No Grammar? No Problem!

Towards Fuzzing the Linux Kernel without System-Call Descriptions

Alexander Bulekov, Bandan Das, Stefan Hajnoczi, Manuel Egele
Boston University Seclab, Red Hat

oss-fuzz oss-fuzz New issue All issues Sign in

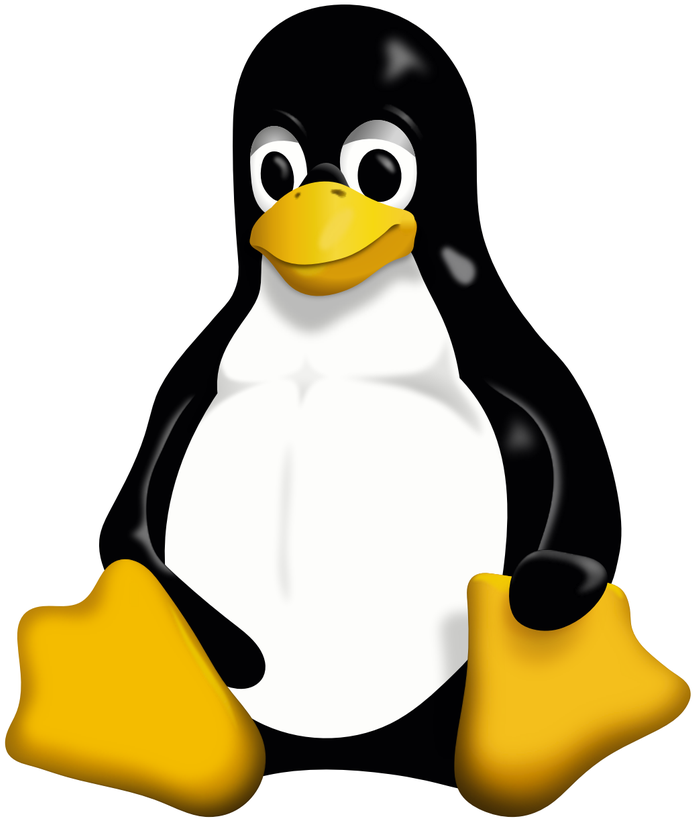
1 - 100 of 54579 Next › List Grid Chart

ID	Type	Component	Status	Proj	Reported	Owner	Summary + Labels
82	Bug	—	Verified	sqlite3	—	—	ash in sqlite3ExprV ClusterFuzz Reproducible
84	Bug-Security	—	Verified	libchewing	—	—	libchewing: Heap-buffer-overflow in ChoiceTheSame ClusterFuzz Reproducible
85	Bug-Security	—	Verified	libchewing	—	—	libchewing: Heap-buffer-overflow in _Inner_InternalSpecialSymbol ClusterFuzz Reproducible
86	Bug-Security	—	Verified	libchewing	—	—	libchewing: Heap-buffer-overflow in CheckUserChoose ClusterFuzz Reproducible
87	Bug-Security	—	Verified	libchewing	—	—	libchewing: Heap-buffer-overflow in ChewingIsChiAt ClusterFuzz Reproducible
88	Bug-Security	—	Verified	libchewing	—	—	libchewing: Heap-buffer-overflow in _Inner_InternalSpecialSymbol ClusterFuzz Reproducible
89	Bug	—	Verified	libchewing	—	—	libchewing: Floating-point-exception in HaninSymbolInput ClusterFuzz Reproducible
90	Bug-Security	—	Verified	pcre2	—	—	pcre2: Crash in match ClusterFuzz Reproducible
91	Bug	—	Verified	libchewing	—	—	libchewing: Floating-point-exception in HaninSymbolInput ClusterFuzz Reproducible
92	Bug	—	Verified	libchewing	—	—	libchewing: Floating-point-exception in SetChoiceInfo ClusterFuzz Reproducible
93	Bug	—	Verified	libchewing	—	—	libchewing: Floating-point-exception in OpenSymbolChoice ClusterFuzz Reproducible
94	Bug-Security	—	Verified	libchewing	—	—	libchewing: Crash in GetUint24 ClusterFuzz Reproducible
96	Bug-Security	—	Verified	libxml2	2016-12-27	—	libxml2: Heap-buffer-overflow in xmlDictComputeFastKey ClusterFuzz

oss-fuzz oss-fuzz New issue All issues Sign in

1 - 100 of 54579 Next › List Grid Chart

ID	Type	Component	Status	Proj	Reported	Owner	Summary + Labels
82	Bug	—	Verified	sqlite3	—	—	ash in sqlite3ExprV ClusterFuzz Reproducible
84	Bug-Security	—	Verified	libchewing	—	—	libchewing: Heap-buffer-overflow in ChoiceTheSame ClusterFuzz Reproducible
85	Bug-Security	—	Verified	libchewing	—	—	libchewing: Heap-buffer-overflow in _Inner_InternalSpecialSymbol ClusterFuzz Reproducible
86	Bug-Security	—	Verified	libchewing	—	—	libchewing: Heap-buffer-overflow in CheckUserChoose ClusterFuzz Reproducible
87	Bug-Security	—	Verified	libchewing	—	—	libchewing: Heap-buffer-overflow in ChewingIsChiAt ClusterFuzz Reproducible
88	Bug-Security	—	Verified	libchewing	—	—	libchewing: Heap-buffer-overflow in _Inner_InternalSpecialSymbol ClusterFuzz Reproducible
89	Bug	—	Verified	libchewing	—	—	libchewing: Floating-point-exception in HaninSymbolInput ClusterFuzz Reproducible
90	Bug-Security	—	Verified	pcre2	—	—	pcre2: Crash in match ClusterFuzz Reproducible
91	Bug	—	Verified	libchewing	—	—	libchewing: Floating-point-exception in HaninSymbolInput ClusterFuzz Reproducible
92	Bug	—	Verified	libchewing	—	—	libchewing: Floating-point-exception in SetChoiceInfo ClusterFuzz Reproducible
93	Bug	—	Verified	libchewing	—	—	libchewing: Floating-point-exception in OpenSymbolChoice ClusterFuzz Reproducible
94	Bug-Security	—	Verified	libchewing	—	—	libchewing: Crash in GetUint24 ClusterFuzz Reproducible
96	Bug-Security	—	Verified	libxml2	2016-12-27	—	libxml2: Heap-buffer-overflow in xmlDictComputeFastKey ClusterFuzz



syzbot

Linux

sign-in | mailing list | source | docs

Open [1169] | Fixed [4335] | Invalid [9732] | Kernel Health | Bug Lifetimes | Fuzzing | Crashes

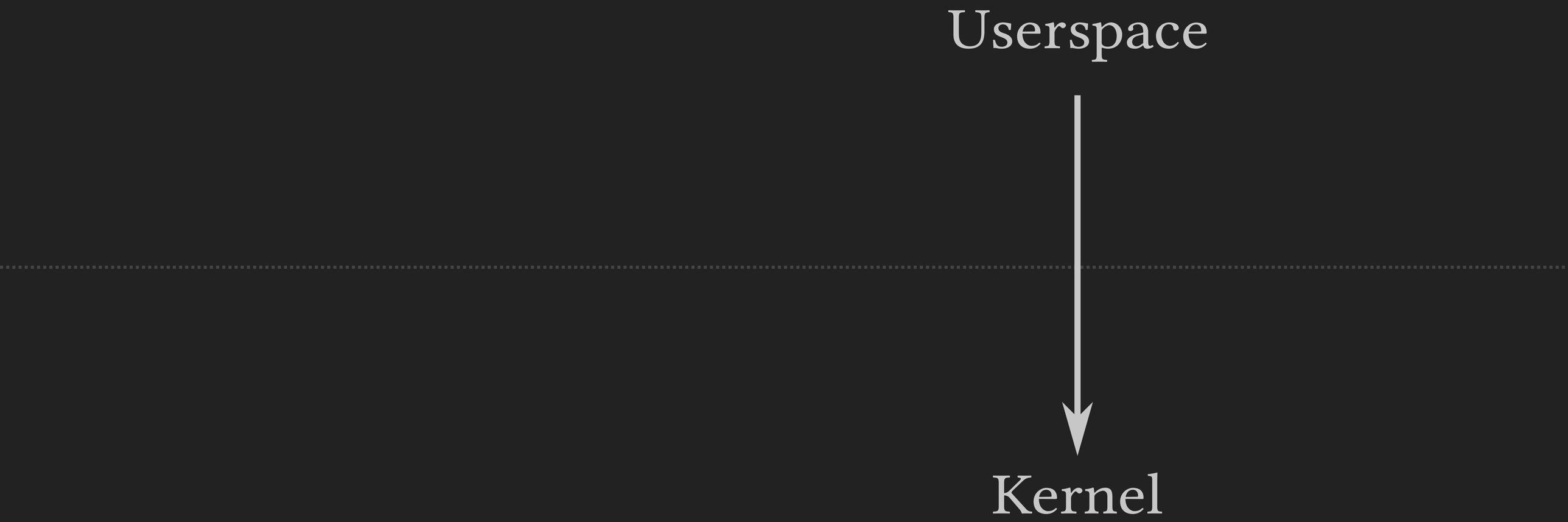
Instances [tested repos]:

Name	Last active	Uptime	Corpus	Coverage	Crashes	Execs	Kernel build				syzkaller build		
							Commit	Config	Freshness	Status	Commit	Freshness	Status
ci-qemu-upstream	now	7h20m	35062	703227	617	650893	ceaa837f96ad	.config	19h05m		957959cb	6h29m	
ci-qemu-upstream-386	now	7h19m	34659	600497	272	1221388	ceaa837f96ad	.config	19h05m		957959cb	6h29m	
ci-qemu2-arm32	now	6h21m	105418	123977	7	1008683	ceaa837f96ad	.config	19h05m		957959cb	6h29m	
ci-qemu2-arm64	now	6h31m	89233	103206	1	438645	0983f6bf2bfc	.config	5d18h	failing	957959cb	6h29m	
ci-qemu2-arm64-compat	now	6h23m	84270	95995	10	306776	0983f6bf2bfc	.config	5d18h	failing	957959cb	6h29m	
ci-qemu2-arm64-mte	now	7h15m	103456	120215	3	732180	0983f6bf2bfc	.config	5d18h	failing	957959cb	6h29m	
ci-qemu2-riscv64	now	6h33m	15391	291972	263	59222	0966d385830d	.config	339d	failing	957959cb	6h29m	
ci-upstream-bpf-kasan-gce	now	6h46m	25205	481003	482	1614328	b963d9d5b943	.config	4d16h		4d66ad72	5h55m	
ci-upstream-bpf-next-kasan-gce	now	6h47m	24374	472601	639	2266880	ab86cf337a5b	.config	2d14h		4d66ad72	5h55m	
ci-upstream-gce-arm64	now	6h39m	84278	627911	441	7277604	2d3827b3f393	.config	2d22h	failing	4d66ad72	5h55m	
ci-upstream-gce-leak	now	5h50m	50488	1069589	267	1444029	ceaa837f96ad	.config	19h05m		4d66ad72	5h55m	
ci-upstream-kasan-gce	now	6h46m	41873	659301	104	3195099	ceaa837f96ad	.config	19h05m		4d66ad72	5h55m	
ci-upstream-kasan-gce-386	now	6h46m	34988	582840	81	940583	ceaa837f96ad	.config	19h05m		4d66ad72	5h55m	
ci-upstream-kasan-gce-root	now	5h38m	54773	992479	326	1591364	ceaa837f96ad	.config	19h05m		4d66ad72	5h55m	
ci-upstream-kasan-gce-selinux-root	now	6h45m	49258	1021042	353	1167165	ceaa837f96ad	.config	19h05m		4d66ad72	5h55m	
ci-upstream-kasan-gce-smack-root	now	6h46m	72568	819537	326	1741572	ceaa837f96ad	.config	19h05m		4d66ad72	5h55m	
ci-upstream-kmsan-gce	now	4h41m	63036	416076	253	1464370	da13c00eebfb	.config	6h47m		4d66ad72	5h55m	
ci-upstream-kmsan-gce-386	now	4h54m	63879	467602	189	510559	da13c00eebfb	.config	6h47m		4d66ad72	5h55m	
ci-upstream-linux-next-kasan-gce-root	now	6h08m	76213	1135783	208	1960136	38d2b86a665b	.config	5d13h	failing	4d66ad72	5h55m	
ci-upstream-net-kasan-gce	now	6h47m	36348	422101	296	6019877	6d86bb0a5cb8	.config	7h38m		4d66ad72	5h55m	
ci-upstream-net-this-kasan-gce	now	6h47m	35608	418087	242	4139974	c68f345b7c42	.config	7h34m		4d66ad72	5h55m	
ci2-upstream-fs	now	6h22m	18534	185314	198	3733936	ceaa837f96ad	.config	19h05m		957959cb	6h29m	
ci2-upstream-kcsan-gce	now	7h16m	56455	396769	220	2917605	ceaa837f96ad	.config	19h05m		957959cb	6h29m	
ci2-upstream-usb	now	7h16m	2171	55346	521	1179700	f87b564686ee	.config	4d06h		957959cb	6h29m	

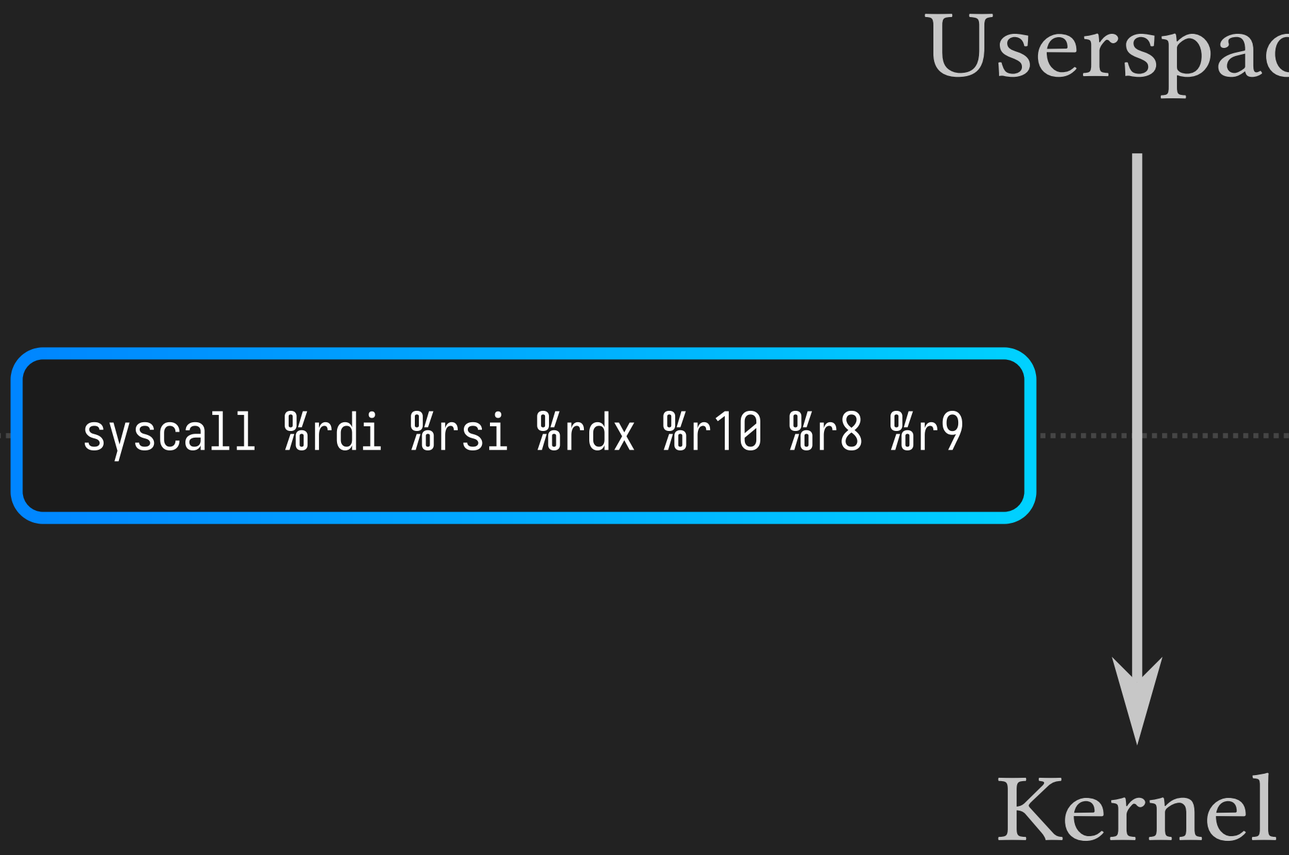
open (1057):

Title	Repro	Cause bisect	Fix bisect	Count	Last	Reported	Last activity
WARNING in udf_prealloc_blocks [udf]				1	4d04h	4h23m	4h23m
KMSAN: kernel-infoleak in iommufd_vfio_ioctl				8	now	6h25m	2h43m
WARNING: suspicious RCU usage in mas_state_walk (2)	C			6	2h01m	10h59m	10h59m
memory leak in vma_node_allow	C			1	4d16h	16h50m	16h50m
WARNING: locking bug in take_dentry_name_snapshot				1	5d10h	1d10h	1d10h
KMSAN: uninit-value in xfs_getfsmap_helper [xfs]				1	8d21h	3d09h	3d09h
KASAN: use-after-free Read in xfs_inode_item_push [xfs]				1	7d11h	3d10h	3d10h
BUG: corrupted list in percpu_counter_destroy [reiserfs]	C	error		1	8d12h	4d12h	4d12h
BUG: unable to handle kernel paging request in folio_fl...	C	error		10	3d14h	4d20h	4d13h
general protection fault in kernfs_link_sibling (3)				5	5d01h	5d01h	5d01h
possible deadlock in exc_page_fault [exfat]				4	3d18h	5d04h	5d04h
INFO: rcu detected stall in ext4_file_write_iter (6)	C	error		6	9d06h	5d06h	3d09h
general protection fault in iomap_dio_bio_iter	C	error		1	9d17h	5d16h	5d13h
general protection fault in bfs_get_block [bfs]				1	11d	5d23h	5d23h
usb-testing build error (3)				1	10d	6d05h	6d05h
WARNING in is_valid_gup_args				1547	5d13h	6d10h	6d04h
WARNING in l2cap_do_send				1	11d	7d07h	7d07h
BUG: unable to handle kernel paging request in vfs_re...				6	5d03h	7d08h	7d08h
general protection fault in __blk_rq_map_sg	C	error		20	15m	8d22h	7d06h
WARNING in kernfs_get (4)	syz	error		3	5d21h	8d23h	8d07h

Fuzzing System Calls



Fuzzing System Calls



Fuzzing System Calls

Fuzzing System Calls

Adding a New System Call

For more sophisticated system calls that involve a larger number of arguments, it's preferred to encapsulate the majority of the arguments into a structure that is passed in by **pointer.** Such a structure can cope with future extension by including a size argument in the structure

...

If your new system call allows userspace to refer to a kernel object, it should use a **file descriptor** as the handle for that object -- don't invent a new type of userspace object handle when the kernel already has mechanisms and well-defined semantics for using file descriptors.

