NetBSD on solid state

Date: 09 December 2003

Author: Roberto Trovò

RCSID="$Header$"

Disclaimer

Table of Contents

1 Document aim 4
2 Introduction to CF 4
3 HW & SW requirements 4
4 Installation 5
4.1 Create the “/tmp” symlink to “/var/tmp/” 5
4.2 Create the “/var” image 6
4.3 Create startup script 6
4.4 Configure the rc.conf 6
4.5 Delete unused files 7
4.6 Make the root file system read-only 7
4.7 Finishing the installation 8
Appendix A 9
Appendix B “dmesg” output 11
Appendix C acknowledgments 12

1 Document aim

The purpose of this document is to build a system running NetBSD on a Compact-Flash card (denoted as CF in the following), treated like to an ATA disk by means of CF-to-IDE adapter. Using other solid state device like DiskOnChip, SmartMedia etc should be possible as long as you take the necessary action.

This document is the dump of my distributed documentation (notes write by hand, downloaded Internet docs, mailing list search, etc.). I hope this writing will help me and any other reader to refine the ideas here contained.

2 Introduction to CF

TO DO!!!

Installing an O.S. on CF should be useful for embedded application/box that run on small motherboard that do not need standard hard disk that are a common failure point for a PC, due to its moving mechanical components. Instead CF is a solid state hard disk device.

The O.S. installed on CF, due to its limits, must have the root file system mounted as read-only. Instead **/var** and **/tmp** are mounted as read-write to permit the normal O.S. activity and therefore as memory file systems.

There are other installation method to achieve the same purpose, for example using compressed file system image, bootable cdrom, but according to me, these are quite hard to update: if you modify any configuration files you should rebuild the image. With our CF & read-only file system you can switch temporary on writable file system:

```
# mount -uw /          (re-mount rootfs rw);
(make the desired configuration changes);
# mount -ur /          (re-mount rootfs ro);
```

This method could be useful to save data periodically, for example log files, collected data etc. But remember CF cards are built with a limited number of write cycle: after that limit they were not operational. Even remember that all data on memory file system, that information will be lost after a reboot. From a certain point of view this could be an advantage: every time the system restart, it will start from the same initial configuration stored on CF. The shutdown operations are actually optional.

3 HW & SW requirements

The requirements are:

- a i386 PC (for embedded box best if fan less) with 64MB RAM (32MB RAM minimum I've tested) with a bootable CDROM (or floppy);
- CF card of 128MB of capacity (minimum 64MB I've tested);
- CF-to-IDE adapter: this passive connector permits O.S. to treat the CF as an ATA disk;
- NetBSD 1.6.1 install CDROM media (the release I used);

4 Installation

Install the CF card with the CF-IDE connector on the IDE controller (follow the instruction accompanying your adapter).

Then you should perform a typical NetBSD install on CF disk recognized as “wd0”, with the following suggestion:

- fdisk: create only one NetBSD slice as big as the whole drive and mark it as active (if you install NetBSD MBM it is optional);
- disklabel: create only a partition, the root one; you could optimize the file system to save space with **`newfs`** by reducing the number of inode (-i option) and to set zero the space reserved for super user (-m option), for example:

  ```
  # newfs -i <inode-#> -m 0 <device>
  ```

- do not create the swap partition (it must not be used on CF);
- perform a minimal install:

for a 64 MB CF:
base.tgz
etc.tgz
kernel.tgz
total install: 52MB

for a 128 MB CF:
base.tgz
etc.tgz
kernel.tgz
man.tgz
total install: 80MB

configure the system as required (host name, IP, default gateway ...) and finishing the installation.

4.1 Create the “/tmp” symlink to “/var/tmp/”

Reboot in single user: this will ensure no program is retain open files under **`/var`** and **`/`**

tmp. Move any files under **/tmp** (not **/tmp** itself) in **/var/tmp**, even the dot ones like **.Xfree86** if exists, then create the symlink:

```
# cd  /
# mv   /tmp/*   /tmp/.*   /var/tmp/
# rmdir tmp
# ln -s  var/tmp  /tmp
```

4.2 Create the "/var" image

Create the tar-gzipped image of **/var** file system:

```
# cd /
# tar -cvzf var-image.tar.gz var
```

and leave the image under **"/"**.

4.3 Create startup script

You must create a startup script under **/etc/rc.d** to create the memory file system (mfs) on "**/var**" and to load the tar-gzipped image just created on it:

```
# cd /etc/rc.d
# vi mount_mfs_fs
(insert the mount_msf_fs script in the appendix A)
```

In short the script mount a mfs on **/var** then load the files contained into the image created.

Then you should set the correct startup script sequence:

root -> mount_mfs_fs -> mountcritlocal

to ensure this edit **mountcritlocal** and use the utility **rcorder** to check the real sequence:

```
# vi  /etc/mountcritlocal
```

and set the required script: **# REQUIRE: mount_mfs_fs**

```
# cd  /etc/rc.d;  rcorder * | less
```

this should show the desired boot script sequence.

