An Introduction to Userspace Filesystem Development

Matt Turner
ACCU Oxford
25th April 2012

Why Filesystems?

- Filesystems are everywhere
- Data in a namespace
- Lots of tooling

```
matt@frogstar:~$ mount
/dev/sda1 on / type ext4 (rw,errors=remount-ro,commit=0)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
fusectl on /sys/fs/fuse/connections type fusectl (rw)
none on /sys/kernel/debug type debugfs (rw)
none on /sys/kernel/security type securityfs (rw)
udev on /dev type devtmpfs (rw,mode=0755)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=0620)
tmpfs on /run type tmpfs (rw,noexec,nosuid,size=10%,mode=0755)
none on /run/lock type tmpfs (rw,noexec,nosuid,nodev,size=5242880)
none on /run/shm type tmpfs (rw,nosuid,nodev)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc
       (rw,noexec,nosuid,nodev)
gvfs-fuse-daemon on /home/matt/.gvfs type fuse.gvfs-fuse-daemon
       (rw,nosuid,nodev,user=matt)
```

Why Filesystems?

- In Unix, everything is a file
- Lifts data into the filesystem namespace
- Can be queried and manipulated with common tools

Sex in the Filesystem

```
unzip; strip; touch; finger; mount;
fsck; more; yes; umount; make
clean; sleep
```

Filesystems to the Extreme

- Under Plan9, **everything** is a file
- Processes have their own mount tables
- Individual processes can be “chroot()ed”
- 9P protocol allows for easy remote filesystem access
- Unionfs replaces $PATH
- Network represented as /net

Why Filesystems?

- UNIX inspired by Plan9 – more and more data in the filesystem
- smbclient, ftp replaced by filesystems
- New protocols should come with filesystems!

Example – A VM Filesystem

- VM hdd image is one big file
- Most blocks not accessed
- Some blocks accessed all the time
- Local modifications like log files not wanted globally
- Domain knowledge meant files could be handled cleverly, but couldn’t alter VMWare

Filesystems in Kernelspace

- Filesystems used to be in the Kernel
- Kernel development is hard!

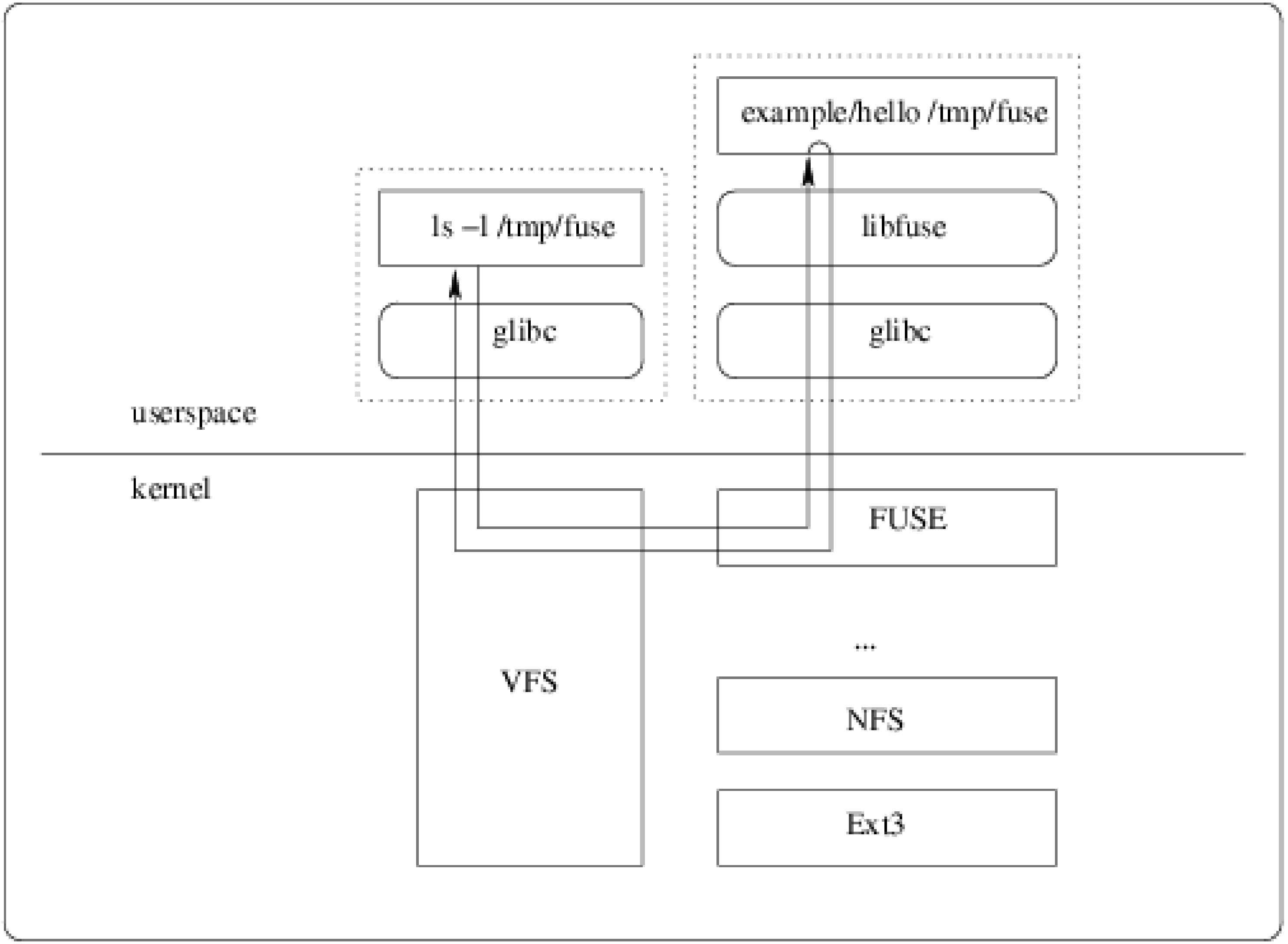
Filesystems in Userspace

- Much easier to develop, test, etc
- “filesystems” in libraries:
  - gfs (C++ API),
  - HDFS (Java API)
- Filesystem frameworks:
  - FUSE (& CUSE),
  - Dokan[.Net]

Filesystem in USErspace (FUSE)

- Not the spectrum emulator
- Fuse currently at version 2.9.0 (released Friday)
- Shipping with Linux kernel since 2.6.14
- Multiple OSes
- Native C, bindings to many other languages
- Lots of successful projects
- Tarballs at fuse.sourceforge.net
- git://fuse.git.sourceforge.net/gitroot/fuse/fuse

