Webcamd

a modern userspace Linux kernel driver framework for FreeBSD

by

Hans Petter Selasky
hselasky @ freebsd . org

Master in Technology, in Information and Communication Technology, at Agder University College in Norway, Factulty of Engineering and Science

EuroBSDcon October 2011

Table of Contents

Webcamd - About

- A good compromise between functionality and technical excellence
- Mini or micro Linux kernel for userspace
- Webcamd might be renamed in the future
- The version number will follow the Linux kernel version used
- Complies to the GPLv2
- Many ideas came from porting my ISDN4BSD from FreeBSD 8.x to NetBSD 3.x
- Works on multiple platforms (i386, amd64, ...)

Webcamd - The official process

1. Get the source code (webcamd + Linux kernel code)
   1. /usr/ports/multimedia/webcamd
   2. svn co ...
   3. git clone git://xxx.git
   4. ...
2. Edit config file
   1. ee config
   2. vi config
3. Create Makefiles using Webcamd's linux_make
   1. make configure
4. Build code
   1. make -jX all
5. Install resulting binary
   1. make install

Webcamd - tools/linux_make

- Lightweight GNU make implementation
- Does not execute any shell commands
- Outputs BSD Makefiles
- Supports modules
- Supports multiple input directories
- Currently only the montolith mode is used
- Example syntax:
  - linux_make -c config -i media_tree/drivers/input -i media_tree/drivers/media -o build/

Webcamd - kernel/linux_defs.[ch]

- C-macros is your friend when porting software
- Macros for non-existing functions makes the code compile
- Defines Linux types like u8, u16, u32 and so on

Webcamd - kernel/linux_file.[ch]

- Contains the glue between CUSE4BSD and the Linux character devices
- Linux uses old style Major/Minor numbering
- No mknod
- Mapping of Major/Minors into /dev/xxx name

Webcamd - kernel/linux_firmware.[ch]

- Contains support for loading firmware files for drivers through lib-C
- Default firmware directory: /boot/modules/

Webcamd - kernel/linux_func.[ch]

- Various functions needed for compilation
- Math
- Endianness
- Superfluous definitions like __used, __user ...

Webcamd - kernel/linux_i2c.[ch]

- Stripped down version of Linux I2C kernel code
- Full version has too many dependancies which currently cannot be configured away
- Used by many drivers
- Exposes chip specific buses directly via USB

Webcamd - kernel/linux_mod_param.[ch]

- Support for Linux module parameters
- Nice to have when debugging and tuning drivers

Webcamd - kernel/linux_start_section.[ch]

- Very important part to glue all initializers together
- Exported through a separate data-section which is scanned upon startup
- static const struct xxx
  __attribute__((__section__("yyy")));

Webcamd - kernel/linux_struct.[ch]

- Definition of missing and stripped down Linux kernel structures
  - struct file ++
  - struct device ++

Webcamd - kernel/linux_task.[ch]

- Simple re-implementation of Linux “work” and “tasklet”

Webcamd - kernel/linux_thread.[ch] 1/2

- Threads (kthread_xxx)
  - pthreads
- Mutexes (mutex_xxx, sema_xxx, rw_xxx)
  - not destroyed on Linux
  - single lock
  - less deadlock issues
  - no need for more than one lock

Webcamd - kernel/linux_thread.[ch] 2/2

- Atomicity (atomic_lock, atomic_unlock)
  - Giant lock principle
- Synchronisation (wait_until)
  - a single condition variable behind the scene

Webcamd - kernel/linux_timer.[ch]

- Simple timer implementation
- Sleeps when no character devices are opened
- Linux jiffies = BSD ticks

Webcamd - kernel/linux_usb.[ch]

- Complete reimplemenation of the Linux Host USB stack API behind LibUSB v2.0 which is currently specific to FreeBSD 8+
- A few limitations
- Works in most cases

Webcamd - dummy

- Empty header files which should not be included from the Linux kernel sources
- Machine specific code

Webcamd - config

- Standard Linux ".config" file containing definitions of all CONFIG_XXX keywords which should be enabled.
- Valid keyword definition values: y | n | m

CUSE4BSD

- Character devices in USErspace for(4) BSD
- A library and kernel module
- Supports regular device permissions (Read, Write and eXecute) and ownership
- Limited support for process signal delivery
- Client of devfs
- No tricks
- /usr/ports/multimedia/cuse4bsd-kmod

