FreeBSD and NetBSD on Small x86 Based Systems

Dr. Adrian Steinmann <ast@marabu.ch>
Asia BSD Conference in Tokyo, Japan
March 17th, 2011
Introduction

Who am I?

• Ph.D. in Mathematical Physics (long time ago)
• Webgroup Consulting AG (now)
• IT Consulting Open Source, Security, Perl
• FreeBSD since version 1.0 (1993)
• NetBSD since version 3.0 (2005)
• Traveling, Sculpting, Go
Focus on Installing and Running FreeBSD and NetBSD on Compact Flash Systems

(1) Overview of suitable SW for small x86 based systems with compact flash (CF)

(2) Live CD / USB dists to try out and bootstrap onto a CF

(3) Overview of HW for small x86 systems

(4) Installation strategies: what needs special attention when doing installations to CF

(5) Building your own custom Install/Maintenance RAMdisk
FreeBSD for Small HW

Many choices! – Too many?

- PicoBSD / TinyBSD
- miniBSD & m0n0wall
- pfSense
- FreeBSD livefs, memstick
- NanoBSD
- STYX.

Others: druidbsd, Beastiebox, Cauldron Project, ...
PicoBSD & miniBSD

• PicoBSD (1998): Initial import into src/release/picobsd/ by Andrzej Bialecki <abial@freebsd.org>

  Geared towards floppy-based systems: “Building picobsd is still a black art. The biggest problem is determining what will fit on the floppies, and the only practical method is trial and error”

• TinyBSD in src/tools/tools/tinybsd (since 2006)

  http://martenvijn.nl/trac/wiki/TinyBSD

• miniBSD (2002): by Manual Kaspar as a precursor to m0n0wall (https://neon1.net/misc/minibsd.html) FreeBSD 4.x

• Other people made modifications to miniBSD to make it work for FreeBSD 5.x and 6.x
• Full-fledged firewall with VPN, traffic shaping, VLAN, captive portal capabilities all in less than 12MB
• Boots into steady state in less than 25 seconds on PC Engines
• Configuration via a PHP web GUI and stored in XML
• No access to a shell nor file system, system is run from RAM
• Very much “end-user” oriented (i.e., burn & install CF, configure IP on console and rest in GUI, forget)
• CF images for PC Engines and Soekris, early versions ran on 4.x and now currently runs on 6.x (2-3 versions /year since 2005) see m0n0.ch/wall/oldversions.php
• Website m0n0.ch/wall/ 2003-2010 Manuel Kasper
pfSense

• Started in 2004 as a fork of m0n0wall; As the name suggests, pfsense uses pf as default firewall filter (m0n0wall uses ipfw and more recently ipfilter)

• Contrary to m0n0wall, you can easily get shell access and modify the file system yourself; it also has packaging system to add additional features

• Very active development (1.x runs on 7.x) the forthcoming pfSense 2.0 will run on 8.x

• Website: www.pfsense.org/index.php

2004-2011 Chris Buechler & Scott Ullrich, now BSD Perimeter LLC wwwbsdperimeter.com
NanoBSD

- Part of FreeBSD source tree since 2004 in
  `src/tools/tools/nanobsd/
by Poul-Henning Kamp <phk@freebsd.org>

  “Nanobsd should make it very simple for people to create
  (CF-)disk images for embedded use of FreeBSD”

- Rewrite from Makefile to Shell Script in 2005

- Geared to 256MB - 4GB CF, with up to three
  partitions “live”, “fallback”, and “config”

- CF geometry needs to be specified case-by-case
  because fdisk is done on vnode device
• A remote managed firewall service since 1998 by Adrian Steinmann <ast@styx.ch>

• Customers have a “read-only” web GUI for status of their “firewall appliance”

• Remote administration via SSH cmd-line
Revision control: www.webgroup.ch/pi

• Remote OS upgrades via Secure Shell maintenance RAMdisk

• Tracks FreeBSD since 3.x, runs on 8.x
As usual, NetBSD is simple and straightforward – the recipe boils down to these command-line steps:

fdisk, disklabel, newfs, installboot, untar base.tgz and etc.tgz sets, make devices, fixup /etc/fstab – DONE!

Cookbook to install NetBSD onto a USB stick (2008):

wwwbsd nexus.com/NetBSD_onastick/install_guide.php
The NetBSD LiveKey project (2006) is a non-destructive NetBSD/i386 on a USB stick. It is composed of a tarball or zipfile to be uncompressed on a USB key without changing the original Filesystem (usually VFAT).

Uses grub to make USB stick bootable

Runs X11 with ion3, customizable

Developement has apparently ceased

... you will probably need about 256MB RAM to run the USB key smoothly ...
Jibbed is a NetBSD-based Live CD, and the version number tracks NetBSDs and is on NetBSD 5.1.