Architecture



[fuse.sourceforge.net]

```
Thread 1 (Thread 0x7ffff7fe5700 (LWP 15341)):
#0  sem_wait () from /lib64/libpthread.so.0
#1  fuse_session_loop_mt () at fuse_loop_mt.c:242
#2  fuse_loop_mt () at fuse_mt.c:117
#3  fuse_main_common () at helper.c:353
#4  __libc_start_main () from /lib64/libc.so.6
#5  _start ()
```

```
Thread 2 (Thread 0x7ffff71ee700 (LWP 15344)):
#0  read () from /lib64/libpthread.so.0
#1  read () at /usr/include/bits/unistd.h:45
#2  fuse_kern_chan_receive () at fuse_kern_chan.c:28
#3  fuse_ll_receive_buf () at fuse_lowlevel.c:2643
#4  fuse_do_work () at fuse_loop_mt.c:81
#5  start_thread () from /lib64/libpthread.so.0
#6  clone () from /lib64/libc.so.6
```

```
Breakpoint 4, xmp_access (path=0x623da0 "/", mask=4) at
fusexmp.c:48
48      {
(gdb) bt
Thread 3 (Thread 0x7ffff69ed700 (LWP 15345)):
#0  xmp_access (path=0x623da0 "/", mask=4) at fusexmp.c:48
#1  fuse_lib_access () at fuse.c:2765
#2  fuse_ll_process_buf () at fuse_lowlevel.c:2416
#3  fuse_do_work () at fuse_loop_mt.c:117
#4  start_thread () from /lib64/libpthread.so.0
#5  clone () from /lib64/libc.so.
```