CUSE4BSD - Startup

- API: cuse_init, cuse_uninit

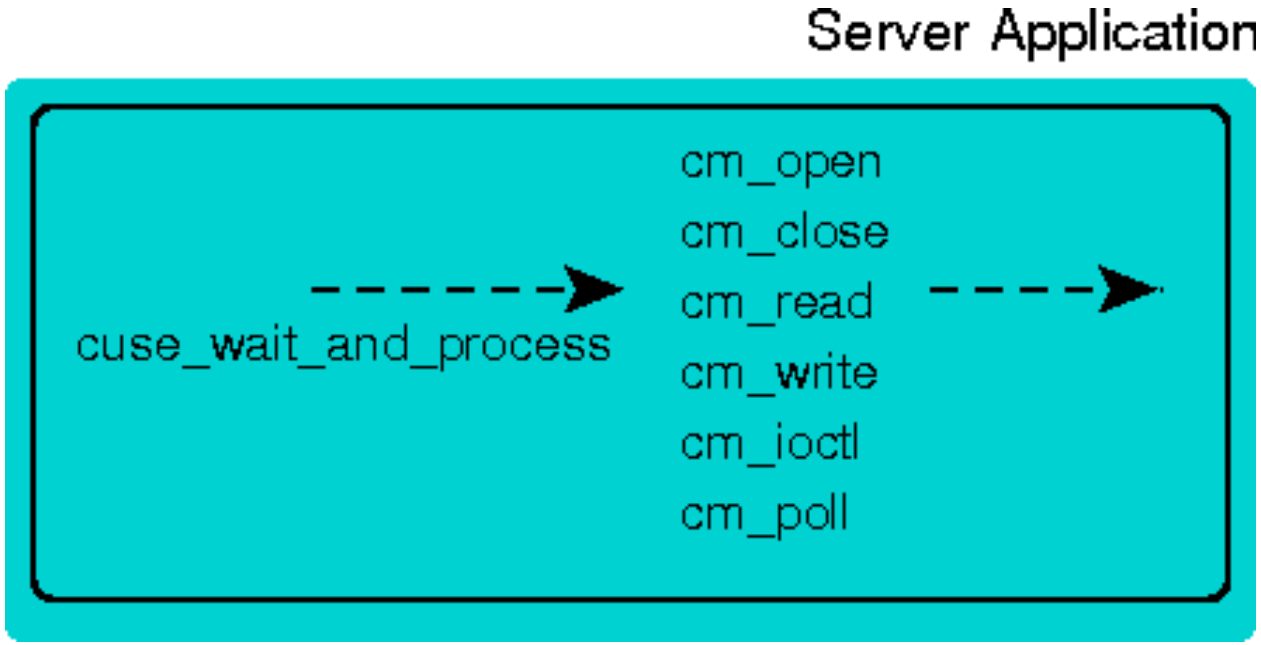
CUSE4BSD - Unit management

- API: cuse_alloc_unit_number, cuse_free_unit_number

CUSE4BSD - Create device

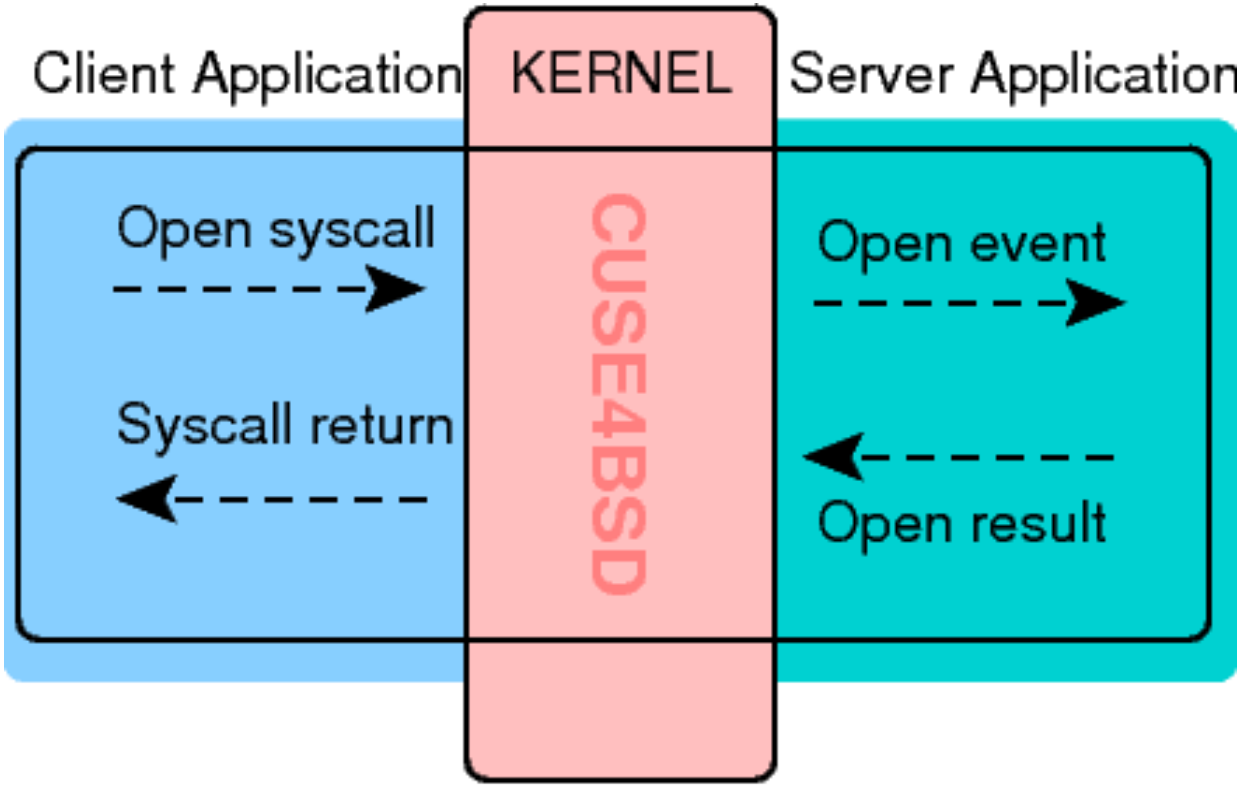
- API: cuse_dev_create, cuse_dev_destroy
- Kernel cdevpriv
- Resonable performance