“... features select packages from pkgsrc, as well as auto-configuration for networking and graphics cards. This version contains the xfce4 window manager and uses Xorg (base). It features vnd compression and is only 400 MB in size. The minimum requirement is now an i686 or compatible CPU and 128 MB RAM ...”
Creating NetBSD USB installation media

pbraun.nethence.com/doc/sysutils_bsd/netbsd_usb.html

based on

ftp.netbsd.org/pub/NetBSD/misc/jmcneill/mkmemstick.sh

basic idea is to take CD image or binary sets from release and create a bootable image file which can be copied raw to a USB stick with dd.

```
makefs netbsd.img path_to_root
installboot netbsd.img /usr/mdec/bootxx_ffsv1
disklabel -R -F netbsd.img disklabel.conf
dd if=netbsd.img of=/dev/rsd1d bs=1024k
```
Summary on SW

- Lots of images for FreeBSD available, some of them a bit dated – the earliest ones tried to fit on 1.4MB floppies, with today’s kernel sizes that is impossible.

- Less images for NetBSD, mainly USB Stick cookbooks – probably because NetBSD is already modular and “small” enough (<100MB).

- Live CD distributions and USB stick images are now in standard ‘make release’ on FreeBSD as of 7.x.

- Search for lists like bengross.com/smallunix/ for small unix distributions.

- Join lists.freebsd.org/pipermail/freebsd-embedded/

- Read hubertf’s NetBSD blog at www.feyrer.de/NetBSD/bx/blosxom.cgi/index.front?-tags=embedded
Small SW calls for small HW!

• Search for “Embedded Systems” — albeit a misnomer (traditional embedded systems are something different)

• What is (was) PC/104 based HW?

• Advantages and disadvantages of PC/104 based systems
What is PC/104?

- PC/104 is simply an ISA bus in another, more compact and versatile form factor

- The bus doubles as the structural backbone for the system

- Some good starting points:
  - www.smallformfactors.com
  - www.controlled.com/pc104faq/
  - www.pc104.com/whatis.html
PC/104 “Stacks”

Tri-M Systems PC/104 CAN-TAINER™
PC/104 Container Designed For Hostile Environments

www.dpie.com/pc104/cantainer.html
PC-104 versus PC-104+

- PC-104+ is the PCI bus version of PC/104
- Additional connector (PC/104-compatible)
- But the modules are often quite expensive!
Single Board Computers (SBC)
3.5 inch “Biscuit PCs”

iEi WAFER-LUKE SBC with a fanless, on-board
VIA® LUKE 533MHz or 1GHz CPU, 2 x SATA with RAID 0,1, and JBOD function
support, VGA, CF Type II socket, PC/104 socket and Dual RTL8110SC GbE chipsets
Advantech, iEi, ... – SBC, PCM-58xx, WAFER, ...

- AMD LX; VIA C3, C5, C7; Intel LV/ULV; Intel Atom
- "Passive" cooling
- AT kbd, VGA/LCD, 2-4 COMs, [Audio]
- ATA HD support
- 1-2 Ethernet [Realtek or Intel], sometimes Gbit
- PC/104 socket, [USB]
- Some vendors known to sell such small x86-based hardware:
VIA EPIA Embedded Boards

Mini-ITX Form Factor

For example
VIA EPIA-EN12000EG 1200MHz Mini-ITX Fanless

Mini - ITX Form Factor

- Intel Atom (single and multicore) low power
- PCIe to support Gbit network and SATA speeds
- Today, Mini-ITX is a commodity available from well known PC motherboard manufacturers like Asus, AsRock, DFI, Gigabyte, MSI, SuperMicro, ...

- Often the power supply or the companion chips require active cooling, so ask yourself:

  Will the complete system really be fanless (quiet)?
  Will the system be stable when operated fanless?
One of the Smallest (not x86-based)

“Gumstix” form factor www.gumstix.com

www.feyrer.de/NetBSD/blog.html?-tags=gumstix
Kiyohara Takashi <kiyohara@netbsd.org> has worked on porting NetBSD to Gumstix, SheevaPlug and Plathome's openblocks.plathome.com OpenBlockS600 (AMCC PowerPC 405EX) tracker.netbsd.org/pub/NetBSD/misc/kiyohara/tmp/
“The world is changing very fast. Big will not beat small anymore. It will be the fast beating the slow.”

– Rupert Murdoch (Chairman of News Corporation)

Researchers at Harvard and MITRE produce world’s first programmable nanoprocessor

February 9, 2011

Nanowire tiles can perform arithmetic and logical functions and are fully scalable

The versatile, nanoscale circuits are assembled into tiny tile-like nanoprocessors from sets of precisely engineered and fabricated germanium-silicon wires with functional oxide shells, having a total diameter of only 30 nanometers. Shown here are atomic force (left) and optical microscopy (center) images of a programmable nanowire nanoprocessor, and a corresponding schematic (right) of the nanowire circuit architecture.
net5501-70

500 Mhz Geode LX CPU, 512 Mbyte DDR-SDRAM, 4 Ethernet, 2 Serial, USB connector, CF socket, 44 pins IDE connector, SATA connector, 1 Mini-PCI socket, 3.3V PCI connector.
Hi Everybody,

Maybe it's time to tell a little more....The net6501 is moving forward and I do expect to have hardware ready in Q1 2010, although I am known to be an optimist :-)

The net6501 will basically be like the other boards, just faster, with more memory and PCI Express expansion.

