Single User
Secure Shell
Tutorial

EuroBSD Conference
Basel, Switzerland
Friday, November 25th, 2005
9:00-13:00, 14:00-17:00
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
• Traveling, Sculpting, Go
Schedule for the day

• Part 1
  Overview SW and HW for small systems

• Part 2
  The single user secure shell explained

• Part 3
  The maintenance RAMdisk in action

• Part 4 (afternoon)
  You install and use the maintenance RAMdisk on your own systems
FreeBSD for Small HW

Many choices!

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

— Too many?
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”

• “Small FreeBSD Home Page”
  http://people.freebsd.org/~picobsd/
  – (still) a good starting point for small systems!
miniBSD

• Manuel Kasper’s <mk@neon1.net> precursor to m0n0wall in 2002 for FreeBSD 4.x: https://neon1.net/misc/minibsd.html

• David Courtney <minibsd@ultradesic.com> in 2005 for 5.x: www.ultradesic.com/index.php?section=86

• Cookbooks on how to whittle down the FreeBSD base system using a chroot environment

• A few utility scripts (for example, to find shared object dependencies)
• Full-fledged firewall based on m0n0BSD, a stripped down 4.x-based FreeBSD for CF

• Configuration via a PHP web GUI, and stored as one big XML file (!)

• Very much “end-user” oriented

• Distributed mainly as CF images for PC-Engines and Soekris platforms

• http://m0n0.ch/wall/
  2003-2005 Manuel Kasper <mk@neon1.net>
• By Italian FreeBSD User Group in 2004-2005 (Gruppo Utenti FreeBSD Italia, www.gufi.org):
  Davide D'Amico <dave@FreeSBIE.org>
  Dario Freni <saturnero@FreeSBIE.org>
  Massimiliano Stucchi <stucchi@FreeSBIE.org>

• Currently 5.3-based

• Appears to be “Knoppix” inspired

• Not really small, but a good RAMdisk model to study

• www.freesbrie.org
NanoBSD

• In tree 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 (see talk by PHK at this conference)

• 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
• 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 6.x
Small SW calls for small HW

- Look for “Embedded Systems” — albeit a misnomer
- What is 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.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”

Tri-M Systems PC/104 CAN-TAINER™
PC/104 Container Designed For Hostile Environments
“Priced for Everyday Use”
(if you’re millionaire, that is)
http://www.dpie.com/pc104/cantainer.html
PC-104 vs 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”

Advantech’s PCM-5825: 3.5” SBC with an on-board 586-class NS GX-300 processor VGA/LCD, Audio, CF socket, PC/104 sockets and Intel 10/100 Mbps Ethernet (fxp0)
Advantech, iEi, ... PCM-58xx

- NS Geode 200MHz-300MHz
- “Passive” cooling
- AT kbd, VGA/LCD, 2-4 COMs, [Audio]
- ATA HD support
- 1-2 Ethernet [Realtek or Intel]
- PC/104 socket, [USB]

- Example sources:
  - 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 boards available
  FreeBSD ISA device drivers usually work

• Well established in 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 ATA DMA falls back to PIO

Yesterday’s PeeCee
Today’s Alternatives

- No PC keyboard, video, floppy
- Not “passive” cooling – NO cooling needed!
- Systems have CF socket
- Support PCI or mini-PCI yet cost significantly less than PC/104+
- “Cool”

Affordable and reliable HW for Open Source OSs
NET4801
NSC SC1100 266 Mhz CPU, 128 Mbyte SDRAM, 3 Ethernet, 2 serial USB connector, CF socket, 44 pins IDE connector, Mini-PCI socket, 3.3V PCI connector
here with Sangoma A101u E1/T1 PCI interface board
PC Engines: www.pcengines.ch

WRAP:2C
AMD Geode
SC1100 266 MHz
128MB SDRAM
1 serial, 1 Ethernet,
CF socket
2 Mini-PCI sockets
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
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
What’s different in userland

- Serial console:
  
  ```
  $ cat /boot.config
  -h
  ```

- No AT Keyboard, no video
  
  ```
  $ cat /boot/loader.conf
  hint.atkbdc.0.disabled="1"
  hint.sc.0.disabled="1"
  hint.vga.0.disabled="1"
  ```
What’s different in userland

