

NanoBSD 简介

摘要

这篇文档提供了关于 NanoBSD 工具的介绍信息，这一工具可以用来创建用于嵌入式应用的 FreeBSD 系统映像，以适应存放到袖珍闪存 (Compact Flash) 卡 (或其它大容量存储介质) 上的需要。

目录

1. NanoBSD 简介	1
2. 如何使用 NanoBSD	1

1. NanoBSD 简介

NanoBSD 是 Poul-Henning Kamp 目前正在开发的一项工具。它可以用来创建用于嵌入式应用的 FreeBSD 系统映像，以便配合袖珍闪存 (Compact Flash) 卡 (或其他大容量存储介质) 使用。

这一工具也可以用来构建定制的安装映像，以简化通常称为 "计算设备 (computer appliances)" 的系统的安装和维护工作。计算设备通常在产品中将捆绑硬件和软件，或者换言之，所有的应用程序都是预先装好的。这些设备可以直接插到暨存的网络中，并 (几乎是) 立即投入使用。

NanoBSD 提供的功能包括：

- 可以和 FreeBSD 一样使用 Ports 和预编译包- 所有的应用程序都可以在 NanoBSD 映像中直接使用，而方式与 FreeBSD 完全一样。
- 不减少功能 - 能够使用 FreeBSD 做的任何工作，都可以在 NanoBSD 中使用，除非您在创建 NanoBSD 映像时，明确地删去它们。
- 所有对象在运行时均是只读的 - 可以安全地拔掉电源插销。在系统非正常关闭之后，无需运行 `fsck(8)`。
- 便于联编和定制 - 只需使用一个 shell 脚本和一个配置文件，您可以很容易地裁减和定制适于任意需求的映像。

2. 如何使用 NanoBSD

2.1. NanoBSD 的设计

一旦将映像存入介质，就可以用它来引导 NanoBSD 了。默认情况下，大容量存储器会划分为三个区：

- 两个映像区：`code#1` 和 `code#2`。
- 一个配置文件区，运行环境中，可以将其挂接到 `/cfg` 目录下。

这些分区默认情况下以只读方式挂接。

`/etc` 和 `/var` 目录均为 `md(4)` (malloc) 盘。

配置文件分区保存在 `/cfg` 目录。它包含了用于 `/etc` 目录的文件，在启动之后暂时以只读方式挂接。因此，在需要从 `/etc` 向 `/cfg` 目录复制所进行的、希望在重启时保持不变的配置时，需要进行一些额外的操作。

例 1. 在 `/etc/resolv.conf` 中进行需要保持的修改

```
# vi /etc/resolv.conf
```

[...]

```
# mount /cfg  
# cp /etc/resolv.conf /cfg  
# umount /cfg
```



只有在系统启动过程中，以及需要修改配置文件的场合，才需要挂载包含 /cfg 的那个分区。

在任何时候都保持挂载 /cfg 不是一个好主意，特别是当您把 NanoBSD 放在不适合进行大量写操作的分区时（由于文件系统的同步进程会定期向系统盘写一些数据）。

2.2. 构建 NanoBSD 映像

NanoBSD 映像是通过使用非常简单的 `nanobsd.sh` shell 脚本来构建的，这个脚本可以在 `/usr/src/tools/tools/nanobsd` 目录中找到。这个脚本建立的映像文件，可以用 [dd\(1\)](#) 工具复制到存储介质上。

