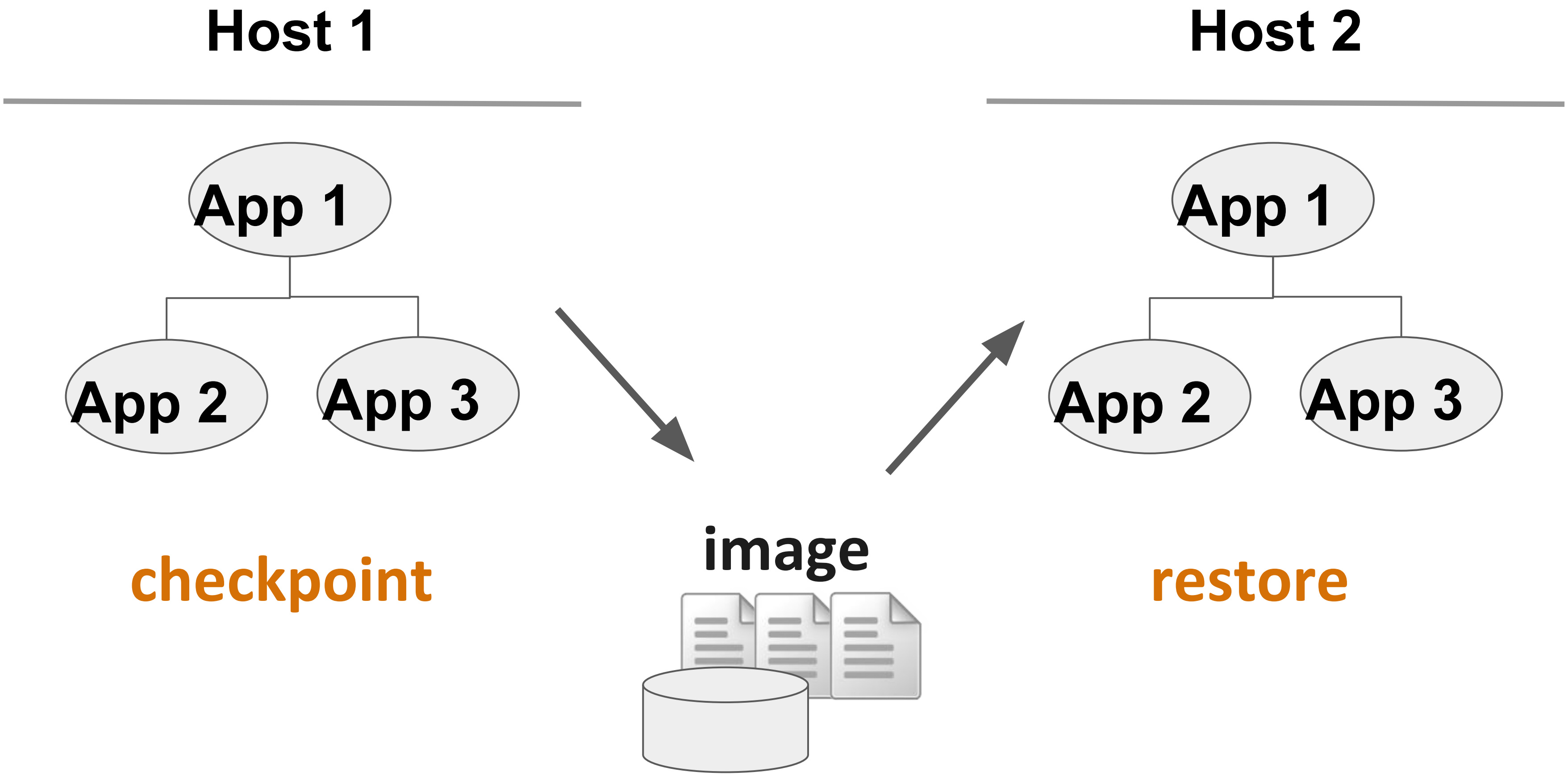
Software Engineering Conference Russia
October 2017, St. Petersburg

CRIU: Checkpoint and Restore & file locks

- Pavel Begunkov (Silence)
  SpbAU RAS

- Failure tolerance
  - Network
  - Hardware
  - Programmer
- Scalability
- Flexibility
- etc.

