# Introdução ao NanoBSD

Resumo

Este documento fornece informações sobre as ferramentas NanoBSD, que podem ser usadas para criar imagens de sistema FreeBSD para aplicações embarcadas, adequadas para uso em uma chave USB, cartão de memória ou outro meio de armazenamento em massa.

# Índice

1. Introdução ao NanoBSD . . . . . 1
2. Guia do NanoBSD . . . . . 1

# 1. Introdução ao NanoBSD

O NanoBSD é uma ferramenta desenvolvida por Poul-Henning Kamp <phk@FreeBSD.org> e agora mantida por Warner Losh <imp@FreeBSD.org>. Ela cria uma imagem do sistema FreeBSD para aplicações embarcadas, adequada para uso em uma chave USB, cartão de memória ou outro meio de armazenamento em massa.

Pode ser usado para construir imagens de instalação especializadas, projetadas para facilitar a instalação e manutenção de sistemas comumente chamados de "computer appliances". Os computer appliances têm seu hardware e software empacotados no produto, o que significa que todos os aplicativos estão pré-instalados. O aparelho é conectado a uma rede existente e pode começar a funcionar (quase) imediatamente.

As características do NanoBSD incluem:

- Os Ports e pacotes funcionam como no FreeBSD - Cada aplicação pode ser instalada e usada em uma imagem NanoBSD, da mesma forma que no FreeBSD.
- Nenhuma funcionalidade faltante - Se é possível fazer algo com o FreeBSD, é possível fazer a mesma coisa com o NanoBSD, a menos que a funcionalidade específica tenha sido explicitamente removida da imagem do NanoBSD quando ela foi criada.
- O sistema opera em modo read-only em tempo de execução - É seguro puxar o plugue de energia. Não há necessidade de executar o fsck(8) após um desligamento abrupto do sistema.
- É fácil de compilar e personalizar - Usando apenas um script de shell e um arquivo de configuração, é possível criar imagens reduzidas e personalizadas, satisfazendo qualquer conjunto arbitrário de requisitos.

# 2. Guia do NanoBSD

## 2.1. O Design do NanoBSD

Uma vez que a imagem está presente no meio de armazenamento, é possível inicializar o NanoBSD. O meio de armazenamento em massa é dividido em três partes por padrão:

- Duas partições de imagem: `code#1` e `code#2`.
- A partição do arquivo de configuração, que pode ser montada sob o diretório /cfg em tempo de execução.

Essas partições são normalmente montadas em modo read-only (somente leitura).

Os diretórios /etc e /var são discos md(4) (malloc).

A partição do arquivo de configuração persiste no diretório /cfg. Ele contém arquivos para o diretório /etc e é montado brevemente como somente leitura logo após a inicialização do sistema. Portanto, é necessário copiar os arquivos modificados do diretório /etc de volta para o diretório /cfg se as alterações precisarem ser mantidas após a reinicialização do sistema.

*Exemplo 1. Fazendo alterações persistentes no arquivo /etc/resolv.conf*

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



A partição que contém o diretório /cfg deve ser montada somente no momento da inicialização e enquanto sobrescreve os arquivos de configuração.

Manter o /cfg montado o tempo todo não é uma boa idéia, especialmente se o sistema NanoBSD rodar em um meio de armazenamento em massa que pode ser afetado adversamente por um grande número de gravações na partição (como quando o sincronizador do sistema de arquivos libera dados para os discos do sistema).

## 2.2. Construindo uma imagem NanoBSD

Uma imagem NanoBSD é compilada usando um simples shell script nanobsd.sh, que pode ser encontrado no diretório /usr/src/tools/tools/nanobsd. Este script cria uma imagem, que pode ser copiada no meio de armazenamento usando o utilitário dd(1).