---

linux-kernel/Documentation/process/adding-syscalls.rst

Fuzzing System Calls

Fuzzing System Calls

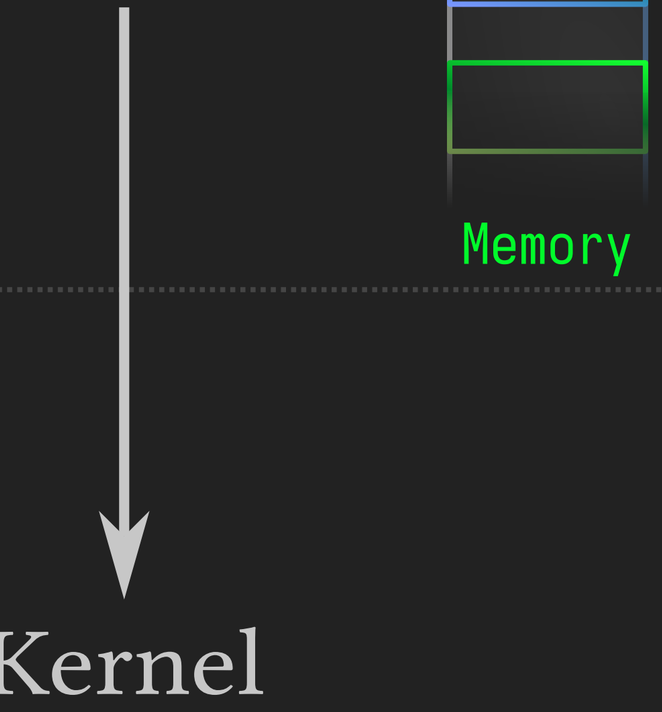
```
fd = open("/var/log/messages", O_RDONLY)
read(fd, buf, 100);
```

Fuzzing System Calls

```
fd = open("/var/log/messages", O_RDONLY)
read(fd, buf, 100);

syscall 0x2 0xce51020 0x0 0x0 0x0 0x0
syscall 0x0     0x55a4e3c2a000 100 0x0 0x0
```

Fuzzing System Calls

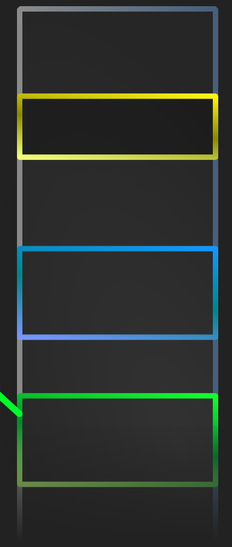
```
fd = open("/var/log/messages", O_RDONLY)
read(fd, buf, 100);

syscall 0x2 0xce51020 0x0 0x0 0x0 0x0
syscall 0x0     0x55a4e3c2a000 100 0x0 0x0
```



Fuzzing System Calls

```
0ce51010 5548 89e5 9090 5dc3 3030 3030 3030 0a00  UH....].000000..
0ce51020 2f76 6172 2f6c 6f67 2f6d 6573 7361 6765  /var/log/message
0ce51030 7300 4572 726f 7220 6f70 656e 696e 6720  s.Error opening
0ce51040 6669 6c65 2e00 5265 6164 696e 6720 7468  file..Reading th
```

```
fd = open("/var/log/messages", O_RDONLY)
read(fd, buf, 100);

syscall 0x2 0xce51020 0x0 0x0 0x0 0x0
syscall 0x0     0x55a4e3c2a000 100 0x0 0x0
```

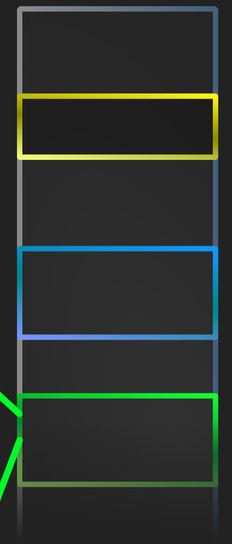


Kernel

Fuzzing System Calls

```
0ce51010 5548 89e5 9090 5dc3 3030 3030 3030 0a00  UH....].000000..
0ce51020 2f76 6172 2f6c 6f67 2f6d 6573 7361 6765  /var/log/message
0ce51030 7300 4572 726f 7220 6f70 656e 696e 6720  s.Error opening
0ce51040 6669 6c65 2e00 5265 6164 696e 6720 7468  file..Reading th
```

```
fd = open("/var/log/messages", O_RDONLY)
read(fd, buf, 100);

syscall 0x2 0xce51020 0x0 0x0 0x0 0x0
syscall 0x0     0x5ba4e3c2a000 100 0x0 0x0
```

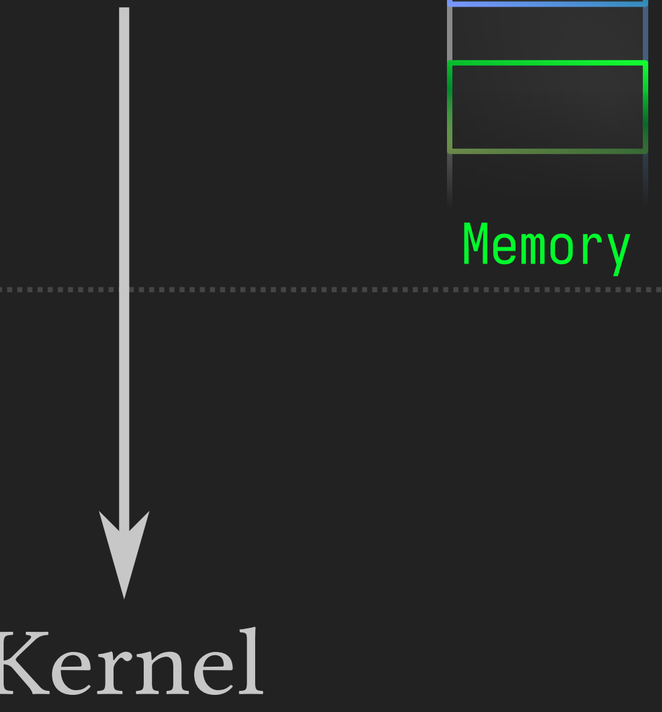


Memory

Kernel

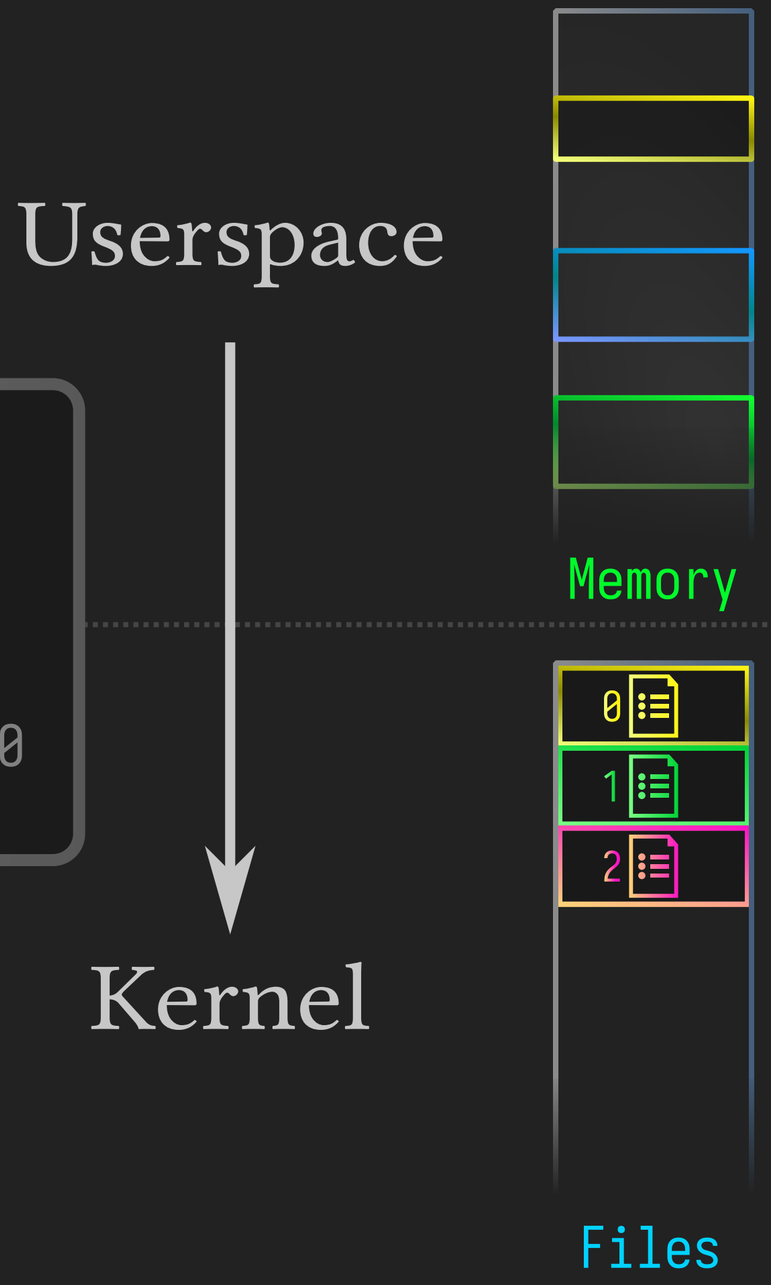
Fuzzing System Calls

```
fd = open("/var/log/messages", O_RDONLY)
read(fd, buf, 100);

syscall 0x2 0xce51020 0x0 0x0 0x0 0x0
syscall 0x0     0x55a4e3c2a000 100 0x0 0x0
```



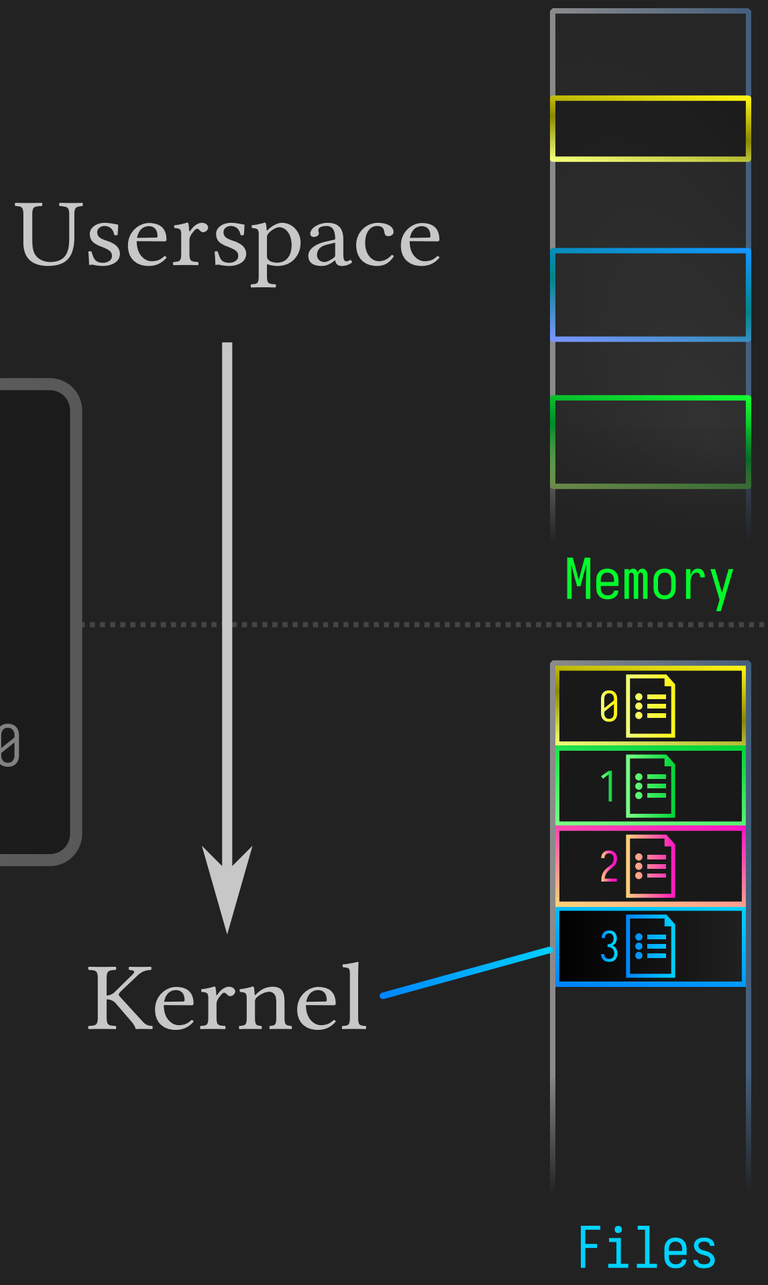
Fuzzing System Calls

```
fd = open("/var/log/messages", O_RDONLY)
read(fd, buf, 100);

syscall 0x2 0xce51020 0x0 0x0 0x0 0x0
syscall 0x0     0x55a4e3c2a000 100 0x0 0x0
```



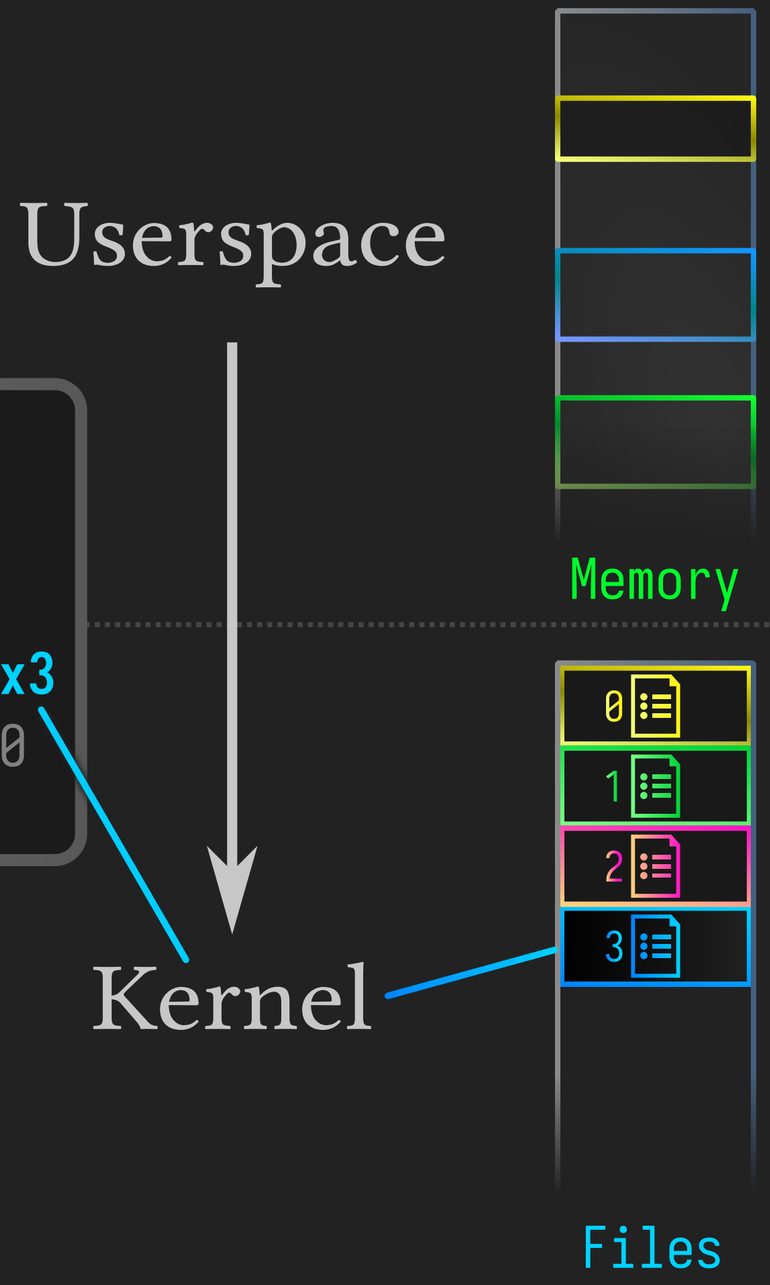
Fuzzing System Calls

```
fd = open("/var/log/messages", O_RDONLY)
read(fd, buf, 100);

syscall 0x2 0xce51020 0x0 0x0 0x0 0x0
syscall 0x0     0x55a4e3c2a000 100 0x0 0x0
```



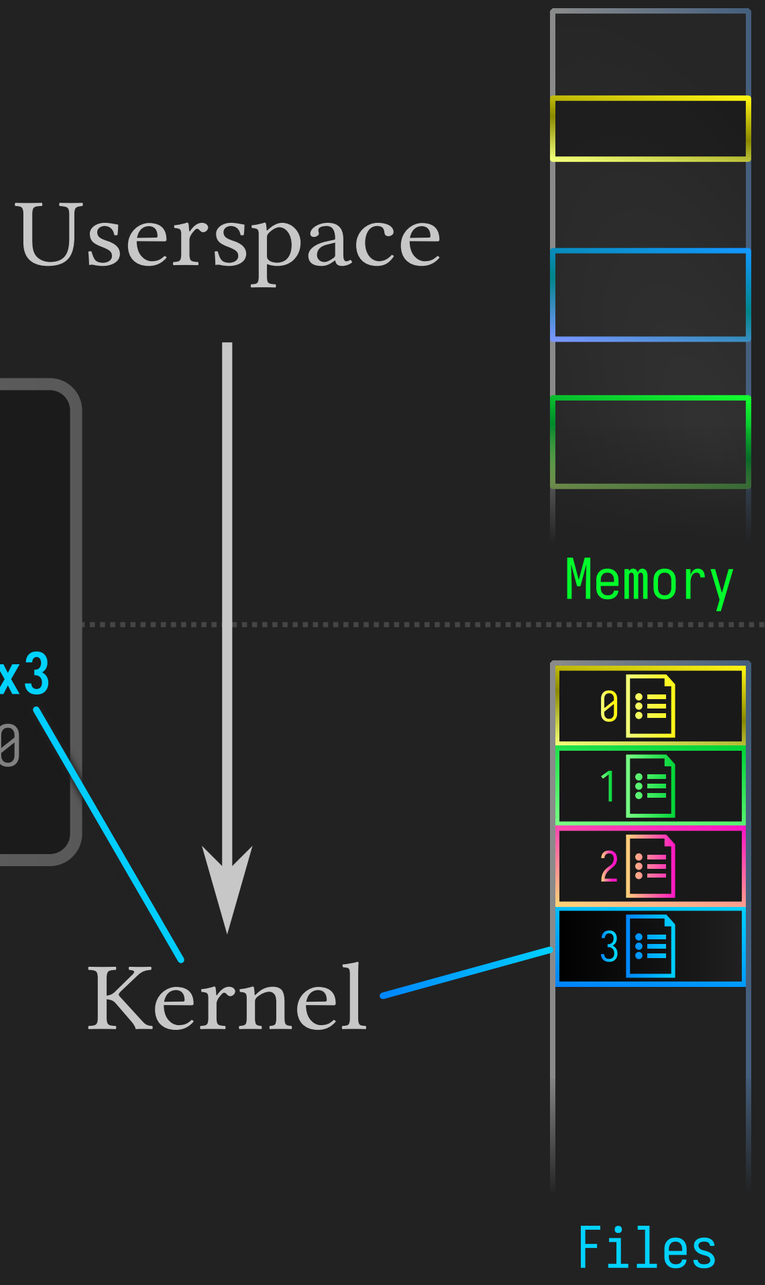
Fuzzing System Calls

```
fd = open("/var/log/messages", O_RDONLY)
read(fd, buf, 100);

syscall 0x2 0xce51020 0x0 0x0 0x0 0x0 = 0x3
syscall 0x0     0x55a4e3c2a000 100 0x0 0x0
```