- All devices reside under /dev/xxx
- Can only use alpanumerical characters and a few others to avoid bad device names
- Supports directories

CUSE4BSD - Server process



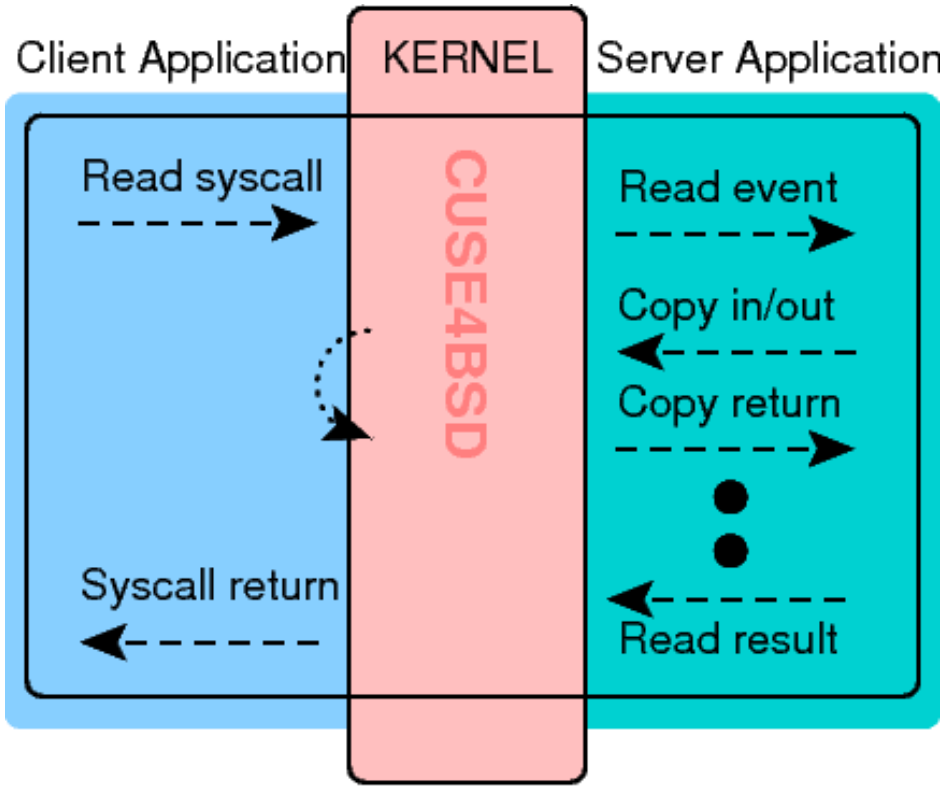
- API: cuse_wait_and_process, cuse_dev_get_current
- Need one thread per concurrent blocking/sleeping event