Os comandos necessários para construir uma imagem NanoBSD são:

```
# 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 Altere o diretório atual para o diretório base do script de compilação do NanoBSD.

② Inicie o processo de compilação.

③ Altere o diretório atual para o local onde as imagens compiladas estão localizadas.

④ Instale o NanoBSD no meio de armazenamento.

### 2.2.1. Opções ao compilar uma imagem NanoBSD

Ao construir uma imagem NanoBSD, várias opções de compilação podem ser passadas para nanobsd.sh na linha de comando. Essas opções podem ter um impacto significativo no processo de compilação.

Algumas opções são para fins de verbosidade:

- `-h`: imprime a página resumida de ajuda.
- `-q`: torna a saída mais silenciosa.
- `-v`: torna a saída mais detalhada (verbose)

Algumas outras opções podem ser usadas para restringir o processo de compilação. Às vezes, não é necessário reconstruir tudo a partir das fontes, especialmente se uma imagem já foi compilada e apenas poucas alterações foram feitas.

- `-k`: não compilar o kernel
- `-w`: não compilar o mundo (world)
- `-b`: não compilar nem o kernel e nem o mundo (world)
- `-i`: não compilar uma imagem de disco. Como um arquivo não será criado, não será possível usá-lo com o comando dd(1) para gravá-lo em um dispositivo de armazenamento.
- `-f`: não compilar uma imagem de disco para a primeira partição (o que é útil para fins de atualização)
- `-n`: adiciona `-DNO_CLEAN` as etapas `buildworld` e `buildkernel`. Além disso, todos os arquivos que já foram compilados em uma execução anterior são mantidos.

Um arquivo de configuração pode ser usado para ajustar quantos elementos desejar. Carregue-o com `-c`

As últimas opções são:

- `-K`: não instalar um kernel. Uma imagem de disco sem um kernel não poderá executar uma sequência de inicialização normal.

### 2.2.2. O Processo Completo de Compilação de Imagens

O processo completo de compilação de imagens passa por muitas etapas. As etapas exatas dependerão das opções selecionadas ao iniciar o script. Supondo que o script seja executado sem opções específicas, isso é o que acontecerá.

1. `run_early_customize`: comandos que são definidos em um arquivo de configuração fornecido.
2. `clean_build`: Apenas limpa o ambiente de compilação excluindo os arquivos previamente compilados.
3. `make_conf_build`: Monta o arquivo make.conf a partir das variáveis `CONF_WORLD` e `CONF_BUILD`.
4. `build_world`: Compila o mundo (world).
5. `build_kernel`: Compila os arquivos do kernel.
6. `clean_world`: Limpa o diretório de destino.
7. `make_conf_install`: Monta o arquivo make.conf a partir das variáveis `CONF_WORLD` e `CONF_INSTALL`.
8. `install_world`: Instala todos os arquivos construídos durante o `buildworld`.
9. `install_etc`: Instala os arquivos necessários no diretório /etc, com base no comando `make distribution`.
10. `setup_nanobsd_etc`: a primeira configuração específica do NanoBSD ocorre nesta etapa. O diretório /etc/diskless é criado e o sistema de arquivos raiz é definido como somente leitura.
11. `install_kernel`: os arquivos do kernel e dos módulos são instalados.
12. `run_customize`: todas as rotinas de personalização definidas pelo usuário serão chamadas.
13. `setup_nanobsd`: uma estrutura especial de diretórios de configuração é configurada. O diretório /usr/local/etc é movido para /etc/local e um link simbólico é criado de volta de /etc/local para /usr/local/etc.
14. `prune_usr`: os diretórios vazios de /usr são removidos.
15. `run_late_customize`: os últimos scripts personalizados podem ser executados neste ponto.
16. `fixup_before_diskimage`: Lista todos os arquivos instalados em um metalog
17. `create_diskimage`: cria a imagem de disco, com base nos parâmetros de geometria de disco fornecidos.
18. `last_orders`: não faz nada por enquanto.

## 2.3. Personalizando uma imagem NanoBSD

Esta é provavelmente a característica mais importante e mais interessante do NanoBSD. É também onde você passará a maior parte do tempo ao desenvolver com o NanoBSD.

A invocação do seguinte comando forçará o nanobsd.sh a ler sua configuração de myconf.nano localizado no diretório atual:

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

A personalização é feita de duas maneiras:

- Opções de configuração
- Funções personalizadas

