Installing and Running FreeBSD and NetBSD on Small x86-based Systems

Dr. Adrian Steinmann <ast@marabu.ch>

Asia BSD Conference in Tokyo, Japan
Thursday, March 12th, 2009
10:00-12:30, 14:00-16:30

Installing and Running FreeBSD and NetBSD on Small x86-based Systems

Dr. Adrian Steinmann <ast@marabu.ch>

Asia BSD Conference in Tokyo, Japan
Thursday, March 12th, 2009
10:00-12:30, 14:00-16:30

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

Introductions

Who are you?

- Name and where you come from
- Your (favorite) work and play
- Why you're here today
- Do you have small x86 system experience?
- If so, which one and what OS did you use?

Schedule for the day

1. Overview SW and HW for small systems
2. Secrets about Compact Flash (CF) Installations
3. The Maintenance RAMdisk in Action (Demos)
4. You install and use a Maintenance RAMdisk on your own systems

FreeBSD for Small HW

Many choices! – Too many?

- PicoBSD
- miniBSD
- m0n0wall
- pfSense
- Freesbie Live CD
- NanoBSD
- STYX.

PicoBSD

- Initial import into src/release/picobsd/ in 1998 by Andrzej Bialecki <abial@freebsd.org>
- Geared towards floppy-based systems
- man picobsd(8):

> *"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"*

people.freebsd.org/~picobsd

> *" The PicoBSD pages have been removed since they were seriously out-of-date"*

miniBSD

- Manuel Kasper's <mk@neon1.net> precursor to m0n0wall in 2002 for FreeBSD 4.x: **`neon1.net/misc/minibsd.html`**
- Cookbooks on how to whittle down the FreeBSD base system using a chroot(8) environment
- A few utility scripts (for example, to find shared object dependencies)
- Making a comeback?

m0n0wall

- Full-fledged firewall with VPN, traffic shaping, VLAN, and captive portal capabilities
- Configuration via a PHP web GUI and stored in XML
- No access to a shell nor to the 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)
- Distributed as CF images for PC Engines and Soekris platforms, currently runs on 6.x

- **`m0n0.ch/wall/`** 2003-2008 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)
- Web GUI inspired by m0n0wall (i.e., very slick)
- 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 developement (runs on 7.x)
- **www.pfsense.org/index.php**
  2004-2009 Chris Buechler & Scott Ullrich

FREESBIE
FREEBSD LIVE CD

- By Italian FreeBSD User Group in 2004-2008 (Gruppo Utenti FreeBSD Italia, **www.gufi.org**) Main developer Matteo Riondata has stopped working on it (has to finish his studies)
- Currently 6.x-based
- Not really small, but a good RAMdisk model to study
- **www.freesbie.org** (when up)
- A revived miniBSD appears to be near this community now

NanoBSD

- In FreeBSD since 2004 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 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

STYX.

DER WIRKSAME FIREWALL.

- A remote managed firewall *service* since 1998 by Adrian Steinmann <ast@styx.ch>
- Customers have a mainly-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 7.x

NetBSD on a Stick

"Of course it runs NetBSD"

Short cookbook to install NetBSD onto a USB stick:

www.bsdnexus.com/NetBSD_onastick/install_guide.php

As usual, NetBSD is simple and straight-forward – 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!

polyBSD (NetBSD on USB Stick)

www.fosstools.org/README.txt

polyBSD is a "multi"-purpose (hence "poly") framework for building embedded systems that address certain aspects of information assurance. Essentially, it is a minimalistic install of NetBSD (i386) designed to run from a 256MB flash card or USB memory stick.

polyBSD: **www.tdisecurity.com/iso/polyBSD-0.1.img.gz**

pocketSAN uses polyBSD as a basis and builds on top of that to provide a functional, secure and completely free NAS/SAN solution with RAID and encrypted virtual disk support that can fit in your shirt pocket. Thus it can be used to address the data at-rest aspect of information assurance.

pocketSAN: **www.tdisecurity.com/iso/pocketSAN-0.1.img.gz**

NetBSD Live Key

NetBSD LiveKey

imil.net/nlk

The NetBSD LiveKey project 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).

You will probably need about 256MB RAM to run the USB key smoothly.

Live NetBSD CD

`www.jibbed.org`

Jibbed is a NetBSD-based Live CD, and the version number indicates that it's based on the NetBSD 5.0_BETA release. From the website:

“It features select packages from pkgsrc, as well as autoconfiguration 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”

Beastiebox (Busybox for BSD)

beastiebox.sourceforge.net

BeastieBox is an attempt to bring a Busybox-like from Linux tool to the BSD world with a BSD license

Three modes are currently available: a *semi-static mode*, where all commands will be statically linked to the main executable, still dynamically linked over libc and libm, a *full static mode*, where the produced binary is statically linked over all needed libraries, and a *dynamic mode*, where commands are available as shared objects.

As of now, the following commands are available :

ifconfig, route, sh, ls, init, ln, mount, mount_ffs, df, cat, rm, fsck, fsck_ffs, ps, kill, dmesg, hostname, cp, mv, test, [, sed, ping, less, more, sysctl, pfctl, wiconfig, traceroute, stty, date, reboot, halt, poweroff, chmod, umount, ex, vi, fdisk, disklabel, tar, getty, login, mksh

Most of these commands are ports of NetBSD 4.0 commands, but some of them, in order to minimize dependencies and size, are older NetBSD versions, older BSD versions (i.e., 4.4BSD Lite2), or **BSD-license compatible** software. The goal is to obtain a functional BSD UN*X system fitting into ***500K*** in semi-static mode, in order to be used in embedded hardware like Wireless routers, ADSL boxes, multimedia hard drives and such. As of today, BeastieBox is about ***700K***... – OK.