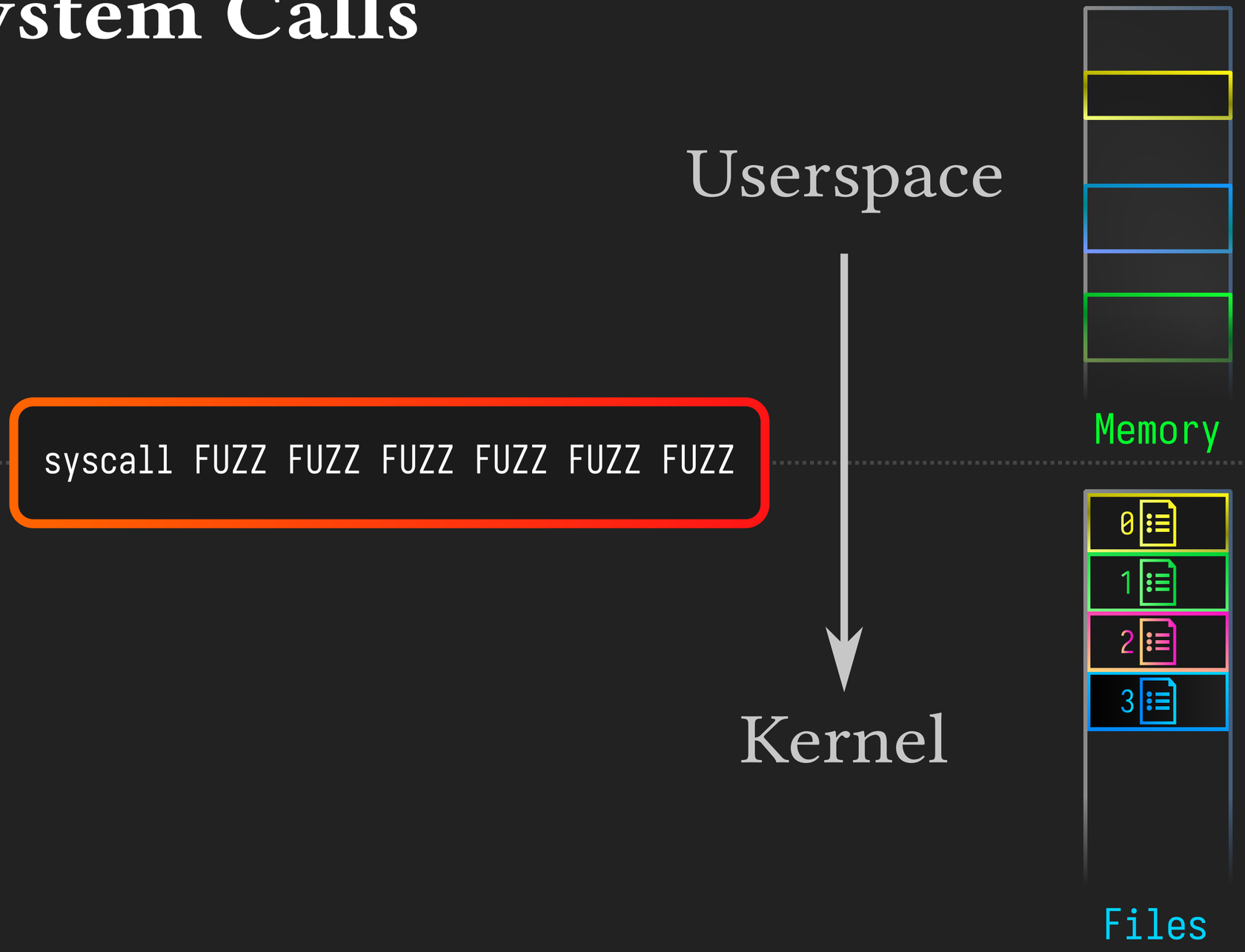


Fuzzing System Calls

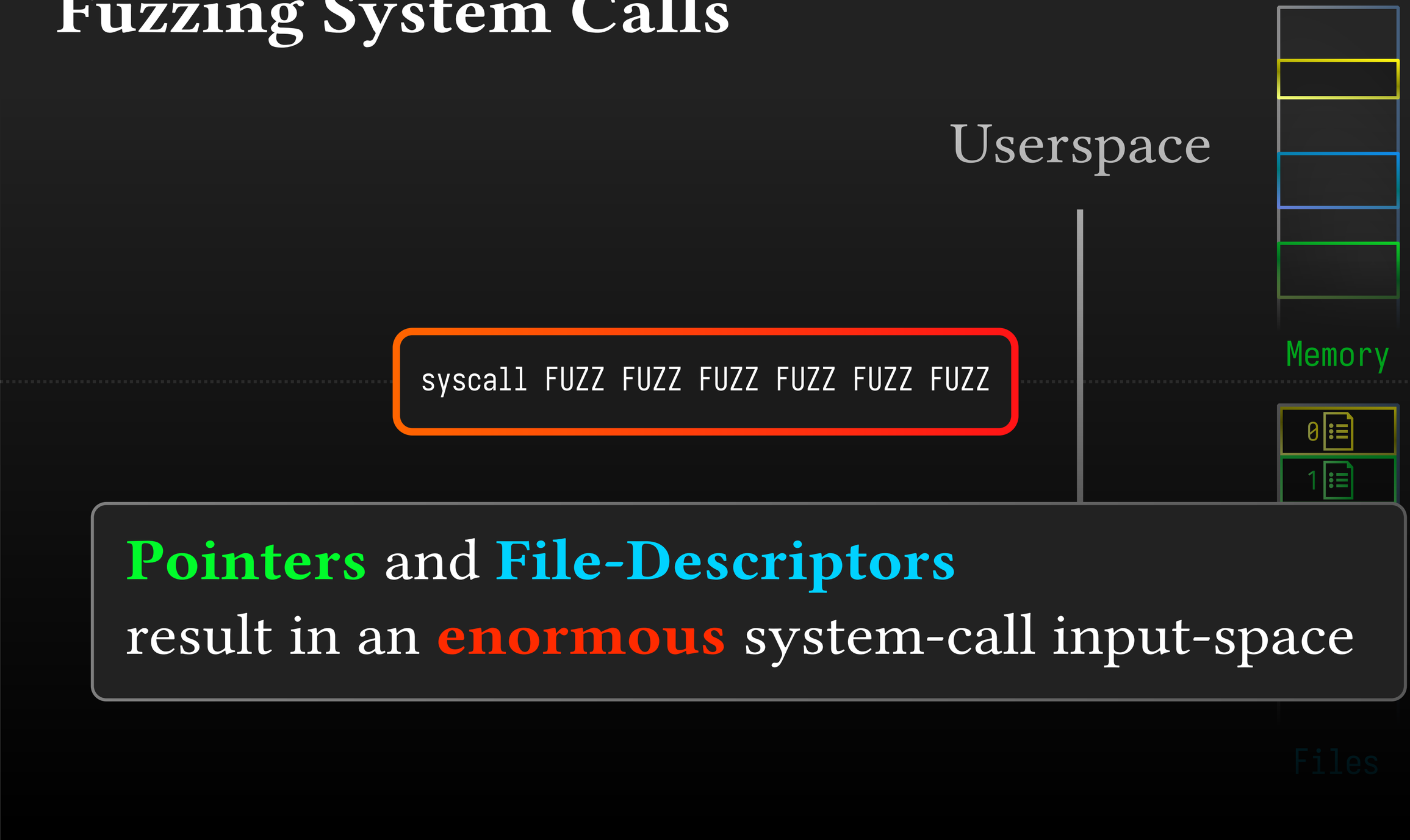
```
fd = open("/var/log/messages", O_RDONLY)
read(fd, buf, 100);

syscall 0x2 0xce51020 0x0 0x0 0x0 0x0 = 0x3
syscall 0x0 0x3 0x55a4e3c2a000 100 0x0 0x0
```



Fuzzing System Calls



Fuzzing System Calls



Pointers and **File-Descriptors**
result in an **enormous** system-call input-space

System-Call Grammars

```
syz_io_uring_setup(entries int32[1:IORING_MAX_ENTRIES], params ptr[inout, io_uring_params], addr_ring vma, addr_sqes vma, ring_ptr ptr[out, ring_ptr], sqes_ptr ptr[out, sqes_ptr]) fd_io_uring

io_uring_setup(entries int32[1:IORING_MAX_ENTRIES], params ptr[inout, io_uring_params]) fd_io_uring
io_uring_enter(fd fd_io_uring, to_submit int32[0:IORING_MAX_ENTRIES], min_complete int32[0:IORING_MAX_CQ_ENTRIES], flags flags[io_uring_enter_flags], sigmask ptr[in, sigset_t], size len[sigmask])
io_uring_register$IORING_REGISTER_BUFFERS(fd fd_io_uring, opcode const[IORING_REGISTER_BUFFERS], arg ptr[in, array[iovec_out]], nr_args len[arg])
io_uring_register$IORING_UNREGISTER_BUFFERS(fd fd_io_uring, opcode const[IORING_UNREGISTER_BUFFERS], arg const[0], nr_args const[0])
io_uring_register$IORING_REGISTER_FILES(fd fd_io_uring, opcode const[IORING_REGISTER_FILES], arg ptr[in, array[fd]], nr_args len[arg])
io_uring_register$IORING_UNREGISTER_FILES(fd fd_io_uring, opcode const[IORING_UNREGISTER_FILES], arg const[0], nr_args const[0])
io_uring_register$IORING_REGISTER_EVENTFD(fd fd_io_uring, opcode const[IORING_REGISTER_EVENTFD], arg ptr[in, fd_event], nr_args const[1])
io_uring_register$IORING_UNREGISTER_EVENTFD(fd fd_io_uring, opcode const[IORING_UNREGISTER_EVENTFD], arg const[0], nr_args const[0])
io_uring_register$IORING_REGISTER_FILES_UPDATE(fd fd_io_uring, opcode const[IORING_REGISTER_FILES_UPDATE], arg ptr[in, io_uring_files_update], nr_args len[arg:fds])
io_uring_register$IORING_REGISTER_EVENTFD_ASYNC(fd fd_io_uring, opcode const[IORING_REGISTER_EVENTFD_ASYNC], arg ptr[in, fd_event], nr_args const[1])
io_uring_register$IORING_REGISTER_PROBE(fd fd_io_uring, opcode const[IORING_REGISTER_PROBE], arg ptr[inout, io_uring_probe], nr_args len[arg:ops])
io_uring_register$IORING_REGISTER_PERSONALITY(fd fd_io_uring, opcode const[IORING_REGISTER_PERSONALITY], arg const[0], nr_args const[0]) ioring_personality_id
io_uring_register$IORING_UNREGISTER_PERSONALITY(fd fd_io_uring, opcode const[IORING_UNREGISTER_PERSONALITY], arg const[0], nr_args ioring_personality_id)

# The mmap'ed area for SQ and CQ rings are really the same -- the difference is
# accounted for with the usage of offsets.
mmap$IORING_OFF_SQ_RING(addr vma, len len[addr], prot flags[mmap_prot], flags flags[mmap_flags], fd fd_io_uring, offset const[IORING_OFF_SQ_RING]) ring_ptr
mmap$IORING_OFF_CQ_RING(addr vma, len len[addr], prot flags[mmap_prot], flags flags[mmap_flags], fd fd_io_uring, offset const[IORING_OFF_CQ_RING]) ring_ptr
mmap$IORING_OFF_SQES(addr vma, len len[addr], prot flags[mmap_prot], flags flags[mmap_flags], fd fd_io_uring, offset const[IORING_OFF_SQES]) sqes_ptr

# If no flags are specified(0), the io_uring instance is setup for interrupt driven IO.
io_uring_setup_flags = 0, IORING_SETUP_IOPOLL, IORING_SETUP_SQPOLL, IORING_SETUP_SQ_AFF, IORING_SETUP_CQSIZE, IORING_SETUP_CLAMP, IORING_SETUP_ATTACH_WQ
io_uring_enter_flags = IORING_ENTER_GETEVENTS, IORING_ENTER_SQ_WAKEUP
_ = __NR_mmap2

# Once an io_uring is set up by calling io_uring_setup, the offsets to the member fields
# to be used on the mmap'ed area are set in structs io_sqring_offsets and io_cqring_offsets.
# Except io_sqring_offsets.array, the offsets are static while all depend on how struct io_rings
# is organized in code. The offsets can be marked as resources in syzkaller descriptions but
# this makes it difficult to generate correct programs by the fuzzer. Thus, the offsets are
# hard-coded here (and in the executor).
define SQ_HEAD_OFFSET 0
define SQ_TAIL_OFFSET 64
define SQ_RING_MASK_OFFSET 256
define SQ_RING_ENTRIES_OFFSET 264
define SQ_FLAGS_OFFSET 276
```

System-Call Grammars

```
io_uring_setup(entries int32[1:IORING_MAX_ENTRIES],
               params ptr[inout, io_uring_params]) fd_io_uring
```

System-Call Grammars

```
io_uring_setup(entries int32[1:IORING_MAX_ENTRIES],
               params ptr[inout, io_uring_params]) fd_io_uring

io_uring_register$IORING_REGISTER_PROBE(fd fd_io_uring,
                                        opcode const[IORING_REGISTER_PROBE],
                                        arg ptr[inout, io_uring_probe], nr_args len[arg:ops])
```

System-Call Grammars

```
io_uring_setup(entries int32[1:IORING_MAX_ENTRIES],
               params ptr[inout, io_uring_params]) fd_io_uring

io_uring_register$IORING_REGISTER_PROBE(fd fd_io_uring,
                                        opcode const[IORING_REGISTER_PROBE],
                                        arg ptr[inout, io_uring_probe], nr_args len[arg:ops])
io_uring_probe {
    last_op const[0, int8]
    ops_len const[0, int8]
    resv    const[0, int16]
    resv2   array[const[0, int32], 3]
    ops array[io_uring_probe_op, 0:IORING_OP_LAST]
}
io_uring_probe_op {
    op  const[0, int8]
    resv    const[0, int8]
    flags   const[0, int16]
    resv2   const[0, int32]
}
```

System-Call Grammars

io_uring_enter(2) Linux Programmer's Manual io_uring_enter(2)

NAME

io_uring_enter - initiate and/or complete asynchronous I/O

SYNOPSIS

```
#include <liburing.h>

int io_uring_enter(unsigned int fd, unsigned int to_submit,
                   unsigned int min_complete, unsigned int
flags,
                   sigset_t *sig);

int io_uring_enter2(unsigned int fd, unsigned int to_submit,
                    unsigned int min_complete, unsigned int
flags,
                    sigset_t *sig, size_t sz);
```

DESCRIPTION

io_uring_enter(2) is used to initiate and complete I/O using the shared submission and completion queues setup by a call to io_uring_setup(2). A single call can both submit new I/O and wait for completions of I/O initiated by this call or previous calls to io_uring_enter(2).

fd is the file descriptor returned by io_uring_setup(2). *to_submit* specifies the number of I/Os to submit from the submission queue. *flags* is a bitmask of the following values:

IORING_ENTER_GETEVENTS
: If this flag is set, then the system call will wait for the specified number of events in *min_complete* before returning. This flag can be set along with *to_submit* to both submit and complete events in a single system call.

System-Call Grammars

io_uring_enter(2) Linux Programmer's Manual io_uring_enter(2)

NAME

io_uring_enter - initiate and/or complete asynchronous I/O

SYNOPSIS

```
#include <liburing.h>

int io_uring_enter(unsigned int fd, unsigned int to_submit,
                   unsigned int min_complete, unsigned int
flags,
                   sigset_t *sig);

int io_uring_enter2(unsigned int fd, unsigned int to_submit,
                    unsigned int min_complete, unsigned int
flags,
                    sigset_t *sig, size_t sz);
```

DESCRIPTION

io_uring_enter(2) is used to initiate and complete I/O using the shared submission and completion queues setup by a call to io_uring_setup(2). A single call can both submit new I/O and wait for completions of I/O initiated by this call or previous calls to io_uring_enter(2).

fd is the file descriptor returned by io_uring_setup(2). *to_submit* specifies the number of I/Os to submit from the submission queue. *flags* is a bitmask of the following values:

IORING_ENTER_GETEVENTS
: If this flag is set, then the system call will wait for the specified number of events in *min_complete* before returning. This flag can be set along with *to_submit* to both submit and complete events in a single system call.

```
SYSCALL_DEFINE6(io_uring_enter, unsigned int, fd, u32, to_submit,
        u32, min_complete, u32, flags, const void __user *, argp,
        size_t, argsz)
{
    struct io_ring_ctx *ctx;
    long ret = -EBADF;
    int submitted = 0;
    struct fd f;

    io_run_task_work();

    if (flags & ~(IORING_ENTER_GETEVENTS | IORING_ENTER_SQ_WAKEUP |
            IORING_ENTER_SQ_WAIT | IORING_ENTER_EXT_ARG))
        return -EINVAL;

    f = fdget(fd);
    if (!f.file)
        return -EBADF;

    ret = -EOPNOTSUPP;
    if (f.file->f_op != &io_uring_fops)
        goto out_fput;

    ret = -ENXIO;
    ctx = f.file->private_data;
    if (!percpu_ref_tryget(&ctx->refs))
        goto out_fput;
```

System-Call Grammars

System-Call Grammars

Li, Dan, and Hua Chen. "**FastSyzkaller**: Improving fuzz efficiency for linux kernel fuzzing." Journal of Physics: Conference Series. Vol. 1176. No. 2. IOP Publishing, 2019.

Wang, Daimeng, et al. "**SyzVegas**: Beating Kernel Fuzzing Odds with Reinforcement Learning." USENIX Security Symposium. 2021.

Pailoor, Shankara, Andrew Aday, and Suman Jana. "**MoonShine**: Optimizing OS Fuzzer Seed Selection with Trace Distillation." USENIX Security Symposium. 2018.

Sun, Hao, et al. "**KSG**: Augmenting Kernel Fuzzing with System Call Specification Generation." 2022 USENIX Annual Technical Conference (USENIX ATC 22). 2022.

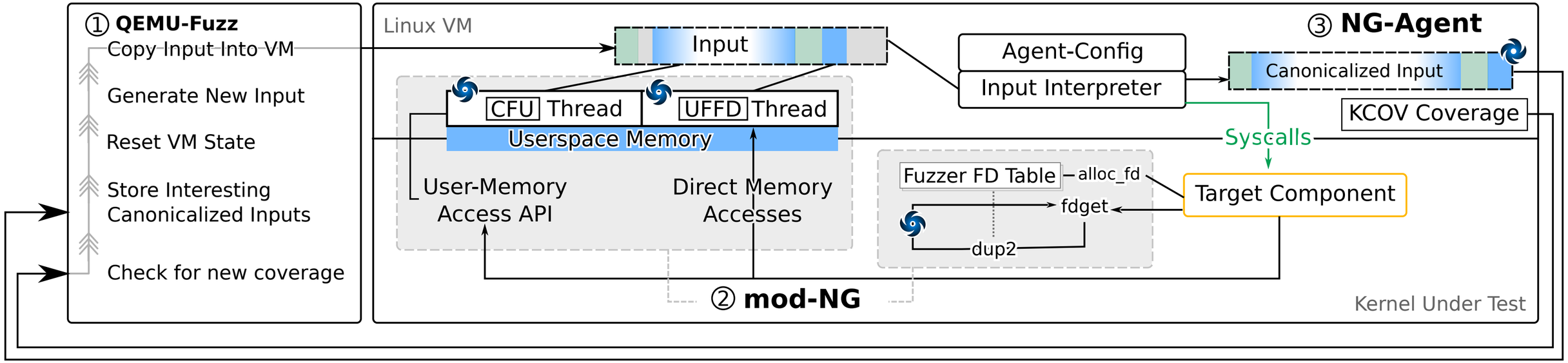
Sun, Hao, et al. "**HEALER**: Relation learning guided kernel fuzzing." Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles. 2021.