Checkpoint & Restore



Host 1

App 1

App 2

App 3

checkpoint

image

Host 2

App 1

App 2

App 3

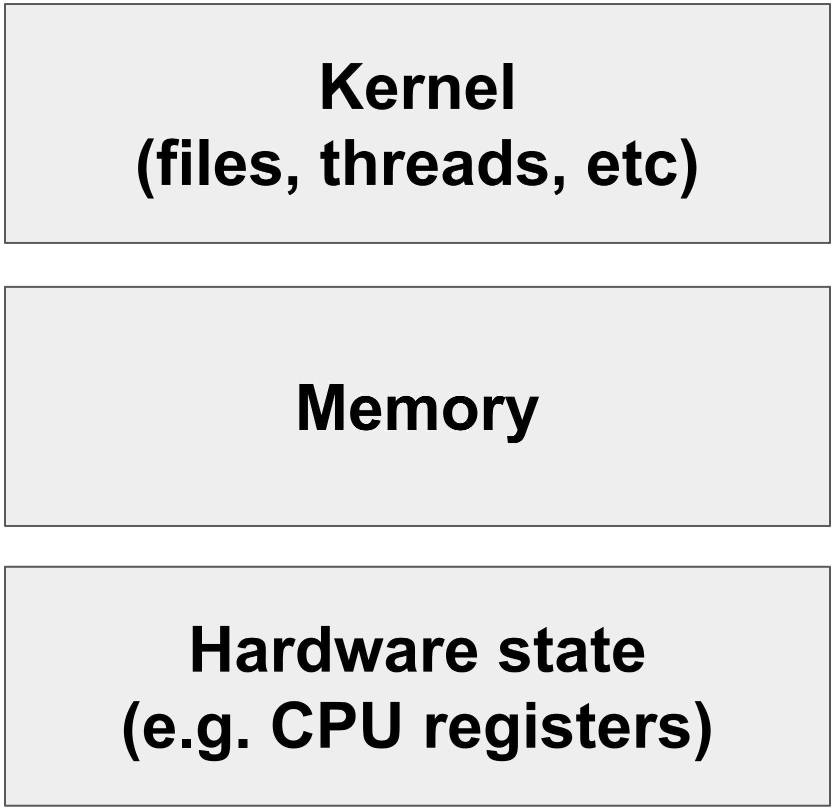
restore

What for?

- **Live-migration**
- **Load-balancing**
- **Failure tolerance & recovery**
- **etc.**

criu.org/Usage_scenarios

Program state



Kernel
(files, threads, etc)

Memory

Hardware state
(e.g. CPU registers)

How to do c/r?

- Code instrumentation
- LD_PRELOAD
- Kernel modification
- Kernel module
- Userspace solution
- etc.