Current work is done under NetBSD, but should easily be ported to FreeBSD, OpenBSD and DragonFlyBSD.

The Cauldron Project

code.google.com/p/cauldron/

(Formerly known the bsd-appliance project)

Architecure Paper by Brian A. Seklecki at Collaborative Fusion, Inc.

“A Scalable Framework for Compact Flash

Booting NetBSD Network Appliances”

people.collaborativefusion.com/~seklecki/cf_nbsd_CFMDRD_odt.pdf

List of BSD-friendly hardware vendors:

code.google.com/p/cauldron/wiki/HardwareVendors

Summary on SW

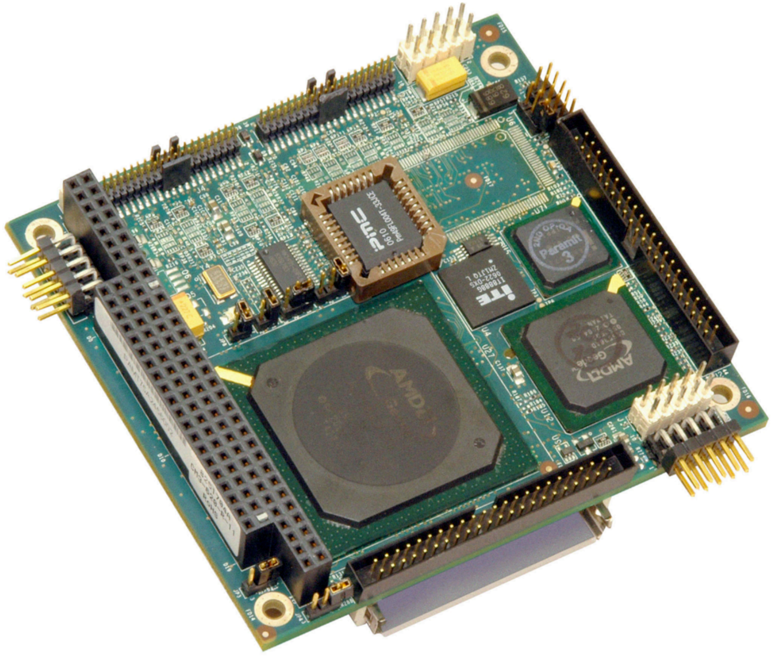
- Lots of images for FreeBSD available, some of them a bit dated – the earliest ones tried to fit on 1.4MB floppies, with today's kernel sizes that is impossible
- Less images for NetBSD, mainly USB Stick cookbooks – probably because NetBSD is already modular and "small" enough (80MB)
- Live CD distributions quite current (actually, in standard 'make release' on FreeBSD 7.x)

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

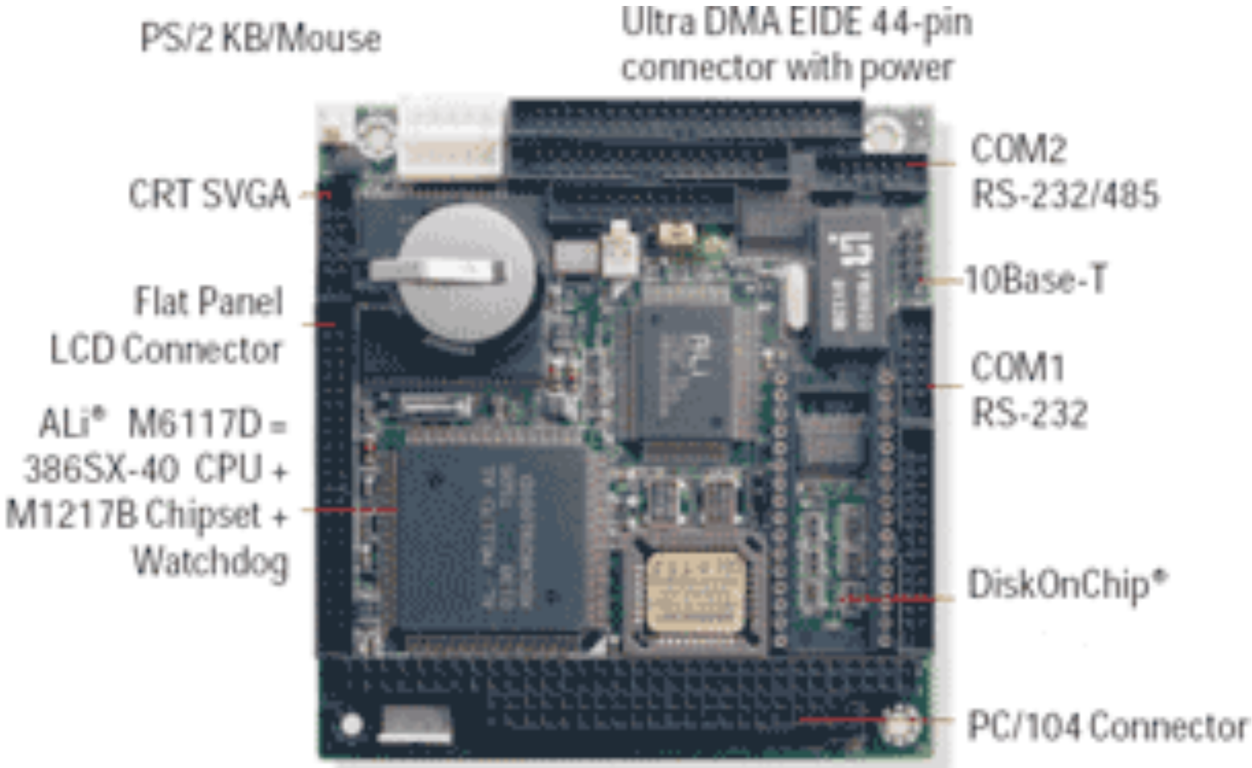
What is PC/104 ?