CUSE4BSD - Open



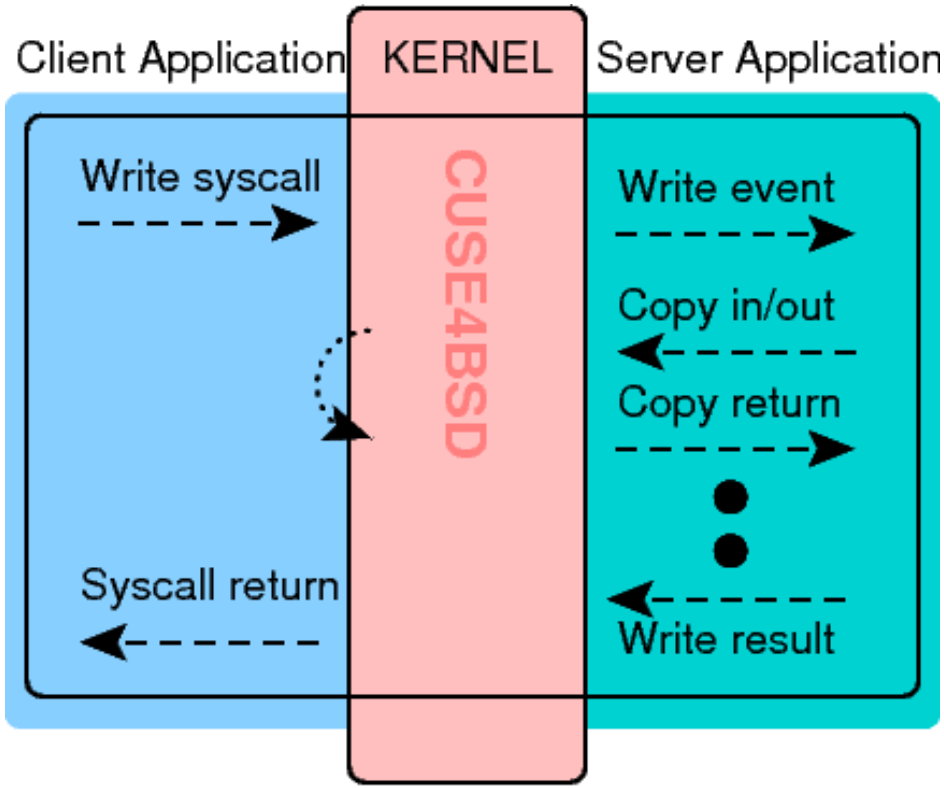
- API: cuse_dev_set_per_file_handle, cuse_dev_get_per_file_handle, cuse_dev_get_privX, cuse_dev_set_privX