There will be both a 2 and 4 port gigabit ethernet version, using Intel controllers, where the 2 ports version will be targeting small servers, with up to 4 of them in a 1U case....

And yes, it will be based on the next generation Intel Atom processor, the Pineview platform, in both single and dual core versions and with up to 2 Gbyte DDR2-SDRAM soldered on.

And it will still be low power and high reliability, with passive cooling. Ok, a tiny server with two 2.5" 10K rpm SATA drives will need a small fan....

A new goodie will be onboard NiMH batteri charger/controller.

Best Regards,

Soren Kristensen

CEO & Chief Engineer
Soekris Engineering, Inc.
www.pcengines.ch

**alix2d3**

3 LAN / 1 miniPCI / LX800 / 256 MB / USB

- CPU: 500 MHz AMD Geode LX800
- DRAM: 256 MB DDR DRAM
- Storage: CompactFlash socket, 44 pin IDE header
- Power: DC jack or passive POE, min. 7V to max. 20V
- Three front panel LEDs, pushbutton
- Expansion: 1 miniPCI slot, LPC bus
- Connectivity: 3 Ethernet channels (Via VT6105M 10/100)
- I/O: DB9 serial port, dual USB port
- Board size: 6 x 6" (152.4 x 152.4 mm)
- Firmware: tinyBIOS
PC Engines: Even smaller!

alix3d3 = 1 LAN / 2 miniPCI / LX800 / 256 MB / USB / VGA / audio - designed for thin clients or networked audio players.
Default Serial BIOS parameters for PC Engines and Soekris

- PC Engines factory default parameters
  
  **38400 8N1**
  
  Type “S” at power-on for BIOS

- Soekris factory default parameters
  
  **19200 8N1**
  
  Type “Control-P” at power-on for BIOS
Default Serial BIOS parameters for PC Engines and Soekris

• **PC Engines** factory default parameters
  
  38400 8N1

  Type “S” at power-on for BIOS

• **Soekris** factory default parameters
  
  19200 8N1

  Type “Control-P” at power-on for BIOS
Compact Flash (CF)

- Most are good for a million write/erase cycles
  CF/SD Performance Database at
  www.robgalbraith.com/bins/multi_page.asp?cid=6007

- Superblocks are saved often so a million writes are not enough (noatime option to mount read-write)

- Best is to mount read-only - never a fsck again!

- Mounting CF read-only is easy on FreeBSD:
  touch /etc/diskless
  /conf/base/... for /etc/rc.initdiskless

- This same script also works on NetBSD!
Mount CF read-only, and then mount RAMdisk for read-write areas

On FreeBSD /dev is a devfs, i.e. ‘writable’

For others:

/sbin/mdmfs -S -i 4096 -s size -M md mount_point

When /dev/console is missing: NetBSD creates a new /dev on a RAMdisk using /dev/MAKEDEV

For others:

/sbin/mount_mfs -i 4096 -s size swap mount_point
How to Install onto a CF system without a CD or Floppy drive, video console, nor keyboard?

- First install and setup the OS on (laptop) hard disk then install from there onto CF for target system

- Essential: **PCMCIA** CF/IDE adapter (aka CF/ATA adapter) to initialize the CF via the laptop

- USB CF Adapters do not work well in all cases because they often assume a non-BIOS geometry (not corresponding to real C/H/S addressing). This results in the feared “no operating system on disk” message when booting the CF on the target system
How to Install onto a CF system without a CD or Floppy drive, video console, nor keyboard?

Installation via PXE netboot!

BIOS and NIC needs to support Intel® PXE support

“FreeBSD Jumpstart Guide”

jdc.parodius.com/freebsd/pxeboot_serial_install.html

people.freebsd.org/~alfred/pxe/en_US.ISO8859-1/articles/pxe/article.html

Diskless NetBSD HOW-TO

www.netbsd.org/docs/network/netboot/intro.i386.html

bsdsupport.org/2007/01/netbsd-pxe-boot-install-without-nfs/
Setting Serial Console

Serial console NetBSD:

```
# installboot -v -m i386 -o
timeout=3,console=com0,speed=38400 -t
ffs /dev/rwd1a /usr/mdec/bootxx_ffsv1
```

Serial console FreeBSD:

```
$ cat /boot.config
-h -S38400
```

Disable AT Keyboard, no video:

```
$ cat /boot/loader.conf
hint.atkbdc.0.disabled="1"
hint.sc.0.disabled="1"
hint.vga.0.disabled="1"
```
FreeBSD Kernel tuning
GEODE and “SOEKRIS”

For older Geode (pre AMD Geode-LX) CPUs
options CPU_GEODE
options CPU_SOEKRIS

• Creates watchdog device (\texttt{/dev/fido}) on Advantech, PC Engines, and Soekris

• Creates LED devices (\texttt{/dev/led/*}) on PC Engines and Soekris

– see \texttt{/usr/src/sys/i386/i386/geode.c}
Kernel Configuration for Crypto Accelerators

Enable in-kernel cryptography (hardware or software)

pseudo-device crypto
pseudo-device swcrypto

Geode LX Security Block crypto accelerator (i.e., PC Engines ALIX, Soekris net5501)

glxsb* at pci?