- 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



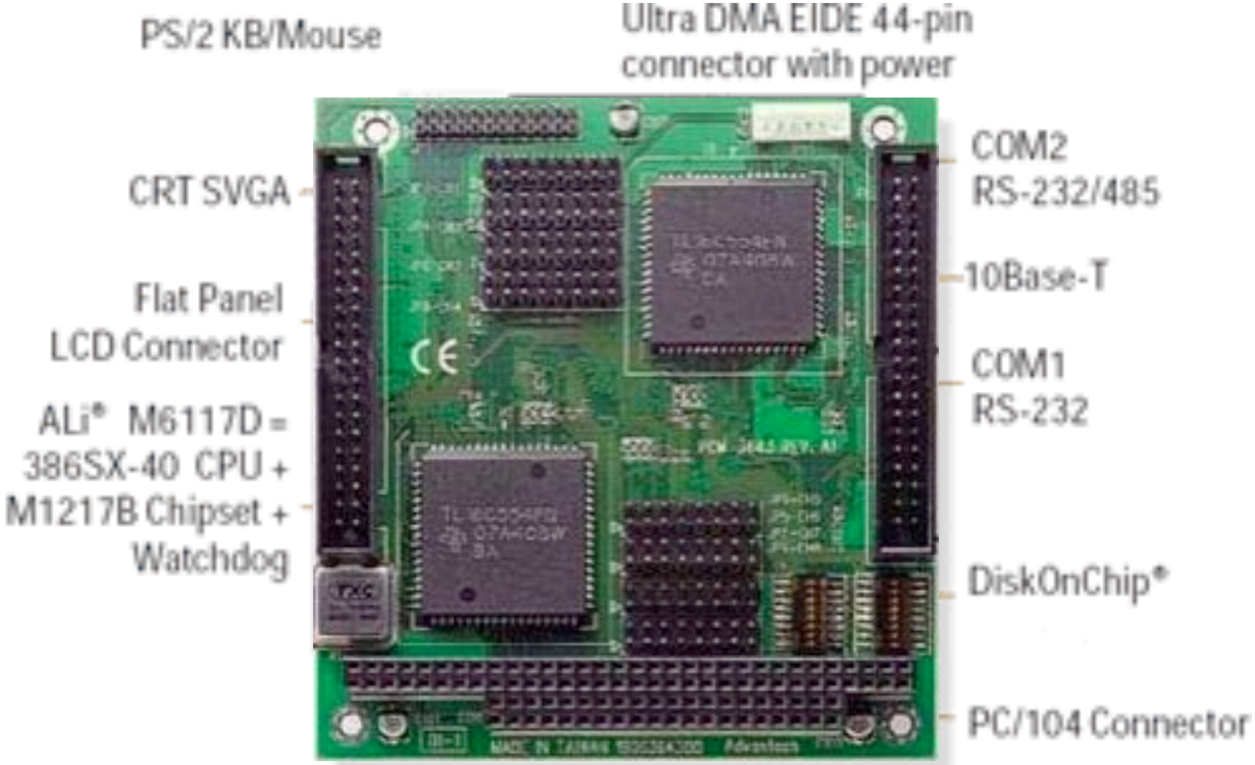
- Some good starting points:
  www.smallformfactors.com
  www.controlled.com/pc104faq/
  www.pc104.com/whatis.html

Example PC/104 CPU module



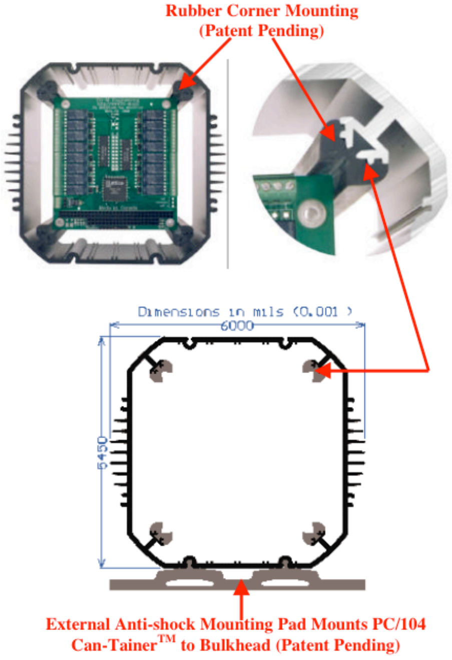
3.78" x 3.54" (96mm x 90mm) PC/104 CPU Module with
Embedded FANLESS 386 class ALI M6117D 40 MHz CPU,
ALI 5113 chipset, 4 MB EDO RAM, LCD/CRT/TFT/DSTN/VGA, ATA 33,
Realtek 8019AS 10 Mbps LAN, 16-bit GPIO and DOC interfaces

Example PC/104 CPU module and peripheral module



3.78" x 3.54" (96mm x 90mm) PC/104 CPU Module with
Embedded FANLESS 386 class ALI M6117D 40 MHz CPU,
ALI 5113 chipset, 4 MB EDO RAM, LCD/CRT/TFT/DSTN/VGA, ATA 33,
Realtek 8019AS 10 Mbps LAN, 16-bit GPIO and DOC interfaces
with "PCM-3643"PC/104 8-Port RS-232 Module

PC/104 “Stacks”