Current System-Call Fuzzers rely on detailed grammars to describe pointer and file-descriptor arguments

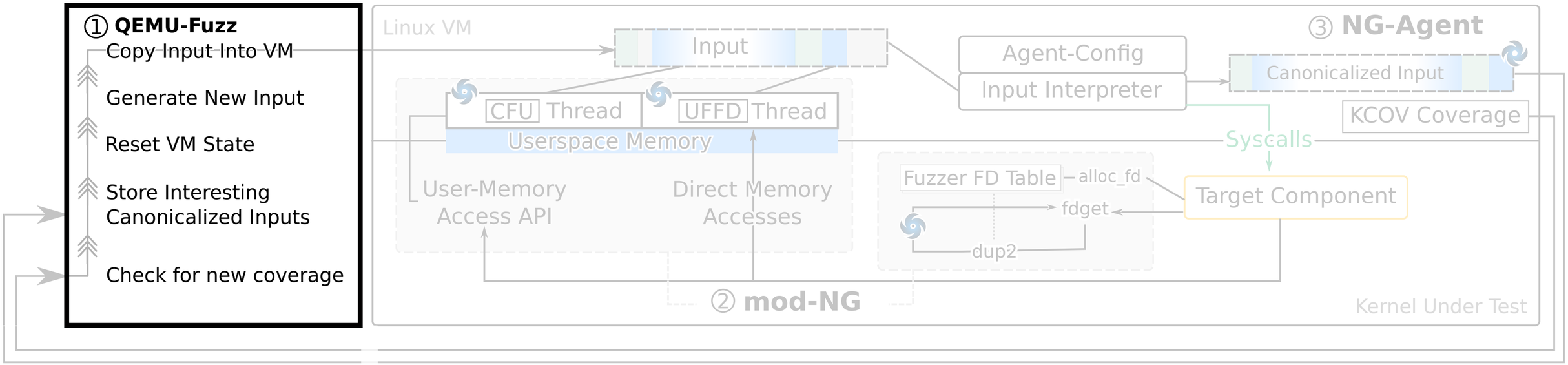
FuzzNG

Reshape the pointer and file-descriptor input-spaces to make system-calls conducive to off-the-shelf fuzzing methods

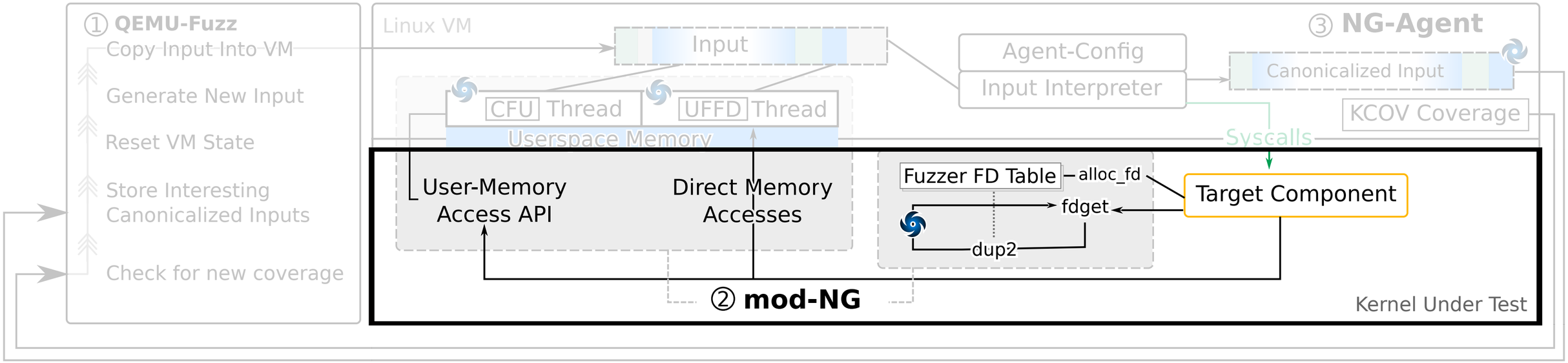
FuzzNG


① QEMU-Fuzz
Copy Input Into VM
Generate New Input
Reset VM State
Store Interesting Canonicalized Inputs
Check for new coverage
Linux VM
Input
CFU Thread
UFFD Thread
Userspace Memory
User-Memory Access API
Direct Memory Accesses
② mod-NG
Agent-Config
Input Interpreter
③ NG-Agent
Canonicalized Input
KCOV Coverage
Syscalls
Fuzzer FD Table
alloc_fd
fdget
dup2
Target Component
Kernel Under Test

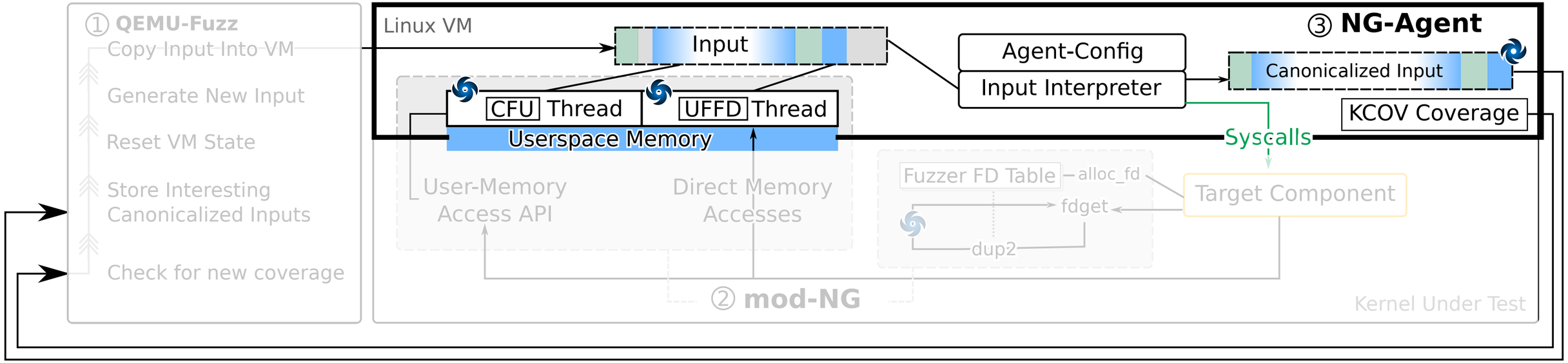
FuzzNG


① QEMU-Fuzz
Copy Input Into VM
Generate New Input
Reset VM State
Store Interesting Canonicalized Inputs
Check for new coverage
Linux VM
Input
CFU Thread
UFFD Thread
Userspace Memory
User-Memory Access API
Direct Memory Accesses
② mod-NG
Agent-Config
Input Interpreter
③ NG-Agent
Canonicalized Input
KCOV Coverage
Syscalls
Fuzzer FD Table
alloc_fd
fdget
dup2
Target Component
Kernel Under Test

FuzzNG


① QEMU-Fuzz
Copy Input Into VM
Generate New Input
Reset VM State
Store Interesting Canonicalized Inputs
Check for new coverage
Linux VM
Input
CFU Thread
UFFD Thread
Userspace Memory
User-Memory Access API
Direct Memory Accesses
Agent-Config
Input Interpreter
③ NG-Agent
Canonicalized Input
KCOV Coverage
Syscalls
Fuzzer FD Table
alloc_fd
fdget
dup2
Target Component
② mod-NG
Kernel Under Test

FuzzNG


① QEMU-Fuzz
Copy Input Into VM
Generate New Input
Reset VM State
Store Interesting Canonicalized Inputs
Check for new coverage
Linux VM
Input
CFU Thread
UFFD Thread
Userspace Memory
User-Memory Access API
Direct Memory Accesses
Agent-Config
Input Interpreter
③ NG-Agent
Canonicalized Input
KCOV Coverage
Syscalls
Fuzzer FD Table
alloc_fd
fdget
dup2
Target Component
② mod-NG
Kernel Under Test

Reshaping the Pointer and File-Descriptor Input Spaces

```
syscall FUZZ FUZZ FUZZ FUZZ FUZZ FUZZ
```

Reshaping the Pointer and File-Descriptor Input Spaces

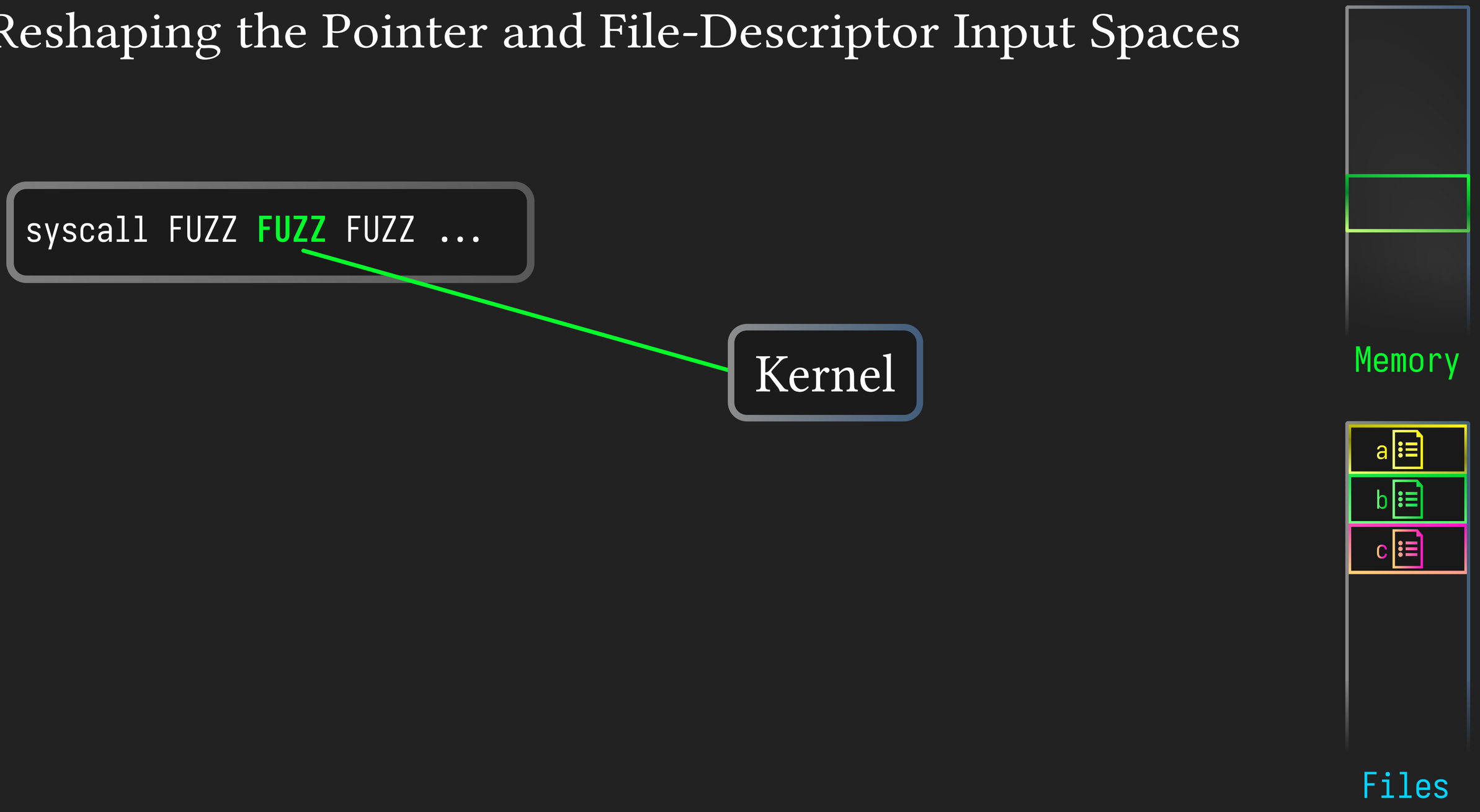
```
syscall FUZZ FUZZ FUZZ FUZZ FUZZ FUZZ
```

What will it take to make **fuzzer-generated pointers and file-descriptors** result in meaningful target behaviors?

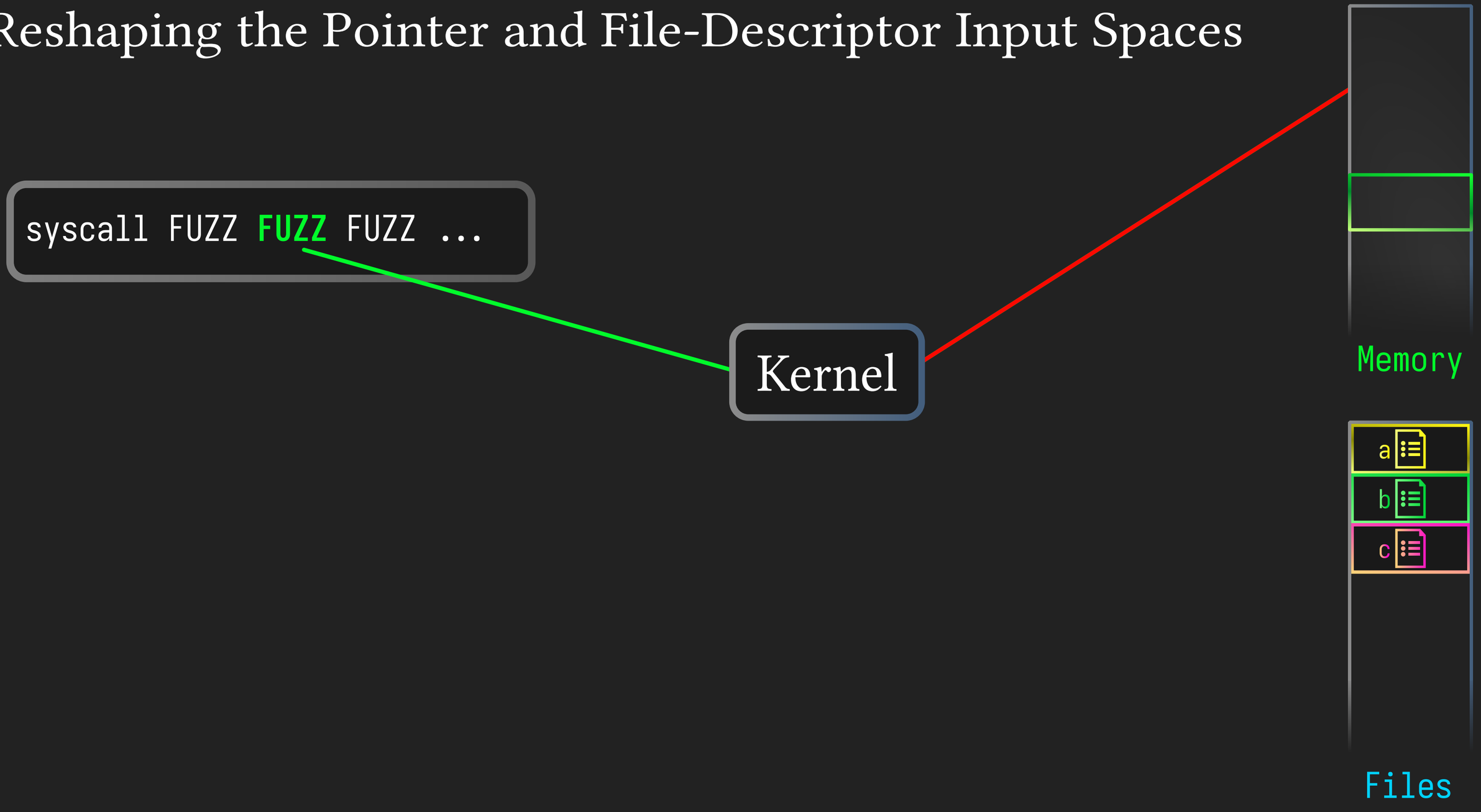
Reshaping the Pointer and File-Descriptor Input Spaces

Kernel

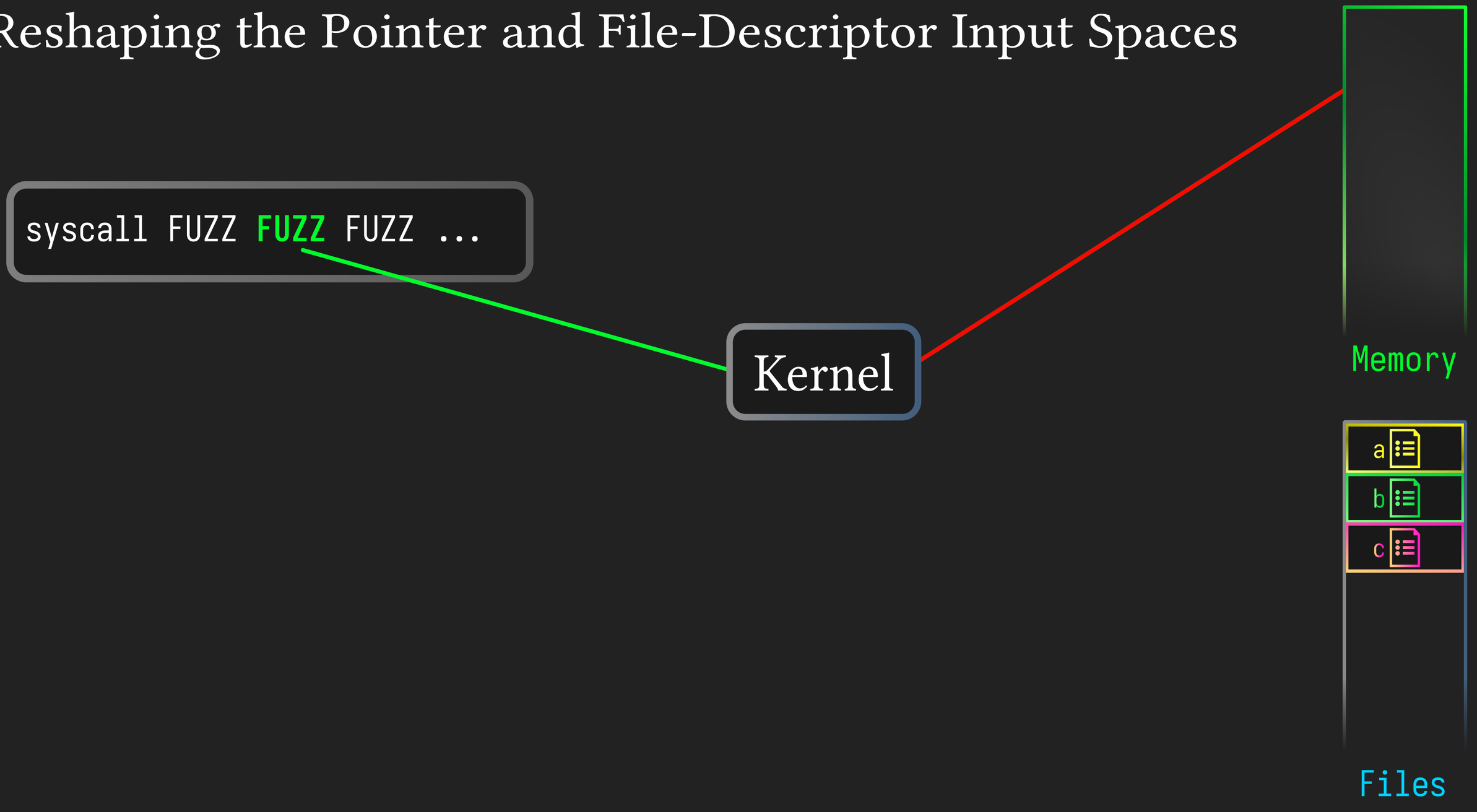
Reshaping the Pointer and File-Descriptor Input Spaces



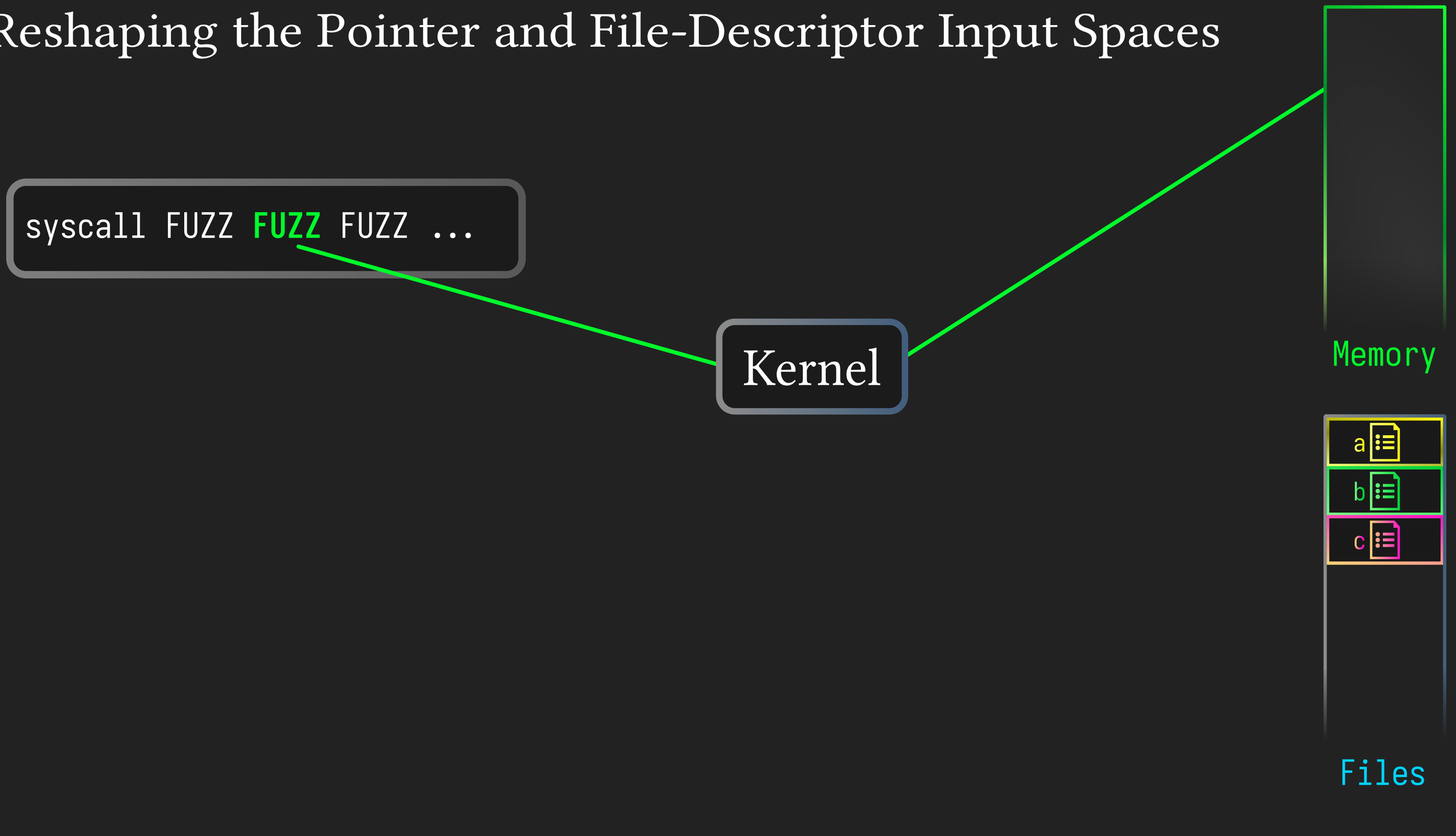
Reshaping the Pointer and File-Descriptor Input Spaces