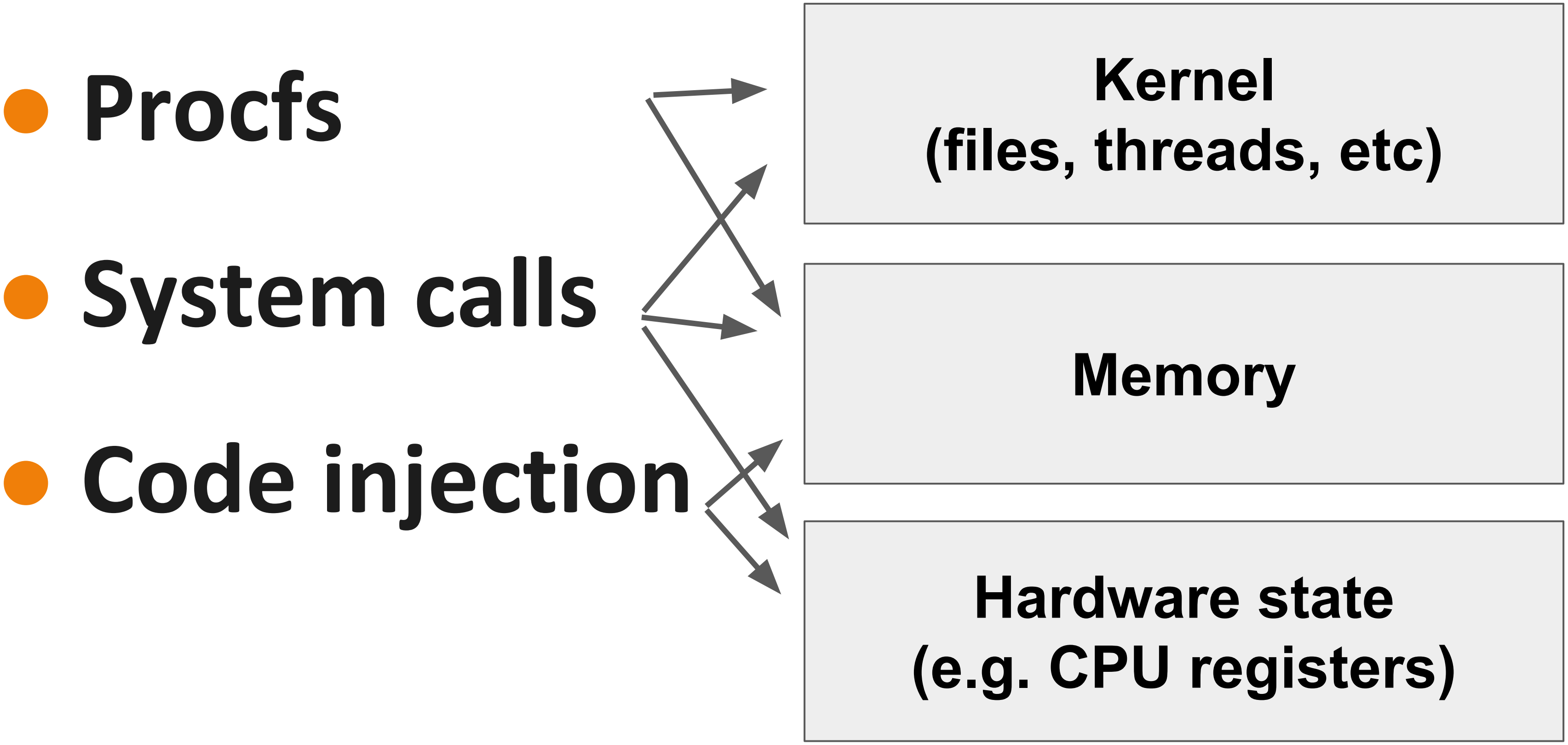
CRIU: Checkpoint and Restore In Userspace

- checkpoint/restart feature work.

“A note on this: this is a project by various mad Russians to perform c/r mainly from userspace, with various oddball helper code added into the kernel where the need is demonstrated.”