Hifn 7751, 7951, 7811, 7955, and 7956 chipsets (i.e. Soekris vpn1211)

hifn* at pci? dev ? function ?

Crypto and RNG in VIA C3, C7 and Eden processors (i.e. VIA EPIA Mini-ITX)

options VIA_PADLOCK
Kernel Configuration for Crypto Accelerators

Enable in-kernel cryptography (hardware or software)

```
device crypto
device cryptodev
```

Geode LX Security Block crypto accelerator (i.e., PC Engines ALIX, Soekris net5501)
```
device glxsb
```

Hifn 7751, 7951, 7811, 7955, and 7956 chipsets (i.e. Soekris vpn1211)
```
device hifn
```

Crypto and RNG in VIA C3, C7 and Eden processors (i.e.VIA EPIA Mini-ITX)
```
device padlock
```
Kernel Configuration for Crypto Accelerators

Enable in-kernel cryptography (hardware or software)

pseudo-device crypto
go to
pseudo-device swcrypto
go to
devcrypto
go to
dev cryptodev

Geode LX Security Block crypto accelerator (i.e., PC Engines ALIX, Soekris net5501)

glxsb* at pci?

device glxsb

Hifn 7751, 7951, 7811, 7955, and 7956 chipsets (i.e. Soekris vpn1211)

hifn* at pci? dev ? function ?

device hifn

Crypto and RNG in VIA C3, C7 and Eden processors (i.e. VIA EPIA Mini-ITX)

options VIA_PADLOCK

device padlock
Building kernel and userland

See also [http://www.netbsd.org/docs/updating.html](http://www.netbsd.org/docs/updating.html)

```bash
$ cd /usr/src

$ cvs up -Pd

$ ./build.sh -O ../obj -T ../tools tools

$ ./build.sh -O ../obj -T ../tools

kernel=MYKERN

The kernel configuration file is in

/usr/src/sys/arch/i386/conf/MYKERNEL

Note that these userland- and kernel- compilation steps do not require superuser privileges
Installing kernel and userland

Installation does require superuser privileges:

```
$ su

# mv /netbsd /netbsd.old
# mv /usr/obj/sys/arch/i386/compile/MYKERNEL/netbsd /
# shutdown -r now
```

Test if new kernel is working, otherwise boot /netbsd.old from boot loader

