A Remote Rescue Environment for FreeBSD Systems

LinuxTag 2006; Wiesbaden, Germany
Friday May 5, 2006; 10:00-11:00

Adrian Steinmann
Webgroup Consulting AG, Zürich
`<ast@webgroup.ch>`

Agenda

- What is a Remote Rescue Environment
- Introduction to RAMdisks and their uses
- Compact Flash and “Small” Hardware
- Building and Deploying the Rescue RAMdisk
- Status of Work in Progress
- Demonstration and Questions & Answers

FreeBSD Rescue Environment
Traditional versus Something New

Traditional

- Serial console
- **`boot -s`**
- **`fsck -y /`**
- **`/rescue`** (statically linked)

FreeBSD Rescue Environment
Traditional versus Something New

<u>Traditional</u>	<u>New Idea</u>
▸ Serial console	★ Serial console optional
▸ **`boot -s`**	★ **`ssh root@sickhost`**
▸ **`fsck -y /`**	★ RAMdisk root **always clean**
▸ **`/rescue`** (statically linked)	★ RAMdisk "mostly static"

“Single User Secure Shell”

The goal is
to be able to login remotely
onto a system with SSH
even when
the harddisk where the
root filesystem resides
is acting up!

RAMdisk to the Rescue

Swap-backed filesystems (i.e., /tmp)

Malloc-backed filesystems for read-write area in **read-only environments** (i.e., /var on compact flash or mfsroot on install CD)

RAMdisk to the Rescue

Swap-backed filesystems (i.e., /tmp)

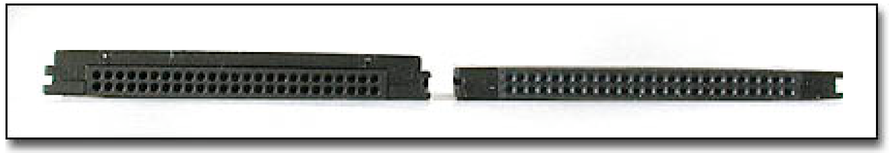
Malloc-backed filesystems for read-write area in **read-only environments** (i.e., /var on compact flash or mfsroot on install CD)

Create “disk images” to build custom distributions

```
dd if=/dev/zero of=somebackingfile bs=1k count=5k
mdconfig -a -t vnode -f somebackingfile -u 0
bsdlabel -w md0 auto
newfs md0c
mount /dev/md0c /mnt
```

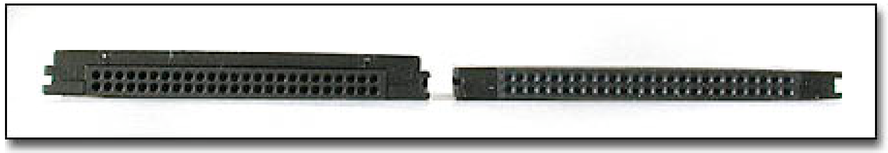
Linux often boots using “initrd” (initial ramdisk)

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
  Solution: mount filesystems **read-only!**

Compact Flash (CF)