Rubber Corner Mounting
(Patent Pending)
Dimensions in mils (0.001)
6000
6450
External Anti-shock Mounting Pad Mounts PC/104
Can-Tainer™ to Bulkhead (Patent Pending)

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

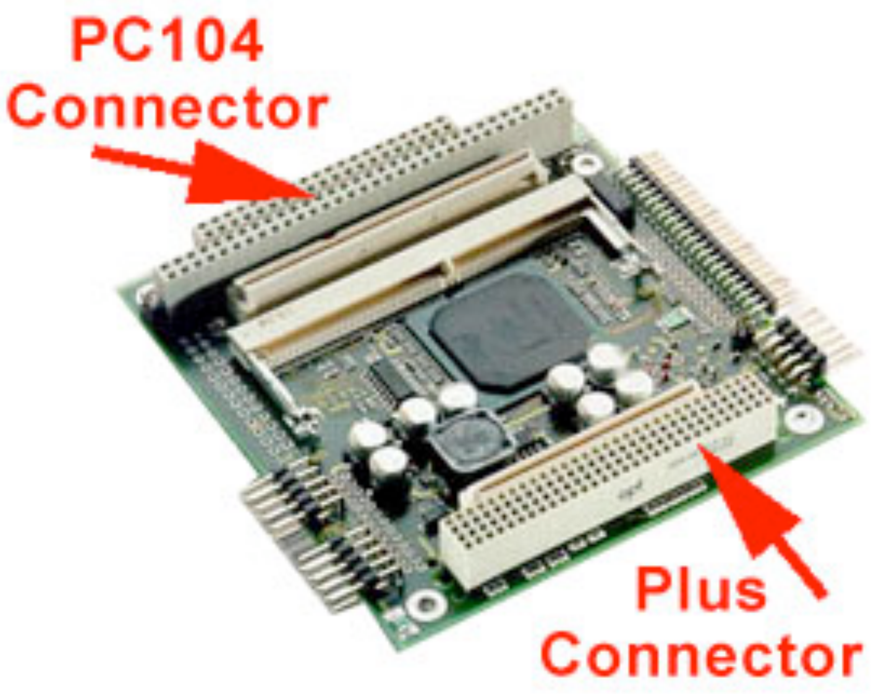
“Priced for Everyday Use”
(if you’re millionaire, that is)