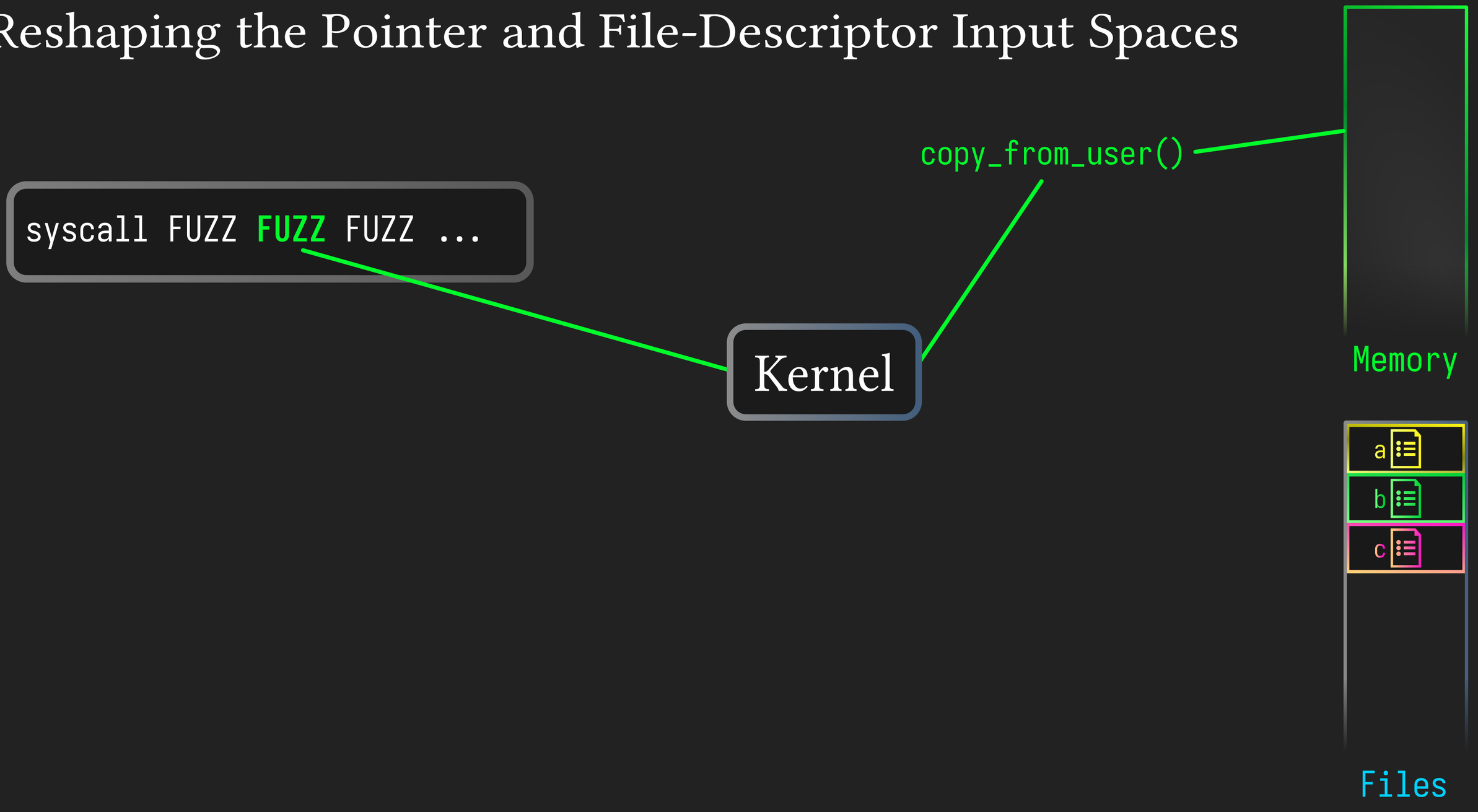
Reshaping the Pointer and File-Descriptor Input Spaces



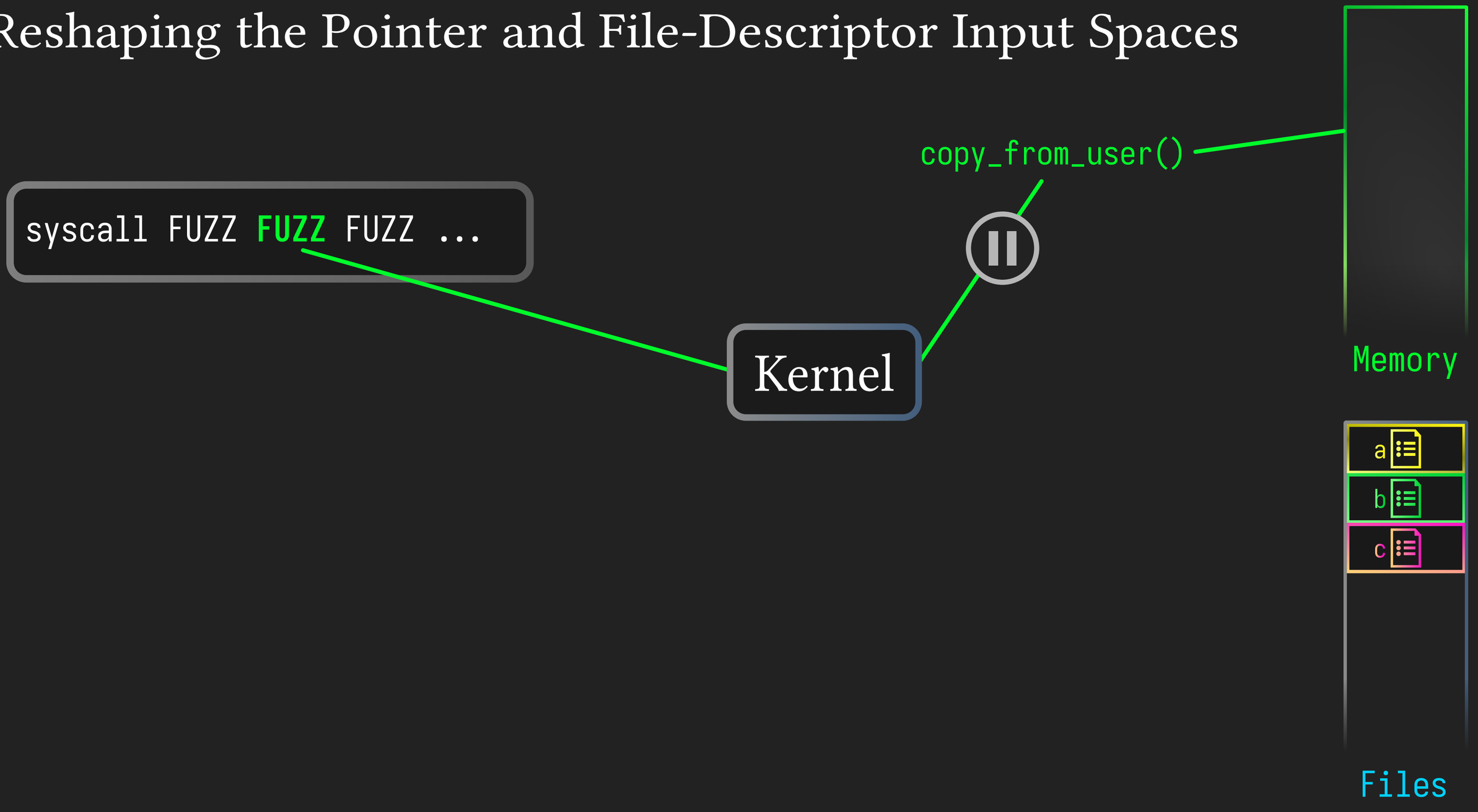
Reshaping the Pointer and File-Descriptor Input Spaces



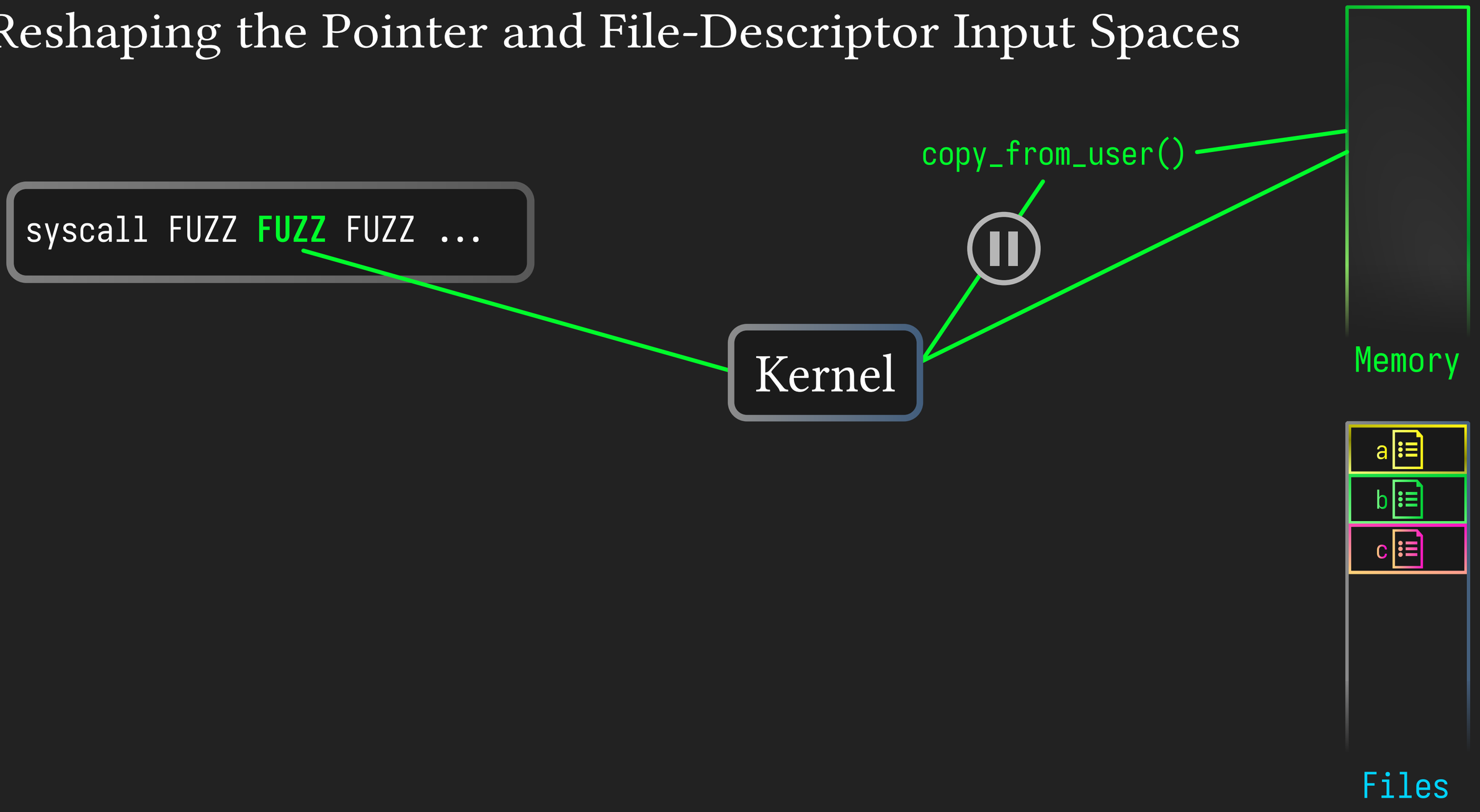
Reshaping the Pointer and File-Descriptor Input Spaces



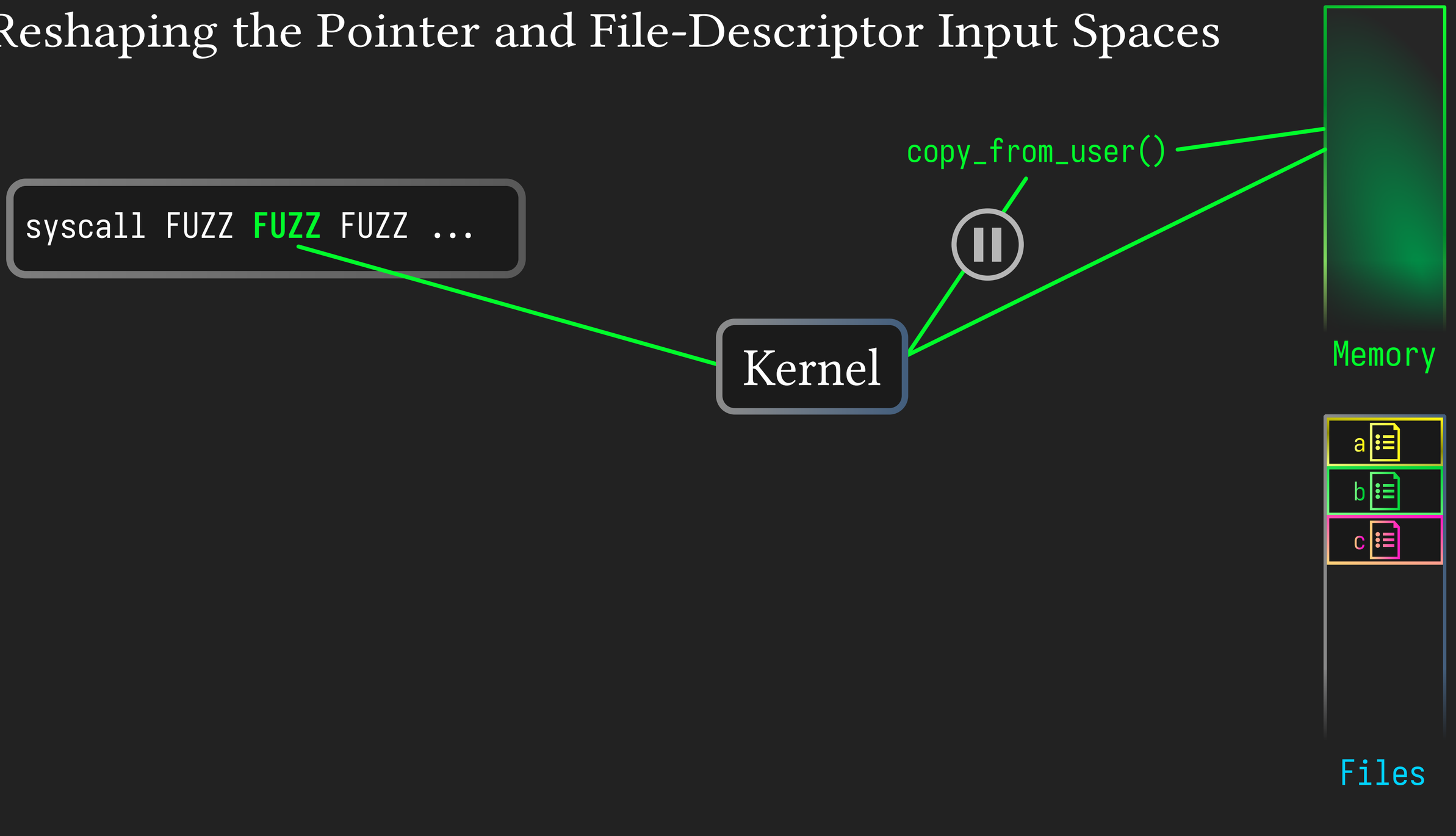
Reshaping the Pointer and File-Descriptor Input Spaces



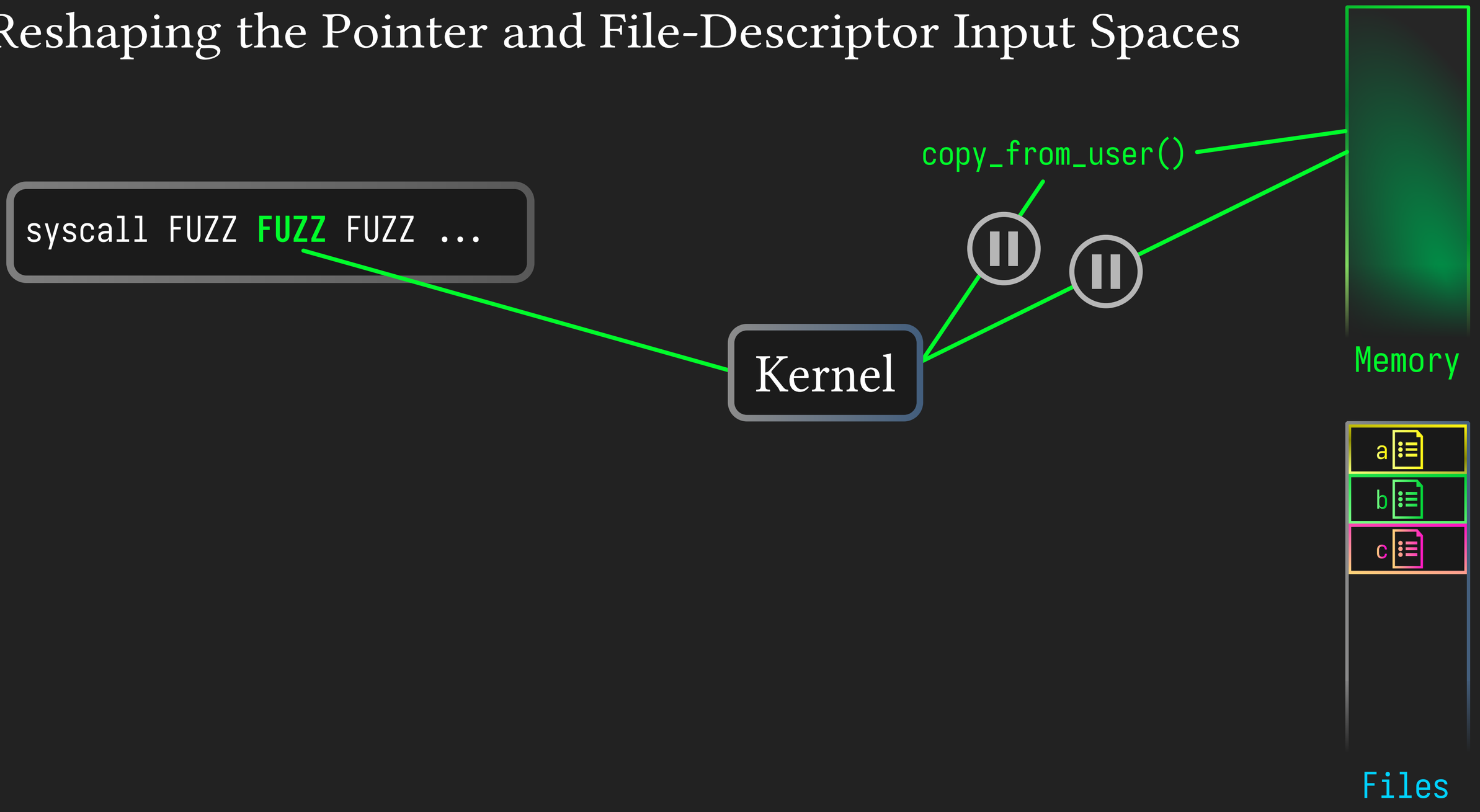
Reshaping the Pointer and File-Descriptor Input Spaces



