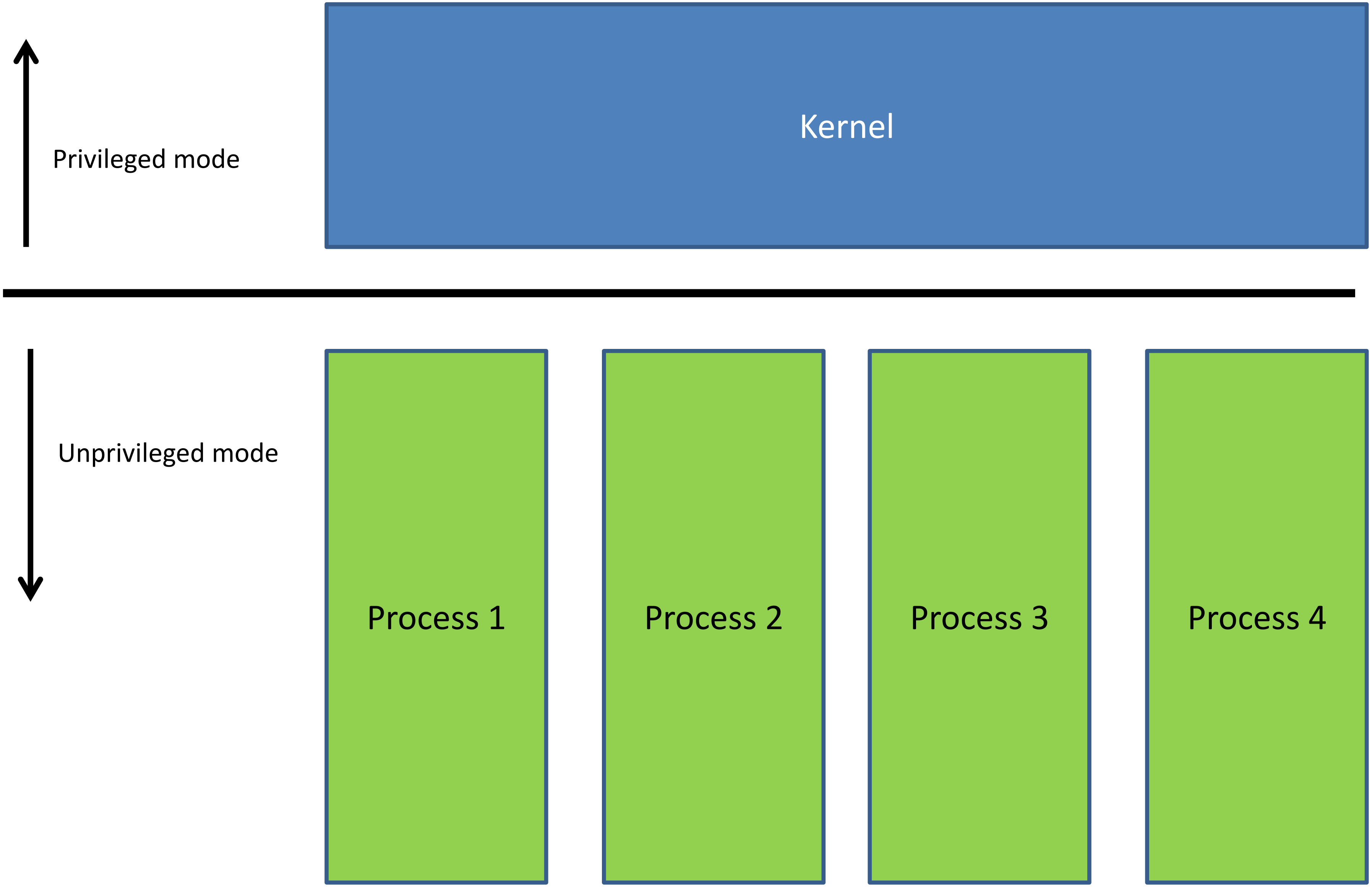


Privileged mode

Kernel

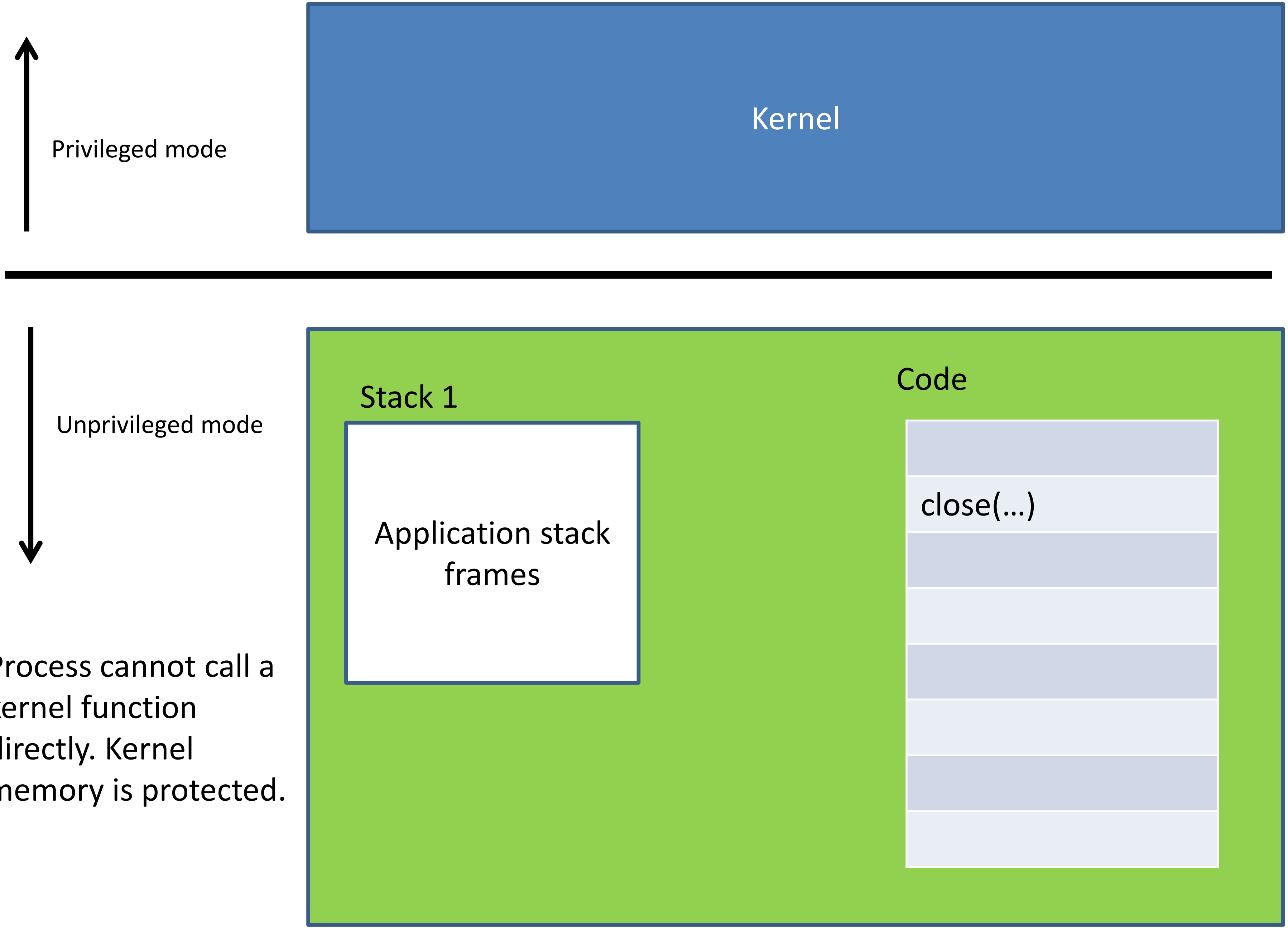
Unprivileged mode