www.dpie.com/pc104/cantainer.html

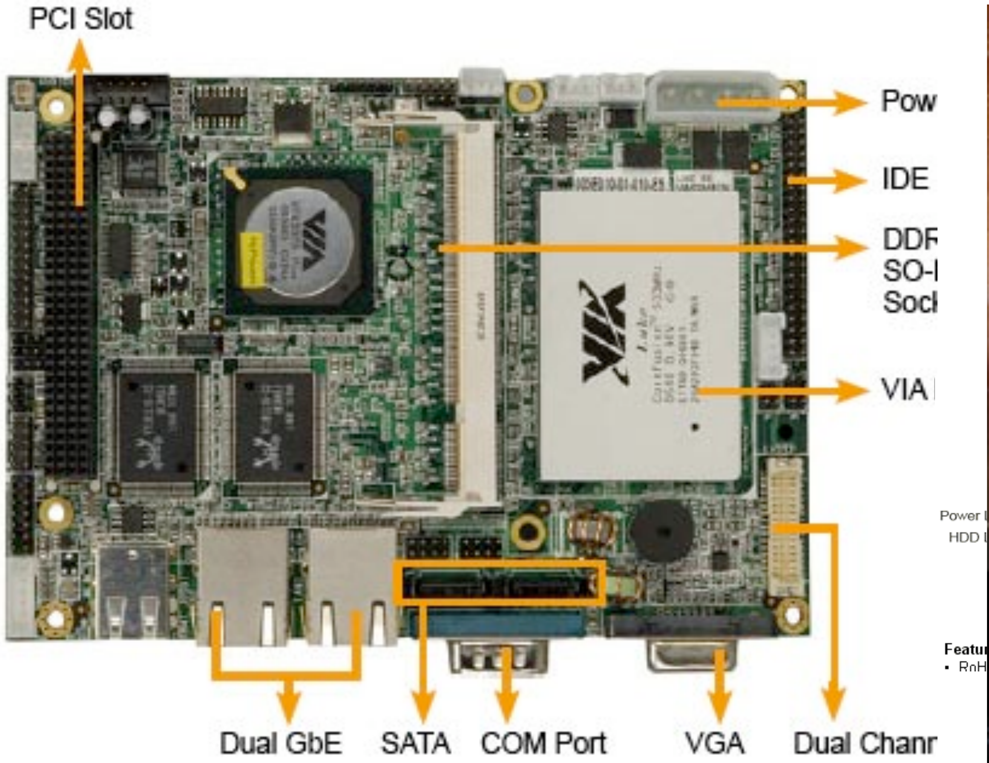
www.pc104.nl/ct104.pdf

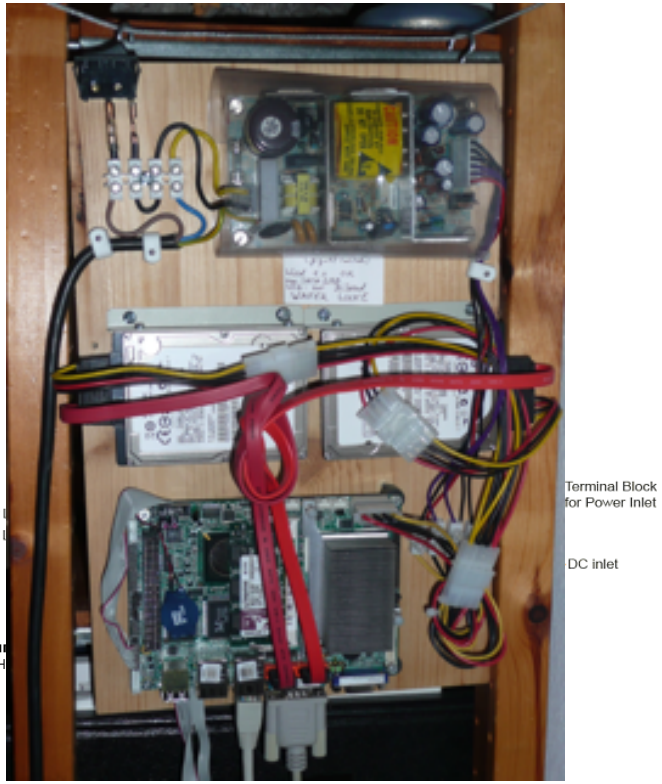
PC-104 versus PC-104+

- 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 “Bisquit PCs”

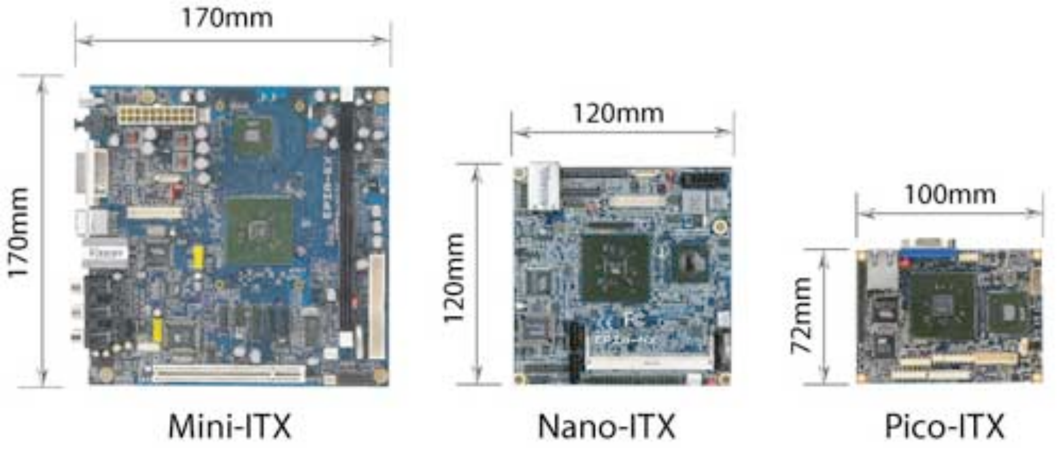




iEi WAFER-LUKE SBC with a fanless, on-board
VIA® LUKE 533MHz or 1GHz CPU, 2 x SATA with RAID 0,1, and JBOD function
support ,VGA, CF Type II socket, PC/104 socket and Dual RTL8110SC GbE chipsets

VIA EPIA Embedded Boards

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



www.via.com.tw/en/products/mainboards/

秋葉原電気街

Akihabara Denki Gai (Akihabara Electric Town)

EPIA-NL Board
with
VIA Luke CPU

Mini-ITXcases galore!

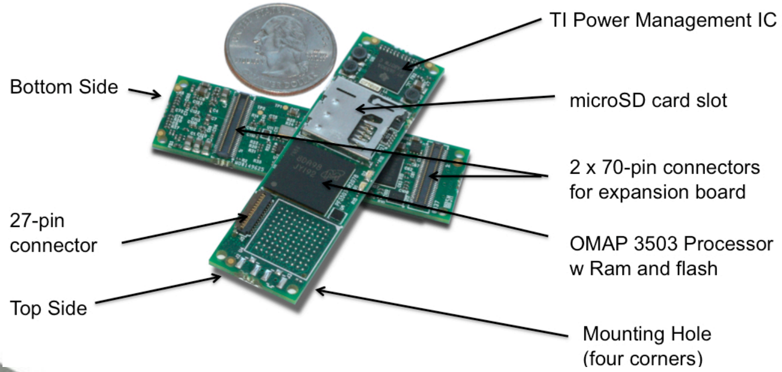
Pi-BOX Sizka Pinon Pi-BOX Sizka Pinon
No.77
Pi-BOX Sizka Basic
PiNON
AMD Geode
CPU
Geode LX800 500MHz
冷却
FAN付き ヒートシンク +FAN 標準装着
メモリー
DDR-256MB 増設不可
IDE/SATA
2. 5" IDE x1 CF x1
VGA
チップセット内蔵
出力
D-Sub
LAN 10/100Mbps
USB 4ポート
IEEE 1394 なし
スピーカー、マイク端子装備
上級者向けです！
Geode LX800@0. 9Wを使った超コンパクトマシン電源も12Wと超低消費電力です。お世辞にも速いとは言えませんがシンクライアントなどにオススメ！なお、ドライバーとBIOSの入手にユーザー登録が必要
付属品
・ACアダプター
・フロントアクリルパネル
・デコレーションキット
¥56800 税込
PI-BOX SIZKA/B2 4937699008016
EPIA-NX

An Expensive LX800 Box

The Smallest (2009, but not x86-based)

"Gumstix" form factor **www.gumstix.com**

www.feyrer.de/NetBSD/blog.html?-tags=gumstix



gumstix verdex board with micro sd wifii and audiostix2 squeezed into dlink dub-h4 usb hub
hubflamebot.com/cgi-bin/pyblosxom.cgi

Advantech, iEi, ... – SBC, PCM-58xx, WAFER, ...