- Serial console:
  ```
  $ cat /boot.config
  -h
  ```

- No AT Keyboard, no video, ...
  ```
  $ cat /boot/loader.conf
  hint.atkbdc.0.disabled="1"
  hint.sc.0.disabled="1"
  hint.vga.0.disabled="1"
  hint.sio.1.disabled="1"
  hint.fdc.0.disabled="1"
  hint.pcic.0.disabled="1"
  hint.ppc.0.disabled="1"
  hint.ata.1.disabled="1"
  hint.ata.ata_dma="0"
  ```
Installing without CD drive, video, keyboard

• Installation via PXE netboot?
  BIOS and NIC needs to support Intel® PXE support
  “FreeBSD Jumpstart Guide”:

• Install and setup FreeBSD on (laptop) harddisk on another system, then install on target system

• Essential for systems which only have CF:
  PCCard or USB adapter to initialise CF via laptop
Compact Flash (CF)

• Most are good for a million write/erase cycles
  
  www.robgalbraith.com/bins/multi_page.asp?cid=6007

• Superblocks of filesystems get written (saved) often, so a million writes is still not enough!

• Mount CF read-only, easy:
  
  • touch /etc/diskless
  
  • /conf/base/... for /etc/rc.initdiskless
RAMdisks

mdmfs(8)

    mkdir /foo
    mdmfs -X -s 32m md /foo

    umount /foo
    mdconfig -d -u #
    rm -rf /foo

mdconfig(8)

    dd if=/dev/zero of=bar bs=1k count=5k
    mdconfig -a -t vnode -f bar
    bsdlabel -w md# auto
    newfs md#a
    mkdir /bar
    mount /dev/md#a /bar

    umount /bar
    mdconfig -d -u #
    rm -rf /bar bar
Kernel tuning
GEODE and “SOEKRIS”

For Geode 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
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
  http://phk.freebsd.dk/soekris/
Summary Part 1

- FreeBSD for small platforms:
  Decide between tools or an all-in-one distribution

- 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 are your friends

- Build custom kernels on a fast “build” system to take full advantage of HW features
Outlook Part II

i. A closer look at how FreeBSD boots/installs the install CD
   the boot sequence
   building crunched binaries

ii. The missing bits needed for building a networked maintenance RAMdisk (Single User Secure Shell)

iii. Some details of building and installing the maintenance RAMdisk (time permitting)

iv. Using the “Single User Secure Shell” to install/upgrade OS (demonstration)
Tutorial sets

- Set includes: PC Engine WRAP.1E-2 (3 LAN / 1 miniPCI / 128 MB DRAM), 128MB CF, Nullmodem serial cable, RJ45 crossover patch cable, 14VA power brick, and enclosure (red, alu, or black)

- Preorders can all be fulfilled (there are a few left)

- Most sets are preassembled, people who pre-ordered 2 sets get one unassembled

- People who really want to assemble their own set, must populate the CF first (and will need tools)
  `/cdrom/6.0-STYX/EuroBSD05Tutorial/stagecf.sh`

- Tutorial price is 140 Euro or 230 CHF in cash

- ITGarage is selling WiFi Upgrade options at conference
Booting FreeBSD


i. BIOS POST “executes” mbr, boot0, or boot0sio, or ...
   F1, ‘no operating system on disk’, F1-loop, ...
   F1, 'no operating system on disk', F1-loop, ...
   fdisk -B|-b
   boot0cfg

ii. boot2 loads /boot/loader from active BIOS partition
   >>FreeBSD/i386 BOOT
   Default: 1:ad(1,a)/boot/loader
   boot:
   bsdlabel -B

iii. loader sets kernel environment, loads kernel and modules
     and boots FreeBSD
     BTX loader 1.0 BTX version is 1.01
     ...
     Hit [Enter] to boot immediately, or any other key ...
     OK
/boot/loader

- help
- show
- set
- ls
- more
- 1000 ms
- words
FreeBSD Install CD

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

$ zcat /cdrom/boot/mfsroot > /tmp/m
# mdconfig -a -t vnode -f /tmp/m
md0
# mount /dev/md0 /mnt
$ ls -isl /mnt/stand
$ ldd /mnt/stand/*
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
A straightforward plan

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

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

iii. Use the boot loader to preload the RAMdisk

iv. Either mount it as the root filesystem for maintenance ...