CUSE4BSD - Read



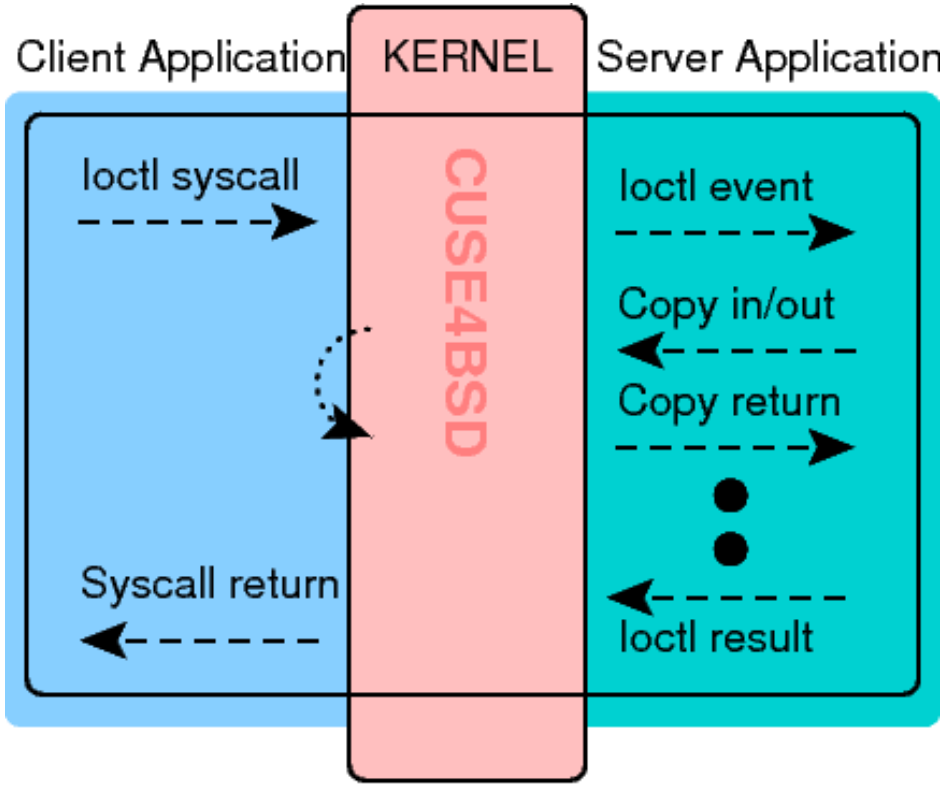
- API: cuse_copy_out, cuse_copy_in, cuse_got_peer_signal
- Data is copied twice

CUSE4BSD - Write



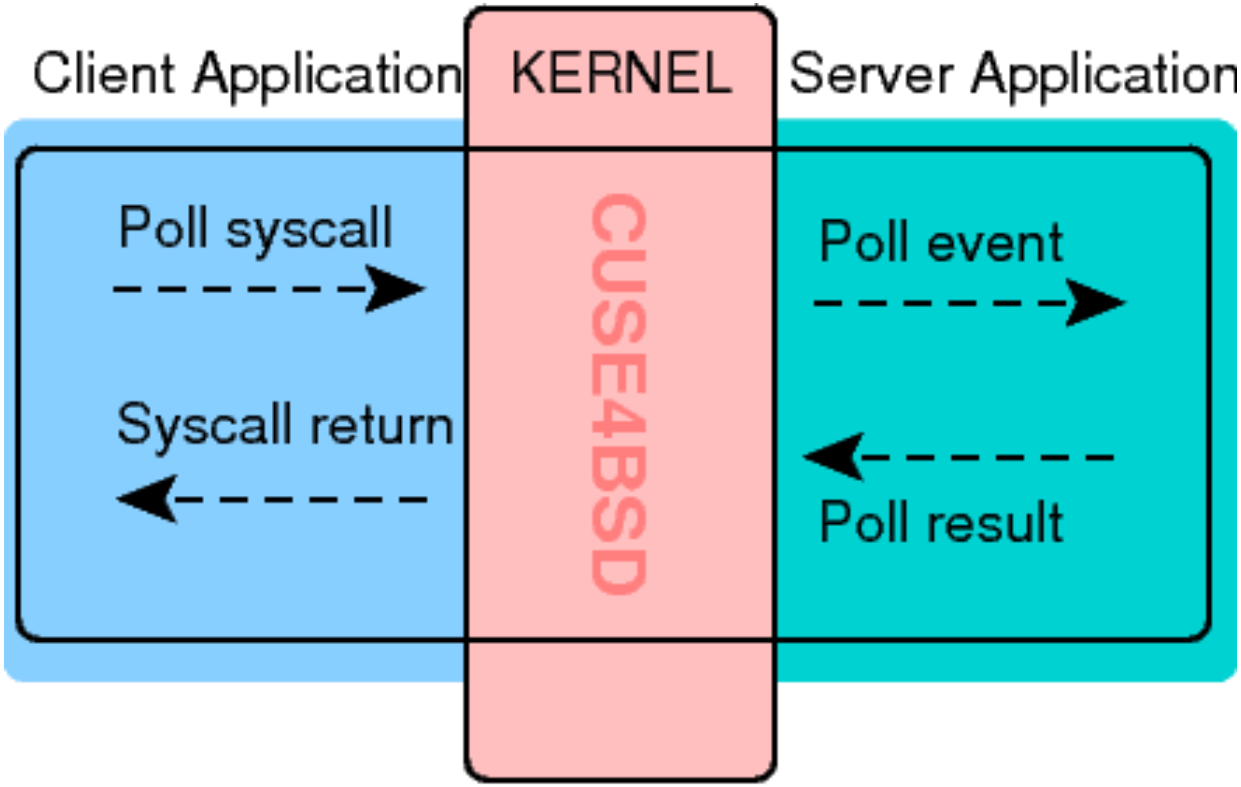
- API: cuse_copy_out, cuse_copy_in, cuse_got_peer_signal
- Data is copied twice