Bash Filesystem

- Has bit-rot ☹

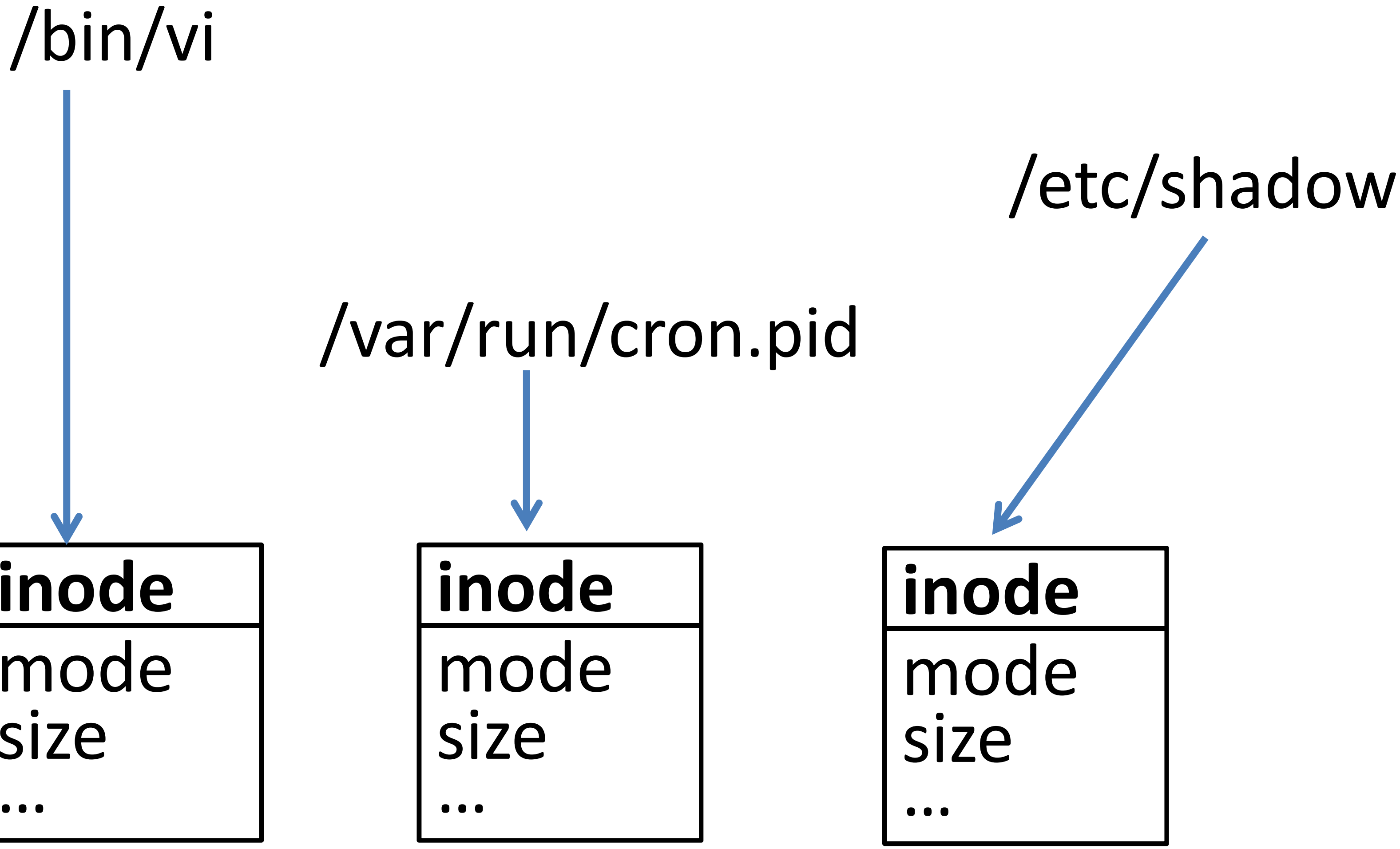
POSIX Filesystem API

/bin/vi

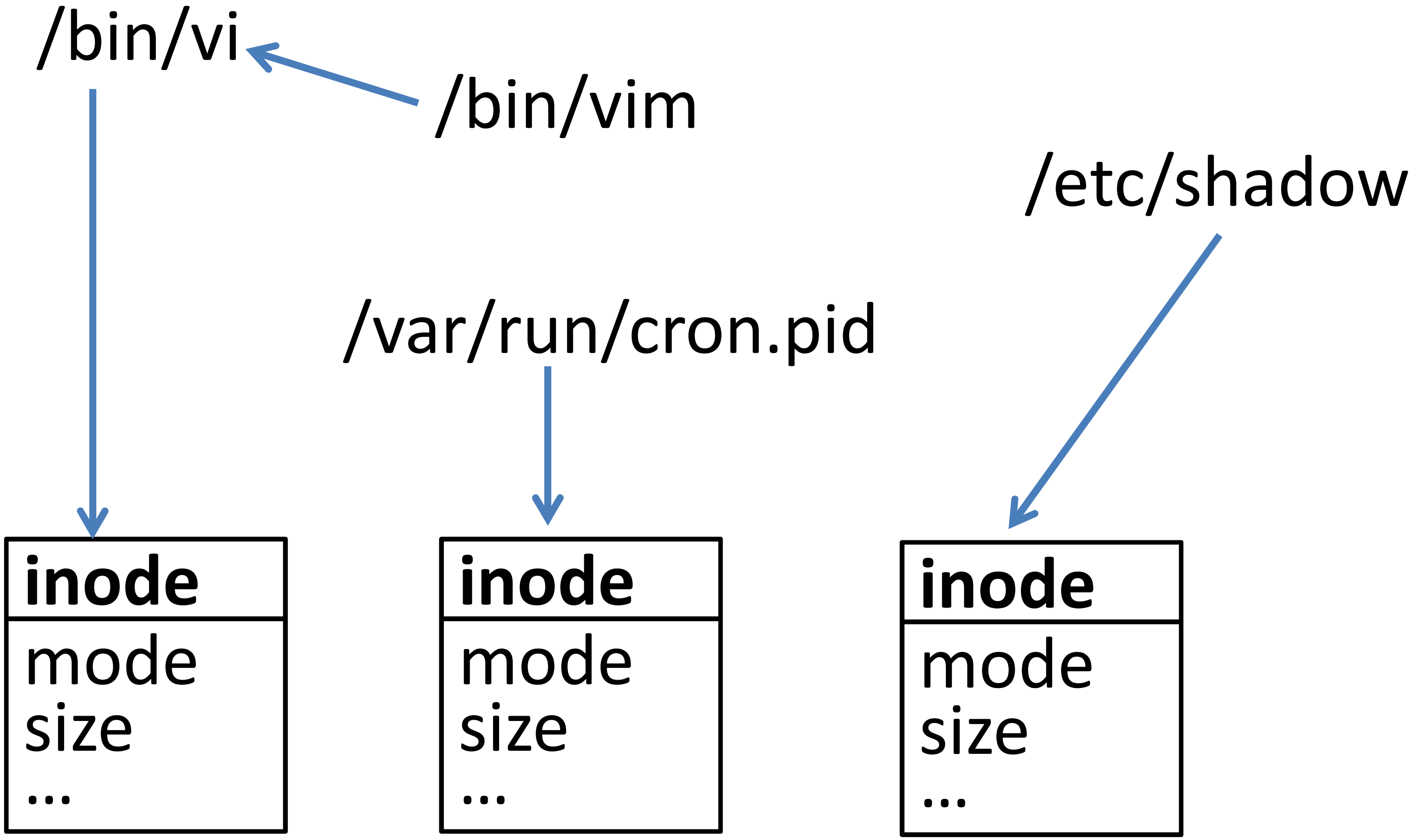
/etc/shadow

/var/run/cron.pid

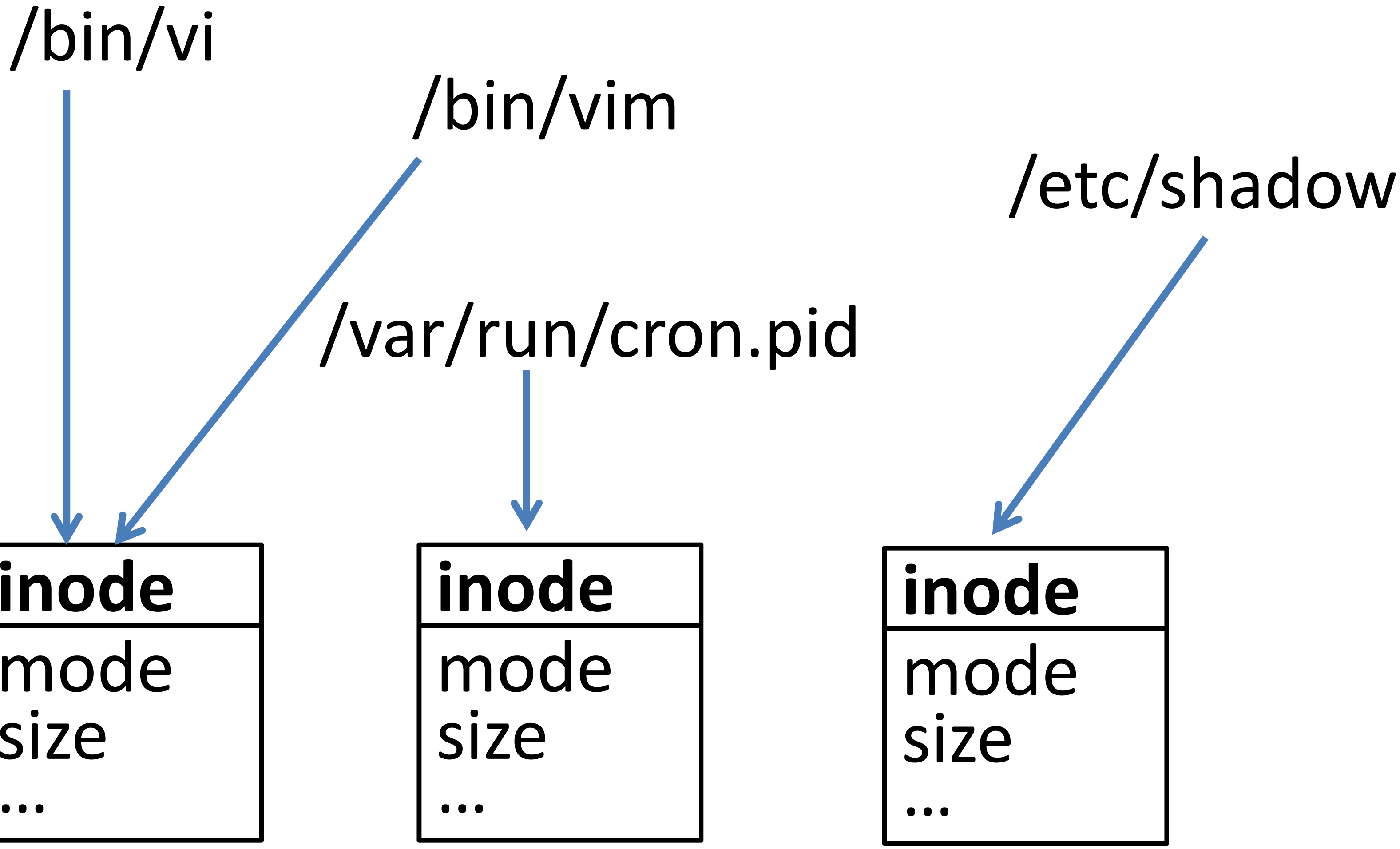
POSIX Filesystem API



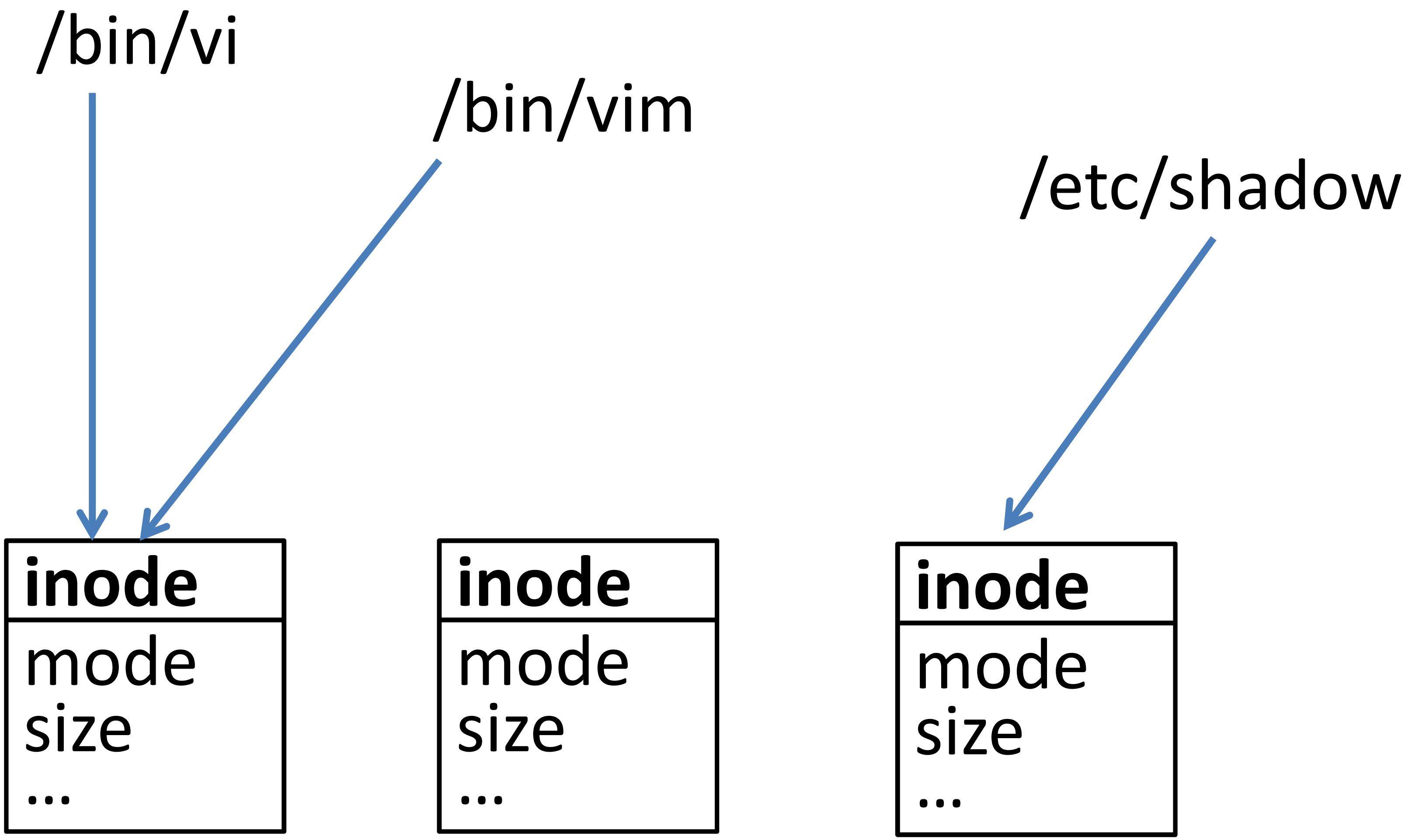
POSIX Filesystem API



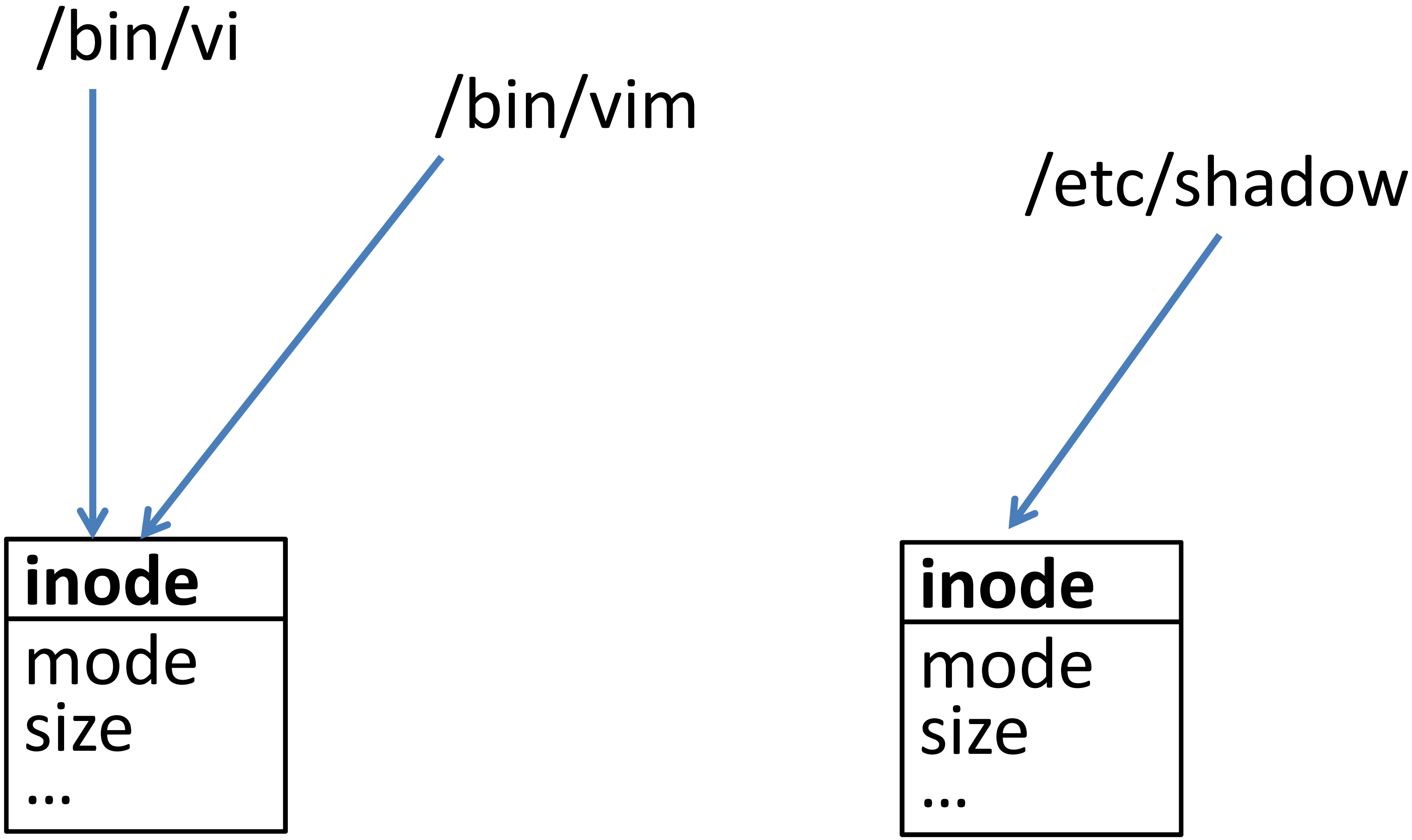
POSIX Filesystem API



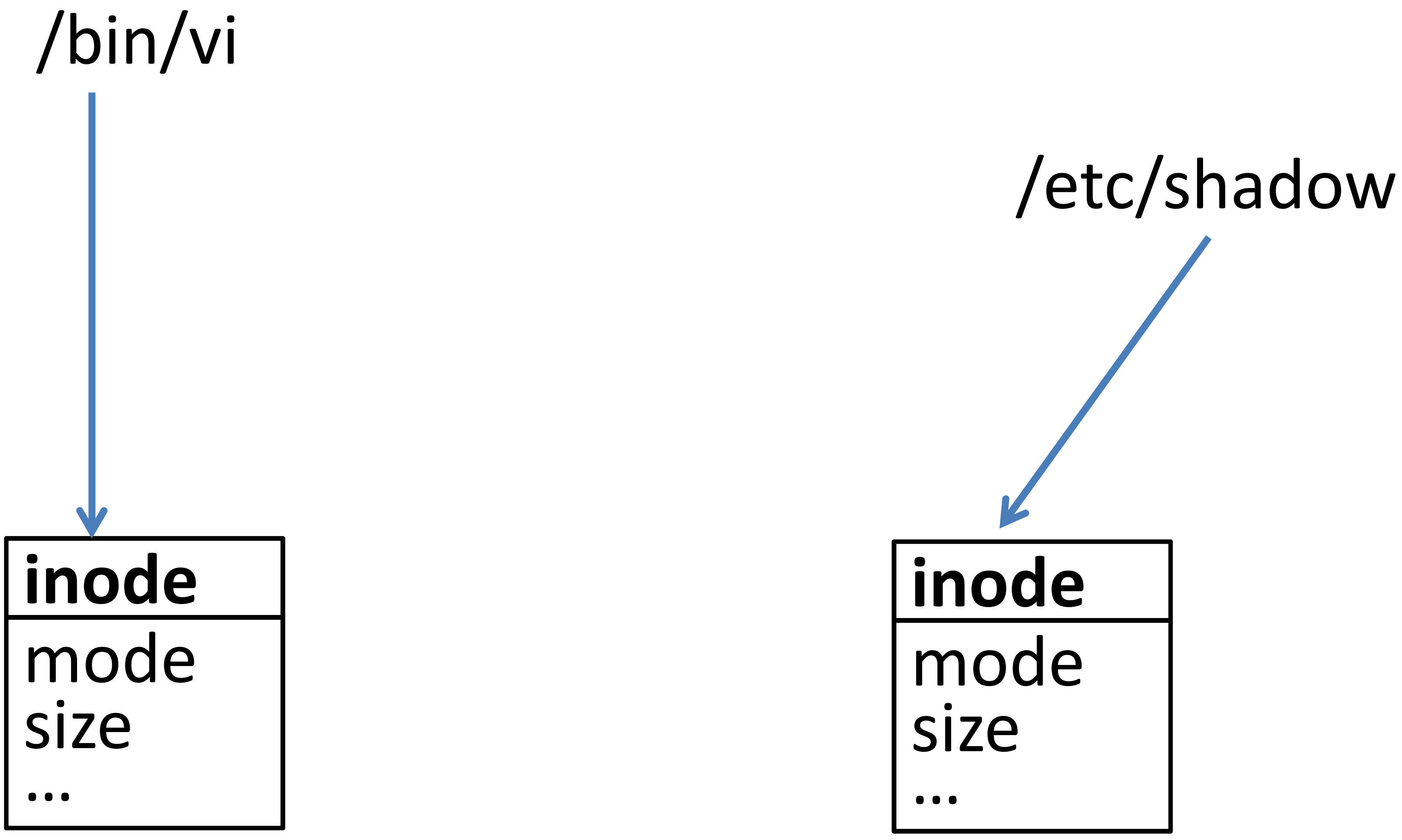
POSIX Filesystem API



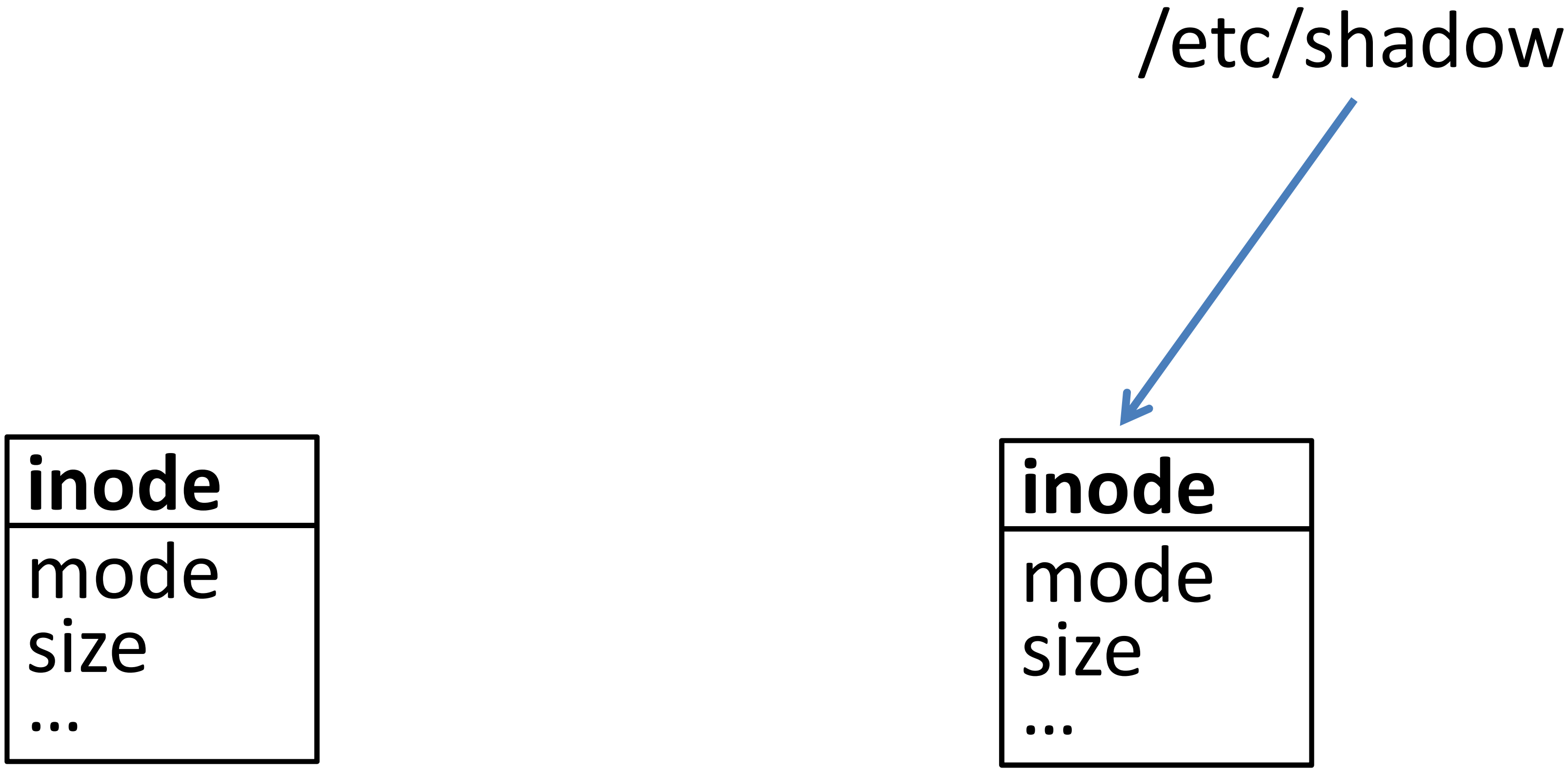