v. ... or mount it very early form a startup script to check filesystem integrity
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:

```
buildopts -DNO_KERBEROS
buildopts -DNO_PAM
srcdirs /usr/src/secure/usr.bin
srcdirs /usr/src/secure/usr.sbin
progs scp ssh 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
  ```
GEOM uses dlopen()

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

geom(8), gconcat(8), glabel(8), gmirror(8), gnop(8), graid3(8), gshsec(8), gstripe(8)

...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 with a twist

• Linking “mostly static” is for now mentioned in crunchgen.conf as a comment:

    # LIBS_SO
    -lmd -lcrypto -lgeom -lsbuf -lbsdxml

• Before running make, the crunchgen.mk is fixed by replacing all
  $(CC) -static ...

  with
  $(CC) -Xlinker -Bstatic ...
  -Xlinker -Bdynamic $LIBS_SO
What’s on the RAMdisk?

- sh
[ du
  expr
  hostname
  chflags
  chgrp
  chmod
  chown
  chroot
  cp
  date
  df
  link
  ln
  ls
  chflags
  fsck
  fsck_4.2bsd
  fsck_ffs
  fsck_ufs
  chmod
  chown
  chroot
  clri
  dhclient
  dhclient-script
  disklabel
diskinfo
dd
dcpc
dgaggregate
dgatel
dlabel
dkldconfig
dkldload
dkldunload
dkuser

mkdir
move
ps
pwd
realpath
rm
rmdir
run
rm
rmdir
sh
sleep
hostname
mv
kill
hostname
mv
kill
hostname
mv
kill
hostname
mv
kill
hostname
mv
kill
host

Basics on RAMdisk

- sh
  sh
  [sh]
  [du]
  [mkdir]
  [expr]
  [hostname]
  [init]
  [mv]
  [kenv]
  [kill]
  [ps]
  [ps]
  [pwd]
  [pwd]
  [realpath]
  [realpath]
  [touch]
  [touch]
  [tset]
  [tset]
  [unlink]
  [unlink]
  cp
  chflags
  chgrp
  chmod
  chown
  chroot
  cat
  date
  df
  ldconfig
  link
  ln
  ls
  rm
  rmdir
### SysAdmin on RAMdisk

<table>
<thead>
<tr>
<th>Command</th>
<th>Command</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>atacontrol</td>
<td>badsect</td>
<td>dumps</td>
</tr>
<tr>
<td>boot0cfg</td>
<td>bsdlabel</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fastboot</td>
<td>fasthalt</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>camcontrol</td>
<td>fdisk</td>
<td>mount_procfs</td>
</tr>
<tr>
<td></td>
<td>ffsinfo</td>
<td>mount_std</td>
</tr>
<tr>
<td></td>
<td>fsck</td>
<td>newfs</td>
</tr>
<tr>
<td></td>
<td>fsck_4.2bsd</td>
<td>swapcmd</td>
</tr>
<tr>
<td></td>
<td>fsck_ffs</td>
<td>swapoff</td>
</tr>
<tr>
<td></td>
<td>fsck_ufs</td>
<td>swapon</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sync</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sysctl</td>
</tr>
<tr>
<td>clri</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>kldconfig</td>
</tr>
<tr>
<td></td>
<td></td>
<td>kldload</td>
</tr>
<tr>
<td></td>
<td></td>
<td>kldstat</td>
</tr>
<tr>
<td></td>
<td></td>
<td>kldunload</td>
</tr>
<tr>
<td>dd</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>reboot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>tunefs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>umount</td>
</tr>
<tr>
<td>diskinfo</td>
<td></td>
<td>mdconfig</td>
</tr>
<tr>
<td>disklabel</td>
<td></td>
<td>mdmfs</td>
</tr>
</tbody>
</table>
Networking on RAMdisk

route

ifconfig

ping

dhclient
dhclient-script
More networking RAMdisk

- route
- scp
- slogin
- ssh
- sshd
- ifconfig
- ipf
- ipfw
- pfctl
- ping
- ggatec
- ggated
- ggateel
- dhclient
- dhclient-script
- mount_nfs
- mount_nfs
- mount_nfs
- mount_nfs
- mount_nfs
- mount_nfs
Archiving tools on RAMdisk