CUSE4BSD - Ioctl



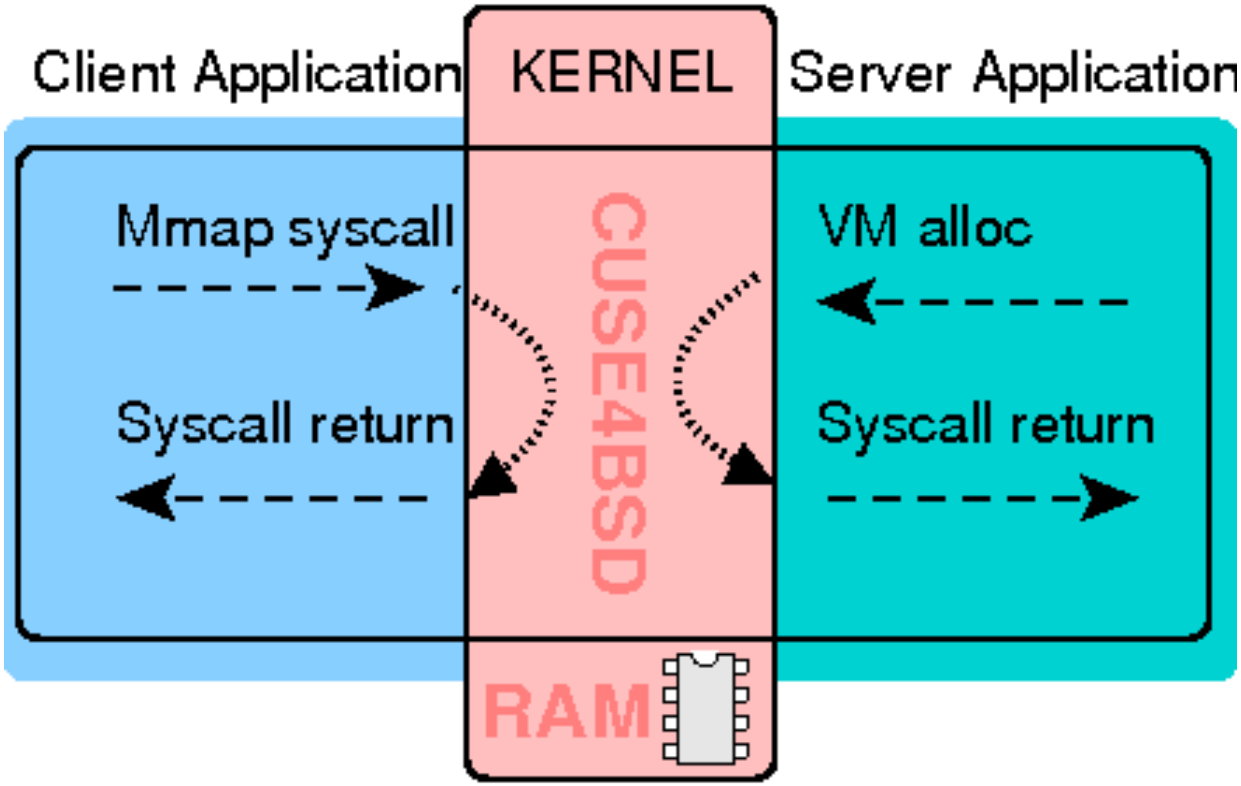
- API: cuse_copy_out, cuse_copy_in, cuse_got_peer_signal
- Linux does not respect R/W flags in IOCTLs.
- Special cases passing an integer.