```
$ cd /usr/src
$ su
# ./build.sh -O ../obj -T ../tools -U install=/
```
Building and installing kernel


# cd /usr/src

# cvs up -Pd

See http://www.freebsd.org/doc/handbook/anoncvs.html on how to keep current on a FreeBSD branch via CVS

# make buildkernel KERNCONF=MYKERNEL

# make installkerneln

The kernel configuration file is in

/usr/src/sys/i386/conf/MYKERNEL
Building and installing world

`# make buildworld`

`# reboot`

Test if new kernel works, otherwise reboot with

`/boot/kernel.old/kernel` from loader on the console

`# mergemaster -p`

`# make installworld`

`# mergemaster`

This should be done in single user if the machine would otherwise be busy
Summary First Half

☑ Many all-in-one FreeBSD images for “small platforms” exist (minimal install of FreeBSD is about 130MB)

☑ NetBSD minimal is small enough for small today’s small platforms (base.tgz + etc.tgz sets requires 80 MB)

☑ Small Hardware
  Look for embedded systems, fanless systems, and don’t be afraid of PC/104 - it’s just an ISA bus

☑ Serial consoles, RAMdisks, and read-only filesystems on CF are your friends

☑ Build custom kernels on a fast “build” system to take full advantage of HW features (crypto accelerators)
Outlook Second Half

1. A closer look at how *BSD boots/installs
   - The install CD
   - The boot sequence
   - Building crunched binaries

2. The missing bits needed for building a
   networked maintenance RAMdisk

3. Some details of building and installing the
   maintenance RAMdisk

4. Using a “RAMdisk maintenance environment”
   to install/upgrade OS (demonstration)
Booting FreeBSD (3 stage boot)


BIOS POST “executes” first 446 bytes of sector 0 – this is a MBR, i.e. the boot0 or a boot0sio program plus the disk slice table (this is also where the dreaded ‘no operating system on disk’ or Fn-key loop can happen). The “active” PC slice is chosen and its first sector, i.e., the first 512 bytes (boot1) in that slice are executed.

(1) boot1 (512 bytes) executes boot2 also in that active PC slice

(2) boot2 understands the FreeBSD disk label as well as the FreeBSD unix file system so it can load /boot/loader from that slice

>>FreeBSD/i386 BOOT
Default: 1:ad(1,a)/boot/loader

boot:

(3) /boot/loader sets kenv(1) variables, loads kernel and modules, and finally boots FreeBSD

BTX loader 1.0 BTX version is 1.01
...
Hit [Enter] to boot immediately, or any other key ...
OK
fdisk & bsdlabel vs gpart

GEOM is the “new” (2004) I/O abstraction for FreeBSD, see www.bsdcan.org/2004/papers/geom-tutorial.pdf it’s modular, stackable, POLA, DWIM, policy-free, by phk@

GEOM gives us all the good things like
Disk Striping, Mirroring, RAID, Encryption, ...

GEOM also gives us transitional grief with fdisk and disklabel versus gpart
a workaround sysctl variable kern.geom.debugflags=16 (the 'foot-shooting' bit)
is sometimes required between FreeBSD 7.x and early 8.x to run the “old” commands
because GEOM now tastes the labels with GEOM_PART_BSD and not (legacy) GEOM_BSD
http://svn.freebsd.org/viewvc/base?view=revision&revision=186240

Worse, sometimes GENERIC kernel complains profusely as
GEOM_PART_BSD tastes the label saying “geometry doesn’t match label”

“Fixes”:
On nbsd before installing fbsd: dd if=/dev/zero of=/dev/wd0 bs=512 count=1024
On FreeBSD: fdisk -a -1 ... becomes gpart set -a active -i 1 ...
/boot/loader

The FreeBSD **loader(8)** is a statically linked standalone executable providing a Forth interpreter and a set of builtin commands to assist in pre-configuration and recovery

‘The main drive behind these commands is user-friendliness’

Today, the main reason for `/boot/loader`’s existence is to set all the kernel environment variables (**kenv**), present a boot menu, and (possibly) a splash image

Some example `/boot/loader` commands

- help
- set
- more
- words
- show
- ls
- 1000 ms
- include
Booting NetBSD (2 stage boot)

www.netbsd.org/docs/guide/en/chap-misc.html#chap-misc-bootmanager

BIOS POST “executes” first 446 bytes of sector 0 – this is a MBR, NetBSD has a few

Normal boot code /usr/mdec/mbr
  Like DOS: just boot from active partition
Bootselector /usr/mdec/mbr_bootsel
  Choice between partitions
Extended Bootselector /usr/mdec/mbr_ext
  Load NetBSD from an extended partition
Serial Bootselector /usr/mdec/mbr_com0
  Same as mbr_ext but will read and write from the first serial port.
  It assumes that the BIOS has initialized the baud rate.
Serial Bootselector /usr/mdec/mbr_com0_9600
  Same as as mbr_com0, additionally it initializes the serial port to 9600 bps.

NetBSD bootstrap consists of two parts: a primary bootstrap written into the disklabel area of the file system by installboot, and a secondary bootstrap that resides as an ordinary file in the file system.

cp /usr/mdec/boot /boot
installboot -v -o timeout=5 /dev/rwd0a /usr/mdec/bootxx_ffsv1
FreeBSD Install CD

$ cat /cdrom/boot/loader.conf
mfsroot_load="YES"
mfsroot_type="mfs_root"
mfsroot_name="/boot/mfsroot"

$ zcat /cdrom/boot/mfsroot > /tmp/m
# mdconfig -a -t vnode -f /tmp/m
md0
# mount /dev/md0 /mnt

$ file /mnt/stand/*

/mnt/stand/-sh:              ELF 32-bit LSB executable, Intel
80386, version 1 (FreeBSD), for FreeBSD 7.1, statically
linked, FreeBSD-style, stripped

... 

/mnt/stand/zcat:            ELF 32-bit LSB executable, Intel
80386, version 1 (FreeBSD), for FreeBSD 7.1, statically
linked, FreeBSD-style, stripped
$ cat boot.cfg
menu=Install NetBSD:load /miniroot.kmod;boot netbsd
menu=Install NetBSD (no ACPI):load /miniroot.kmod;boot netbsd -2
menu=Install NetBSD (no ACPI, no SMP):load /miniroot.kmod;boot netbsd -12
menu=Drop to boot prompt:prompt

$ ls -l miniroot.kmod
-rw-r--r-- 1 root wheel 1019259 Feb  3 02:33 miniroot.kmod

$ file miniroot.kmod
/mnt/miniroot.kmod: gzip compressed data, from Unix, last modified: Tue Feb  3 02:26:42 2009, max compression

$ ls -l netbsd
-rw-r--r-- 1 root wheel 5046737 Feb  3 02:33 netbsd
$ file netbsd
netbsd: gzip compressed data, was "netbsd GENERIC", from Unix, max compression
crunchgen

Makes one statically linked binary for a set of programs (/rescue)

Toy example

i. crunchgen pls.conf

ii. make -f pls.mk

iii. ./pls

Compare sizes of /bin/ps, /bin/ls, ./pls
Build a Maintenance RAMdisk

A Straightforward Plan

i. Make a list of commands we need for system installation via a SSH session

ii. Use crunchgen to combine all commands into one “static” binary

iii. Craft a RAMdisk filesystem image which configures network and starts SSH daemon

iv. Boot into this RAMdisk image like the Install CD


described this method for building “tiny systems” NetBSD in 2003
Yet not so easy, because

• We specifically want some programs on RAMdisk which turn out to be crunchgen-unfriendly:
  • SSH doesn’t crunch “out of the box”
  • By default, SSH links in far too many libraries
  • Programs based on GEOM classes require the runtime loader

• Network parameters should be text-file editable, and the RAMdisk md_image should stay generic
Crunching SSHD fails

- This crunchgen.conf fragment fails with straightforward configuration:

```bash
buildopts -DNO_KERBEROS
buildopts -DNO_PAM
srcdirs /usr/src/secure/usr.bin
srcdirs /usr/src/secure/usr.sbin
progs scp ssh sshd
libs -lssh -lutil -lz -lcrypt
libs -lcrypto -lmd
```

link phase wants `libwrap.a` and `libpam.a` routines
Crunching SSHD fixed

• Change hard-coded #defines directly in

/usr/src/crypto/openssh/config.h

#define LIBWRAP
#define USE_PAM
#define HAVE_LIBPAM
#define HAVE_PAM_GETENVLIST
#define HAVE_SECURITY_PAM_APPL_H
#define XAUTH_PATH
NetBSD crunches using Makefile technology – what else?

Makefile essentials

```makefile
... 

IMAGE= ramdisk-$\{BOOTMODEL\}.fs 
IMAGESIZE= 5000k 

.include "${NETBSDSRCDIR}/distrib/common/Makefile.distrib"

CRUNCHBIN= ramdiskbin 
LISTS= $\{.CURDIR\}/list 
MTREECONF= ${DISTRIBDIR}/common/mtree.common 

PARSELISTENV+= CUSTOM_SSHD=$\{.CURDIR\}/custom_sshd 

# This propagates through to the link of ramdiskbin 
CRUNCHENV += MKSKEY=no MKWRAP=no MKPAM=no MKKERBEROS=no MKSHARE=no RELEASE_CRUNCH=yes 

... 

.include ${DISTRIBDIR}/common/Makefile.crunch 
.include ${DISTRIBDIR}/common/Makefile.image 

MDSETTARGETS= \
  ${NETBSDOBJDIR}/sys/arch/i386/compile/INSTALL_FLOPPY/netbsd ramdisk-custom.fs netbsd-RAMDISK 

.include ${DISTRIBDIR}/common/Makefile.mdset 

.include <bsd.prog.mk> 

and a "list" file (almost like a crunchgen.conf)
Crunching SSHD fixed

Simply remove offending stuff from /usr/src/usr.bin/ssh/sshd/Makefile

$ cat custom_sshd/Makefile

.include <bsd.own.mk>

SSHDIST?= $(NETBSDSRCDIR)/crypto/dist/ssh

.PATH: $(SSHDIST)

CPPFLAGS+=-I$(SSHDIST) -DHAVE_LOGIN_CAP -DHAVE_MMAP -DHAVE_OPENPTY

LDADD+= -lssh -lcrypto -lcrypt -lz -lutil

DPADD+= $(LIBSSH) $(LIBCRYPTO) $(LIBCRYPT) $(LIBZ) $(LIBUTIL)

CPPFLAGS+=-DSUPPORT_UTMP -DSUPPORT_UTMPX

PROG= sshd

MAN= sshd.8

SRCS= sshd.c auth-rhosts.c auth-passwd.c auth-rsa.c auth-rh-rsa.c \
    sshpty.c sshlogin.c servconf.c serverloop.c uidswap.c \
    auth.c auth1.c auth2.c auth-options.c session.c \
    auth-chall.c auth2-chall.c groupaccess.c \
    auth-skey.c auth-bsdauth.c auth2-hostbased.c auth2-kbdint.c \
    auth2-none.c auth2-passwd.c auth2-pubkey.c \
    monitor_mm.c monitor.c monitor_wrap.c \
    kexdhs.c kexgexs.c

.include <bsd.prog.mk>
GEOM and ZFS use `dlopen()`

The GEOM and ZFS commands use `dlopen()` to load classes from `/lib/geom` dynamically

```cpp
gem(4), gconcat(8), geli(8),
glabel(8), gmirror(8), gnop(8),
graid3(8), gshsec(8), gstripe(8),
gvirstor(8), zfs(1M), zpool(1M)
```

...yet it is exactly these commands – among others – that we need most in a maintenance environment!
“Mostly static” linking

Include `rtld(1)` in RAMdisk:

```
/libexec/ld-elf.so.1
```

then, for GEOM classes link dynamically:

```
ldd /lib/geom/*.so
/lib/geom/geom_concat.so
/lib/geom/geom_eli.so
   libmd.so.3 => /lib/libmd.so.3 (0x2815a000)
   libcrypto.so.4 => /lib/libcrypto.so.4 (0x28168000)
/lib/geom/geom_label.so
/lib/geom/geom_mirror.so
   libmd.so.3 => /lib/libmd.so.3 (0x28155000)
/lib/geom/geom_nop.so
/lib/geom/geom_raid3.so
   libmd.so.3 => /lib/libmd.so.3 (0x28154000)
/lib/geom/geom_shsec.so
/lib/geom/geom_stripe.so
```
crunchgen(1) with a twist

Linking “mostly static” from man crunchgen(1)

libs_so libspec ...