POSIX Filesystem API



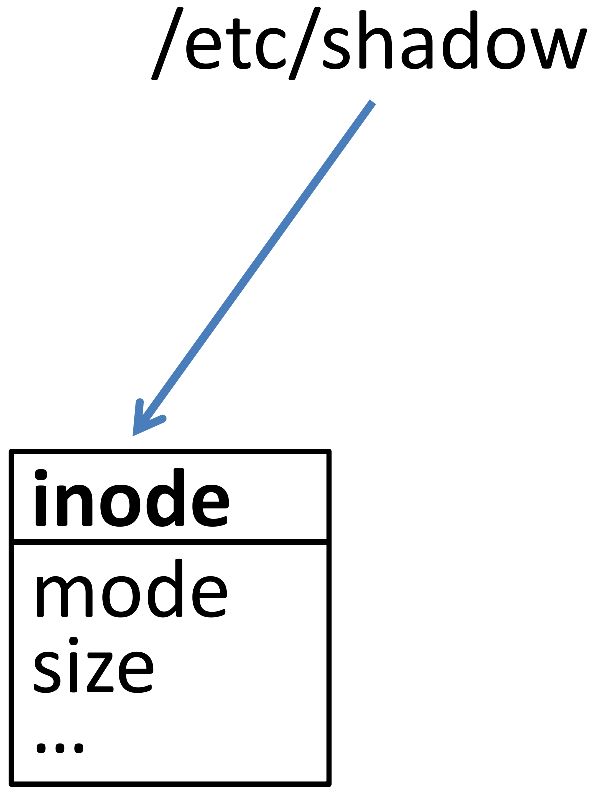
POSIX Filesystem API



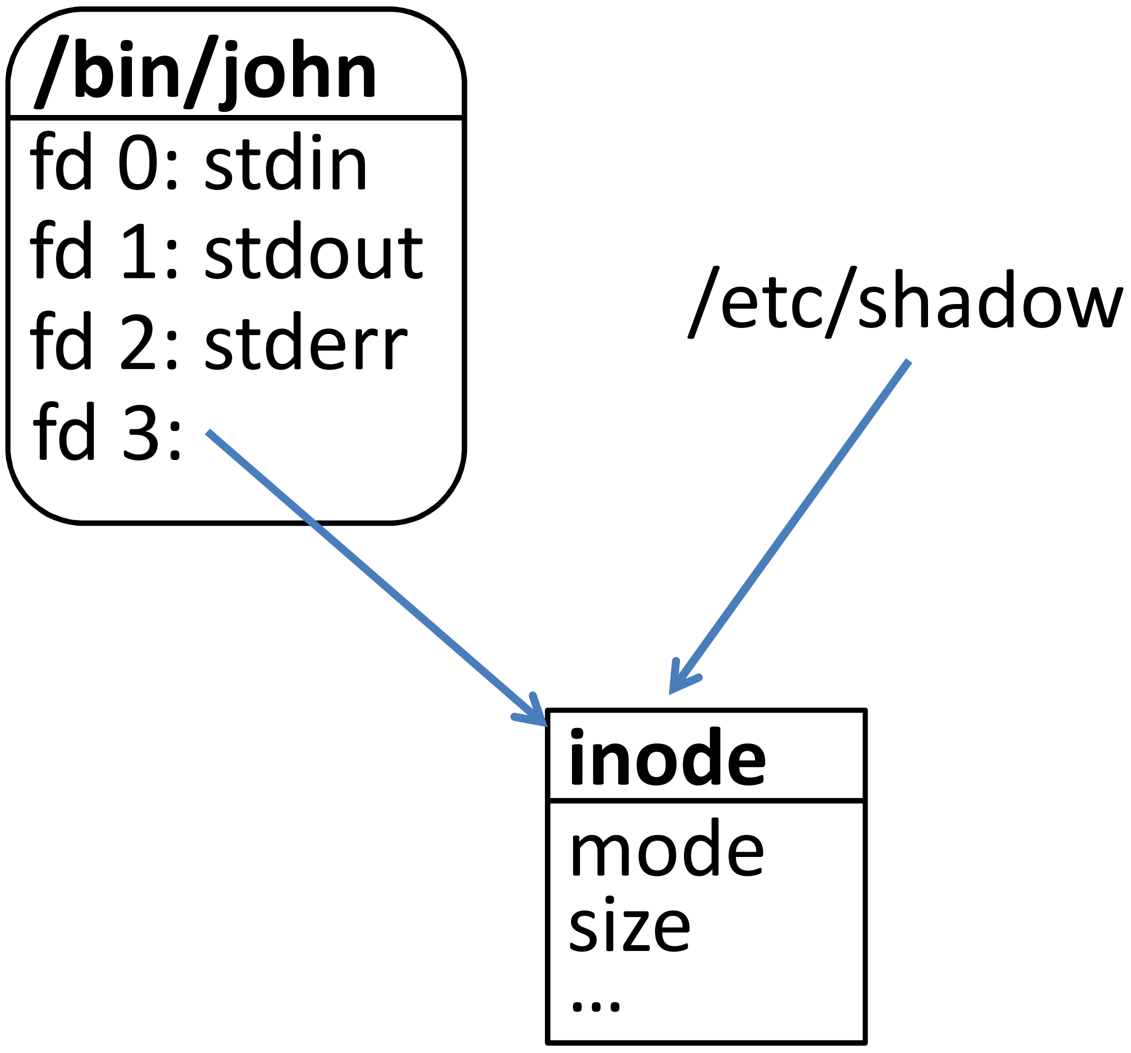
POSIX Filesystem API



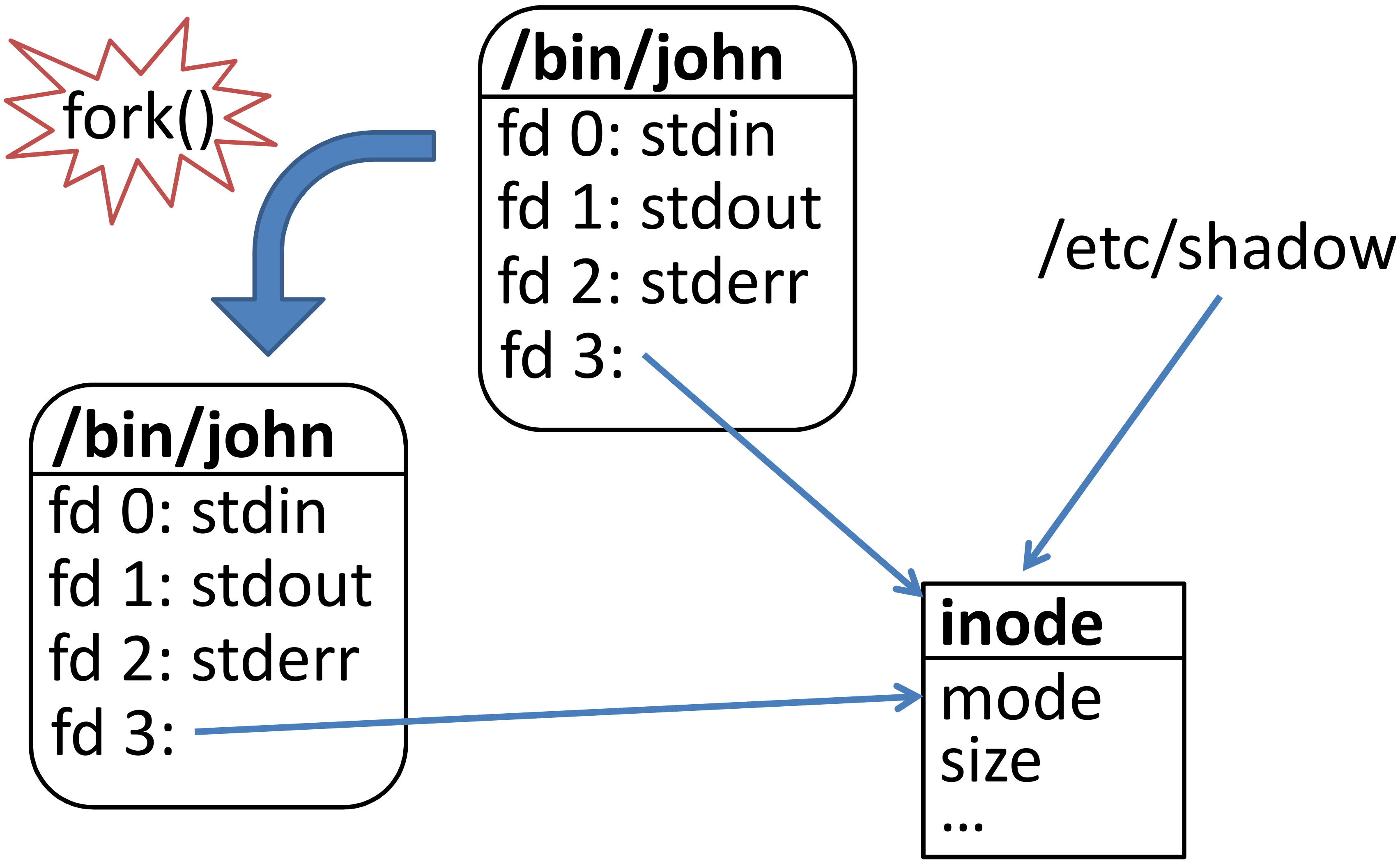
POSIX Filesystem API



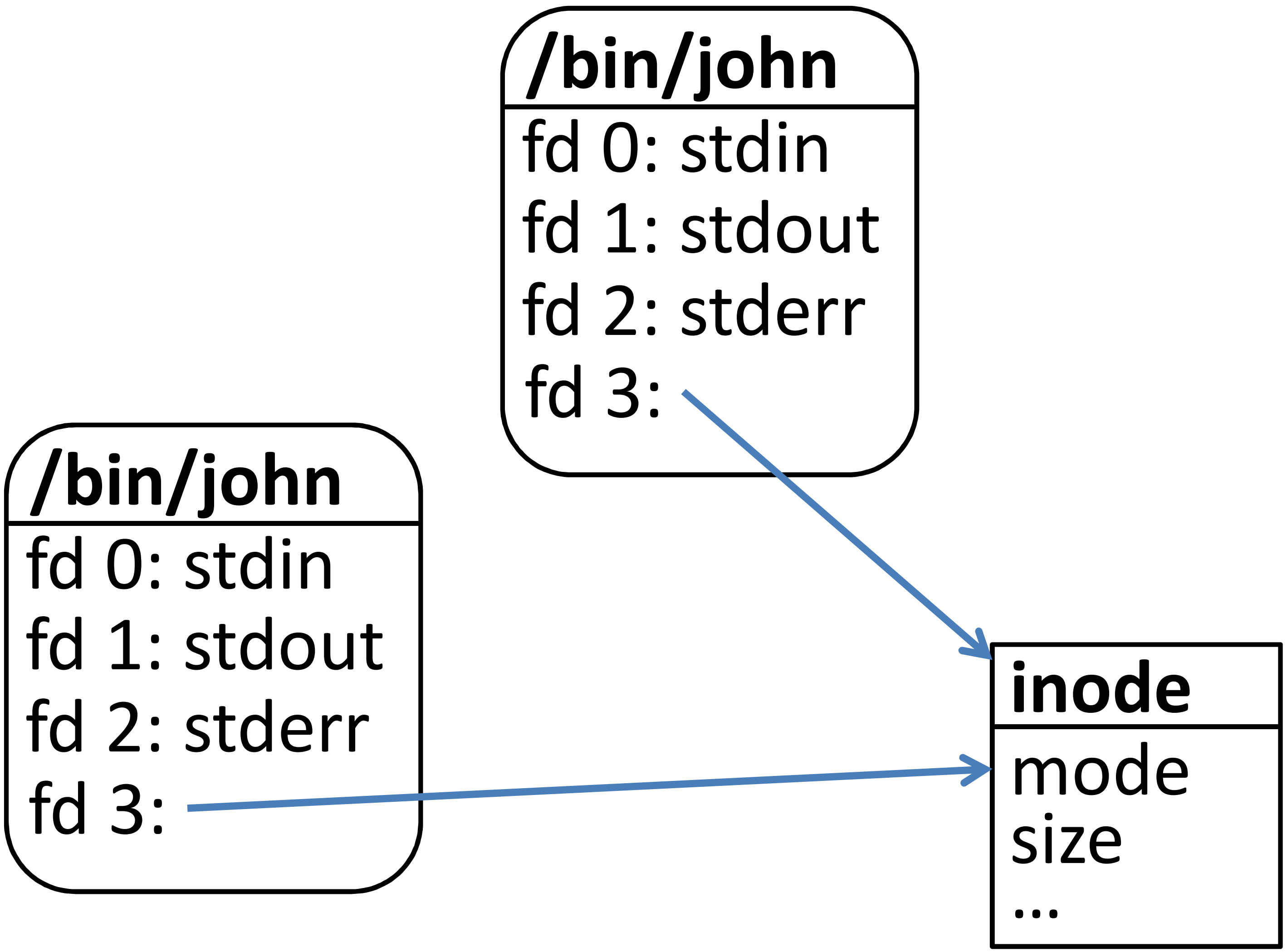
POSIX Filesystem API



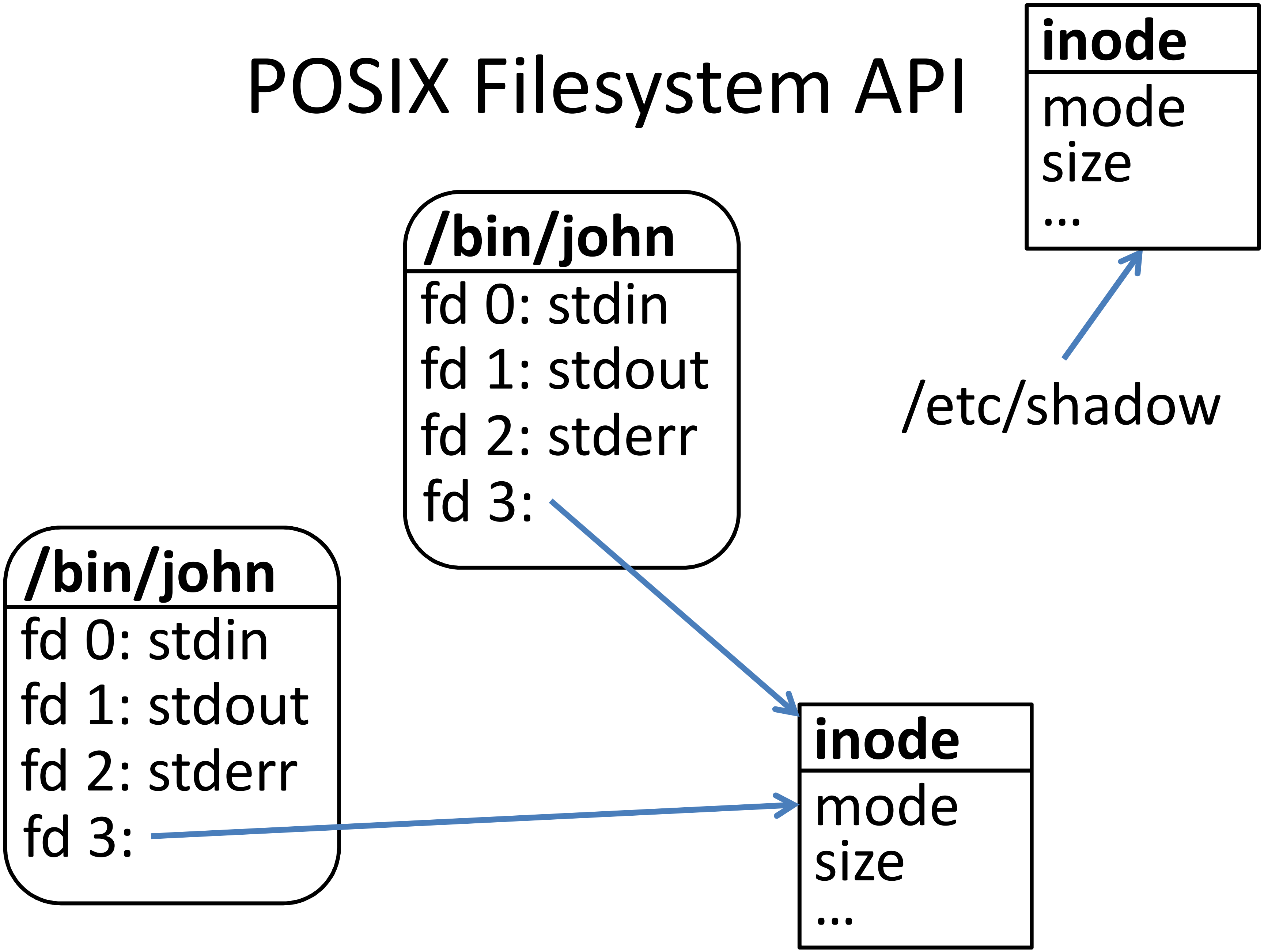
POSIX Filesystem API



POSIX Filesystem API



POSIX Filesystem API



Highlevel API

- Very simple
- Pass control to fuse_main()