构建 NanoBSD 映像所需的命令是：

```
# 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 进入 NanoBSD 构建脚本的主目录。
- ② 开始构建过程。
- ③ 进入构建好的映像文件所在的目录。
- ④ 在存储介质上安装 NanoBSD。

2.3. 定制 NanoBSD 映像

这可能是 NanoBSD 最为重要，同时也是您最感兴趣的功能。同时，您在开发 NanoBSD 应用时，这也是相当耗时的过程。

执行下面的命令将使 `nanobsd.sh` 从当前目录中的 `myconf.nano` 文件读取配置：

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

定制过程包含两步：

- 配置选项
- 定制函数

2.3.1. 配置选项

通过对配置进行设置，可以配置用以传递给 NanoBSD 构建过程中 `buildworld` 和 `installworld` 阶段的联编和安装选项，以及 NanoBSD 的主构建过程中的选项。通过使用这些选项可以削减系统的尺寸，使之能够放入 64MB 的存储。您还可以进一步通过这些选项来削减 FreeBSD，直到它只包含内核以及两三个用户环境文件为止。

配置文件中包含用以代替默认值的配置选项。最重要的语句包括：

- **NANO_NAME** - 本次构建的名称 (用于创建工作目录的名字)。
- **NANO_SRC** - 用以联编和构建映像的源码树的位置。
- **NANO_KERNEL** - 用以联编内核的配置文件的名称。
- **CONF_BUILD** - 用于传递给 **buildworld** 构建阶段的选项。
- **CONF_INSTALL** - 用于传递给 **installworld** 构建阶段的选项。
- **CONF_WORLD** - 用以传递给 **buildworld** 和 **installworld** 这两个构建阶段的选项。
- **FlashDevice** - 定义所用的介质类型。要了解进一步的细节，请参考 **FlashDevice.sub** 文件。

2.3.2. 定制函数

通过在配置文件中使用 shell 函数可以进一步微调 NanoBSD。下面的例子展示了定制函数的基本模式：

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

下面是一个更贴近实际的例子，它将默认的 /etc 目录尺寸，从 5MB 调整为 30MB：

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

除此之外，还有几个默认的预定义定制函数：

- **cust_comconsole** - 在 VGA 设备上禁止 **getty(8)** (/dev/ttyv* 设备节点) 并启用串口 COM1 作为系统控制台。
- **cust_allow_ssh_root** - 允许 **root** 通过 **sshd(8)** 登录。
- **cust_install_files** - 从 nanobsd/Files 目录中安装文件，这包含一些实用的系统管理脚本。

2.3.3. 安装预编译软件包

通过增加自定义的函数，可以在 NanoBSD 增加预编译的软件包。下面的函数会添加位于 /usr/src/tools/tools/nanobsd/packages 的全部预编译软件包：

```
install_packages () (  
    mkdir -p ${NANO_WORLDDIR}/packages  
    cp /usr/src/tools/tools/nanobsd/packages/* ${NANO_WORLDDIR}/packages  
    chroot ${NANO_WORLDDIR} sh -c 'cd packages; pkg_add -v *; cd ..;'  
    rm -rf ${NANO_WORLDDIR}/packages  
)
```

2.3.4. 配置文件举例

下面是一个用于构建定制的 NanoBSD 映像的完整例子：

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

```
CONF_BUILD='
NO_KLDLOAD=YES
NO_NETGRAPH=YES
NO_PAM=YES
'
```

```
CONF_INSTALL='
NO_ACPI=YES
NO_BLUETOOTH=YES
NO_CVS=YES
NO_FORTRAN=YES
NO_HTML=YES
NO_LPR=YES
NO_MAN=YES
NO_SENDMAIL=YES
NO_SHAREDOCS=YES
NO_EXAMPLES=YES
NO_INSTALLLIB=YES
NO_CALENDAR=YES
NO_MISC=YES
NO_SHARE=YES
'
```

```
CONF_WORLD='
NO_BIND=YES
NO_MODULES=YES
NO_KERBEROS=YES
NO_GAMES=YES
NO_RESCUE=YES
NO_LOCALES=YES
NO_SYSCONS=YES
NO_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
```

2.4. 更新 NanoBSD

更新 NanoBSD 相对而言较为简单：

1. 和之前一样构建新的 NanoBSD 映像文件。
2. 将新的映像放入正运行的 NanoBSD 设备中的一个未用的分区。

与之前最初安装 NanoBSD 的步骤相比，这一步骤最重要的区别在于这次不应使用 `_.disk.full` 文件（它包含整个盘的映像），而应安装 `_.disk.image` 映像（这个文件中，只包含一个系统分区）。

3. 重新启动，并从新安装的分区中启动系统。
4. 如果一切顺利的话，升级工作就完成了。
5. 如果发生了任何问题，则可以从先前的分区启动（其中包含了旧的、可用的映像），来尽可能快地恢复系统功能。接下来可以修正新联编的版本中存在的问题，并重复前述步骤。

要在正在运行的 NanoBSD 系统中安装新的映像，可以使用位于 `/root` 目录的 `updatep1` 或 `updatep2` 脚本，具体使用哪一个脚本，取决于正在运行的系统位于那个分区。

随时提供新 NanoBSD 映像所提供的服务，以及采用的传输方法的不同，您可以参考并使用下列三种方式之一：

2.4.1. 使用 `ftp(1)`

如果传输速度是第一要务，采用下面的例子：

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

2.4.2. 使用 `ssh(1)`

如果更倾向于安全传输，应参考下面的例子：

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

2.4.3. 使用 nc(1)

如果远程主机既不提供 [ftp\(1\)](#) 服务，也不提供 [sshd\(8\)](#) 服务：

1. 开始时，在提供映像的主机上开启 TCP 监听，并令其将映像文件发给客户机：

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



请确认您所使用的端口没有通过防火墙阻止来自 NanoBSD 客户机的联接请求。

2. 连接到提供新映像服务的主机，并执行 updatep1 脚本：

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