La variable `NANO_CUSTOMIZE` se utiliza después de que se hayan instalados los ficheros del kernel, world y configuración y de que los ficheros de etc se hayan configurado para que sean una instalación de NanoBSD. Así que es el paso correcto del proceso de construcción para ajustar las opciones de configuración y añadir paquetes como se hace en el ejemplo cust_nobeastie.

La variable NANO_LATE_CUSTOMIZE se usa justo antes de que se cree la imagen de disco, así que es el último momento para poder cambiar algo. Recuerda que la rutina setup_nanobsd ya se ha ejecutado y que los directorios etc, conf and cfg y subdirectorios ya se han modificado, así que no es el momento de cambiar nada. Más bien es posible añadir o eliminar ficheros específicos.

Opciones de Arranque

También hay variables que pueden cambiar la forma en la que arranca una imagen de NanoBSD. Se pasan dos opciones a boot0cfg(8) para inicializar el sector de arranque de la imagen de disco:

- NANO_BOOT0CFG
- NANO_BOOTLOADER

Con NANO_BOOTLOADER se puede escoger un fichero de bootloader. Las opciones posibles más habituales son boot0sio y boot0 dependiendo de si el dispositivo tiene o no puerto serie. Es mejor evitar proporcionar un bootloader diferente, pero es posible. Para hacerlo, lo mejor es consultar el capítulo sobre el proceso de arranque en el FreeBSD Handbook.

Con NANO_BOOT0CFG se puede modificar el proceso de arranque como por ejemplo seleccionar desde qué partición arrancará la imagen de NanoBSD. Es mejor consultar la página boot0cfg(8) antes de cambiar el valor por defecto de esta variable. Una opción que podría ser interesante cambiar es el tiempo de espera del procedimiento de arranque. Para ello, se puede cambiar la variable NANO_BOOT0CFG a "-o packet -s 1 -m 3 -t 36". De este modo el proceso de arranque empezaría después de aproximadamente 2 segundos; porque es raro que se quiera esperar 10 segundos antes de arrancar.

Está bien saber esto: la variable NANO_BOOT2CFG sólo se usa en la rutina cust_comconsole que se puede llamar en el paso NANO_CUSTOMIZE si el dispositivo tiene un puerto serie y toda la entrada y salida de la consola tiene que ir a través de él. Asegúrate de comprobar los parámetros relevantes del puerto serie ya que establecer un parámetro a un valor erróneo puede inhabilitar el puerto.

Creación de una Imagen de Disco

Al final del proceso de arranque se encuentra la creación de la imagen de disco. Con este paso, el script de Nano BSD proporciona un fichero que puede ser copiado simplemente en el disco del dispositivo y que hará que arranque.

Hay muchas variables que se tienen que configurar bien para que el script produzca una imagen de disco que sea utilizable.

- La variable NANO_DRIVE tiene que establecerse al nombre de la unidad del medio en tiempo de ejecución. Normalmente, el valor por defecto ada0, que representa al primer dispositivo IDE/ATA/SATA en la unidad, se espera que sea correcto pero se podría usar un tipo diferente de almacenamiento - como una llave USB, en cuyo caso sería mejor usar da0.
- La variable NANO_MEDIASIZE se tiene que establecer al tamaño (en sectores de 512 bytes) del medio de almacenamiento que se utilizará. Si lo estableces de forma incorrecta, es posible que la imagen de NanoBSD no arranque en absoluto y se mostrará un mensaje durante el arranque avisando de una geometría de disco incorrecta.
- Los directorios /etc, /var, y /tmp se reservan como discos(malloc) md(4) en el arranque; de

forma que sus tamaños se pueden ajustar par satisfacer las necesidades del dispositivo. La variable NANO_RAM_ETCSIZE establece el tamaño de /etc; y la variable NANO_RAM_TMPVARSIZE establece el tamaño de los directorios /var y /tmp puesto que /tmp está enlazado simbólicamente a /var/tmp. Por defecto, los tamaños de ambos discos malloc se establece a 20MB cada uno. Siempre se pueden cambiar, pero normalmente /etc no crece demasiado por lo que 20MB es un buen puntor de partida, mientras que /var y especialmente /tmp pueden crecer mucho más si no ponemos cuidado. Para sistemas con limitaciones de memoria, se pueden escoger sistemas de ficheros más pequeños.