- NS Geode 200MHz-300MHz (non-RoHS)
- AMD LX; VIA C3, C5, C7; Intel LV/ULV
- “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]
- Example vendors:
  www.advantech.com
  www.ieiworld.com

Advantages of PC-104 based HW

- Supports the standard PC components: i.e. Keyboard, Video, Floppies, and ATA HDs
- Usually without fans (Low Power CPUs, passive cooling)
- Lots of PC/104 expansion boards available FreeBSD ISA device drivers usually work
- Well established in the industrial environment

A small, silent PC!

Some disadvantages of most PC/104 based HW

- Has PC Keyboard and Video (cost, security)
- "Passive" cooling may really not be enough
- ISA devices are becoming legacy
- Are still expensive although only i486-like
- ... and Geode GX1 ATA 'DMA' falls back to PIO

Yesterday's PeeCee

Today's Alternatives

- Have no PC keyboard, video, floppy (legacy)
- Not "passive" cooling – NO COOLING needed!
- Systems have CF socket and USB 2.0 on-board
- Support PCI, mini-PCI, or even PCI-Express yet cost significantly less than PC/104+

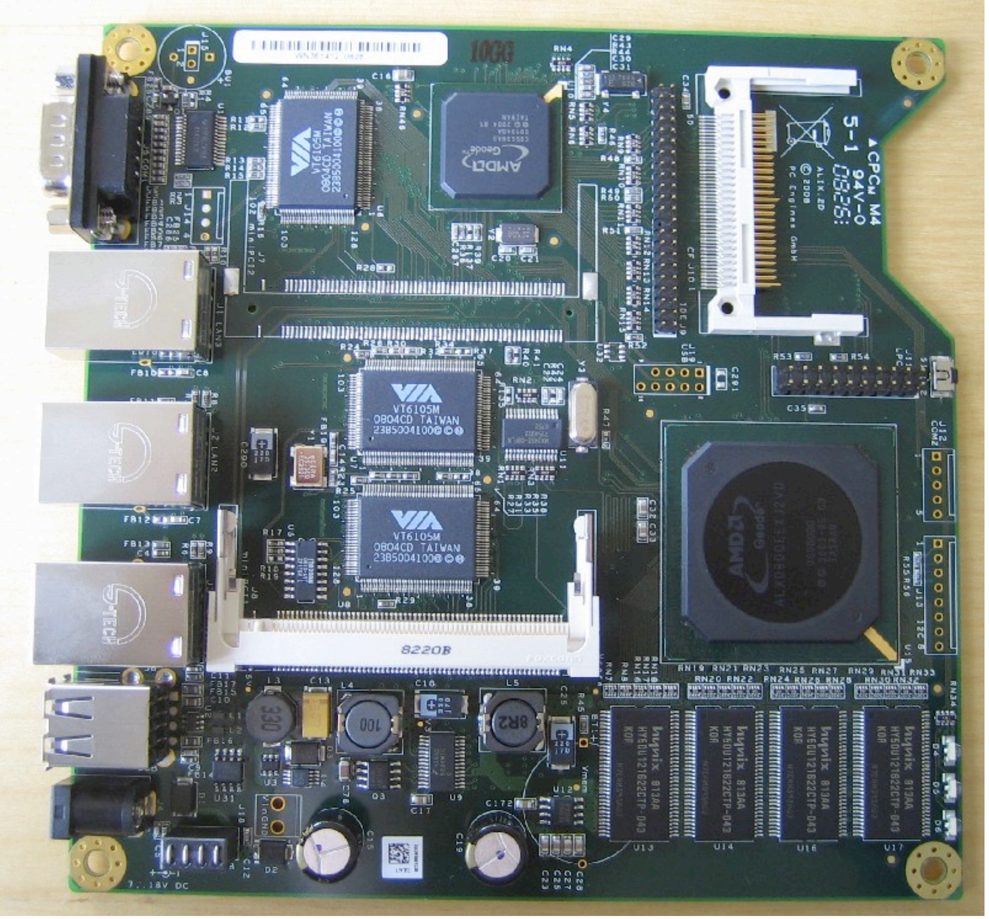
"Cool", Affordable, and Reliable HW for Open Source OS's !

www.soekris.com

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.