Torvalds committed on Jan 13, 2012
(commit 0994695)

Userspace c/r



- Procfs
- System calls
- Code injection

Kernel
(files, threads, etc)

Memory

Hardware state
(e.g. CPU registers)

File locks

Types

1. BSD locks
2. POSIX locks
3. OFD locks
4. *File lease*

Key features

1. Read-Write synchronisation
2. Advisory & Mandatory*

File locks

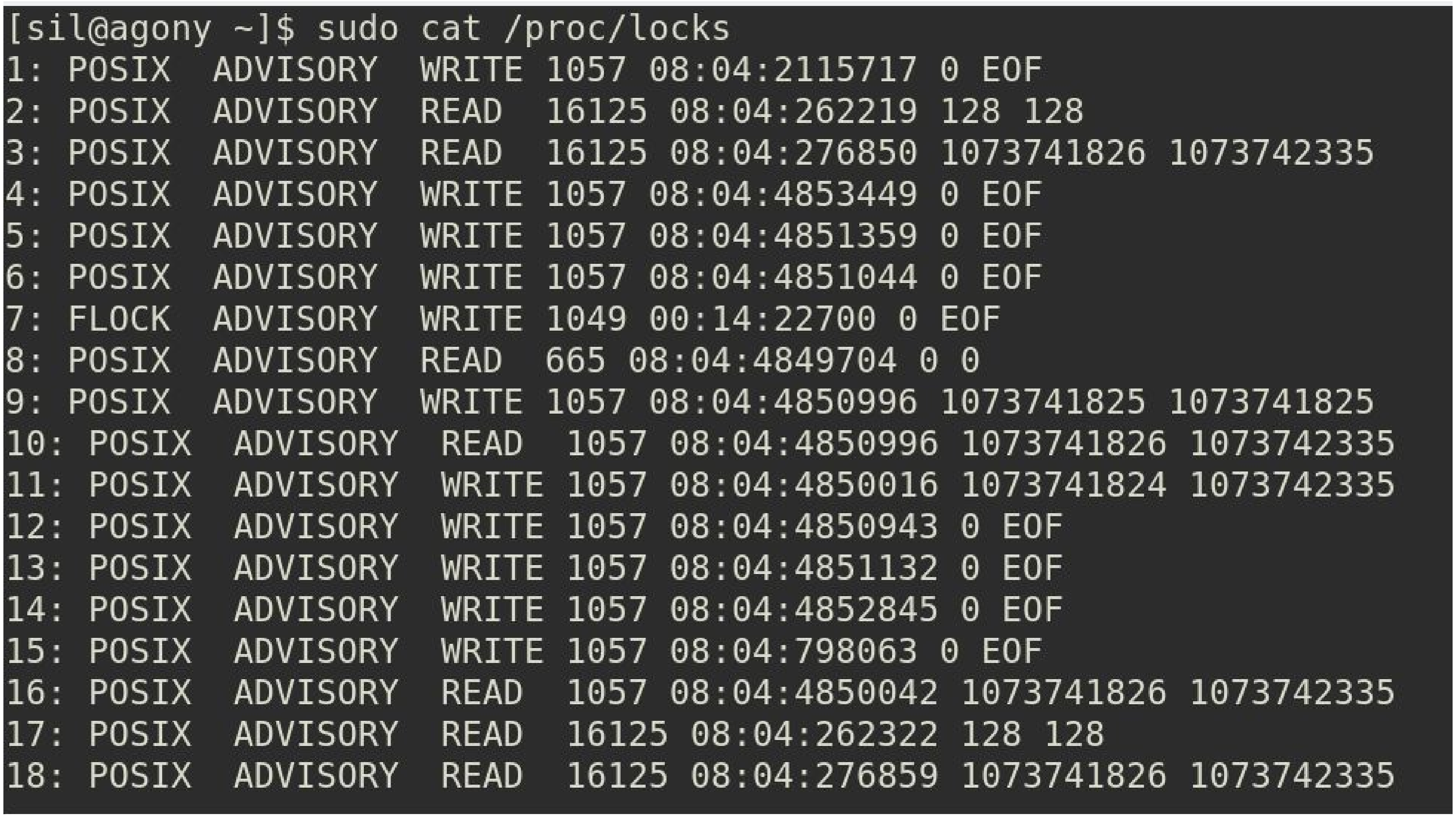
Checkpoint

1. Read locks from ***procfs***
2. Match each ***lock*** with physical file*
3. Match each ***lock*** with **open *file description****
4. Fixup the data
5. Save to image