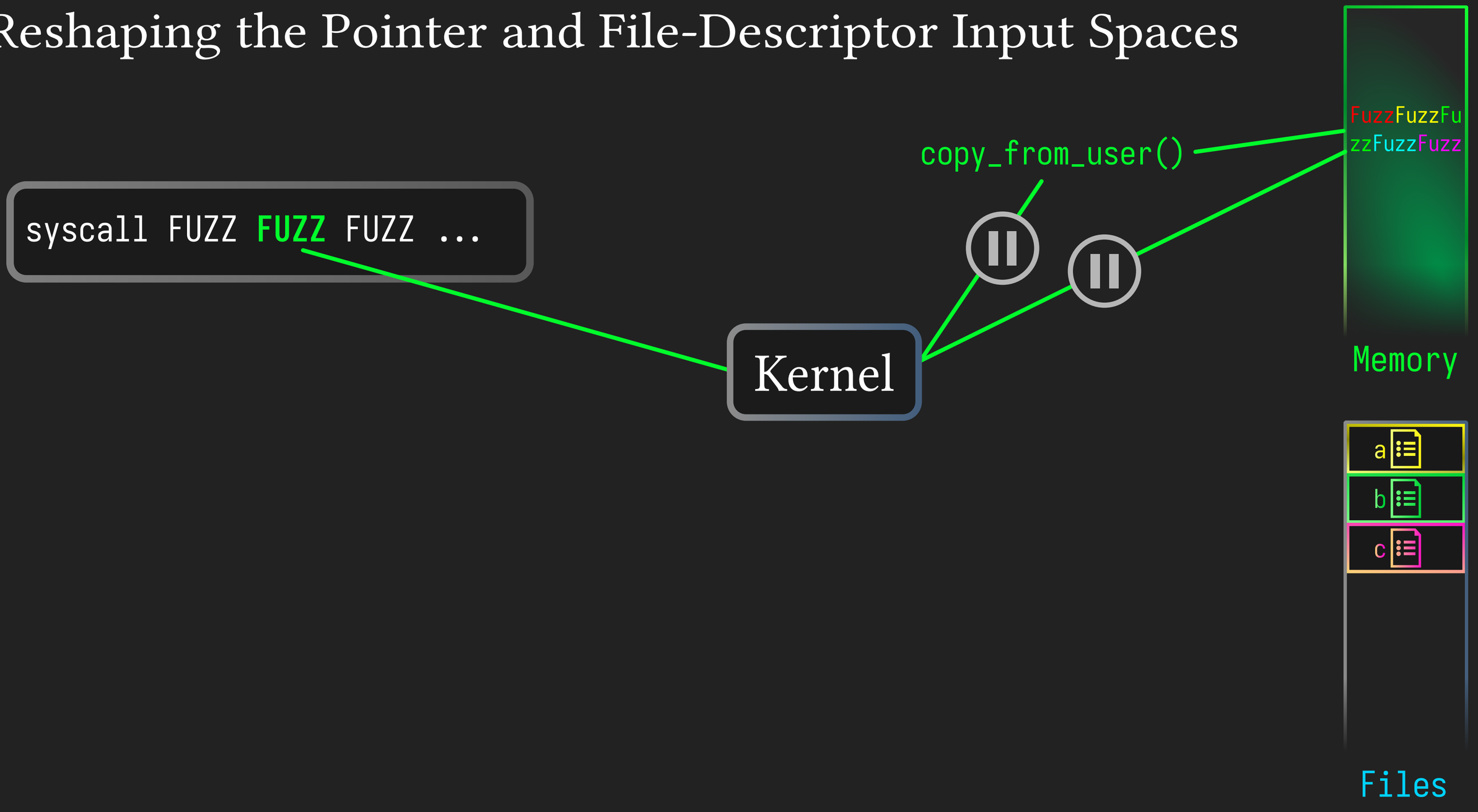
Reshaping the Pointer and File-Descriptor Input Spaces



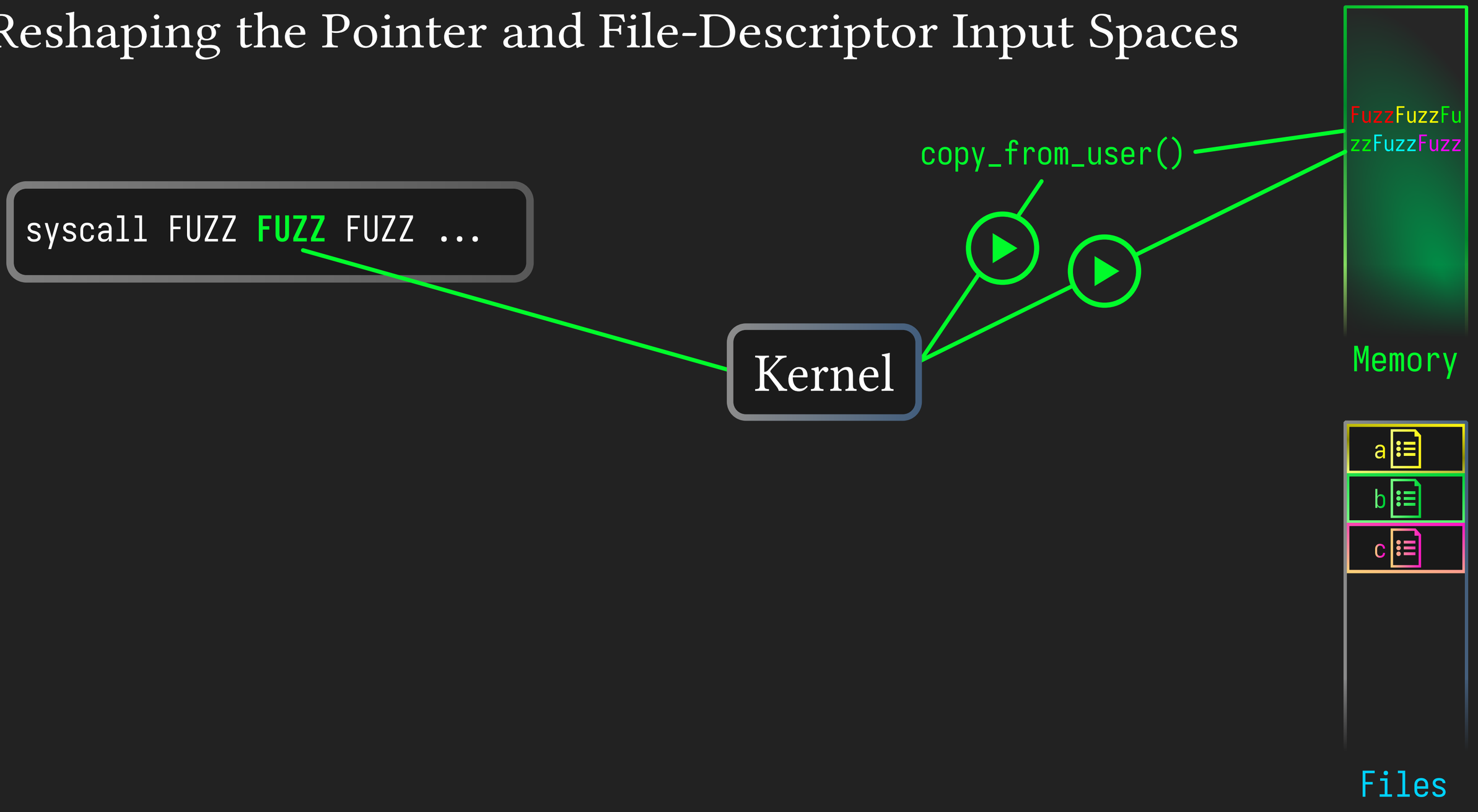
Reshaping the Pointer and File-Descriptor Input Spaces



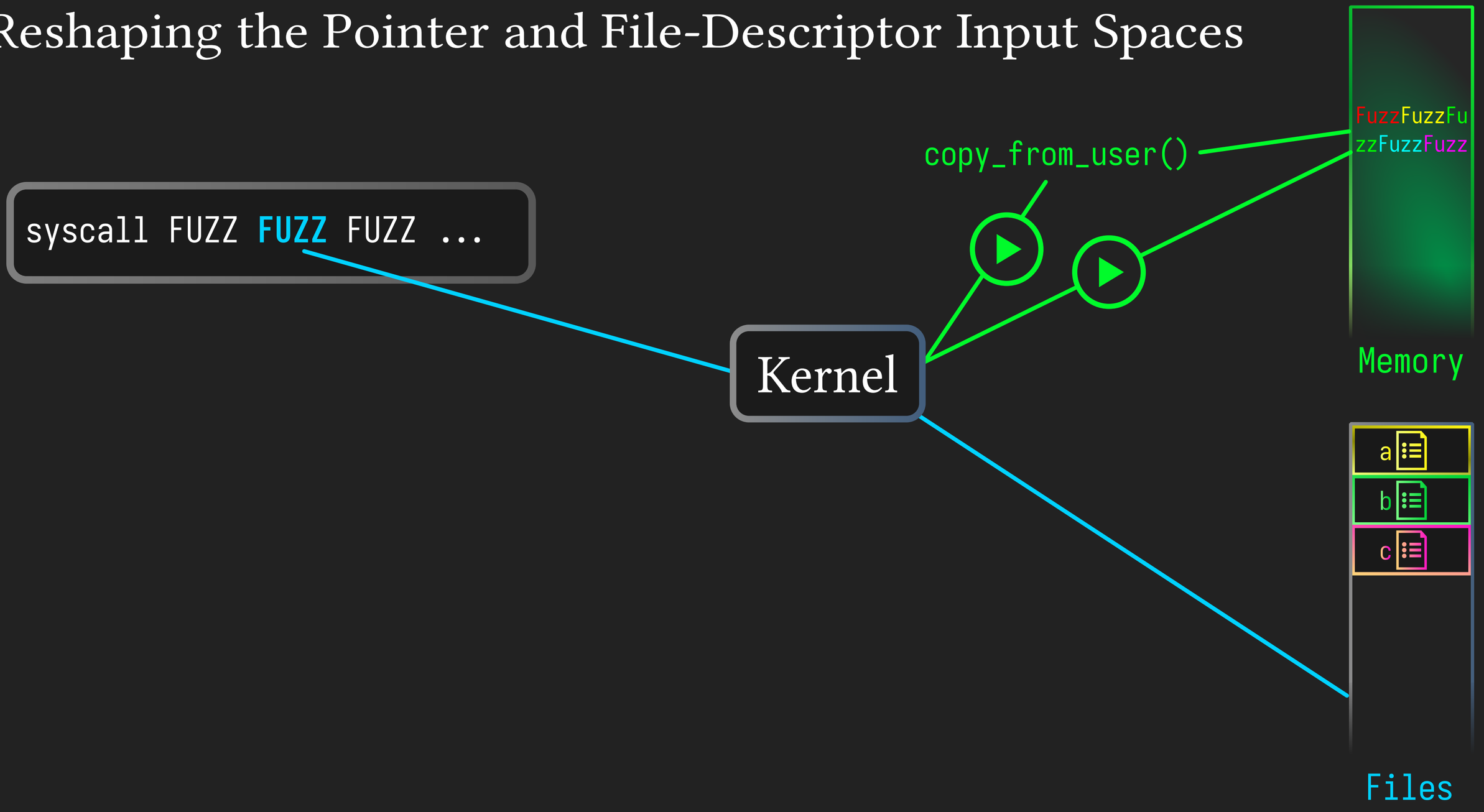
Reshaping the Pointer and File-Descriptor Input Spaces



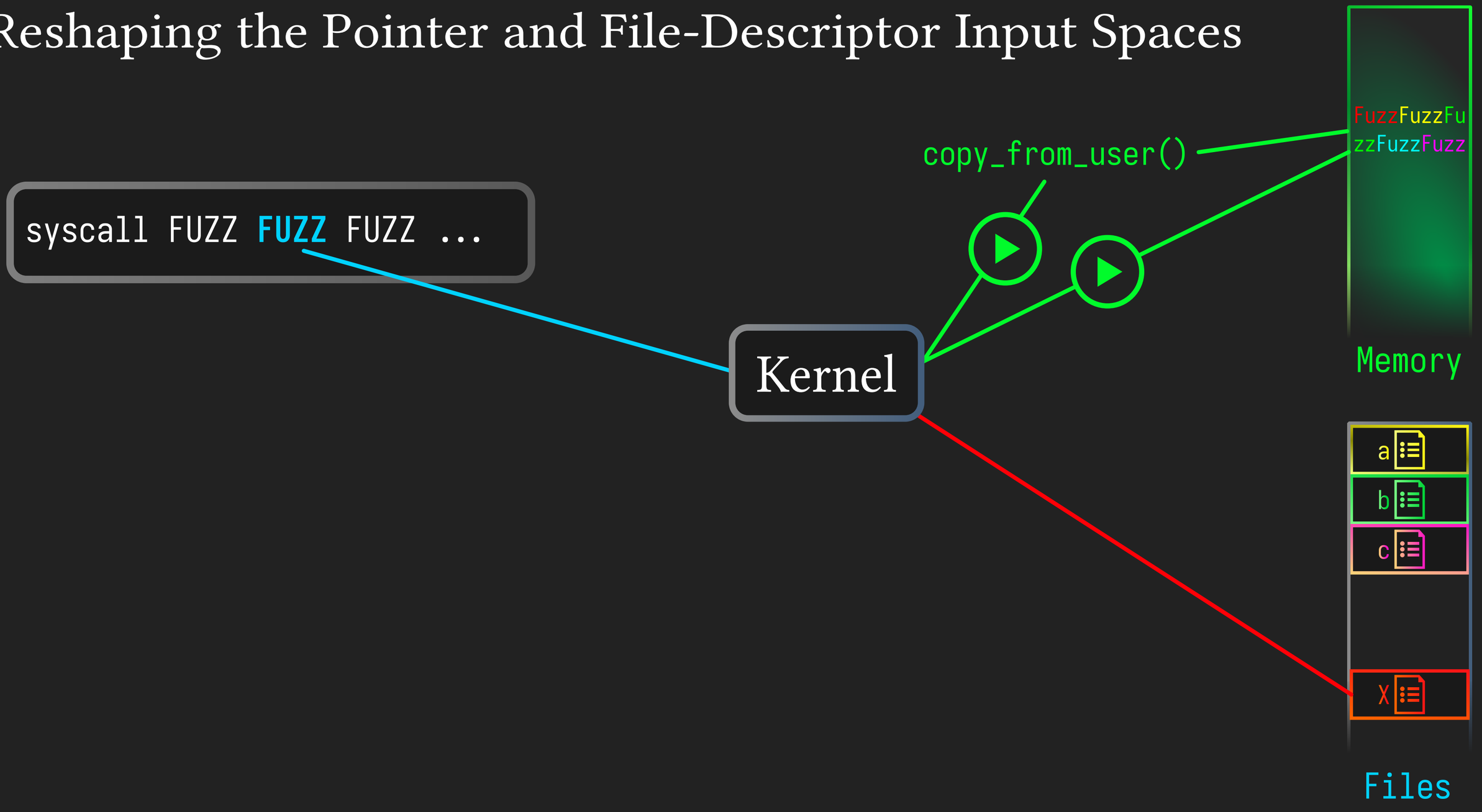
Reshaping the Pointer and File-Descriptor Input Spaces



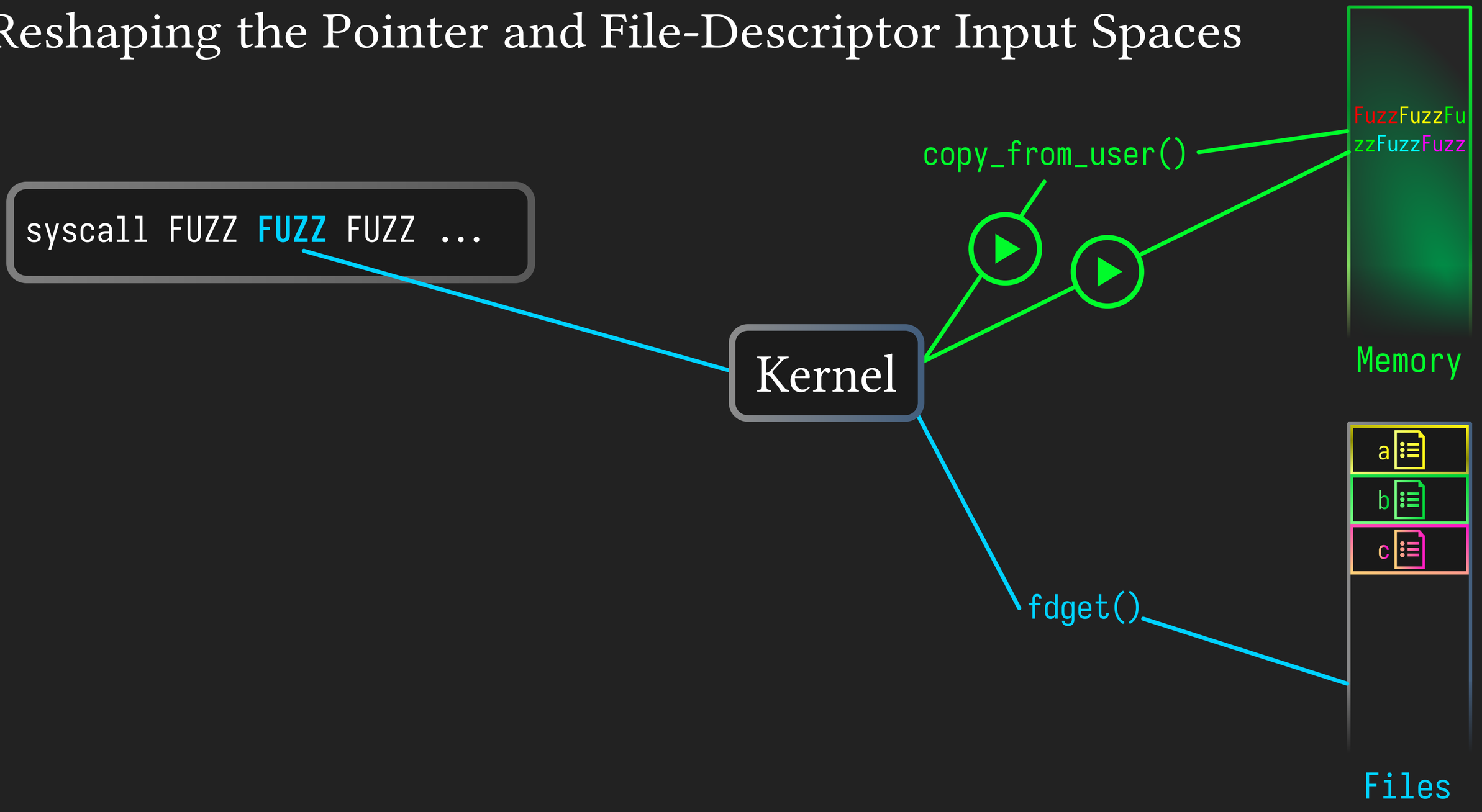
Reshaping the Pointer and File-Descriptor Input Spaces