### 2.3.1. Opções de configuração

Com as configurações de configuração, é possível configurar as opções passadas tanto para as etapas `buildworld` quanto `installworld` do processo de compilação do NanoBSD, bem como opções internas passadas para o processo de compilação principal do NanoBSD. Através dessas opções, é possível reduzir o sistema, para que ele caiba em tão pouco quanto 64MB. Você pode usar as opções de configuração para reduzir ainda mais o FreeBSD, até que consista apenas no kernel e em dois ou três arquivos no espaço do usuário.

O arquivo de configuração consiste em opções de configuração, que substituem os valores padrão. As diretivas mais importantes são:

- `NANO_NAME` - Nome da compilação (usado para construir os nomes dos diretórios de trabalho).
- `NANO_SRC` - Caminho para a árvore de código-fonte usada para compilar a imagem.
- `NANO_KERNEL` - Nome do arquivo de configuração do kernel usado para compilar o kernel.
- `CONF_BUILD` - Opções passadas para o estágio `buildworld` da compilação.
- `CONF_INSTALL` - Opções passadas para o estágio `installworld` da compilação.
- `CONF_WORLD` - Opções passadas tanto para o estágio `buildworld` quanto para o estágio `installworld` da compilação.
- `FlashDevice` - Define qual tipo de mídia usar. Verifique FlashDevice.sub para obter mais detalhes.

Existem muitas outras opções de configuração que podem ser relevantes, dependendo do tipo de NanoBSD desejado.

**Personalização Geral**

Existem três estágios, por design, nos quais é possível fazer alterações que afetam o processo de compilação, apenas configurando uma variável no arquivo de configuração fornecido:

- `run_early_customize`: antes de qualquer outra coisa acontecer.
- `run_customize`: depois que todos os arquivos padrões tiverem sido organizados
- `run_late_customize`: no final do processo, antes da compilação da imagem real do NanoBSD.

Para personalizar uma imagem do NanoBSD em qualquer um desses passos, é melhor adicionar um valor específico a uma das variáveis correspondentes.

A variável `NANO_EARLY_CUSTOMIZE` é usada no primeiro passo do processo de construção. Neste momento, não há um exemplo específico do que pode ser feito usando essa variável, mas isso pode mudar no futuro.

A variável `NANO_CUSTOMIZE` é usada depois que os arquivos de configuração do kernel, world e etc foram instalados, e os arquivos etc foram configurados como uma instalação do NanoBSD. Portanto, é o passo correto no processo de compilação para ajustar opções de configuração e adicionar pacotes, como no exemplo cust_nobeastie.

A variável `NANO_LATE_CUSTOMIZE` é usada imediatamente antes da criação da imagem do disco,

portanto, é o momento final para fazer qualquer alteração. Lembre-se de que a rotina `setup_nanobsd` já foi executada e que os diretórios etc, conf e cfg e seus subdiretórios já foram modificados, então não é o momento de alterá-los neste ponto. Em vez disso, é possível adicionar ou remover arquivos específicos.

### Opções de Inicialização

Também existem variáveis que podem alterar a forma como a imagem do NanoBSD é inicializada. Duas opções são passadas para o boot0cfg(8) para inicializar o setor de boot da imagem do disco:

- `NANO_BOOT0CFG`
- `NANO_BOOTLOADER`

Com o `NANO_BOOTLOADER`, é possível escolher um arquivo de bootloader. As opções mais comuns são entre boot0sio e boot0, dependendo se o aparelho possui uma porta serial ou não. É melhor evitar fornecer um bootloader diferente, mas é possível fazê-lo. Para isso, é recomendável ter verificado o capítulo sobre o processo de inicialização no Handbook do FreeBSD.

Com o `NANO_BOOT0CFG`, é possível ajustar o processo de inicialização, como selecionar em qual partição a imagem do NanoBSD será inicializada. É melhor verificar a página boot0cfg(8) antes de alterar o valor padrão dessa variável. Uma opção interessante para alterar é o tempo limite do procedimento de inicialização. Para fazer isso, a variável `NANO_BOOT0CFG` pode ser alterada para `"-o packet -s 1 -m 3 -t 36"`. Dessa forma, o processo de inicialização começaria após aproximadamente 2 segundos, porque é raro que seja desejado esperar 10 segundos antes de iniciar o boot.

Bom saber: a variável `NANO_BOOT2CFG` é usada apenas na rotina `cust_comconsole`, que pode ser chamada no passo `NANO_CUSTOMIZE` se o aparelho tiver uma porta serial e toda a entrada e saída do console tiver que ser feita por meio dela. Certifique-se de verificar os parâmetros relevantes da porta serial, pois definir um valor incorreto pode torná-la inútil.