Restore

1. Read image
2. Open file
3. Set lock (fcntl, flock, etc)
4. [break lease]

Procfs & locks



```
[sil@agony ~]$ sudo cat /proc/locks
1: POSIX  ADVISORY  WRITE 1057 08:04:2115717 0 EOF
2: POSIX  ADVISORY  READ  16125 08:04:262219 128 128
3: POSIX  ADVISORY  READ  16125 08:04:276850 1073741826 1073742335
4: POSIX  ADVISORY  WRITE 1057 08:04:4853449 0 EOF
5: POSIX  ADVISORY  WRITE 1057 08:04:4851359 0 EOF
6: POSIX  ADVISORY  WRITE 1057 08:04:4851044 0 EOF
7: FLOCK  ADVISORY  WRITE 1049 00:14:22700 0 EOF
8: POSIX  ADVISORY  READ  665 08:04:4849704 0 0
9: POSIX  ADVISORY  WRITE 1057 08:04:4850996 1073741825 1073741825
10: POSIX  ADVISORY  READ  1057 08:04:4850996 1073741826 1073742335
11: POSIX  ADVISORY  WRITE 1057 08:04:4850016 1073741824 1073742335
12: POSIX  ADVISORY  WRITE 1057 08:04:4850943 0 EOF
13: POSIX  ADVISORY  WRITE 1057 08:04:4851132 0 EOF
14: POSIX  ADVISORY  WRITE 1057 08:04:4852845 0 EOF
15: POSIX  ADVISORY  WRITE 1057 08:04:798063 0 EOF
16: POSIX  ADVISORY  READ  1057 08:04:4850042 1073741826 1073742335
17: POSIX  ADVISORY  READ  16125 08:04:262322 128 128
18: POSIX  ADVISORY  READ  16125 08:04:276859 1073741826 1073742335
```

File locks

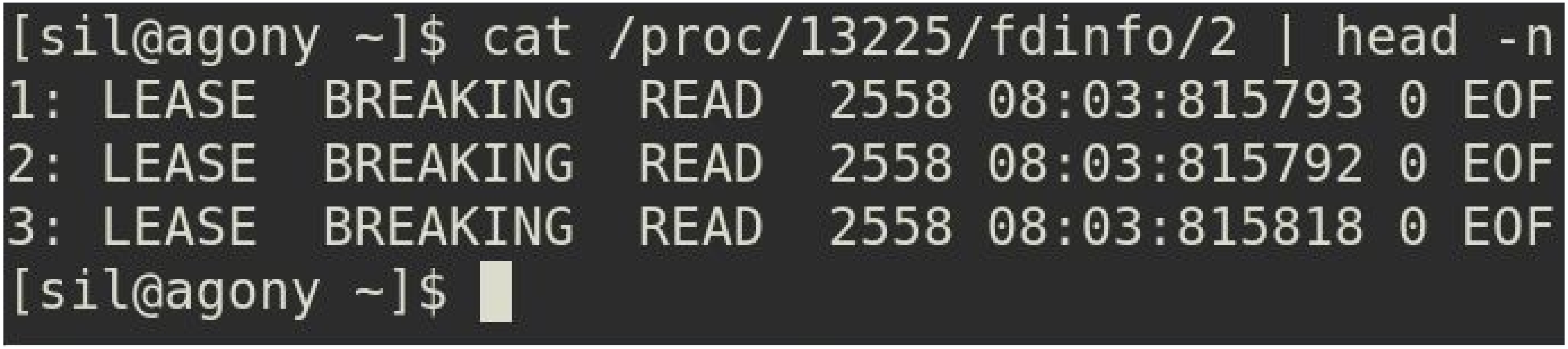
Checkpoint

1. Read locks from ***procfs***
2. Match each ***lock*** with physical file*
3. Match each ***lock*** with **open *file description*** *
4. Fixup the data
5. Save to image