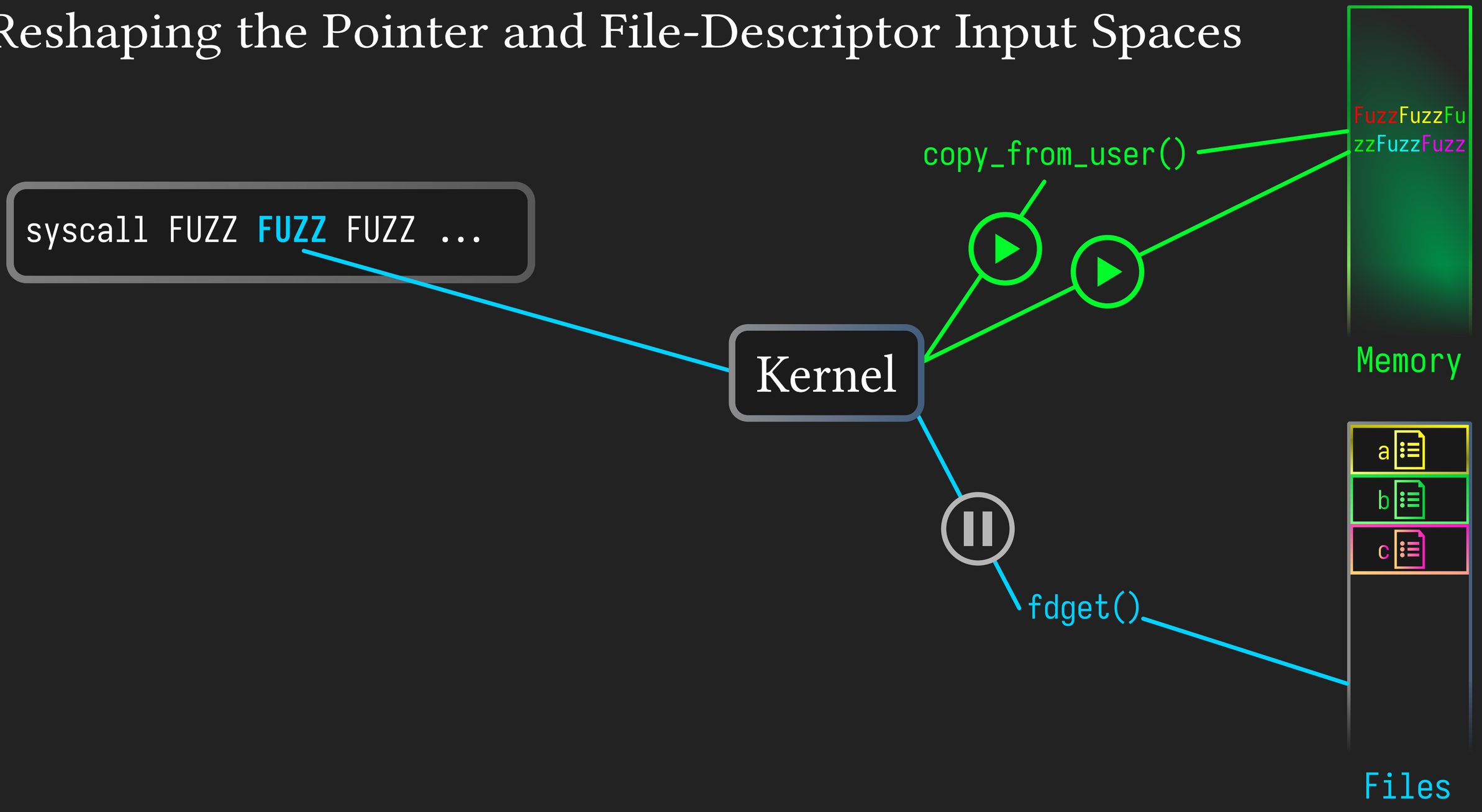
Reshaping the Pointer and File-Descriptor Input Spaces



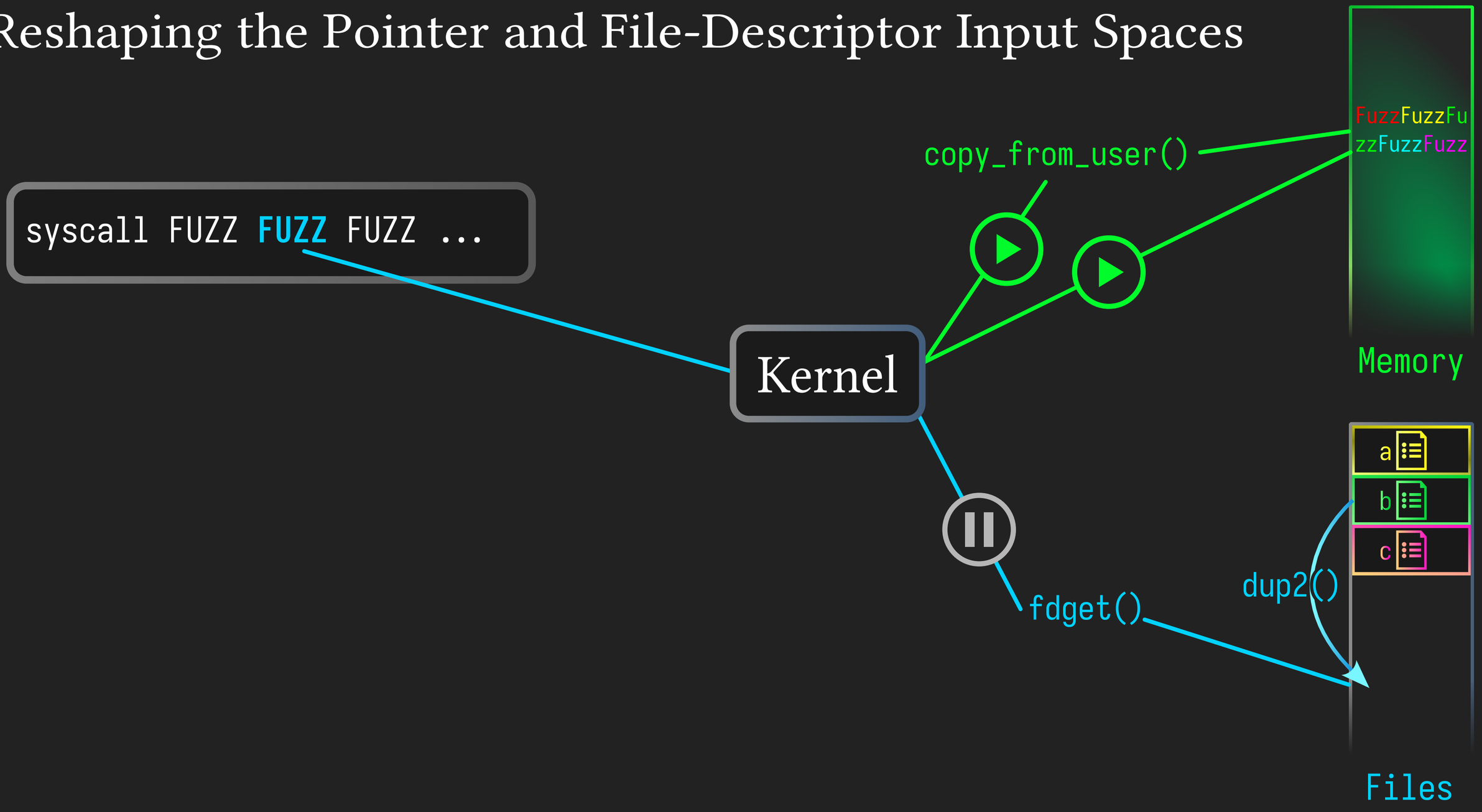
Reshaping the Pointer and File-Descriptor Input Spaces



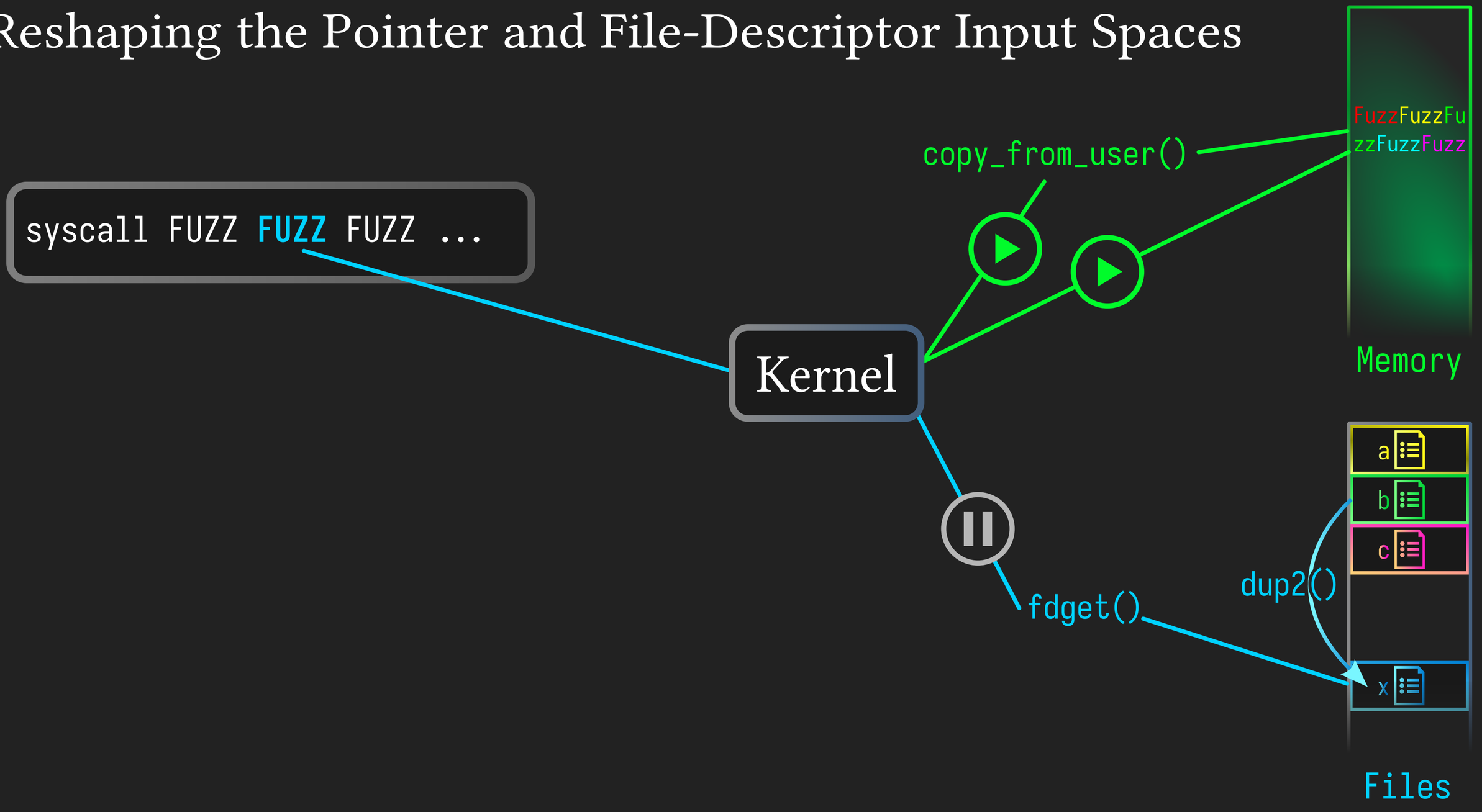
Reshaping the Pointer and File-Descriptor Input Spaces



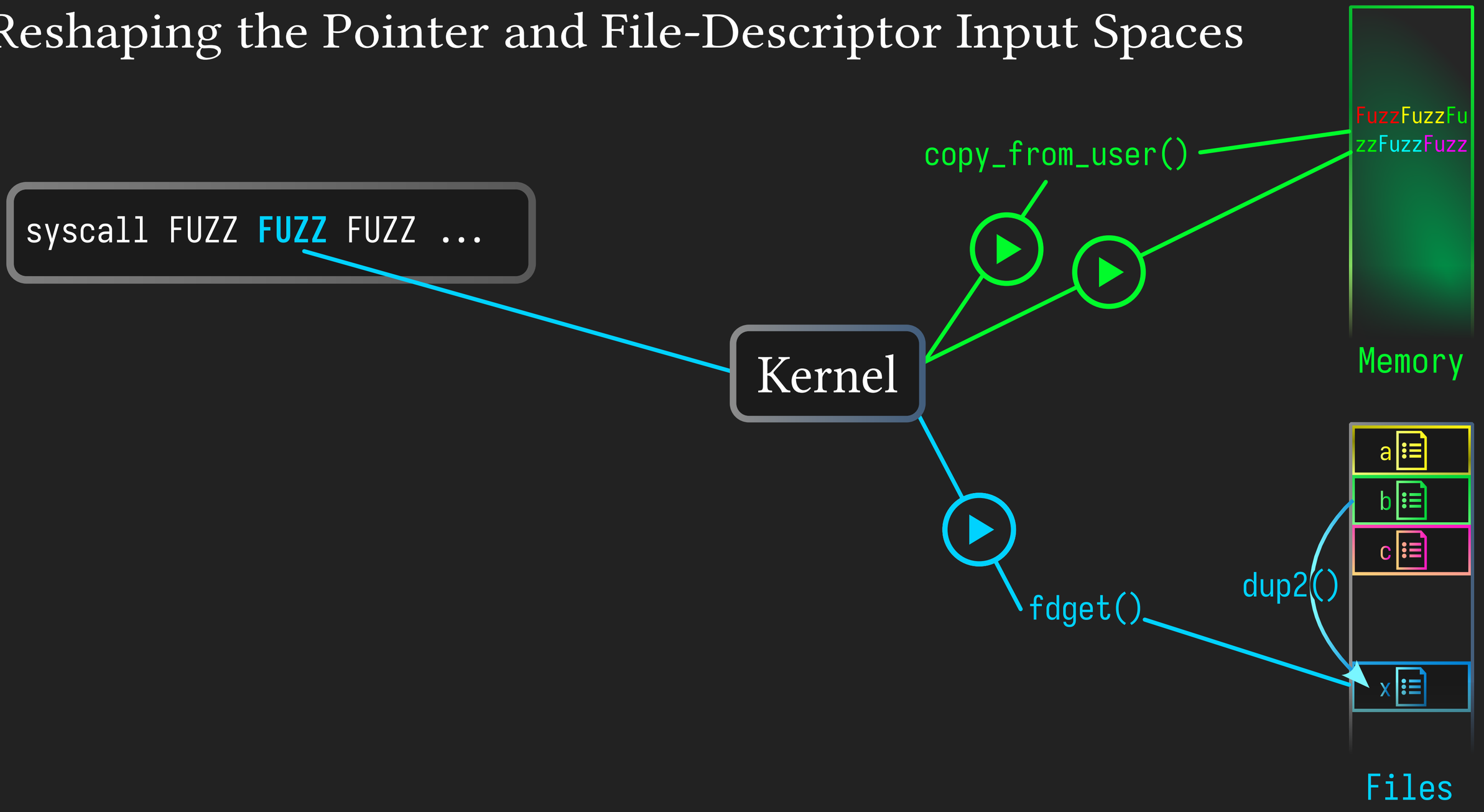
Reshaping the Pointer and File-Descriptor Input Spaces



Reshaping the Pointer and File-Descriptor Input Spaces

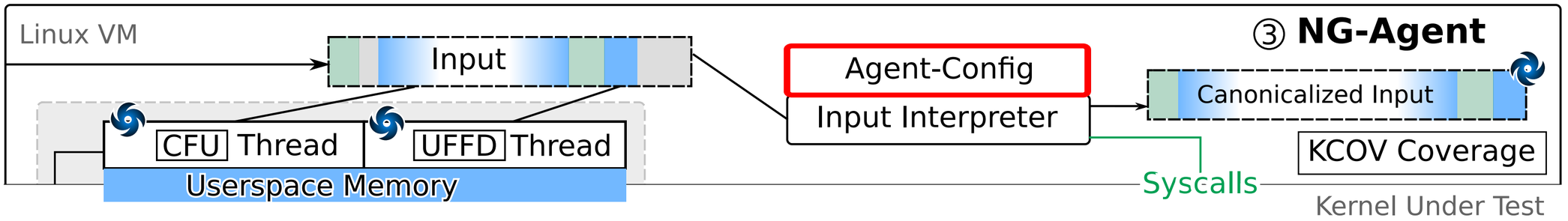


Reshaping the Pointer and File-Descriptor Input Spaces



Config Setup

```
files = "/dev/kvm", O_RDWR
ioctl[-1, -1, -1]
mmap[0, 0xF000, PROT_READ | PROT_WRITE, MAP_SHARED|MAP_POPULATE, 0xFFFF, -1]
close[-1]
fstat[-1, -1]
read[-1, -1, 0xFFFF]
write[-1, -1, 0xFFFF]
```



NG-Agent

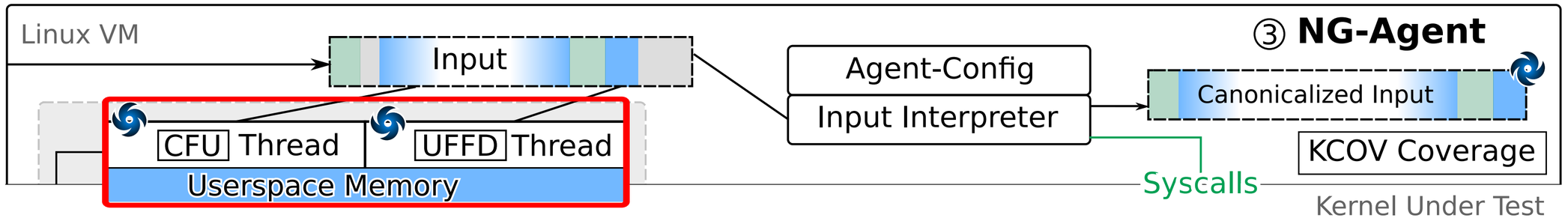
```
files = "/dev/kvm", O_RDWR
ioctl[-1, -1, -1]
mmap[0, 0xF000, PROT_READ | PROT_WRITE, MAP_SHARED|MAP_POPULATE, 0xFFFF, -1]
close[-1]
fstat[-1, -1]
read[-1, -1, 0xFFFF]
write[-1, -1, 0xFFFF]
```

NG-Agent

Setup

Inflate Memory

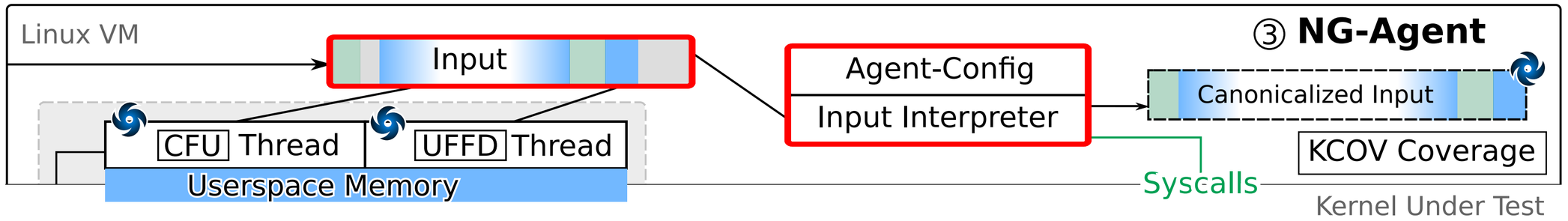
Start up threads to fill memory-accesses



NG-Agent

Interpreter

```
files = "/dev/kvm", O_RDWR
ioctl[-1, -1, -1]
mmap[0, 0xF000, PROT_READ | PROT_WRITE, MAP_SHARED|MAP_POPULATE, 0xFFFF, -1]
close[-1]
fstat[-1, -1]
read[-1, -1, 0xFFFF]
write[-1, -1, 0xFFFF]
```

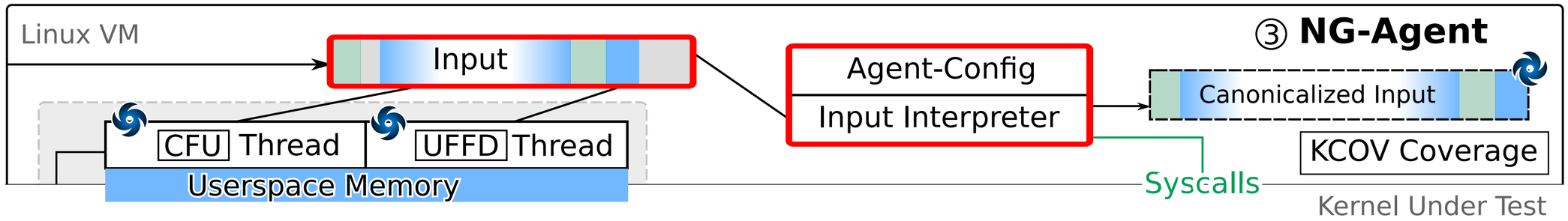


NG-Agent

Interpreter

```
00110010100010000100110101001101101111111111001100101000 1...
```