- dump
- bunzip2
- bzip2
- gunzip
- gzcat
- gzip
- pax
- tar
- rdump
- restore
- zcat
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
- bzcat
- bzip2
- camcontrol
- cat
- chflags
- chgrp
- chmod
- chown
- chroot
- clri
- chmod
- cp
- date
- dd
- df
- dhclient
- dhclient-script
- diskinfo
- disklabel
- disklabel
- diskinfo
- diskinfo
- dmsetup
- dmesg
- du
- dump
- dumpfs
- ed
- ex
- expr
- fastboot
- fasthalt
- fdisk
- ffsinfo
- fsck
- fsck_4.2bsd
- fsck_ffs
- fsck_ufs
- gbde
- gconcat
- geli
- geom
- ggate
- ggatec
- ggated
- ggatel
- glabel
- gmirror
- gnop
-radient
- gd不愿意
- growfs
- gshsec
- gstripe
- gunzip
- gzip
- gzip
- halt
- hostname
- ifconfig
- init
- ipf
- ipfw
- kenv
- kill
- kldconfig
- kldload
- kldstat
- kldunload
- ldconfig
- link
- ln
- ls
- mdconfig
- mdmfs
- mini_crunchn
- route
- mkdir
- mknod
- mount
- sed
- mount_cd9660
- mount_devfs
- mount_fdescfs
- mount_linprocfs
- mount_nfs
- mount_procfs
- mount_std
- mv
- newfs
- swapoff
- pax
- swapon
- pfctl
- sync
- ping
- sysctl
- ps
- test
- pwd
- touch
- redbank
- rpm
- rm
- rmdir
- restore
- umount
- unlink
- vi
- ssh
- slogin
- sv
- ssfd
- mount
- sftp
RAMdisk disk usage 5MB

$ du -sk .
5186 .

$ du -sk * | sort -rn
2682 bin
2218 lib
136 libexec
78 etc
26 boot
22 usr
12 var
6 root
2 mnt
2 dev
0 tmp
0 sbin
On-disk 2.5 MB

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

• Additional on-disk (CF) usage is minimal

$ du -ks fs.6.0-RAMdisk.gz
2352    fs.6.0-RAMdisk.gz
On-disk 2.5 MB / RAM 7MB

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

- Additional on-disk (CF) usage is minimal

  $ du -ks fs.6.0-RAMdisk.gz
  2352    fs.6.0-RAMdisk.gz

- In RAM currently defined as 7MB md0

  # mdconfig -l -u 0
  md0     preload   7.0M
## RAMdisk versus /rescue

### Additional on RAMdisk (today)

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>boot0cfg</td>
</tr>
<tr>
<td>chgrp</td>
</tr>
<tr>
<td>chown</td>
</tr>
<tr>
<td>diskinfo</td>
</tr>
<tr>
<td>du</td>
</tr>
<tr>
<td>ffsinfo</td>
</tr>
<tr>
<td>gconcat</td>
</tr>
<tr>
<td>geli</td>
</tr>
<tr>
<td>geom</td>
</tr>
<tr>
<td>ggatec</td>
</tr>
<tr>
<td>ggated</td>
</tr>
<tr>
<td>ggate1</td>
</tr>
<tr>
<td>glabel</td>
</tr>
<tr>
<td>gmirror</td>
</tr>
<tr>
<td>gnop</td>
</tr>
<tr>
<td>graid3</td>
</tr>
<tr>
<td>growfs</td>
</tr>
<tr>
<td>gshsec</td>
</tr>
<tr>
<td>gstripe</td>
</tr>
<tr>
<td>ipfw</td>
</tr>
<tr>
<td>pfctl</td>
</tr>
<tr>
<td>scp</td>
</tr>
<tr>
<td>sed</td>
</tr>
<tr>
<td>sleep</td>
</tr>
<tr>
<td>slogin</td>
</tr>
<tr>
<td>ssh</td>
</tr>
<tr>
<td>sshd</td>
</tr>
<tr>
<td>styxinstall</td>
</tr>
<tr>
<td>swapctl</td>
</tr>
<tr>
<td>swapoff</td>
</tr>
<tr>
<td>touch</td>
</tr>
<tr>
<td>tset</td>
</tr>
</tbody>
</table>