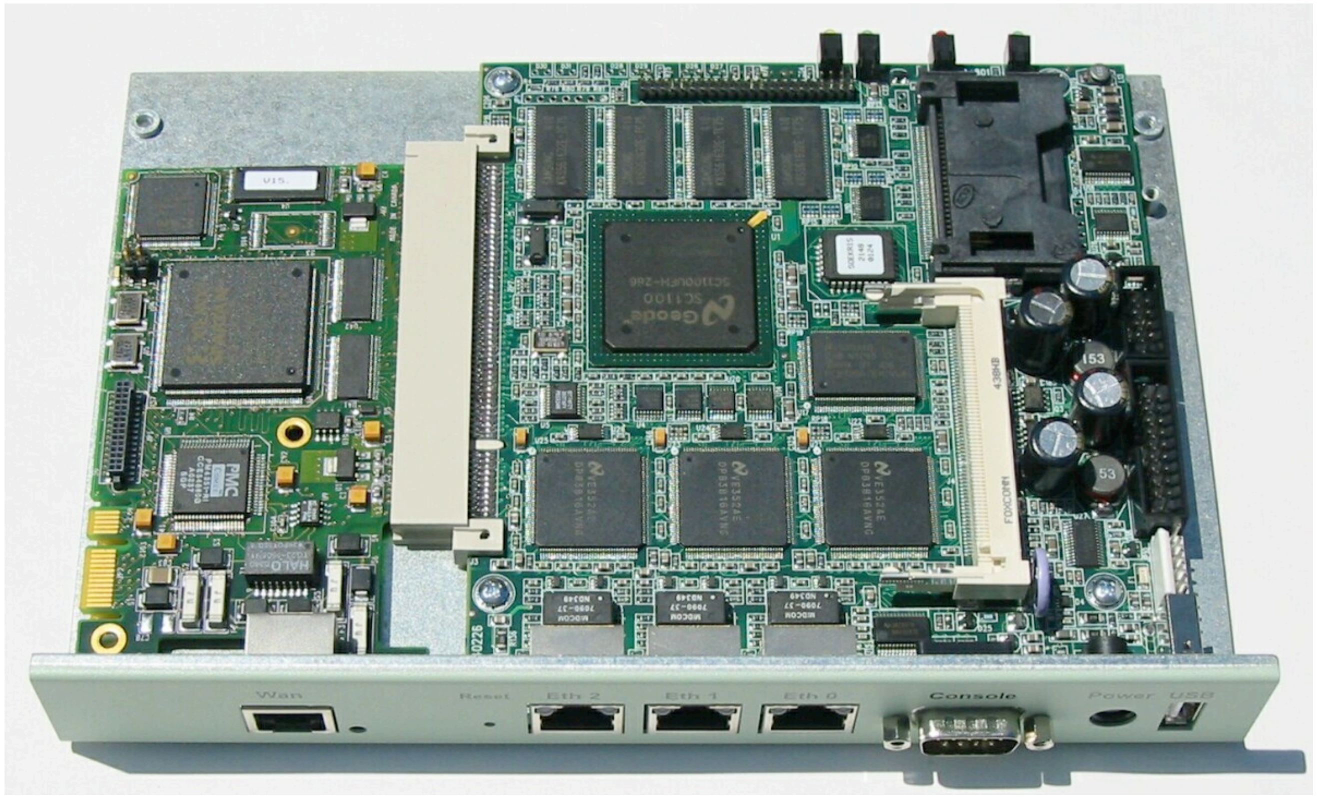


- Most are good for a million write/erase cycles
  www.robgalbraith.com/bins/multi_page.asp?cid=6007
- Superblocks of filesystems get written (saved) often, so a million writes is still not enough
  Solution: mount filesystems **read-only!**