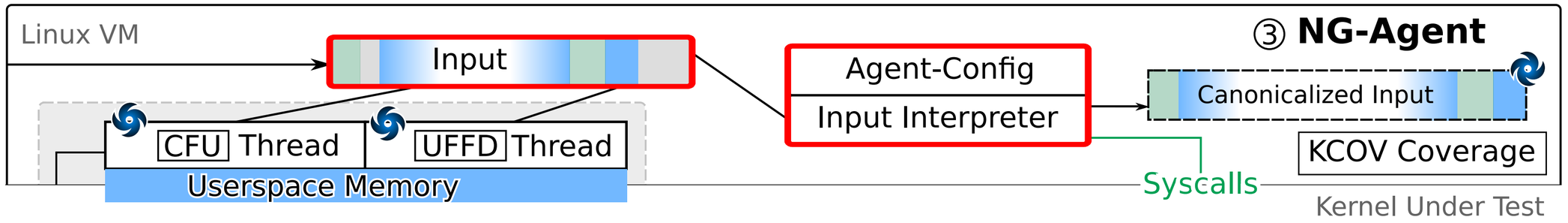
```
files = "/dev/kvm", O_RDWR
ioctl[-1, -1, -1]
mmap[0, 0xF000, PROT_READ | PROT_WRITE, MAP_SHARED|MAP_POPULATE, 0xFFFF, -1]
close[-1]
fstat[-1, -1]
read[-1, -1, 0xFFFF]
write[-1, -1, 0xFFFF]
```



NG-Agent

Interpreter

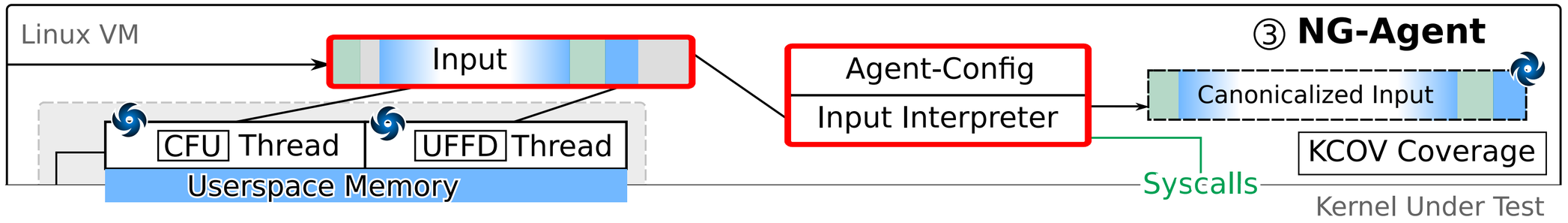
```
files = "/dev/kvm", O_RDWR
ioctl[-1, -1, -1]
mmap[0, 0xF000, PROT_READ | PROT_WRITE, MAP_SHARED|MAP_POPULATE, 0xFFFF, -1]
close[-1]
fstat[-1, -1]
read[-1, -1, 0xFFFF]
write[-1, -1, 0xFFFF]
```



NG-Agent

Interpreter

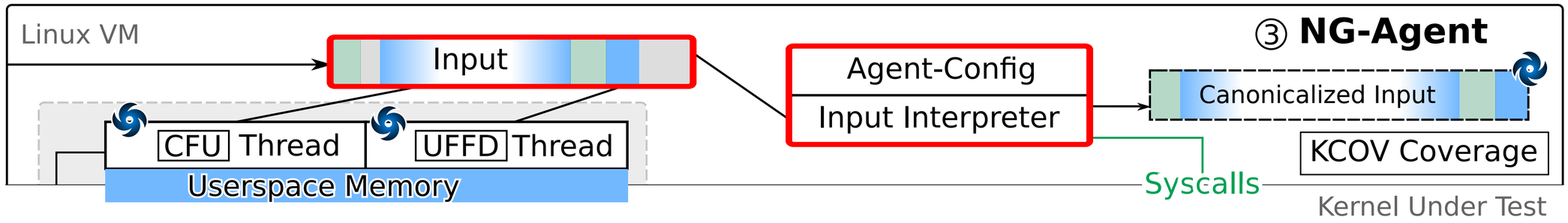
```
files = "/dev/kvm", O_RDWR
ioctl[-1, -1, -1]
mmap[0, 0xF000, PROT_READ | PROT_WRITE, MAP_SHARED|MAP_POPULATE, 0xFFFF, -1]
close[-1]
fstat[-1, -1]
read[-1, -1, 0xFFFF]
write[-1, -1, 0xFFFF]
```



NG-Agent

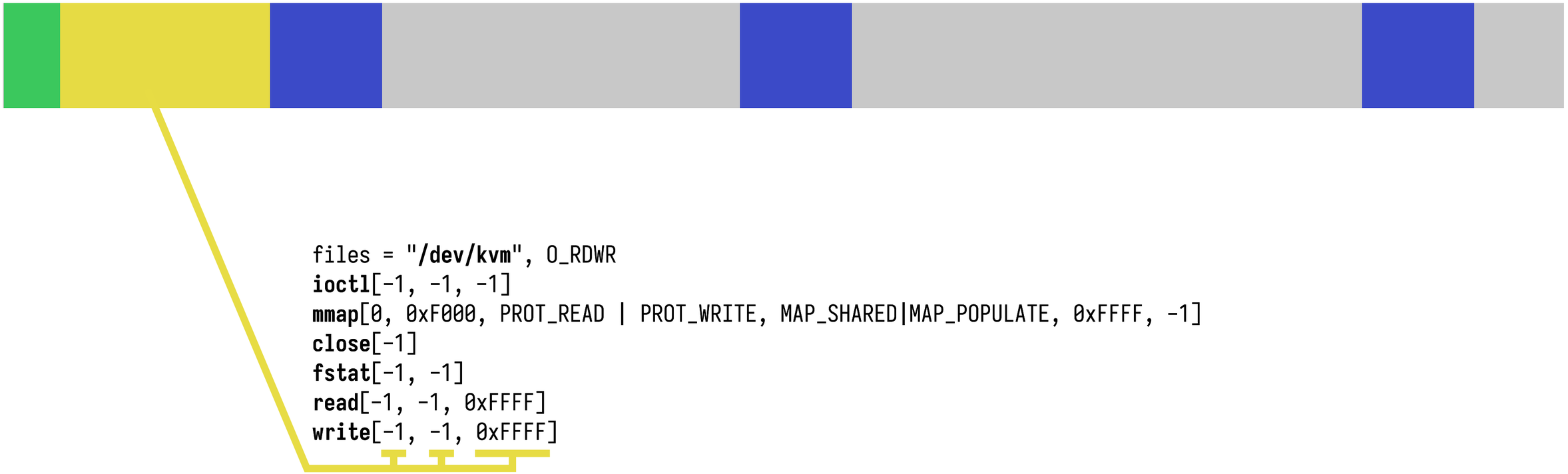
Interpreter

```
files = "/dev/kvm", O_RDWR
ioctl[-1, -1, -1]
mmap[0, 0xF000, PROT_READ | PROT_WRITE, MAP_SHARED|MAP_POPULATE, 0xFFFF, -1]
close[-1]
fstat[-1, -1]
read[-1, -1, 0xFFFF]
write[-1, -1, 0xFFFF]
```

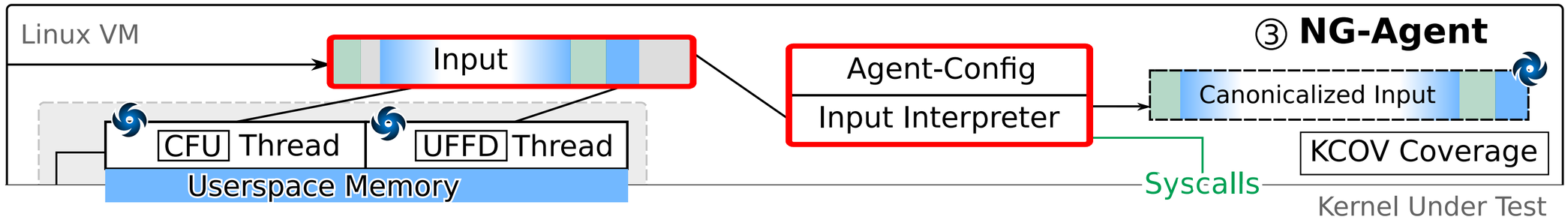


NG-Agent

Interpreter



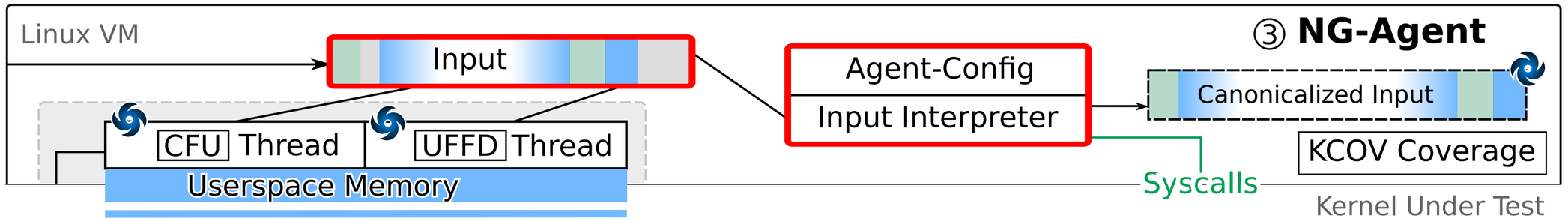
```
files = "/dev/kvm", O_RDWR
ioctl[-1, -1, -1]
mmap[0, 0xF000, PROT_READ | PROT_WRITE, MAP_SHARED|MAP_POPULATE, 0xFFFF, -1]
close[-1]
fstat[-1, -1]
read[-1, -1, 0xFFFF]
write[-1, -1, 0xFFFF]
```



NG-Agent

Interpreter

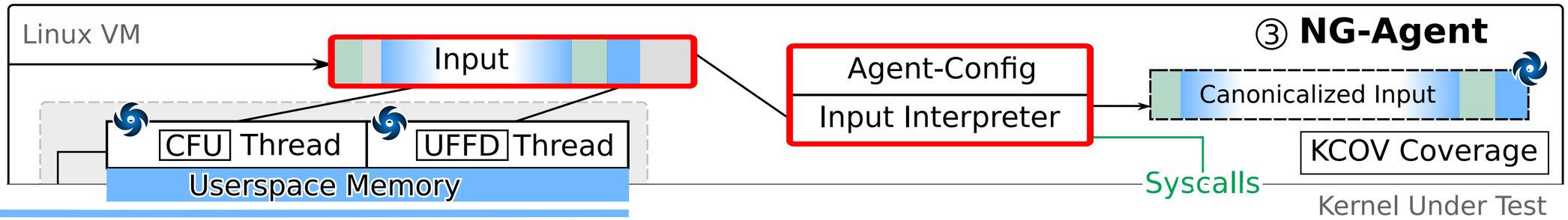
```
files = "/dev/kvm", O_RDWR
ioctl[-1, -1, -1]
mmap[0, 0xF000, PROT_READ | PROT_WRITE, MAP_SHARED|MAP_POPULATE, 0xFFFF, -1]
close[-1]
fstat[-1, -1]
read[-1, -1, 0xFFFF]
write[-1, -1, 0xFFFF]
```



NG-Agent

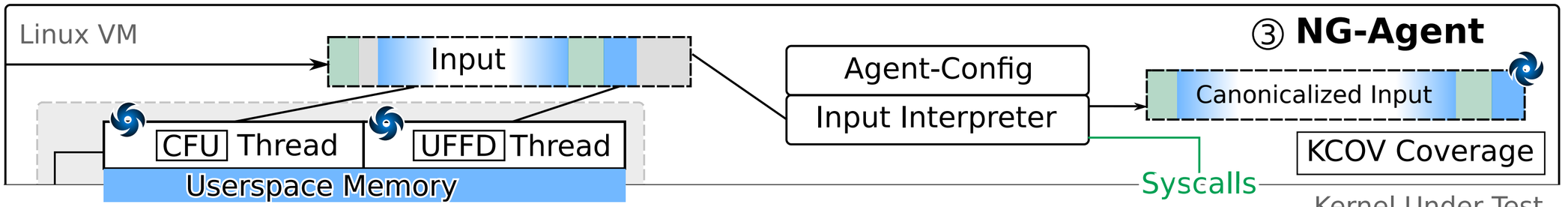
Interpreter

```
files = "/dev/kvm", O_RDWR
ioctl[-1, -1, -1]
mmap[0, 0xF000, PROT_READ | PROT_WRITE, MAP_SHARED|MAP_POPULATE, 0xFFFF, -1]
close[-1]
fstat[-1, -1]
read[-1, -1, 0xFFFF]
write[-1, -1, 0xFFFF]
```



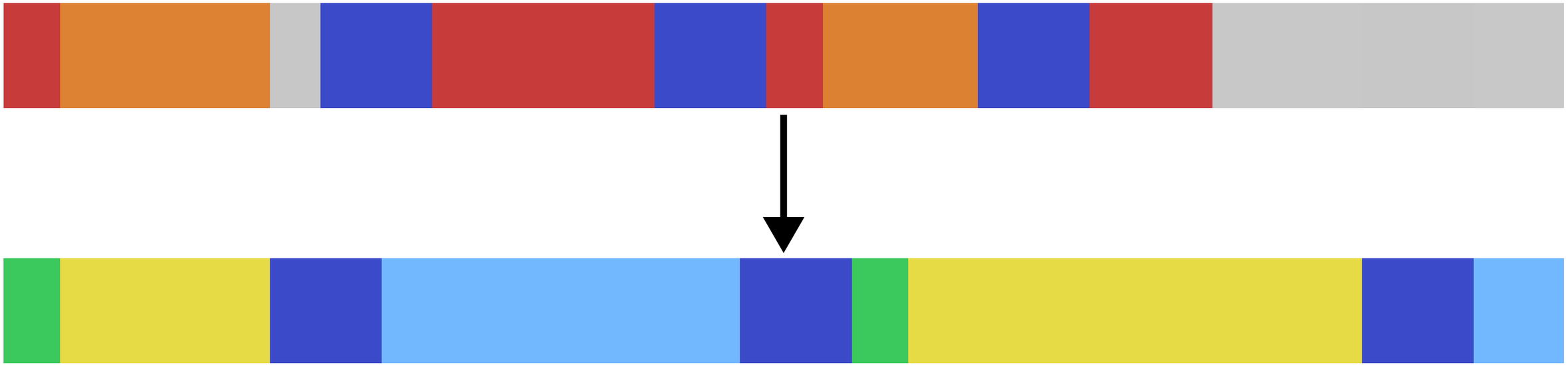
NG-Agent

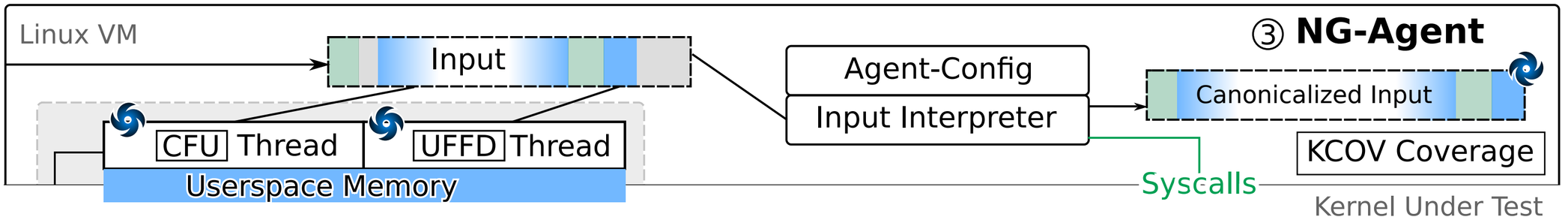
"Canonicalization"



NG-Agent

"Canonicalization"





Results

- Linux 5.12

Results

- Linux 5.12
- 13 Components

```
bpf
video4linux
rdma
binder
cdrom
kvm
vhost_net
drm
io_uring
vt_ioctl
ptmx
rdma
vhost
```

Results

- Linux 5.12
- 13 Components
- Coverage
  - Fuzzed for 7 Days on 20 Cores
  - 102.5% of Syzkaller's Coverage
  - Configurations <1.7% of Syzkaller's

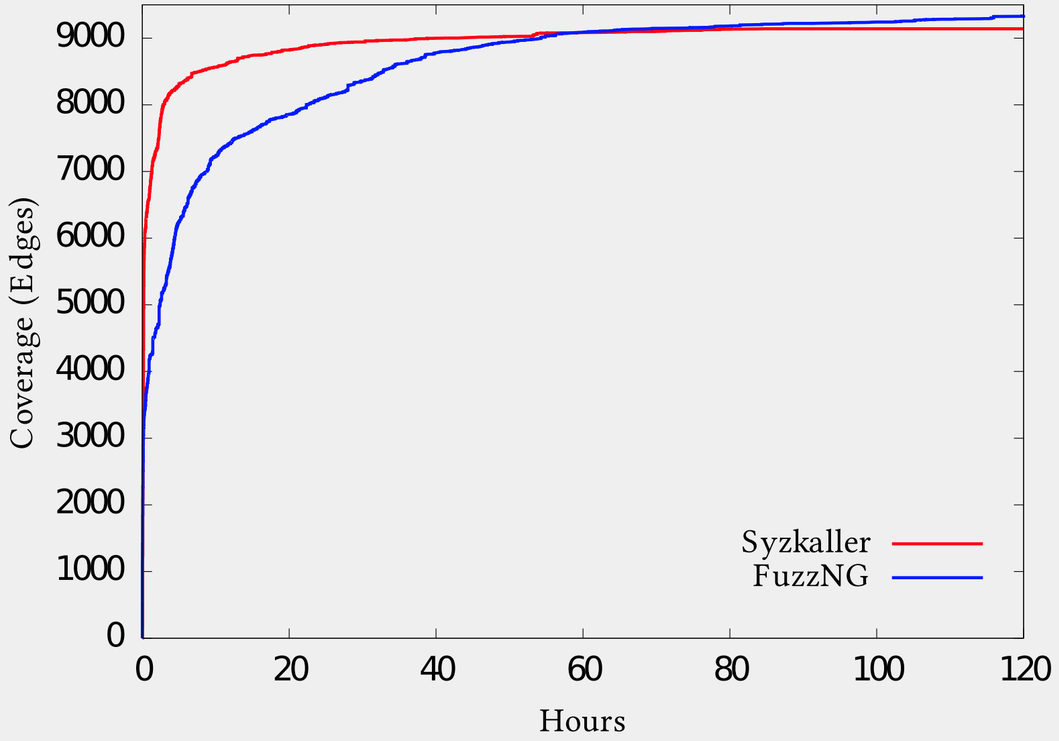
```
bpf
video4linux
rdma
binder
cdrom
kvm
vhost_net
drm
io_uring
vt_ioctl
ptmx
rdma
vhost
```

Results

- Linux 5.12
- 13 Components
- Coverage
  - Fuzzed for 7 Days on 20 Cores
  - 102.5% of Syzkaller's Coverage
  - Configurations <1.7% of Syzkaller's
- Bugs
  - 9 New bugs
  - 5 in components already fuzzed by syzkaller
  - Found **bugs in lines covered by syzkaller**

```
bpf
video4linux
rdma
binder
cdrom
kvm
vhost_net
drm
io_uring
vt_ioctl
ptmx
rdma
vhost
```

Results



Component	Max Cov	Syzkaller Edge Count	Syzkaller Syzlang LoC	FUZZNG Edge Count	FUZZNG Config LoC
bpf	15359	3623	864	3572	1
video4linux	1004	563	381	567	4
rdma	4014	562	1474	591	5
binder	2506	340	272	344	6
cdrom	956	138	351	144	5
kvm	34924	9213	891	9468	7
vhost_net	415	218	157	225	9
drm	12503	2296	745	2138	7
io_uring	3413	982	343	1003	6
vt_ioctl	332	142	381	162	9
Average				**102.53%**	**1.67%**
Geo. Mean				**102.41%**	**1.09%**

FuzzNG

Bend the system-call input space to make it conducive to fuzzing

FuzzNG

Bend the system-call input space to make it conducive to fuzzing

Use time-tested **off-the-shelf fuzzers**

FuzzNG

Bend the system-call input space to make it conducive to fuzzing

Use time-tested **off-the-shelf fuzzers**

Competitive fuzzing performance with **tiny component configs**

FuzzNG

Bend the system-call input space to make it conducive to fuzzing

Use time-tested **off-the-shelf fuzzers**

Competitive fuzzing performance with **tiny component configs**

FuzzNG

alxndr@bu.edu

Looking for Industry Jobs

BOSTON UNIVERSITY

Red Hat