

Operating Systems Design

12a. Special File Systems

Paul Krzyzanowski
pxk@cs.rutgers.edu

Generic Interfaces via VFS

- VFS gives us a generic interface to file operations
- We don't need to have persistent storage underneath ... or even storage!

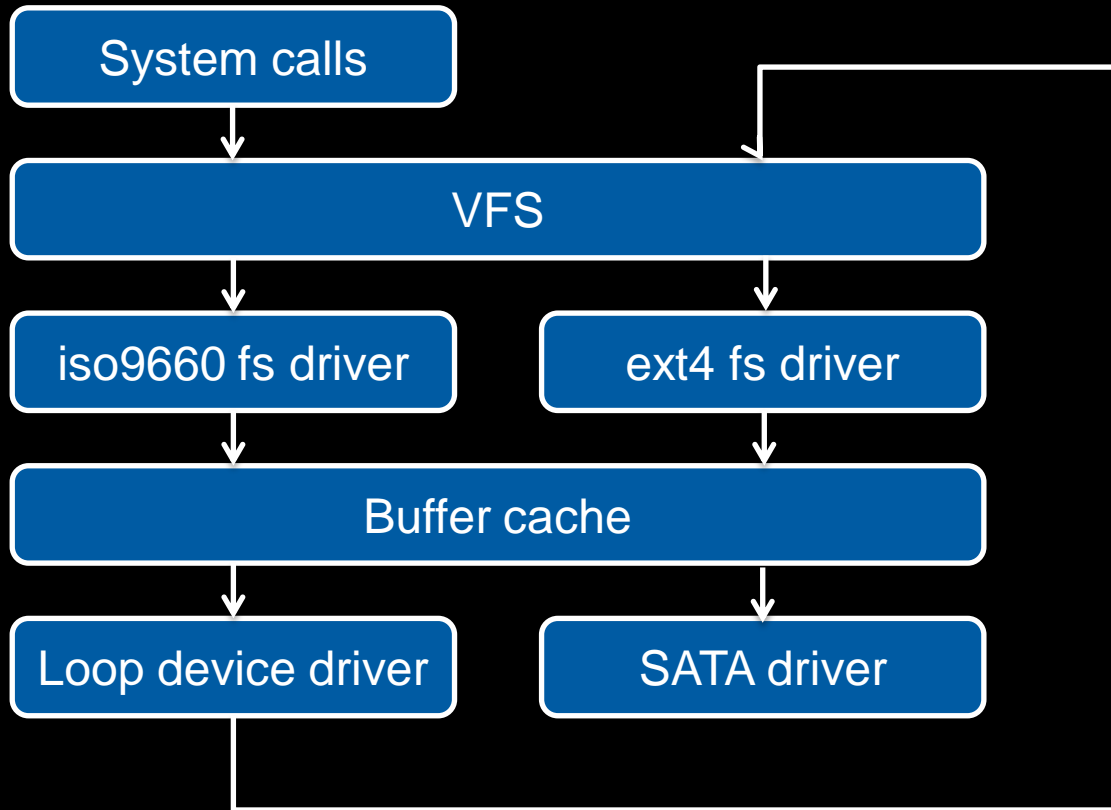
Simple special-function files

- `/dev/null` *Null device*
 - Throw away anything written to it; return EOF on reads
- `/dev/zero` *Zero device*
 - Return 0x00 bytes for each read operation
- `/dev/random`, `/dev/urandom` *Random numbers*
 - `urandom` is non-blocking
 - `\Device\KsecDD` on Windows NT

Loop device

- Pseudo device
 - Provides a block device interface to a file
 - Register as a block device
 - Let the buffer cache know:
 - *request (strategy)* procedure for read/write
 - block size
 - The file can then be formatted with a file system and mounted
 - See the *losetup* command in Linux
 - Common uses
 - installation software
 - CD/DVD images
 - Encrypted file systems

Loop device



Example: (1) Create a loop device

Create a 10MB file named file.img

```
# dd if=/dev/zero of=file.img bs=1k count=10000
10000+0 records in
10000+0 records out
```

Associate loop device /dev/loop0 with the file file.img

```
# losetup /dev/loop0 file.img
```