Process 1

Process 2

Process 3

Process 4



Privileged mode

Kernel

Unprivileged mode

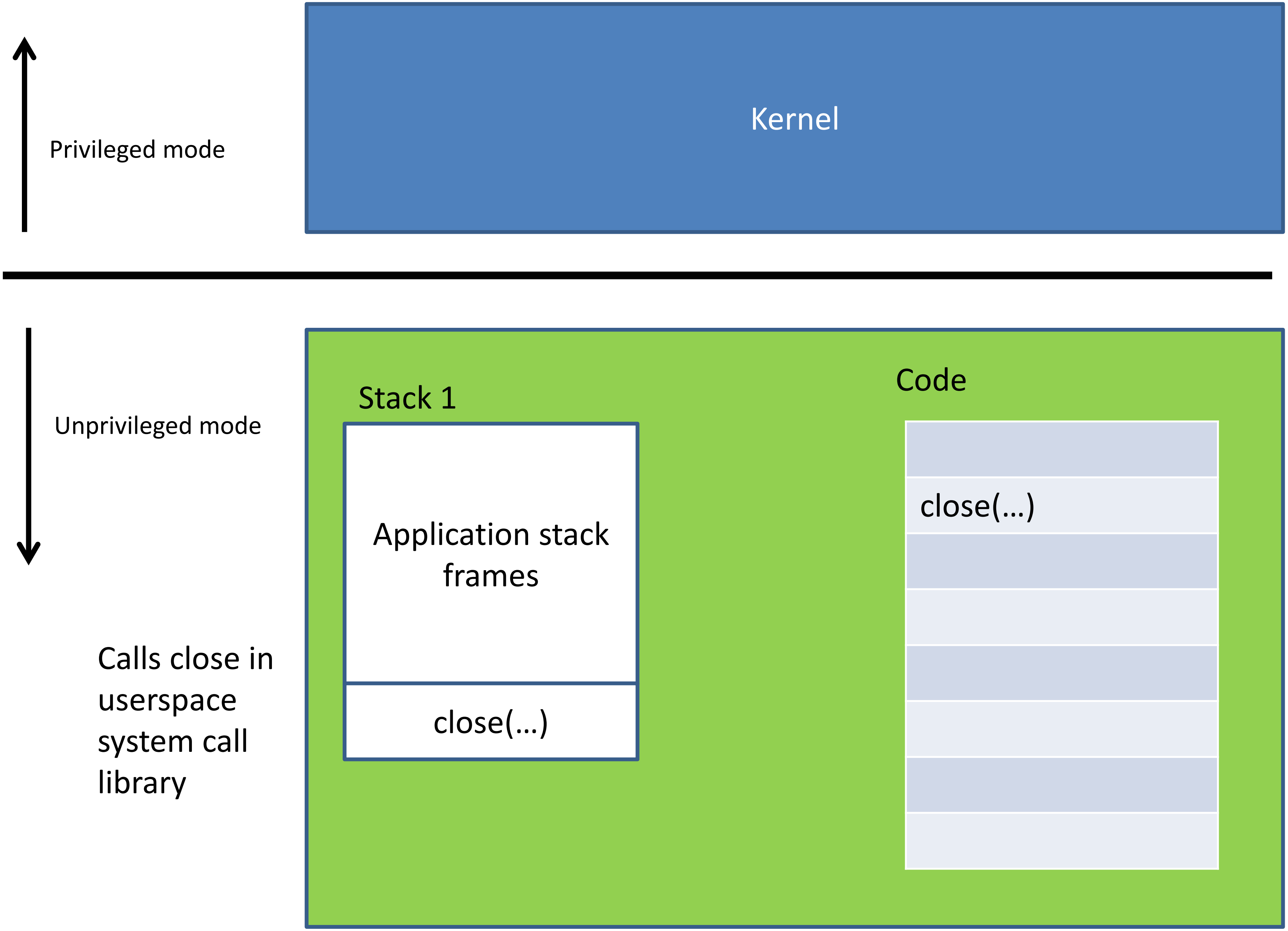
Stack 1

Application stack frames

Code

close(…)

Process cannot call a kernel function directly. Kernel memory is protected.