www.pcengines.ch

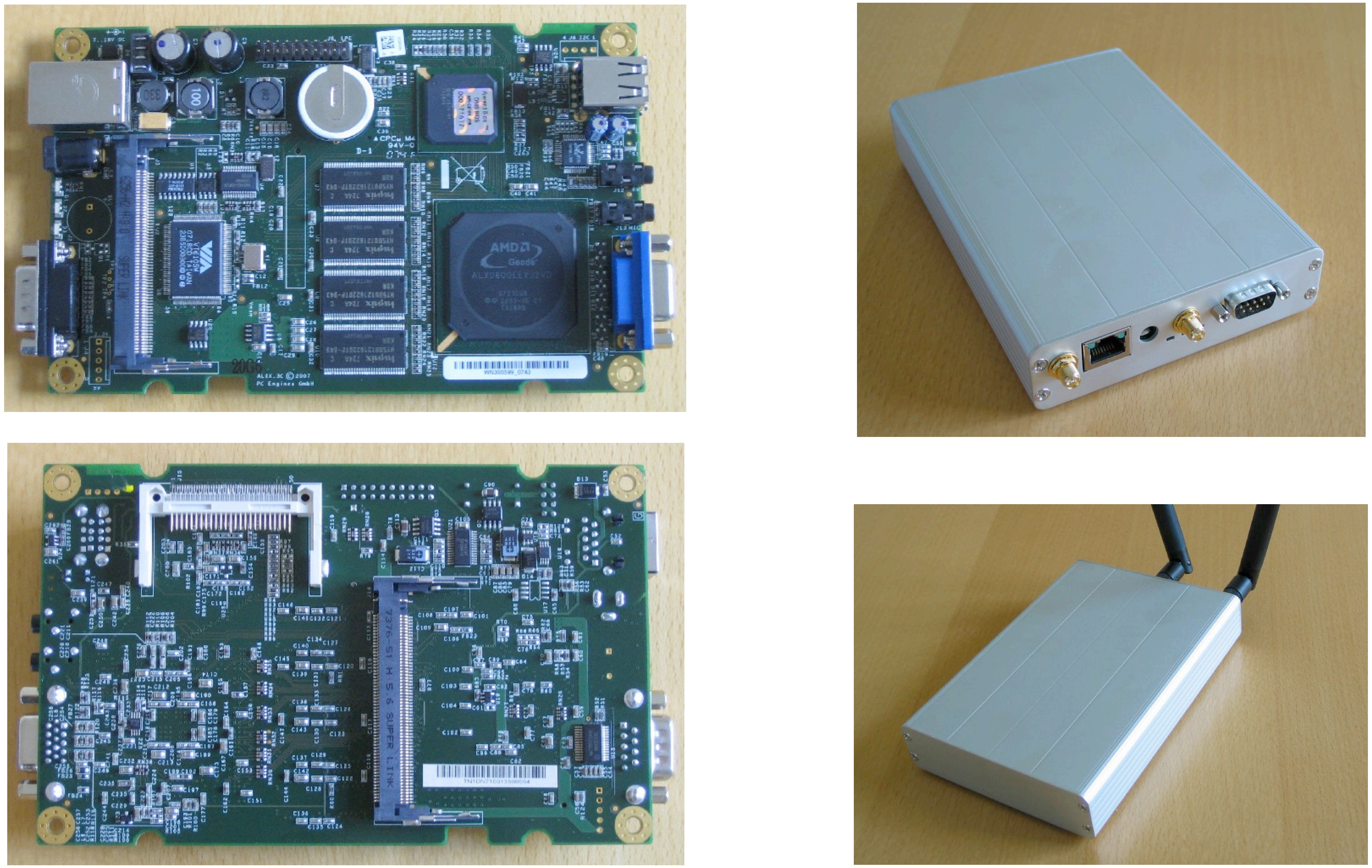


alix2d3

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

The AMD Geode LX

Not to be confused with older Geode GX1 (AMD SC1100) which had no on-chip encryption (whilst VIA C3 already had 'ACE' padlock at the time)

en.wikipedia.org/wiki/Geode_(processor)#Geode_GX1

Yesterday's Soekris net4xxx and PC Engines WRAP series

2004 2005 2006

As of July 1, 2006, the European Union introduced "The Restriction of the use of Certain Hazardous Substances" (**RoHS**) in Electrical and Electronic equipment regarding , limiting the use of 6 chemicals.

The AMD Geode LX has an integrated on-chip security block for (AES CBC/ECB) 128-Bit Advanced Encryption Standard including a true RNG 2007 no HW 2008 2009 2010?

en.wikipedia.org/wiki/Geode_(processor)#Geode_LX

Today's Soekris net55xx and PC Engines ALIX series

Compact Flash (CF)

- Most are good for a million write/erase cycles
  www.robgalbraith.com/bins/multi_page.asp?cid=6007
- Superblocks of filesystems are written (saved) often, so a million writes is not enough (hence, use **`noatime`** option when mounting 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 without CD drive, Floppy drive, video console, nor keyboard?

- First install and setup the OS on (laptop) harddisk 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 without CD drive, 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
```

- 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 (**`/dev/fido`**) on Advantech, PC Engines, and Soekris
- Creates LED devices (**`/dev/led/*`**) on PC Engines and Soekris
  - see **`/usr/src/sys/i386/i386/geode.c`**

FreeBSD Kernel tuning for AMD ELAN 520 CPU

- For ELAN CPUs
  `options CPU_ELAN`
  enables watchdog and LED (on Soekris net4501)
  – see man CPU_ELAN(4), led(4) and
  `src/sys/i386/i386/elan-mmcr.c`

- For timestamping external signals and attaching an LCD display on GPIO Soekris 4xxx, see
  `phk.freebsd.dk/soekris/`

Kernel Configuration for Crypto Accelerators

Enable in-kernel cryptography (hardware or software)

```
pseudo-device crypto
pseudo-device swcrypto
```

```
device crypto
device 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
```

Summary Part 1

- ☑ 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 Part 2

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)

www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/boot.html
www.freebsd.org/doc/en_US.ISO8859-1/books/arch-handbook/boot.html

BIOS POST "executes" first 446 bytes of sector 0 – this is a MBR, boot0, or a boot0sio program plus the disk slice table (This is also where the 'no operating system on disk' or Fn-loop can happen). The "active" PC slice is chosen and the first sector, i.e. the first 512 bytes (boot1) there are executed.

fdisk -B
|| boot0cfg

/dev/ad0

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

bsdlabel -B

(2) **boot2** understands the FreeBSD disklabel 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:
```

/dev/ad0s1

(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
```

/boot/loader.rc
/boot/loader.conf

/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 (besides booting) is to set all the kernel environment variables and display a splash image

Some example **`loader`** commands

- **`help`**
- **`show`**
- **`set`**
- **`ls`**
- **`more`**
- **`1000 ms`**
- **`words`**
- **`include`**

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
```

Booting NetBSD (2 stage boot)

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

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.

```
fdisk -B || mbrlabel
     /dev/wd0
```

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
```

NetBSD Install CD

```
$ 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`**

```
srcdirs /usr/src/bin
progs ls
libs -lncurses -lutil
progs ps
libs -lm -lkvm
```

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 does

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:

```
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

#undef LIBWRAP
#undef USE_PAM
#undef HAVE_LIBPAM
#undef HAVE_PAM_GETENVLIST
#undef HAVE_SECURITY_PAM_APPL_H
#undef XAUTH_PATH
```

NetBSD crunches using Makefile technology – what else?