This makes /dev/loop0 a block device whose contents are file.img

```
# ls -l /dev/loop0
```

```
brw-rw---- 1 root disk 7, 0 2011-03-06 17:17 /dev/loop0
```

Example: (2) Put a file system on the file

Create a file system on /dev/loop0

```
# mke2fs -c /dev/loop0 10000
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
...
```

Example: (3) Mount it

Create a directory that will be the mount point

```
# mkdir /mnt/here
```

Mount the file system

```
# mount -t ext2 /dev/loop0 /mnt/here
```

Test it out!

```
# ls -l /mnt/here
```

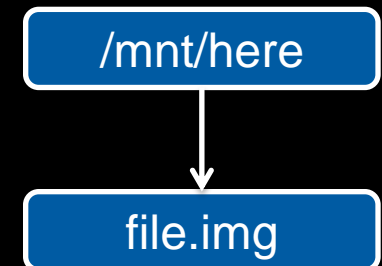
```
total 12
```

```
drwx----- 2 root root 12288 2011-03-06 17:18 lost+found
```

```
# echo hello >/mnt/here/hello.txt
```

```
# cat /mnt/here/hello.txt
```

```
hello
```



Example: Do it recursively!

Create a 1000KB file called another.img within the file.img file system

```
# dd if=/dev/zero of=/mnt/here/another.img bs=1k count=1000
1000+0 records in
1000+0 records out
```

Make /dev/loop1 be a loop device that points to another.img

```
# losetup /dev/loop1 /mnt/here/another.img
```

Create a file system

```
# mke2fs -c /dev/loop1 1000
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
...

#
```

Example: Do it recursively!

Create a directory (/mnt/there) that will be the mount point

```
# mkdir /mnt/there
```

mount the file system

```
# mount -t ext2 /dev/loop1 /mnt/there
```

Test it!

```
# echo hey! >/mnt/there/test
```

```
# ls -l /mnt/there
```

```
total 13
drwx----- 2 root root 12288 2011-03-06 17:22 lost+found
-rw-r--r-- 1 root root      5 2011-03-06 17:22 test
```

It works! Another.img is a file system within file.img which is a file system on the disk

```
# ls -l /mnt/here
```

```
total 1018
-rw-r--r-- 1 root root 1024000 2011-03-06 17:23 another.img
-rw-r--r-- 1 root root      6 2011-03-06 17:19 hello.txt
drwx----- 2 root root 12288 2011-03-06 17:18 lost+found
```

```
# umount /mnt/there
```

/mnt/there/text:
File in a file system (/mnt/there)
that is a file (another.img)
within a file system (/mnt/here)
that is a file (file.img)
within a file system (top-level)

procfs: process file system

- /proc
 - View & control processes as files & other kernel structures
- Origins: Plan 9 from Bell Labs
- Does not exist in the file system
- procfs is a file system driver
 - Registers itself with VFS
 - When VFS calls to request inodes as files & directories are accessed, /proc creates them from info within kernel structures.
 - /proc/devices

proafs: process file system

- Remove the need for system calls to get info, read config parameters, and inspect processes
- Simplify scripting
- Just a few items:
 - /proc/cpuinfo info about the cpu
 - /proc/devices list of all character & block devices
 - /proc/diskstats info on logical disks
 - /proc/meminfo info on system memory
 - /proc/net directory containing info on the network stack
 - /proc/swaps list of swap partitions
 - /proc/uptime time the system has been up
 - /proc/version kernel version
- Plan 9 allowed remote access to /proc

proafs: process info

```
$ ls /proc/27325
```

```
attr          cwd          loginuid     oom_adj      smaps
auxv          environ     maps        oom_score    stack
cgroup       exe         mem         pagemap      stat
clear_refs   fd          mountinfo   personality   statm
cmdline      fdinfo     mounts      root         status
comm        io         mountstats  sched        syscall
coredump_filter latency    net         schedstat    task
cpuset      limits    numa_maps   sessionid    wchan
```

devfs: device file system

- Special file system mounted on /dev
 - Presents device files
 - Avoids having to create device special files in /dev
 - Obsolete since Linux 2.6; still used in OS X and others
- udev device manager
 - User space manager reads device events from /sbin/hotplug
 - Persistent device naming

The End