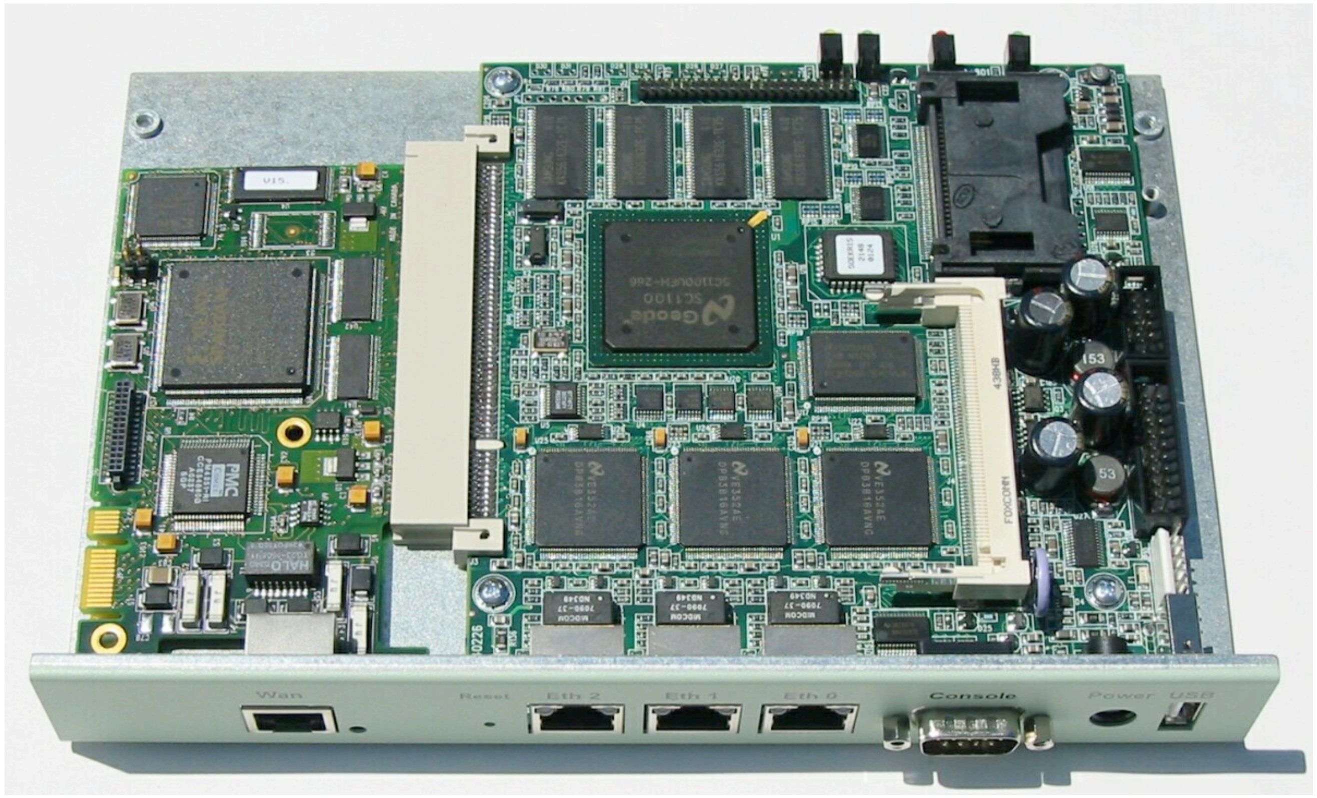
Mount read-write over read-only is automatic!

`touch /etc/diskless` activates startup script **`/etc/rc.initdiskless`** which copies **`/conf/base/<fs>`** RAMdisk templates