It is not necessary to create the **/dev** image as it will be created automatically by init, but could be useful make a copy for **/dev/MAKEDEV** and **/dev/MAKEDEV.local** under **/sbin** (even they do not seem used by init).

4.4 Configure the rc.conf

This is the main startup configuration file: you should edit it to disable the swap device and optionally set the hostname and other stuff you need:

```
# vi /etc/rc.conf
```

then insert the lines:

```
no_swap="YES"
hostname="your_desired_name"
```

4.5 Delete unused files

Remove unused files under **/var** and **/dev** because you will create them as mfs. Remember that we do not create the image dev-tar.gz as we did for **/var** because the init program will do it automatically upon reboot.

Then:

```
# cd /
# rm -rf /var/*
```

Do not delete the /var directory as it will be the mount point;

```
# mv /dev  /dev-old; mkdir /dev
```

Do not delete the special files because they (maybe) are open or used under the next shutdown;

Before reboot the system in single user again, please check that the mount point **/dev** has the right permission as **/dev-old**: if not use **chmod** to fix it.

On the next reboot you should see the following message:

```
...
warning: no /dev/console
init: Creating mfs /dev
...
Mounting memory file system: /var:
...
Mounting memory file systems: Done.
...
```

Remove **/dev-old** to free space:

```
# rm -rf /dev-old
```

4.6 Make the root file system read-only

After the reboot in single user, edit the startup script **/etc/rc.d/root** and **/etc/fstab** to mount read-only the whole system:

```
# vi  /etc/fstab
```

then change the "**rw**" into "**ro**": **/dev/wd0a / ffs ro 1 1**

```
# vi  /etc/rc.d/root
```

then change "**mount -w /**" into "**mount -r /**";

4.7 Finishing the installation

Then perform the final reboot: check the boot message to ensure all is correct.

Then login as root and check the mounted file system: you should see root as read-only, **/dev** and **/var** as memory file system.

Example for **mount** and **df**:

```
# mount
/dev/wd0a on / type ffs (read-only, local)
mfs:7 on /dev type mfs (synchronous, local)
mfs:2601 on /var type mfs (synchronous, local)
```

```
# df
Filesystem  1K-blocks   Used   Avail Capacity iused  ifree  %iused
Mounted on
/dev/wd0a      121023  81240   33731    70%    5503  24959    18%   /
mfs:7             103     51      47    52%    1642    372    81%   /dev
mfs:2601         7903   1399    6108    18%     104   1942     5%   /var
```

Appendix A

This is the mount_mfs_fs script:

```
#!/bin/sh
#
# mount_mfs_fs: mount memory file system /var and /dev
# by roby, 23 jun 2003

# PROVIDE: mount_mfs_fs
# REQUIRE: root

. /etc/rc.subr

name="mount_mfs_fs"
start_cmd="mount_mfs_fs_start"
stop_cmd=":"

mount_mfs_fs_start()
{
     echo "Mounting memory file system: /var:"
     #    Mount /dev fs
     #
     #mount_mfs none /dev
     #
     # build device files
     #cd /dev
     #/sbin/MAKEDEV all

     #    Mount /var fs
     #
     mount_mfs none /var
     #
     # build dir/files in var
     tar -xvzpf /var-image.tar.gz -C /
     echo "Mounting memory file systems: Done."
     sleep 5
}

load_rc_config $name
run_rc_command "$1"
```

END mount_mfs_fs script

This is the mountcritlocal modified script:

```
#!/bin/sh
#
```

```
# $NetBSD: mountcritlocal,v 1.7 2002/04/29 12:29:53 lukem Exp $
#

# PROVIDE: mountcritlocal
# REQUIRE: mount_mfs_fs

. /etc/rc.subr

name="mountcritlocal"
start_cmd="mountcritlocal_start"
stop_cmd=":"

mountcritlocal_start()
{
     #    Mount critical filesystems that are `local'
     #    (as specified in $critical_filesystems_local)
     #    This usually includes /var.
     #
     mount_critical_filesystems local

     #    clean up left-over files.
     #    this could include the cleanup of lock files and /
var/run, etc.
     #
     rm -f /etc/nologin /var/spool/lock/LCK.* /
var/spool/uucp/STST/*
     (cd /var/run && rm -rf -- *)
}

load_rc_config $name
run_rc_command "$1"
```

END mount_mfs_fs script

This is the **root** modified script:

```
#!/bin/sh
#
# $NetBSD: root,v 1.2 2000/05/13 08:45:09 lukem Exp $
#

# PROVIDE: root
# REQUIRE: fsck

. /etc/rc.subr

name="root"
start_cmd="root_start"
stop_cmd=":"

root_start()
```

```
{
        #umount -a >/dev/null 2>&1
        #mount /
        umount -a
        mount -r /
        rm -f /fastboot
}

load_rc_config $name
run_rc_command "$1"
```