A list of library specifications to be dynamically linked in the crunched binary. These libraries will need to be made available via the run-time link-editor rtld(1) when the component program that requires them is executed from the crunched binary. Multiple libs_so lines can be specified.

$ ls -RF lib libexec
lib:
  geom/
  libbsdxml.so.3
  libc.so.7
  libcrypto.so.5
  ld-elf.so.1*

lib/geom:
  geom_cache.so
  geom_concat.so
  geom_eli.so
  geom_journal.so
  geom_label.so
  geom_multicat.so
  geom_raid3.so

libexec:
  ld-elf.so.1*
What’s on the RAMdisk?

- sh
  - du
  - mkdir

  - expr
    - hostname

  - cat
  - chflags
  - chgrp
  - chmod
  - chown
  - chroot

  - cp
  - date
  - df
  - link
  - ln
  - ls

  - sh
  - sleep

  - mv

  - ps
  - pwd
  - realpath
  - test
  - touch
  - tset
  - unlink

  - rm
  - rmdir
Basics on RAMdisk

- sh
  [ du
  mkdir
  expr
  hostname
  init
  mv
  kenv
  kill
  ps
  pwd
  mkdir
  realpath
  touch
  test
  tset
  chflags
  chmod
  chgrp
  chown
  chroot
  cp
  date
  df
  ldconfig
  ln
  ls
  link
  ln
  ls
  rm
  rmdir
  sh
  sleep
  stty
  test
  tset
  vi
  disklabel
  gnop
  mdmfs
  rmdir
  zcat
SysAdmin on RAMdisk

atacontrol  badsect  dumpfs  mknod  mount
boot0cfg    bsdlabel     mount_cd9660  mount_devfs  mount_fdescfs
                mount_linprocfs
fastboot  fasthalt  halt
fsck  fsck_4.2bsd  fsck_ffs  fsck_ufs
gbde  kldconfig
clri
cli
gei  kldload  kldstat  kldunload

dd

diskinfo  disklabel

mdconfig  mdmfs

reboot  tunefs  umount

swapctl  swapoff  swapon  sync  sysctl
newfs
sync

disklabel

zfs
zpool
Networking on RAMdisk

- `route`
- `ifconfig`
- `ping`
- `dhclient`
- `dhclient-script`
More networking RAMdisk

route
scp

mount_nfs

slogin
ssh
sshd

ifconfig

ip
ipf
ipfw

pfctl
ping

dhclient
dhclient-script
ggatemc
ggated

ggatel

df
dhclient-script

"Installing and Running FreeBSD and NetBSD on Small x86 Based Systems"

AsiaBSDCon Tutorial March 17, 2011 in Tokyo, Japan

Dr. Adrian Steinmann <ast@marabu.ch>
Archiving tools on RAMdisk

- dump
- bunzip2
- bzcat
- bzip2
- gunzip
- gzcat
- gzip
- pax
- tar
- rdump
- restore
- zcat
- rrestore
Editors on the RAMdisk

ed
ex

sed

red
and last but not least ...

Requires a (small) /usr/share/misc/termcap

Only 5306 bytes (not 204798 bytes!) supporting vt100, vt220, xterm, screen, ansi, AT386

Being on RAMdisk, the required /var/tmp exists

vi
Maintenance RAMdisk

- sh
  [ atacontrol
  badsect
  boot0cfg
  bsdlabel
  bunzip2
  bzip2
  camcontrol
  cat
  chflags
  chgrp
  chmod
  chown
  chroot
  cli
  cp
  date
  dd
  df
  dhclient
  dhclient-script
diskinfo
disklabel
  dmesg
du
dump
dumpfs
growfs
gshsec
gstripe
  gunzip
gvirstor
gzcat
gzip
halt
hostname
ifconfig
init
ipf
ipfw
kenv
kill
kldconfig
kldload
kldstat
kldunload
link
ln
ls
mdconfig
mdmfs
mini_crunch
mkdir
  mkfs
mount
mount_cd9660
mount_devfs
mount_fdescfs
mount_linprocfs
mount_nfs
mount_procfs
mount_std
mv
newfs
nex
nice
nvi
nview
pax
pfctl
ping
ps
pwd
rdump
realpath
reboot
recoverdisk
red
restore
rm
rmdir
route
rrestore
scp
sed
sh
sleep
slogin
ssh
sshd
stty
styxinstall
swapctl
swapoff
swapon
sync
sysctl
tar
test
touch
tset
tunefs
unmount
unlink
vi
view
zcat
zfs
zpool
NetBSD 5 custom RAMdisk

netbsd-RAMDISK# df -h
Filesystem      Size   Used    Avail %Cap Mounted on
/dev/md0a       4.8M    4.5M     381K  92% /
mdfs:16         1.0M    36K     975K  3% /dev

netbsd-RAMDISK# ls /*bin /usr/*
netbsd-RAMDISK# ls /*bin /usr/*

/bin:
- sh    cp    echo    kill    mv    rm    stty
[    date    ed    ln    pax    rmdir    sync
cat    dd    expr    ls    ps    sh    test
chmod    df    hostname    mkdir    pwd    sleep