Small HW requires CF



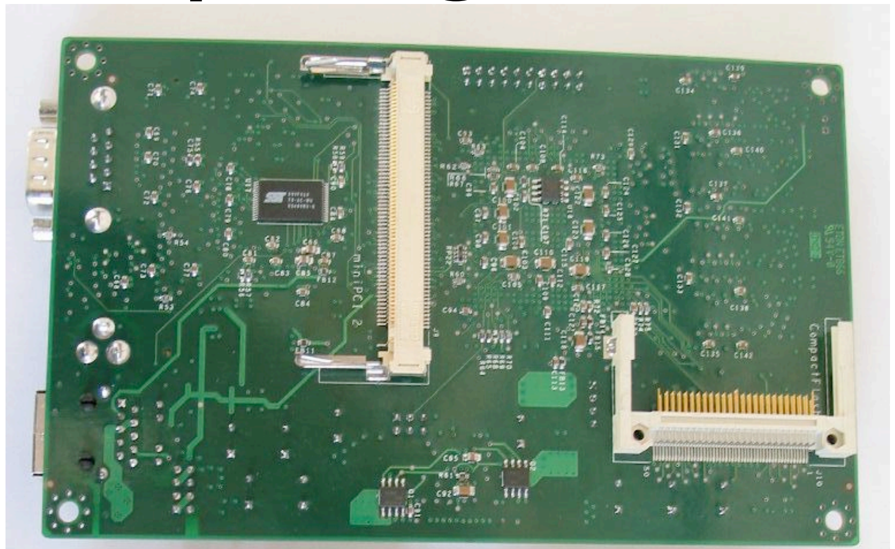
Soekris: www.soekris.com

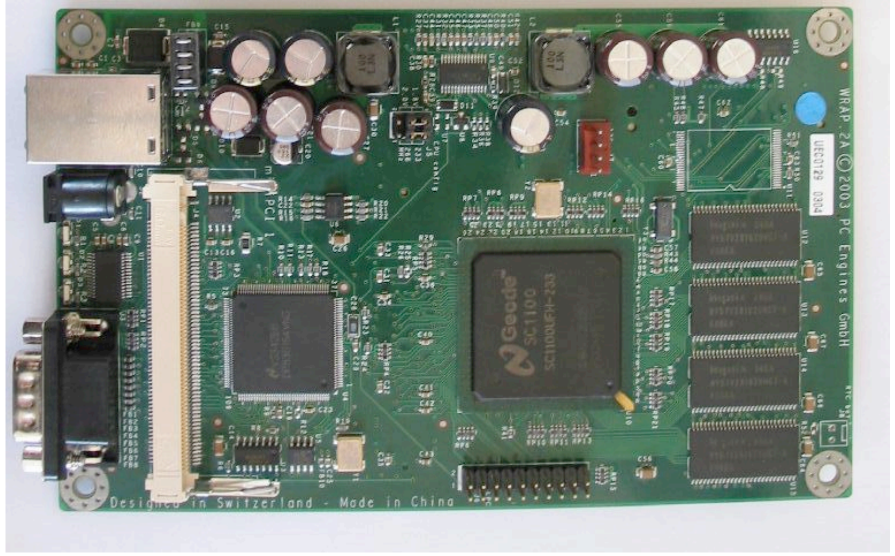


NET4801
NSC SC1100 266 Mhz CPU, 128 Mbyte SDRAM, 3 Ethernet, 2 serial
USB connector, CF socket, 44 pins IDE connector,
Mini-PCI socket, 3.3V PCI connector
here with Sangoma A101u E1/T1 PCI interface board

PC Engines: www.pcengines.ch

WRAP.2C
AMD Geode
SC1100 266 MHz
128MB SDRAM
1 serial, 1 Ethernet,
CF socket
2 Mini-PCI sockets




WRAP 2A © 2003 PC Engines GmbH
Geode SC1100
mini PCI 1
Designed in Switzerland - Made in China

FreeBSD for Small HW

Many choices!

- PicoBSD
- miniBSD
- m0n0wall
- pfsense
- NanoBSD
- STYX.

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
- 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 “Single User Secure Shell” Rescue/Maintenance RAMdisk
- Tracks FreeBSD since 3.x (-stable, -current)

Pit Stop

- ☑ What is a Remote Rescue Environment
- ☑ Introduction to RAMdisks and their uses
- ☑ Compact Flash and "Small" Hardware
- ● Building and Deploying the Rescue RAMdisk
  - Using **`crunchgen`** to make a "busybox" binary
  - Link "mostly static" instead of fully static
  - RAMdisk image generic, textfile configurable

The Deployment Plan

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

ii. Craft a RAMdisk filesystem image which can configure network and start SSH daemon

iii. Use the boot loader to preload the RAMdisk

The Deployment Plan

i. Use **`crunchgen`** to combine all commands into one "mostly static" binary

ii. Craft a RAMdisk filesystem image which can configure network and start 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 early from a **`/etc/rc.d`** startup script to check filesystem integrity or launch "maintenance SSH daemon" on alternate port

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 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


/lib/geom/geom_label.so
/lib/geom/geom_mirror.so

/lib/geom/geom_nop.so
/lib/geom/geom_raid3.so

/lib/geom/geom_shsec.so
/lib/geom/geom_stripe.so
```

“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 dynamically for “mostly static” crunched binaries via new **`libs_so`** keyword in **`crunchgen.conf`**:

```
libs_so -lmd -lcrypto -lgeom -lsbuf -lbsdxml
```

cruncgen with a twist

Linking dynamically for “mostly static” crunched binaries via new `libs_so` keyword in `crunchgen.conf`:

```
libs_so -lmd -lcrypto -lgeom -lsbuf -lbsdxml

progs geom
libs -lutil
special geom srcdir /usr/src/sbin/geom/core
ln geom gconcat
ln geom geli
ln geom glabel
ln geom gmirror
ln geom gnop
ln geom graid3
ln geom gshsec
ln geom gstripe
```

What’s on the RAMdisk ?

```
-sh
[               du                              mkdir


                                                                sh
                                                                sleep
                expr

                                hostname
                                                                stty
cat
chflags                                         mv
chgrp
chmod
chown                           kill
chroot
                                                ps
cp                                              pwd             test
date                                                            touch
                                                realpath        tset
df                              link
                                ln
                                ls                              unlink
                                                rm
                                                rmdir
```

Basics on RAMdisk

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

SysAdmin on RAMdisk

```
atacontrol                                      mknod
badsect         dumpfs                          mount
boot0cfg                                        mount_cd9660
bsdlabel                                        mount_devfs
                                                mount_fdescfs
                fastboot        halt            mount_linprocfs
                fasthalt
camcontrol      fdisk                           mount_procfs
                ffsinfo                         mount_std
                fsck                                            swapctl
                fsck_4.2bsd                     newfs           swapoff
                fsck_ffs                                        swapon
                fsck_ufs                                        sync
                                kldconfig                       sysctl
clri                            kldload
                                kldstat
                                kldunload
dd
                                                reboot          tunefs
                                                                umount

diskinfo                        mdconfig
disklabel                       mdmfs
```

Networking on RAMdisk

`route`

`ifconfig`

`ping`

```
dhclient
dhclient-script
```

More networking RAMdisk

```
route

scp

slogin
ssh
sshd

mount_nfs

ifconfig

ipf
ipfw

pfctl
ping

ggatec
ggated
ggatel

dhclient
dhclient-script
```

Archiving tools on RAMdisk

```
rrestore
dump
gunzip
gzcat
bunzip2
gzip
bzcat
bzip2
pax
tar
rdump
restore
zcat
```

Editors on the RAMdisk

`sed`

`ed`
`ex`

`red`

and last but not least ...

`vi`

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`

The Full Rescue RAMdisk

```
-sh              dmesg          graid3         mini_crunch      route
[                du             growfs         mkdir            rrestore
atacontrol       dump           gshsec         mknod            scp
badsect          dumpfs         gstripe        mount            sed
boot0cfg         ed             gunzip         mount_cd9660     sh
bsdlabel         ex             gzcat          mount_devfs      sleep
bunzip2          expr           gzip           mount_fdescfs    slogin
bzcat            fastboot       halt           mount_linprocfs  ssh
bzip2            fasthalt       hostname       mount_nfs        sshd
camcontrol       fdisk          ifconfig       mount_procfs     stty
cat              ffsinfo        init           mount_std        styxinstall
chflags          fsck           ipf            mv               swapctl
chgrp            fsck_4.2bsd    ipfw           newfs            swapoff
chmod            fsck_ffs       kenv           pax              swapon
chown            fsck_ufs       kill           pfctl            sync
chroot           gbde           kldconfig      ping             sysctl
clri             gconcat        kldload        ps               tar
cp               geli           kldstat        pwd              test
date             geom           kldunload      rdump            touch
dd               ggatec         ldconfig       realpath         tset
df               ggated         link           reboot           tunefs
dhclient         ggatel         ln             red              umount
dhclient-script  glabel         ls             restore          unlink
diskinfo         gmirror        mdconfig       rm               vi
disklabel        gnop           mdmfs          rmdir            zcat
```

RAMdisk versus /rescue

Additional on RAMdisk (today)

`boot0cfg`	`geli`	`gnop`	`scp`	`swapctl`
`chgrp`	`geom`	`graid3`	`sed`	`swapoff`
`chown`	`ggatec`	`growfs`	`sleep`	`touch`
`diskinfo`	`ggated`	`gshsec`	`slogin`	`tset`
`du`	`ggatel`	`gstripe`	`ssh`	
`ffsinfo`	`glabel`	`ipfw`	`sshd`	
`gconcat`	`gmirror`	`pfctl`	`styxinstall`	

Additional in /rescue (6.x)

`atm`	`fsdb`	`md5`	`nos-tun`	`setfacl`
`atmconfig`	`fsirand`	`mount_ext2fs`	`ping6`	`slattach`
`ccdconfig`	`getfacl`	`mount_msdosfs`	`raidctl`	`spppcontrol`
`chio`	`groups`	`mount_ntfs`	`rcorder`	`startslip`
`csh`	`id`	`mount_nullfs`	`rcp`	`tcsh`
`devfs`	`ilmid`	`mount_udf`	`routed`	`vinum`
`dumpon`	`ipfs`	`mount_umapfs`	`rtquery`	`whoami`
`echo`	`ipfstat`	`mount_unionfs`	`rtsol`	
`fore_dnld`	`ipmon`	`newfs_msdos`	`savecore`	
`fsck_msdosfs`	`ipnat`	`nextboot.sh`	`sconfig`	

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 ***
```