- Supply set of callbacks for fs operations
- Callbacks are called as userspace performs operations on the filesystem
- Callbacks are passed a path, return data or error code
- Single or multi-threaded

The Request

```
int (*open)(
    const char *path,
    struct fuse_file_info *fi
);
```

Highlevel Example

Highlevel Notes

- Return plausible link-count for /
- readdir() operates in pages
- Start additional threads from init callback

Lowlevel API

- Deals in inode numbers not paths
  - / is inode 1
  - Everything else is looked up name->inode
  - Directories are just lists of names
- Co-ordinates are Pair<inode,generation>
- Async – you can return from request and call reply later

Why the Lowlevel API?

- More performant in some cases
- More statefull
- More control over memory usage

The Request

```
int (*open)(
   fuse_req_t req,
   fuse_ino_t ino,
   struct fuse_file_info *fi
);
```

Lifecycles

- inodes are reference-counted
  - Incremented by LOOKUP, decremented by FORGET
  - You must field requests on any non-forgotten inode
- OPEN / FLUSH / RELEASE cycle also gives hints

Lowlevel Example

Debugging

- Same as any other user-space programme
- Can use gdb / ddd / $IDE / etc
- Can use valgrind (needs setup)
- Force unmount after crash with

```
fusermount –u /mnt/point
```

Testing

- fuse/test/test.c
- Linux Test Tools (ltp.sf.net/tooltable.php)
- POSIX compliance test suite at http://www.itl.nist.gov/div897/ctg/posix_form.htm
- Ntfs-3g has a couple

Performance

- FUSE filesystems can be very fast
- Be careful of memory usage – obey release and forget
- Inherently lots of copies – try to minimise these

Conclusions

- Write filesystems!
- It’s easy in userspace!
- Highlevel interface and scripting bindings make it trivial for simple tasks
- Lowlevel interface gives surprising performance and control

Thank You

Any Questions?