CUSE4BSD - Poll



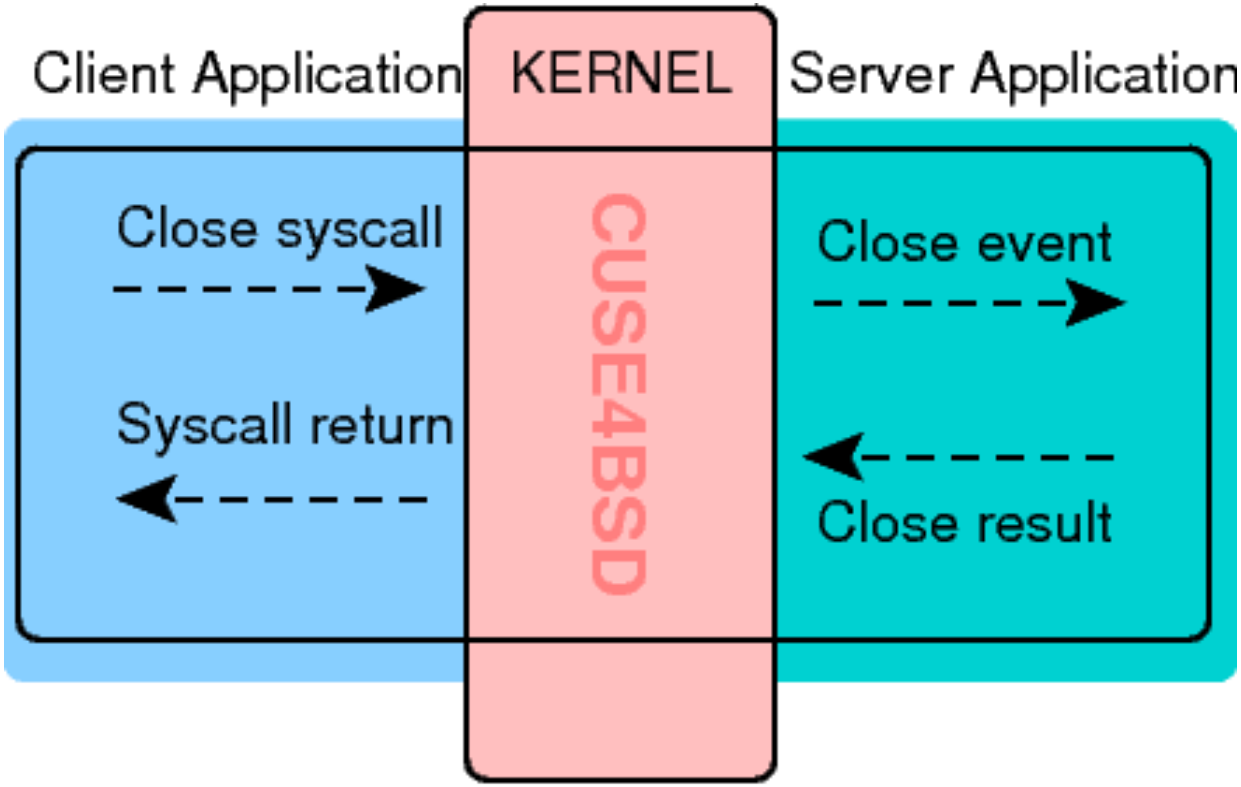
- API: cuse_poll_wakeup, cuse_got_peer_signal

CUSE4BSD - Mmap



- API: cuse_vmalloc, cuse_vmfree, cuse_vmoffset
- Memory mapped memory is never freed

CUSE4BSD - Close



- Terminates a file handle

Advantages

- Can debug [Linux] kernel code from userspace. Something similar has been done in NetBSD.
- Reduces the need for writing new drivers.

Disadvantages

- Overhead with regard to task switching
- Overhead with regard to additional data copy unless mmap is used

Conclusion

- Opens up new possibilities
- Less “license fighting”
- More fun on the FreeBSD platform

Questions

Any questions ?