### Additional in /rescue (6.x)

<table>
<thead>
<tr>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>atm</td>
</tr>
<tr>
<td>atmconfig</td>
</tr>
<tr>
<td>ccdconfig</td>
</tr>
<tr>
<td>chio</td>
</tr>
<tr>
<td>csh</td>
</tr>
<tr>
<td>devfs</td>
</tr>
<tr>
<td>dumpon</td>
</tr>
<tr>
<td>echo</td>
</tr>
<tr>
<td>fore_dnld</td>
</tr>
<tr>
<td>fsck_msdosfs</td>
</tr>
<tr>
<td>fsdb</td>
</tr>
<tr>
<td>fsirand</td>
</tr>
<tr>
<td>getfacl</td>
</tr>
<tr>
<td>groups</td>
</tr>
<tr>
<td>id</td>
</tr>
<tr>
<td>ilmid</td>
</tr>
<tr>
<td>ipfs</td>
</tr>
<tr>
<td>ipfstat</td>
</tr>
<tr>
<td>ipmon</td>
</tr>
<tr>
<td>ipnat</td>
</tr>
<tr>
<td>md5</td>
</tr>
<tr>
<td>mount_ext2fs</td>
</tr>
<tr>
<td>mount_msdosfs</td>
</tr>
<tr>
<td>mount_ntfs</td>
</tr>
<tr>
<td>mount_nullfs</td>
</tr>
<tr>
<td>mount_udf</td>
</tr>
<tr>
<td>mount_omapfs</td>
</tr>
<tr>
<td>mount_unionfs</td>
</tr>
<tr>
<td>newfs_msdos</td>
</tr>
<tr>
<td>nextboot.sh</td>
</tr>
<tr>
<td>nos-tun</td>
</tr>
<tr>
<td>ping6</td>
</tr>
<tr>
<td>raidctl</td>
</tr>
<tr>
<td>rcrdr</td>
</tr>
<tr>
<td>rcp</td>
</tr>
<tr>
<td>routed</td>
</tr>
<tr>
<td>rtquery</td>
</tr>
<tr>
<td>rtsol</td>
</tr>
<tr>
<td>savecore</td>
</tr>
<tr>
<td>sconfig</td>
</tr>
<tr>
<td>setfacl</td>
</tr>
<tr>
<td>slattach</td>
</tr>
<tr>
<td>spppcontrol</td>
</tr>
<tr>
<td>startslip</td>
</tr>
<tr>
<td>tcsh</td>
</tr>
<tr>
<td>vinum</td>
</tr>
<tr>
<td>whoami</td>
</tr>
</tbody>
</table>
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
dev | sed -ne 's/^maint\.//p' >> /etc/params
One way into RAMdisk

By replacing `/boot/loader.rc` (remotely) 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
```
Another way into RAMdisk

By starting with a very early script:

```bash
$ cd /etc/rc.d; rcorder * | head -4
rcconf.sh
dumpon
initrandom
maint_sshd

$ head maint_sshd
#!/bin/sh
PATH=/rescue:usr/bin:/bin:/usr/sbin:/sbin
# REQUIRE: initrandom
# PROVIDE: maint_sshd
# KEYWORD: nojail
# BEFORE: disks
```
/etc/rc.d/maint_sshd steps

i. Check for preloaded RAMdisk
   If it hasn’t been preloaded, look for it at
   $maint_sshd_fs_img and mdconfig it

ii. Mount it on /boot/maint and mount devfs
   on /boot/maint/dev

iii. Execute chroot /boot/maint /etc/rc
RAMdisk /etc/rc

i. Configure network and start a
   /usr/sbin/sshd (in RAMdisk)

ii. Check the “real” root filesystem

iii. Check the /usr filesystem as specified
     by /etc/fstab on the “real” root

iv. If called from /etc/rc.d/maint_sshd
    and the filesystems checked well, exit

v. Otherwise, wait for administrator login
Cleaning up after RAMdisk

A. Returning from RAMdisk /etc/rc, we know that the real root filesystem (and /usr) are clean

B. Continue with the startup scripts ...

C. Right before launching the real SSHD:
   i. Kill the SSHD running in RAMdisk
   ii. Unmount /boot/maint/dev and /boot/maint
   iii. Relinquish RAM used by RAMdisk (if possible)
Single User Secure Shell

- A more sophisticated “rescue” environment in a RAMdisk which configures the network and also supports SSH, SSHD, and GEOM commands

- Is launched either stand-alone from boot loader or from /etc/rc.d before filesystems are checked

- Secure Shell remote login for root is possible – even when system is stuck in “Single User”
Overview of “full build”

• Build world
• Build release
• Build sandbox(es) and customize
  ➔ _DEV, _128MB, _64MB tarballs
• Build custom kernel per architecture
• Build mfs image
  customize its /etc directory
  copy appropriate kernel into /boot/maint
  ➔ boot+maint.tbz tarballs
Details building a release

Get FreeBSD CVS repository with CVSUP (/usr/cvs)

cd /usr && cvs -d /usr/cvs co src -r RELENG_6
  && cd src && make buildworld

cd release && make release \
  BUILDNAME=NAME CVSROOT=/usr/cvs \
  CHROOTDIR=/r/BUILD RELEASETAG=RELENG_6

After a while, results in /r/BUILD/R

Common gotchas
  Not enough disk space in /usr/obj or /r/BUILD
  Environment or /etc/make.conf incompatibilities
Details _DEV dist tarball