### Criação da Imagem do Disco

No final do processo de inicialização está a criação da imagem do disco. Com essa etapa, o script do NanoBSD fornece um arquivo que pode ser simplesmente copiado para um disco do aparelho, e isso fará com que ele seja inicializado e iniciado.

Existem muitas variáveis que precisam ser configuradas corretamente para que o script produza uma imagem de disco utilizável.

- A variável `NANO_DRIVE` deve ser definida como o nome da unidade de mídia em tempo de execução. Geralmente, o valor padrão `ada0`, que representa o primeiro dispositivo `IDE`/`ATA`/`SATA` no aparelho, é esperado como o correto, mas um tipo diferente de armazenamento também pode ser usado - como uma chave USB, nesse caso seria `da0`.
- A variável `NANO_MEDIASIZE` deve ser definida como o tamanho (em setores de 512 bytes) da mídia de armazenamento que será usada. Se você definir isso incorretamente, é possível que a imagem do NanoBSD não inicialize e uma mensagem durante a inicialização avisará sobre a geometria de disco incorreta.
- Os diretórios /etc, /var e /tmp são alocados como discos md(4) (malloc) durante a inicialização;

portanto, seus tamanhos podem ser ajustados para atender às necessidades do aparelho. A variável NANO_RAM_ETCSIZE define o tamanho do /etc e a variável NANO_RAM_TMPVARSIZE define o tamanho tanto do diretório /var quanto do diretório /tmp, já que /tmp está simbolicamente vinculado a /var/tmp. Por padrão, ambos os tamanhos dos discos malloc são definidos em 20MB cada. Eles podem ser alterados, mas geralmente o /etc não cresce muito em tamanho, portanto, 20MB é um bom ponto de partida, enquanto o /var e especialmente o /tmp podem aumentar muito de tamanho se não tiver cuidado. Para sistemas com restrição de memória, tamanhos menores de sistema de arquivos podem ser escolhidos.

- Como o NanoBSD é projetado principalmente para criar uma imagem de sistema para um aparelho, presume-se que a mídia de armazenamento usada seja relativamente pequena. Por esse motivo, o sistema de arquivos que é organizado é configurado com um tamanho de bloco pequeno (4Kb) e um tamanho de fragmento pequeno (512b). As opções de configuração do sistema de arquivos podem ser modificadas por meio da variável NANO_NEWFS, mas a sintaxe deve respeitar o formato do comando newfs(8). Além disso, por padrão, o sistema de arquivos tem Soft Updates ativado. O Handbook do FreeBSD pode ser consultado para obter mais informações sobre isso.
- Os diferentes tamanhos das partições podem ser definidos por meio do uso de NANO_CODESIZE, NANO_CONFSIZE e NANO_DATASIZE como múltiplos de setores de 512 bytes. NANO_CODESIZE define o tamanho das duas primeiras partições da imagem: code#1 e code#2. Elas devem ser grandes o suficiente para armazenar todos os arquivos que serão produzidos como resultado dos processos buildworld e buildkernel. NANO_CONFSIZE define o tamanho da partição do arquivo de configuração, portanto, não precisa ser muito grande, mas não a faça tão pequena que não possa armazenar todos os arquivos de configuração. Por fim, NANO_DATASIZE define o tamanho de uma partição opcional que pode ser usada no aparelho. A última partição pode ser usada, por exemplo, para manter arquivos criados dinamicamente no disco.

### 2.3.2. Funções Personalizadas

É possível ajustar o NanoBSD usando funções de shell no arquivo de configuração. O exemplo a seguir ilustra o modelo básico de funções personalizadas:

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

Um exemplo mais útil de uma função de personalização é o seguinte, que altera o tamanho padrão do diretório /etc de 5MB para 30MB:

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

Existem algumas funções de customização pré-definidas por padrão e prontas para uso:

- `cust_comconsole` - Desabilita o getty(8) nos dispositivos VGA (os nós de dispositivo /dev/ttyv*) e habilita o uso da porta serial COM1 como console do sistema.
- `cust_allow_ssh_root` - Permite que o usuário `root` faça login via sshd(8).
- `cust_install_files` - Instala arquivos do diretório nanobsd/Files, que contém alguns scripts úteis para administração do sistema.
- `cust_pkgng` - Instala pacotes do diretório nanobsd/Pkg, (necessita também do pacote pkg-* para inicialização).

### 2.3.3. Adicionando Pacotes

Pacotes podem ser adicionados a uma imagem do NanoBSD para fornecer funcionalidades específicas no aparelho. Para fazer isso, você pode escolher uma das seguintes opções:

- Adicionar `cust_pkgng` à variável `NANO_CUSTOMIZE`, ou
