Pivot Root for BSD

Adrian Steinmann
ast@NetBSD.org
Agenda

Describe the problem

Demonstrate a solution

Outline the implementation
What is pivot_root?

What pivot_root is not

pivot_root system call in action

pivot_root on the command line
pivot_root is ...

not “chroot”

Changes the root fs for a new process and subsequent children

nor is it an “init_chroot”

A sysctl or kenv telling init where to chroot when invoked at boot time

and not a “FreeBSD jail” either
pivot_root on CLI

```
# df
Filesystem   1K-blocks  Used    Avail  %Cap Mounted on
/dev/wd0a    930829976  6838212 877450266  0%  /
/dev/wd1a    945135288  5780444 892098080  0%  /new_root

# pivot_root /new_root /new_root/put_old

# df
Filesystem   1K-blocks  Used    Avail  %Cap Mounted on
/dev/wd1a    945135288  5780444 892098080  0%  /
/dev/wd0a    930829976  6838212 877450266  0%  /put_old
```
Why pivot_root?

Traditional – used over a decade

Modern – avoid reboots of large boxes

Embedded – firmware is a BSD system
Linux initrd

Booting is a two-stage boot process

First boot into the Linux *initial ramdisk*

The initrd has executables and drivers to prepare and mount the root fs

After real root fs is mounted, initrd is unmounted and its memory is freed
Special root fs setups

Encrypted root fs

cgd on NetBSD

geli or gbde on FreeBSD

Boot into a RAMdisk

setup root

pivot_root
OS installs

Mimic a traditional initial install

Install RAMdisk prepares rootfs (chroot)

Reboots into that rootfs

A pivot_root could skip that reboot
Firmware upgrades

1. Clone current rootfs into RAMdisk
2. First pivot_root into this new RAMdisk
3. Upgrade the firmware rootfs
4. Second pivot_root back into upgraded rootfs

Works for all upgrades without kernel updates
Design decisions

Dedicated syscall as a loadable kernel module?

Or piggy-back mount syscall?

```
mount -t pivot /new_root /new_root/put_old
```

What should userland stub need to do additionally?

Allow pivot_root for FreeBSD jails?
/usr/sbin/pivot_root

/* check if root_new exists and normalize it */
if (realpath(argv[1], real_new) == NULL) {
    fprintf(stderr, "invalid root_new '%s'\n", real_new);
    exit(EINVAL);
}
/* ditto for real_old, check path lengths */
...

/* do syscall with normalized pathnames */
if (pivot_root(real_new, real_old) != 0) {
    perror("pivot_root");
    exit(errno);
}

signal(1, (void *) 1);
...
Syscall implementation

VFS kernel programming involving the rootvnode
vnodes and mountpoints
vnode locks and vnode references
name lookup cache management
the mountlist and mountpoint locks
and last but not least, all the running processes
Syscall input validation

* Pseudocode for pivot_root(root_new, root_old)

* Get rootvnode from point of view of process, return EPERM
  * unless it is the real rootvnode (i.e., not in chroot env)

* Get ovp (vnode of root_old)
  * Exit if not a suitable new mountpoint for old root
* Get nvp (vnode of root_new)
  * Exit if not a mountpoint distinct from root
  * Exit if ovp not under nvp

* Chdir to new root if cwd is not under new root

* ...
Syscall: actual pivot

* ... *
* * Lock mountlist *
* * Move nmp (mountpoint of new root) to head of mountlist *
* * Adjust mount information for all mountpoints (use locks) *
* * Adjust nvp, ovp, and original rootvnode information (locks) *
* * Notify all processes that rootvnode has changed *
* * Move /dev (FreeBSD) *
* * Adjust sysctl kern.root* “constants” (NetBSD)
VFS semantics

vnode via \texttt{NDINIT()} and \texttt{namei()} 
or \texttt{namei\_simple\_kernel} / \texttt{namei\_simple\_user()}

vnode references via \texttt{vref()} / \texttt{vrele()}

vnode locks via \texttt{namei()} or \texttt{vn\_lock()}
unlocks via \texttt{vput()} or \texttt{VOP\_UNLOCK()}

mountlist locks via \texttt{mountlist\_lock} mutex

mountpoint locks via \texttt{mp->mnt\_updating} mutex

\texttt{mount\_checkdirs(rootvnode)} in \texttt{vfs\_mount.c}
adjusts cwd of process on \texttt{put\_old} and then
changes rootvnode to \texttt{new\_root}
How to debug *vp

vfs_vnode_print()

VNODE DEBUG rootvnode:
OBJECT 0xffffffff811d212cd0: locked=0, ..., refs=11

VNODE flags 0x31<ROOT,MPSAFE,LOCKSWORK>
mp 0xffffffff811d175000 numoutput 0 size 0x200 writesize 0x200
data 0xffffffff811d237f00 writecount 0 holdcnt 1
tag VT_UFS(1) type VDIR(2) mount 0xffffffff811d175000 ...
v_lock 0xffffffff811d212de0
How to debug *mp

vfs_mount_print()

MOUNTPOINT DEBUG mountlist mp:

vnodecovered = 0xffffffffcfaaf8 syncer = ...
fs_bshift 14 dev_bshift = 9
flag = 0x1000<\texttt{MNT\_LOCAL}>
iflag = 0x1c0<\texttt{IMNT\_MPSAFE, IMNT\_HAS\_TRANS, IMNT\_DTYPE}>
refcnt = 4 unmounting @ 0xffffffff811dd85048 ...

statvfs cache:

...  
mntonname = \texttt{/new\_root}

mntfromname = \texttt{/dev/\texttt{wd2a}}

locked vnodes = ...
What to watch

vnode vp

mountpoint vp->v_mount

mountpoint vp->v_mountedhere

vnode vp->v_mount->mnt_vnodecovered

mountpoint vp->v_mount->mnt_vnodecovered->v_mountedhere
Been there, done that

**EPERM** when invoked from chroot environment!

When validating input, avoid locking against oneself

Locking two vnodes concurrently: potential race

Old root file system busy after pivot_root

It works, but: panic or lockups later or at reboot
Will be doing soon ...

Test and document typical use cases

```
pivot_root.kmod, /usr/sbin/pivot_root
mount -t pivot ...
```

Syscall shim for linux emulation

Update sysctl kern.root* data (NetBSD)

Move /dev back to / (FreeBSD)

What about /dev/pts (NetBSD)
Short Demo

23MB netbsd-RAMDISK (uncompressed)

GENERIC INSTALL Kernel (HEAD)

Custom RAMdisk 8MB with sshd, /root/ ...

pivot_root.kmod is loaded

pivot_root /wd1 /wd1/put_old

Restart all process to reopen file descriptors on new root

umount -f RAMdisk