- Como NanoBSD está diseñado principalmente para construir una imagen de un sistema para un aparato, se asume que el medio de almacenamiento será relativamente pequeño. Por esa razón, el sistema de ficheros que se establece está configurado para tener un tamaño de bloque pequeño (4Kb) y un tamaño de fragmento pequeño (512b). Las opciones de configuración del sistema de ficheros se pueden configurar mediante la variable NANO_NEWFS, pero la sintaxis debe respetar el formato del comando newfs(8) . El sistema de ficheros también tiene Soft Updates activado por defecto. Se puede leer sobre esto en el FreeBSD Handbook.
- Los diferentes tamaños de partición se pueden establecer mediante el uso de NANO_CODESIZE, NANO_CONFSIZE, y NANO_DATASIZE como múltiplos de sectores de 512 bytes. NANO_SIZE define el tamaño de las dos primeras particiones: code#1 and code#2. Tienen que ser suficientemente grandes para contener todos los ficheros que se producirán como resultado de los procesos buildworld y buildkernel. NANO_CONFSIZE define el tamaño de la partición de ficheros de configuración por lo que no tiene que ser muy grande; pero no la hagas tan pequeña que no pueda albergar los ficheros. Por último NANO_DATASIZE define el tamaño de una partición opcional, que se puede usar en el dispositivo. Por ejemplo, se puede utilizar la última partición para mantener en disco ficheros creados al vuelo.

2.3.2. Funciones Personalizadas

Es posible afinar NanoBSD utilizando funciones del shell en el fichero de configuración. El siguiente ejemplo ilustra el modelo básico de las funciones personalizadas:

```
cust_foo () (
    echo "bar=baz" > \
        ${NANO_WORLDDIR}/etc/foo
)
customize_cmd cust_foo
```

Un ejemplo más útil de una función de personalización es el siguiente, el cual cambia el tamaño por defecto del directorio /etc de 5MB a 30MB:

```
cust_etc_size () (
    cd ${NANO_WORLDDIR}/conf
    echo 30000 > default/etc/md_size
)
customize_cmd cust_etc_size
```

Estas son algunas funciones de personalización incluidas por defecto y listas para ser usadas:

- `cust_comconsole` - Deshabilita getty(8) en los dispositivos VGA (los nodos de dispositivo /dev/ttyv*) y habilita el uso del puerto serie COM1 como consola del sistema.
- `cust_allow_ssh_root` - Permite al usuario `root` hacer login vía sshd(8).
- `cust_install_files` - Instala ficheros desde el directorio nanobsd/Files, el cual contiene algunos scripts útiles para la administración del sistema.

2.3.3. Agregando Paquetes

Se pueden agregar paquetes a una imagen de NanoBSD para proporcionar funcionalidades específicas para el dispositivo. Para ello, o bien:

- Añade `cust_pkgng` a la variable `NANO_CUSTOMIZE`, o
- Añade el comando `'customize_cmd cust_pkgng'` en un fichero de configuración personalizado.

Ambos métodos consiguen el mismo resultado: lanzar la rutina `cust_pkgng`. Esta rutina recorrerá el directorio `NANO_PACKAGE_DIR` para encontrar bien todos los paquetes o sólo la lista de paquetes de la variable `NANO_PACKAGE_LIST`.

Cuando se instalan aplicaciones mediante pkg en un entorno FreeBSD estándar, es habitual que el proceso de instalación cree ficheros de configuración en el directorio /usr/local/etc, y scripts de arranque en el directorio /usr/local/etc/rc.d. De modo que después de que se hayan instalado los paquetes necesarios, necesitan ser configurados para que estén listos para usar. Para ello se tienen que instalar los ficheros de configuración necesarios en los directorios correctos. Esto se puede conseguir escribiendo rutinas dedicadas o se puede utilizar la rutina genérica `cust_install_files` para copiar los ficheros desde el directorio /usr/src/tools/tools/nanobsd/Files. Para cada paquete normalmente se necesita añadir una línea (a veces varias) en /etc/rc.conf.