END root script

Appendix B “dmesg” output

```
NetBSD 1.6.1 (GENERIC) #0: Tue Apr  8 12:05:52 UTC 2003
    autobuild@tgm.daemon.org:/autobuild/netbsd-1-6/i386/OBJ/autobuild/netbsd-1-
6/src/sys/arch/i386/compile/GENERIC
cpu0: IDT Pentium Pro compatible (686-class), 532.66 MHz
cpu0: features 803035<FPU,DE,TSC,MSR,MTRR>
cpu0: features 803035<PGE,MMX>
total memory = 247 MB
avail memory = 223 MB
using 3195 buffers containing 12780 KB of memory
BIOS32 rev. 0 found at 0xfb500
mainbus0 (root)
pci0 at mainbus0 bus 0: configuration mode 1
pci0: i/o space, memory space enabled, rd/line, rd/mult, wr/inv ok
pchb0 at pci0 dev 0 function 0
pchb0: VIA Technologies product 0x0601 (rev. 0x05)
agp0 at pchb0: aperture at 0xe0000000, size 0x10000000
ppb0 at pci0 dev 1 function 0: VIA Technologies product 0x8601 (rev. 0x00)
pci1 at ppb0 bus 1
pci1: i/o space, memory space enabled
vga1 at pci1 dev 0 function 0: Trident Microsystems product 0x8500 (rev. 0x6a)
wsdisplay0 at vga1 kbdmux 1: console (80x25, vt100 emulation)
wsmux1: connecting to wsdisplay0
pcib0 at pci0 dev 7 function 0
pcib0: VIA Technologies VT82C686A (Apollo KX133) PCI-ISA Bridge (rev. 0x40)
pciide0 at pci0 dev 7 function 1: VIA Technologies VT82C686A (Apollo KX133) ATA100 controller
pciide0: bus-master DMA support present
pciide0: primary channel configured to compatibility mode
pciide0: primary channel ignored (disabled)
pciide0: secondary channel configured to compatibility mode
wd0 at pciide0 channel 1 drive 0: <SanDisk SDCFB-128>
wd0: drive supports 1-sector PIO transfers, LBA addressing
wd0: 122 MB, 980 cyl, 8 head, 32 sec, 512 bytes/sect x 250880 sectors
wd0: 32-bit data port
pciide0: secondary channel interrupting at irq 15
wd0(pciide0:1:0): using PIO mode 0
VIA Technologies VT82C686A SMBus Controller (miscellaneous bridge, revision 0x40) at pci0 dev 7
function 4 not configured
rtk0 at pci0 dev 8 function 0: RealTek 8139 10/100BaseTX
rtk0: interrupting at irq 12
rtk0: Ethernet address 00:40:f4:8a:03:71
ukphy0 at rtk0 phy 7: Generic IEEE 802.3u media interface
ukphy0: OUI 0x000000, model 0x0000, rev. 0
```

```
ukphy0: 10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
rtk1 at pci0 dev 9 function 0: RealTek 8139 10/100BaseTX
rtk1: interrupting at irq 10
rtk1: Ethernet address 00:40:f4:8a:03:70
ukphy1 at rtk1 phy 7: Generic IEEE 802.3u media interface
ukphy1: OUI 0x000000, model 0x0000, rev. 0
ukphy1: 10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
rtk2 at pci0 dev 11 function 0: RealTek 8139 10/100BaseTX
rtk2: interrupting at irq 11
rtk2: Ethernet address 00:40:f4:8a:03:6f
ukphy2 at rtk2 phy 7: Generic IEEE 802.3u media interface
ukphy2: OUI 0x000000, model 0x0000, rev. 0
ukphy2: 10baseT, 10baseT-FDX, 100baseTX, 100baseTX-FDX, auto
isa0 at pcib0
pckbc0 at isa0 port 0x60-0x64
pckbd0 at pckbc0 (kbd slot)
pckbc0: using irq 1 for kbd slot
wskbd0 at pckbd0: console keyboard, using wsdisplay0
pcppi0 at isa0 port 0x61
midi0 at pcppi0: PC speaker
sysbeep0 at pcppi0
isapnp0 at isa0 port 0x279: ISA Plug 'n Play device support
npx0 at isa0 port 0xf0-0xff: using exception 16
isapnp0: no ISA Plug 'n Play devices found
biomask e3fd netmask fffd ttymask ffff
Kernelized RAIDframe activated
boot device: wd0
root on wd0a dumps on wd0b
root file system type: ffs
warning: no /dev/console
wsdisplay0: screen 1 added (80x25, vt100 emulation)
wsdisplay0: screen 2 added (80x25, vt100 emulation)
wsdisplay0: screen 3 added (80x25, vt100 emulation)
wsdisplay0: screen 4 added (80x25, vt100 emulation)
```

Appendix C acknowledgments

I will thanks all the NetBSD mailing list user that helped me.