- Adicionar o comando `'customize_cmd cust_pkgng'` em um arquivo de configuração personalizado.

Ambos os métodos alcançam o mesmo resultado: executar a rotina `cust_pkgng`. Essa rotina percorrerá o diretório `NANO_PACKAGE_DIR` para encontrar todos os pacotes ou apenas a lista de pacotes na variável `NANO_PACKAGE_LIST`.

É comum, ao instalar aplicativos por meio do pkg em um ambiente FreeBSD padrão, que o processo de instalação coloque arquivos de configuração no diretório usr/local/etc e scripts de inicialização no diretório /usr/local/etc/rc.d. Portanto, após a instalação dos pacotes necessários, eles precisam ser configurados para iniciar imediatamente. Para fazer isso, os arquivos de configuração necessários devem ser instalados nos diretórios corretos. Isso pode ser feito escrevendo rotinas dedicadas ou utilizando a rotina genérica `cust_install_files` para organizar os arquivos corretamente a partir do diretório /usr/src/tools/tools/nanobsd/Files. Geralmente, também é necessário adicionar uma ou várias declarações no arquivo /etc/rc.conf para cada pacote.

### 2.3.4. Exemplo do arquivo de configuração

Um exemplo completo de um arquivo de configuração para construir uma imagem personalizada do NanoBSD pode 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 as opções de compilação e instalação podem ser encontradas na página do manual src.conf(5), mas nem todas as opções podem ou devem ser usadas ao construir uma imagem do NanoBSD. As opções de compilação e instalação devem ser definidas de acordo com as necessidades da imagem sendo construída.

Por exemplo, o cliente e servidor FTP podem não ser necessários. Adicionar `WITHOUT_FTP=TRUE` em um arquivo de configuração na seção `CONF_BUILD` evitará que eles sejam compilados. Além disso, se o aparelho NanoBSD não será usado para compilar programas, é possível adicionar `WITHOUT_BINUTILS=TRUE` na seção `CONF_INSTALL`; mas não na seção `CONF_BUILD`, pois eles serão usados para construir a imagem do NanoBSD.

Não compilar um conjunto específico de programas - por meio de uma opção de compilação - reduz o tempo total de compilação e diminui o tamanho necessário para a imagem do disco, enquanto não instalar o mesmo conjunto específico de programas não reduz o tempo total de compilação.

## 2.4. Atualizando o NanoBSD

O processo de atualização do NanoBSD é relativamente simples:

1. Construa uma nova imagem do NanoBSD, como de costume.
2. Faça o upload da nova imagem para uma partição não utilizada de um aparelho NanoBSD em execução.

   A diferença mais importante deste passo em relação à instalação inicial do NanoBSD é que, em vez de usar o arquivo _.disk.full (que contém uma imagem do disco inteiro), é instalada a imagem _.disk.image (que contém uma imagem de uma única partição do sistema).
3. Reinicie e inicie o sistema a partir da partição recém-instalada.
4. Se tudo correr bem, a atualização está concluída.
5. Se algo der errado, reinicie a partição anterior (que contém a imagem antiga que estava em funcionamento) para restaurar a funcionalidade do sistema o mais rápido possível. Corrija quaisquer problemas da nova compilação e repita o processo.

Para instalar uma nova imagem em um sistema NanoBSD em execução, é possível usar o script updatep1 ou updatep2, localizados no diretório /root, dependendo de qual partição o sistema atual está sendo executado.

Dependendo dos serviços disponíveis no host que está servindo a nova imagem do NanoBSD e do tipo de transferência preferido, é possível examinar uma dessas três opções:

### 2.4.1. Usando o comando ftp(1)

Se a velocidade de transferência estiver em primeiro lugar, use este exemplo:

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

### 2.4.2. Usando o comando ssh(1)

Se uma transferência segura for preferida, considere usar este exemplo:

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

### 2.4.3. Usando o comando nc(1)

Experimente este exemplo se o host remoto não estiver executando os serviços ftpd(8) ou sshd(8):

1. Primeiramente, abra um socket TCP em modo escuta no host que serve a imagem e envie a imagem para o cliente:

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



Verifique se a porta utilizada não está bloqueada pelo firewall para receber conexões originadas do host NanoBSD.

2. Conecte-se ao host que está servindo a nova imagem e execute o script updatep1:

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