Restore

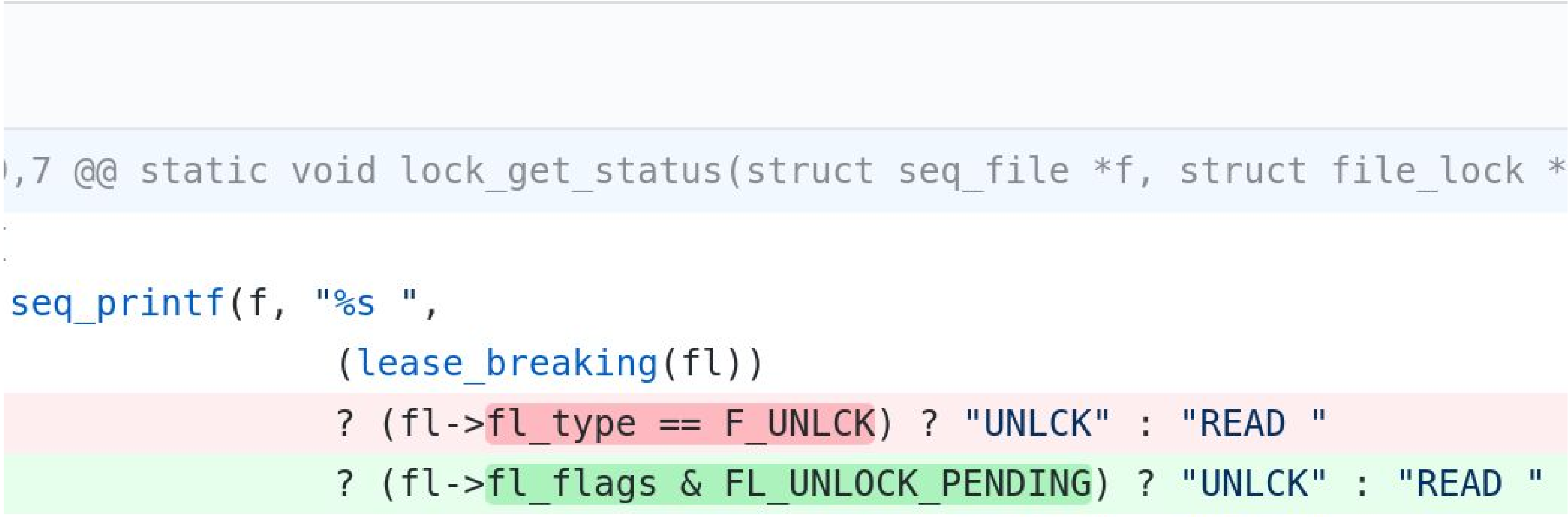
1. Read image
2. Open file
3. Set lock (fcntl, flock, etc)
4. [break lease]

Breaking leases



```
[sil@agony ~]$ cat /proc/13225/fdinfo/2 | head -n
1: LEASE  BREAKING  READ  2558 08:03:815793 0 EOF
2: LEASE  BREAKING  READ  2558 08:03:815792 0 EOF
3: LEASE  BREAKING  READ  2558 08:03:815818 0 EOF
[sil@agony ~]$ 
```

Bugs



```
,7 @@ static void lock_get_status(struct seq_file *f, struct file_lock *

seq_printf(f, "%s ",
               (lease_breaking(fl))
               ? (fl->fl_type == F_UNLCK) ? "UNLCK" : "READ "
               ? (fl->fl_flags & FL_UNLOCK_PENDING) ? "UNLCK" : "READ "
               : (fl->fl_type == F_WRLCK) ? "WRITE" : "READ ");
```

Contact information

Pavel Begunkov (Silence)

- asml.silence [at] gmail.com
- www.linkedin.com/in/isilence
- www.github.com/isilence