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
```

Two ways into RAMdisk

(1) Replacing **`/boot/loader.rc`**

(i.e., for remote re-installations)

```
include /boot/loader.4th
start
unload
load /boot/maint/k.CUSTOM
load -t md_image /boot/maint/fs_img
include /boot/maint/params
set vfs.root.mountfrom=ufs:/dev/md0
autoboot 10
```

(2) Starting from **`/etc/rc.d/maint_ssh`**

(i.e., for serial console replacement)

A Better Rescue

- ☑ 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"

Pit Stop

- ✓ What is a Remote Rescue Environment
- ✓ Introduction to RAMdisks and their uses
- ✓ Compact Flash and “Small” Hardware
- ✓ Building and Deploying the Rescue RAMdisk
- Status of Work in Progress
- Demonstration and Questions & Answers

Work in Progress

- ☑ Shell in "Fixit" Menu Item on Install CD has an additional "go into a rescue RAMdisk" function
- ☑ Rescue RAMdisk as initial root filesystem ("initrd") networked with running sshd() and geom() commands

Work in Progress

- ☑ Shell in "Fixit" Menu Item on Install CD has an additional "go into a rescue RAMdisk" function
- ☑ Rescue RAMdisk as initial root filesystem ("initrd") networked with running sshd() and geom() commands
- ☑ Mount real root on **`/a`**, mount devfs on **`/a/dev`**, and when necessary, mount real **`/usr`** on **`/a/usr`**
- 📌 Then, "exchange" root filesystem with **`/a`**, in other words, **`/a`** hierarchy becomes new root hierarchy, and oldroot (RAMdisk) becomes **`/mnt`** (was empty **`/a/mnt`**)
- ☐ Re-exec sshd() and init() and cleanup RAMdisk

BSD needs pivot_root() syscall

'Exchange root mountpoint with this one'

Linux "pivot_root(new, put_old)" syscall

AIX had it even earlier – there, it goes by the name of "getrootfs" in boot_serv_mode

FreeBSD kernel does something similar in
`kern/vfs_mount.c`
```
     devfs_fixup(struct thread *td)
```
where devfs – initially `/` – is swapped with **`/dev`**

Currently, my implementation does swap the mountpoints, but put_old is not visible/working

Demonstration
Q & A

- Remote Login ssh root@RescueRAMdisk
- Launching Rescue RAMdisk from boot loader
- “Fixit” Shell on Install CD with Rescue RAMdisk
- pivot_root() system call on FreeBSD-current
- Paper and Talk are available at

  `http://www.webgroup.ch/linuxtag2006/`