/sbin:
atactl    dump    mbrlabel    mount_ufs    route
badsect    dump_lfs    mknod    newfs    rrerestore
cdconfig    fdisk    modload    newfs_lfs    scsictl
cgdconfig    fsck    modunload    ping    swapctl
ciri    fsck_ffs    mount    raidctl    swapon
dhclient    fsck_lfs    mount_cd9660    rcorder    sysct1
dhclient-script    halt    mount_ffs    rdump    umount
disklabel    ifconfig    mount_ffs    rdump_lfs    umount
dkctl    init    mount_mfs    reboot
dmesg    ldconfig    mount_nfs    reboott

/usr/bin:
bunzip2    du    gunzip    printf    ssh    tset
bzcat    env    gzcat    scp    ssh-keygen    vi
bzip2    ex    gzip    sed    tar    zcat
chflags    ftp    passwd    slogin    touch

/usr/mdec:
boot    bootxx_ffsv2    mbr_bootsel    mbr_com0_9600
bootxx_ffsv1    mbr    mbr_com0    mbr_ext

/usr/sbin:
chgrp    chroot    installboot    pwd    mkdb    vnconfig
chown    dumpfs    mdconfig    sshd    wiconfig
On-disk: 8 MB / Runs in 42 MB

- The boot loader is able to preload gzip-compressed RAMdisk images

- Additional on-disk (CF) usage is minimal < 8MB
  $ du -h k.GENERIC.gz fs.8.2-RAMDISK.gz
  3.6M k.GENERIC.gz
  4.3M fs.8.2-RAMDISK.gz

- In RAM currently defined as 14.0 MB md0
  # mdconfig -l -u 0
  md0     preload   14.0M
$ ls -sh netbsd-RAMDISK.gz
6.6M netbsd-RAMDISK.gz

- The boot program is able to load gzip-compressed netbsd kernels containing RAMdisk images
- RAMdisk space usage is negligible on today CF sizes
- Running RAMdisk is currently defined as 5.0 MB filesystem of which 4.5 MB is used

On-disk: 6.6 MB / Runs in 20 MB + 4.5 MB
On-disk: 7.9 MB / Runs in 27 MB + 14 MB
The RAMdisk personality

- The compressed RAMdisk image stays generic
- The key idea is to pass all machine-specific parameters via the kernel environment `kenv(1)`
- These can be set in a `/boot/maint/params` file which is an editable textfile and is included by the loader
- Those values are read back into RAMdisk user space via `kenv(1)` calls
Example personality

OK more /boot/maint/params

*** FILE /boot/maint/params BEGIN ***
set maint.ifconfig_sis0="192.168.1.200/24"
set maint.defaultrouter="192.168.1.1"
set maint.domain="mydomain.ch"
set maint.nameservers="192.168.1.1 192.168.1.100"
set maint.sshkey_01a="ssh-dss AAAAB3N.............cZ9"
set maint.sshkey_01b="ucifE5QoUN...(120 chars)..PYik"
...
*** FILE /boot/maint/params END ***

RAMdisk# sed -ne /kenv/p /etc/rc
kenv | sed -ne 's/^maint\.\//p' >> /etc/params
One way into RAMdisk

By replacing /boot/loader.rc with:

```bash
include /boot/loader.4th
start
unload
load /boot/maint/k.CUSTOM
load -t md_image /boot/maint/fs.6.0-STYX
include /boot/maint/params
set vfs.root.mountfrom=ufs:/dev/md0
autoboot 10
```
Booting into RAMdisk

Change `default=1 menu in /boot.cfg`

```
$ cat /boot.cfg
menu=Boot normally:boot netbsd
menu=Boot single user:boot netbsd -s
menu=Disable ACPI:boot netbsd -2
menu=Disable ACPI and SMP:boot netbsd -12
menu=Drop to boot prompt:prompt
menu=Maintenance RAMdisk:boot netbsd-RAMDISK
default=6
timeout=5
```

The RAMdisk needs to setup networking specific to this machine so that sshd will be accessible remotely.
Booting into RAMdisk

Have single user shell execute /etc/rc from .profile

```
ETCRC_DONE=/done_etc_rc

if [ ! -e ${ETCRC_DONE} ]
then
    echo "Running /etc/rc autoboot from .profile"
    /bin/sh /etc/rc autoboot && touch ${ETCRC_DONE}
    exit 0
fi
```

Because init is called with "-s" option and thus would otherwise leave machine in single user mode.
Thank you very much for attending this tutorial!

steinmann.com/AsiaBSDCon2011/SmallBSDTutorial.tbz