2.3.4. Ejemplo de Archivo de Configuración

Un ejemplo completo de un archivo de configuración para construir una imagen personalizada de NanoBSD podría ser:

```
NANO_NAME=custom
NANO_SRC=/usr/src
NANO_KERNEL=MYKERNEL
NANO_IMAGES=2

CONF_BUILD='
WITHOUT_KLDLOAD=YES
WITHOUT_NETGRAPH=YES
WITHOUT_PAM=YES
'

CONF_INSTALL='
WITHOUT_ACPI=YES
WITHOUT_BLUETOOTH=YES
WITHOUT_FORTRAN=YES
WITHOUT_HTML=YES
```

```
WITHOUT_LPR=YES
WITHOUT_MAN=YES
WITHOUT_SENDMAIL=YES
WITHOUT_SHAREDOCS=YES
WITHOUT_EXAMPLES=YES
WITHOUT_INSTALLLIB=YES
WITHOUT_CALENDAR=YES
WITHOUT_MISC=YES
WITHOUT_SHARE=YES
'

CONF_WORLD='
WITHOUT_BIND=YES
WITHOUT_MODULES=YES
WITHOUT_KERBEROS=YES
WITHOUT_GAMES=YES
WITHOUT_RESCUE=YES
WITHOUT_LOCALES=YES
WITHOUT_SYSCONS=YES
WITHOUT_INFO=YES
'

FlashDevice SanDisk 1G

cust_nobeastie() (
    touch ${NANO_WORLDDIR}/boot/loader.conf
    echo "beastie_disable=\"YES\"" >> ${NANO_WORLDDIR}/boot/loader.conf
)

customize_cmd cust_comconsole
customize_cmd cust_install_files
customize_cmd cust_allow_ssh_root
customize_cmd cust_nobeastie
```

Todas las opciones de construcción e instalación se pueden encontrar en la página del manual de src.conf(5), pero no todas las opciones se pueden o se deben usar cuando se construye una imagen de NanoBSD. Las opciones de construcción e instalación se deberían definir de acuerdo a las necesidades de la imagen que se está construyendo.

Por ejemplo, el cliente y el servidor de ftp podrían no ser necesarios. Añadir `WITHOUT_FTP=TRUE` a un fichero de configuración en la sección `CONF_BUILD` evitará compilarlos. También, si el dispositivo NanoBSD no se va a usar para construir programas entonces es posible añadir `WITHOUT_BINUTILS=TRUE` en la sección `CONF_INSTALL`; pero no en la sección `CONF_BUILD` ya que serán usadas para construir la imagen de NanoBSD.

Evitar compilar un conjunto de programas en particular - mediante una opción de compilación - acorta el tiempo total de construcción y reduce el tamaño necesario para la imagen de disco, mientras que no instalar dicho conjunto de programas no disminuye el tiempo total de construcción.

2.4. Actualizando NanoBSD

El proceso de actualización de NanoBSD es relativamente simple:

1. Construye una nueva imagen de NanoBSD de forma habitual.
2. Sube la imagen nueva a una partición sin usar en el dispositivo que esté ejecutando NanoBSD.

   La diferencia más importante entre este paso y la instalación inicial de NanoBSD es que ahora, en lugar de usar _.disk.full (que contiene la imagen completa del disco), se instala la imagen _.disk.image (la cual contiene la imagen de una sola partición del sistema).
3. Reinicia y arranca el sistema desde la partición recién instalada.
4. Si todo termina correctamente, la actualización habrá finalizado.
5. Si algo sale mal, reinicia en la partición anterior (que contiene la antigua imagen que funciona correctamente), para restaurar la funcionalidad del sistema tan rápido como sea posible. Arregla los problemas de la nueva imagen y repite el proceso.

Para instalar una nueva imagen en un sistema que está ejecutando NanoBSD, es posible utilizar los scripts updatep1 o updatep2 que se encuentran en el directorio /root, dependiendo de la partición desde la que se esté ejecutando el sistema actual.

Dependiendo de los servicios disponibles en el host que ofrece la nueva imagen de NanoBSD y el tipo de transferencia preferido, es posible examinar uno de estos tres métodos:

2.4.1. Usar ftp(1)

Si la velocidad de transferencia es una prioridad, utiliza este ejemplo:

```
# ftp myhost
get _.disk.image "| sh updatep1"
```

2.4.2. Usar ssh(1)

Si prefieres una transferencia segura, considera usar este ejemplo:

```
# ssh myhost cat _.disk.image.gz | zcat | sh updatep1
```

2.4.3. Usar nc(1)

Prueba este ejemplo si el host remoto no está ejecutando ni el servicio ftpd(8) ni el servicio sshd(8):

1. En primer lugar, abre un puerto TCP en el host que se encuentra sirviendo la imagen y haz que envíe la imagen al cliente:

```
myhost# nc -l 2222 < _.disk.image
```



Asegúrate de que el puerto que usas no está bloqueado por ningún firewall para recibir conexiones entrantes desde el host NanoBSD.

2. Conéctate al host que sirve la nueva imagen y ejecuta el script updatep1:

```
# nc myhost 2222 | sh updatep1
```