`Makefile` essentials

```
...

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 propogates 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 easily

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

```
geom(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/               libgeom.so.4     libncurses.so.7    libutil.so.7
libbsdxml.so.3      libkvm.so.4      libnvpair.so.1     libuutil.so.1
libc.so.7           libm.so.5        libsbuf.so.4       libz.so.4
libcrypto.so.5      libmd.so.4       libufs.so.4        libzfs.so.1

lib/geom:
geom_cache.so       geom_mirror.so         geom_shsec.so
geom_concat.so      geom_multipath.so      geom_stripe.so
geom_eli.so         geom_nop.so            geom_virstor.so
geom_journal.so     geom_part.so
geom_label.so       geom_raid3.so

libexec:
ld-elf.so.1*
```

What’s on the RAMdisk ?

```
-sh
[
cat
chflags
chgrp
chmod
chown
chroot
cp
date
df
du
expr
hostname
kill
link
ln
ls
mkdir
mv
ps
pwd
realpath
rm
rmdir
sh
sleep
stty
test
touch
tset
unlink
```

Basics on RAMdisk

-sh
[
cat
chflags
chgrp
chmod
chown
chroot
cp
date
df

du
expr

hostname
init
kenv
kill
ldconfig
link
ln
ls

mkdir
mv
ps
pwd
realpath
rm
rmdir

sh
sleep
stty
test
touch
tset
unlink

SysAdmin on RAMdisk

```
atacontrol
badsect
boot0cfg
bsdlabel
camcontrol
clri
dd
diskinfo
disklabel

dumpfs
fastboot
fasthalt
fdisk
ffsinfo
fsck
fsck_4.2bsd
fsck_ffs
fsck_ufs
gbde
geli

halt
kldconfig
kldload
kldstat
kldunload
mdconfig
mdmfs

mknod
mount
mount_cd9660
mount_devfs
mount_fdescfs
mount_linprocfs
mount_procfs
mount_std
newfs
reboot

swapctl
swapoff
swapon
sync
sysctl
tunefs
umount
zfs
zpool
```

Networking on RAMdisk

`route`

`ifconfig`

`ping`

```
dhclient
dhclient-script
```

More networking RAMdisk

dhclient
dhclient-script

ggatec
ggated
ggatel

ifconfig

ipf
ipfw

mount_nfs

pfctl
ping

route

scp

slogin
ssh
sshd

Archiving tools on RAMdisk

```
bunzip2
bzcat
bzip2
```

```
dump
```

```
gunzip
gzcat
gzip
```

```
pax
rdump
restore
```

```
rrestore
tar
zcat
```

Editors on the RAMdisk

`sed`

`ed`
`ex`

`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
bzcat
bzip2
camcontrol
cat
chflags
chgrp
chmod
chown
chroot
clri
cp
date
dd
df
dhclient
dhclient-script
diskinfo
disklabel
dmesg
du
dump
dumpfs
ed
env
ex
expr
fastboot
fasthalt
fdisk
ffsinfo
fsck
fsck_4.2bsd
fsck_ffs
fsck_ufs
gbde
gcache
gconcat
geli
geom
ggatec
ggated
ggatel
gjournal
glabel
gmirror
gmultipath
gnop
gpart
graid3
growfs
gshsec
gstripe
gunzip
gvirstor
gzcat
gzip
halt
hostname
ifconfig
init
ipf
ipfw
kenv
kill
kldconfig
kldload
kldstat
kldunload
ldconfig
link
ln
ls
mdconfig
mdmfs
mini_crunch
mkdir
mknod
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
umount
unlink
vi
view
zcat
zfs
zpool
```

NetBSD 5 custom RAMdisk

```
$ du -sh .
4.5M    .

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

sbin:
atactl          dump            mbrlabel        mount_ufs       rrestore
badsect         dump_lfs        mknod           newfs           scsictl
ccdconfig       fdisk           modload         newfs_lfs       swapctl
cgdconfig       fsck            modunload       ping            swapon
clri            fsck_ffs        mount           raidctl         sysctl
dhclient        fsck_lfs        mount_cd9660    rdump           tunefs
dhclient-script halt            mount_ffs       rdump_lfs       umount
disklabel       ifconfig        mount_lfs       reboot
dkctl           init            mount_mfs       restore
dmesg           ldconfig        mount_nfs       route

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

usr/sbin:
chgrp        chroot       installboot  pwd_mkdb     vnconfig
chown        dumpfs       mdconfig     sshd         wiconfig
```

On-disk: 3.0 MB / In RAM: 9.0 MB

- The boot loader is able to preload *gzip-compressed* RAMdisk images
- Additional on-disk (CF) usage is minimal

```
$ du -h fs.7.x-RAMdisk.gz
3.0M    fs.7.x-RAMdisk.gz
```

- In RAM currently defined as 9.0 MB md0

```
# mdconfig -l -u 0
md0     preload   9.0M
```

```
$ ls -sh netbsd-RAMDISK.gz
3.7M 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
- In RAMdisk is currently defined as 5.0 MB filesystem of which 4.5 MB is used
- Compares favorably to FreeBSD:

On-disk: 3.0 MB / In RAM: 9.0 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:

```
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.

Thank you very much for attending this tutorial!