```
SB=/r/BUILD
STYXTREE=${SB}/STYX/stage/trees
STYXTREEDEV=${STYXTREE}/DEV
mkdir -m 0755 -p ${STYXTREEDEV}

for d in `cd ${SB}/R/stage/trees; echo *` do
do
    cd ${SB}/R/stage/trees/$d && 
    find . -depth | cpio -dump ${STYXTREEDEV}
done

for d in `cd ${SB}/R/stage/mfsfd; echo *`; cd ${SB}/R/stage/mfsfd && 
    find stand -depth | cpio -dump ${STYXTREEDEV}

mount_devfs ${STYXTREEDEV}/dev

for d in `cd /etc/resolv.conf; echo *`; cd /etc/resolv.conf && 
    cp -r $d ${STYXTREEDEV}/etc

mount_devfs ${STYXTREEDEV}/dev

for d in `cd /etc/resolv.conf; echo *`; cd /etc/resolv.conf && 
    cp -r $d ${STYXTREEDEV}/etc

# copy customization scripts into ${STYXTREEDEV}
# run customization scripts in chroot ${STYXTREEDEV}

cd `dirname ${STYXTREEDEV}` && tar cvBjf DEV.tbz DEV
```
Details _generic tarball

SB=/r/BUILD
STYXTREE=${SB}/STYX/stage/trees
STYXTREEGENERIC=${STYXTREE}/GENERIC_generic
mkdir -m 0755 -p ${STYXTREEGENERIC}

# default /boot hierarchy
 cd ${SB}/R/stage/trees/base && 
     find boot -depth | cpio -dump ${STYXTREEGENERIC}

# GENERIC kernel
 cd ${SB}/usr/src
 make buildkernel KERNCONF=GENERIC
 make installkernel DESTDIR=${STYXTREEGENERIC}

# maintenance RAMdisk
 mkmfs.sh

 cd `dirname ${STYXTREEGENERIC}` & & 
     tar cvBjf GENERIC_generic.tbz boot
mkdir ${mfs}
cd ${mfs}
mkdir -m 0755 -p boot dev etc mnt libexec lib/geom \
usr/share/misc var/db var/run root/.ssh mini
mkdir -m 1777 var/tmp
ln -s var/tmp tmp && ln -s mini bin && ln -s mini sbin
cd usr && ln -s ../bin && ln -s ../sbin
cd ${SB}/R/stage/trees/base/boot
cp boot boot0 boot0sio boot1 boot2 mbr ${mfs}
cp -p ${SB}/R/stage/trees/base/libexec/ld-elf.so.[0-9] ${mfs}/libexec
ln -s ${SB}/R/stage/trees/base/lib/geom/geom_*.so ${mfs}/lib/geom
do_crunchgen_with_a_twist && install_mini_crunch_into_mfs_mini
# i.e., /lib/geom/geom_eli.so, requires libmd.so and libcrypto.so
# see also man rtld(1)
ldd -f "cp -p ${SB}/R/stage/trees/base%p ${mfs}%p
lib/geom/* mini/mini_crunch | grep -v : | sort -u | /bin/sh -x
cp_some/etc/files/to/mfs/etc
# make RAMdisk image file fssize=7k (7MB) fsinode=32k (big binaries)
/usr/src/release/scripts/doFS.sh bsdlabel "" \
fs_img ${SB}/R/stage /mnt ${fssize} ${mfs} ${fsinode} auto
gzip -cv9 fs_img > ${STYXTREEGENERIC}/maint/fs.gz