Kernel

Privileged mode

Unprivileged mode

Stack 1

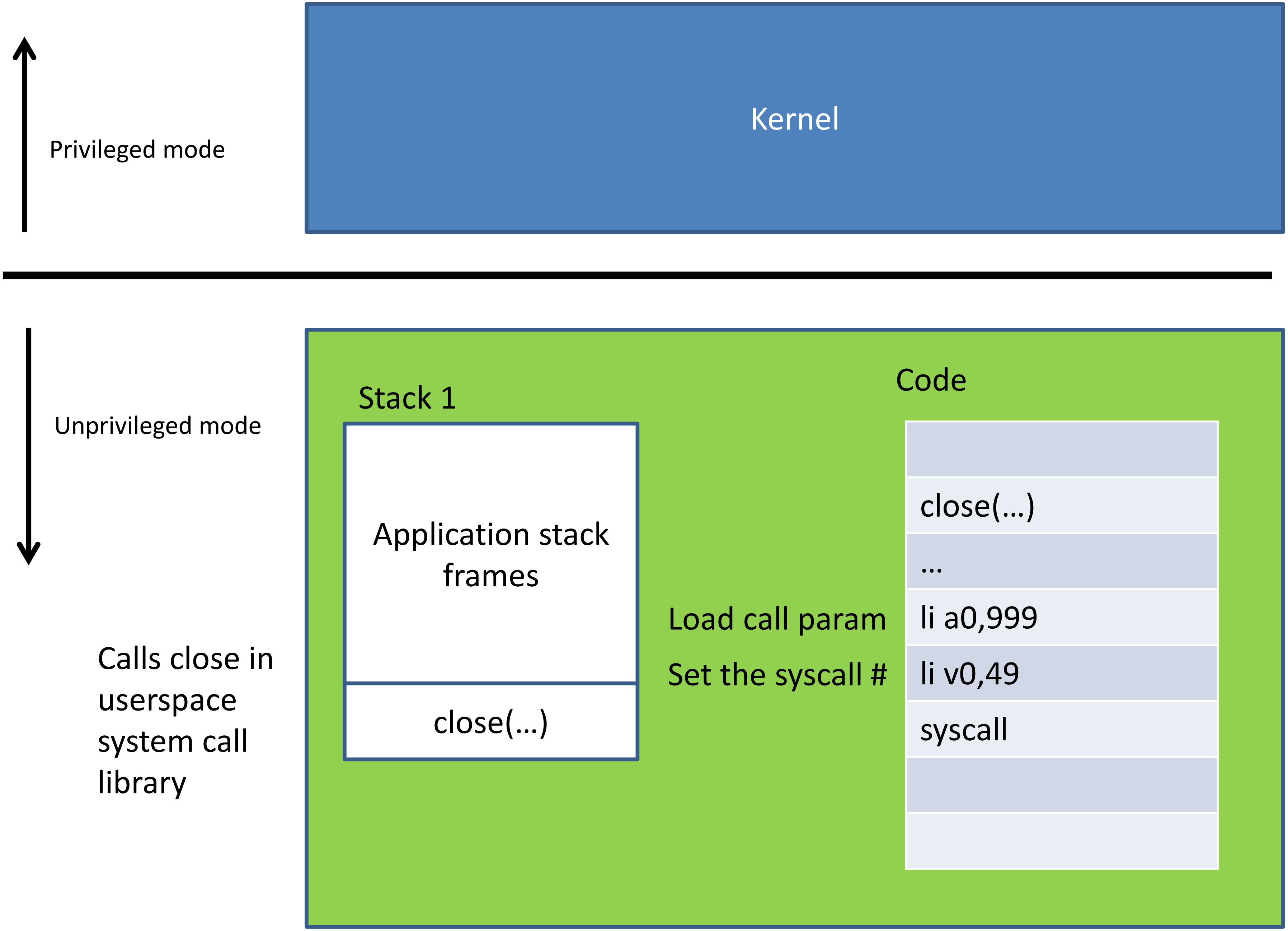
Application stack frames

close(...)

Code

close(...)

Calls close in userspace system call library



Privileged mode

Kernel

Unprivileged mode

Calls close in userspace system call library

Stack 1

Application stack frames

close(…)

Code

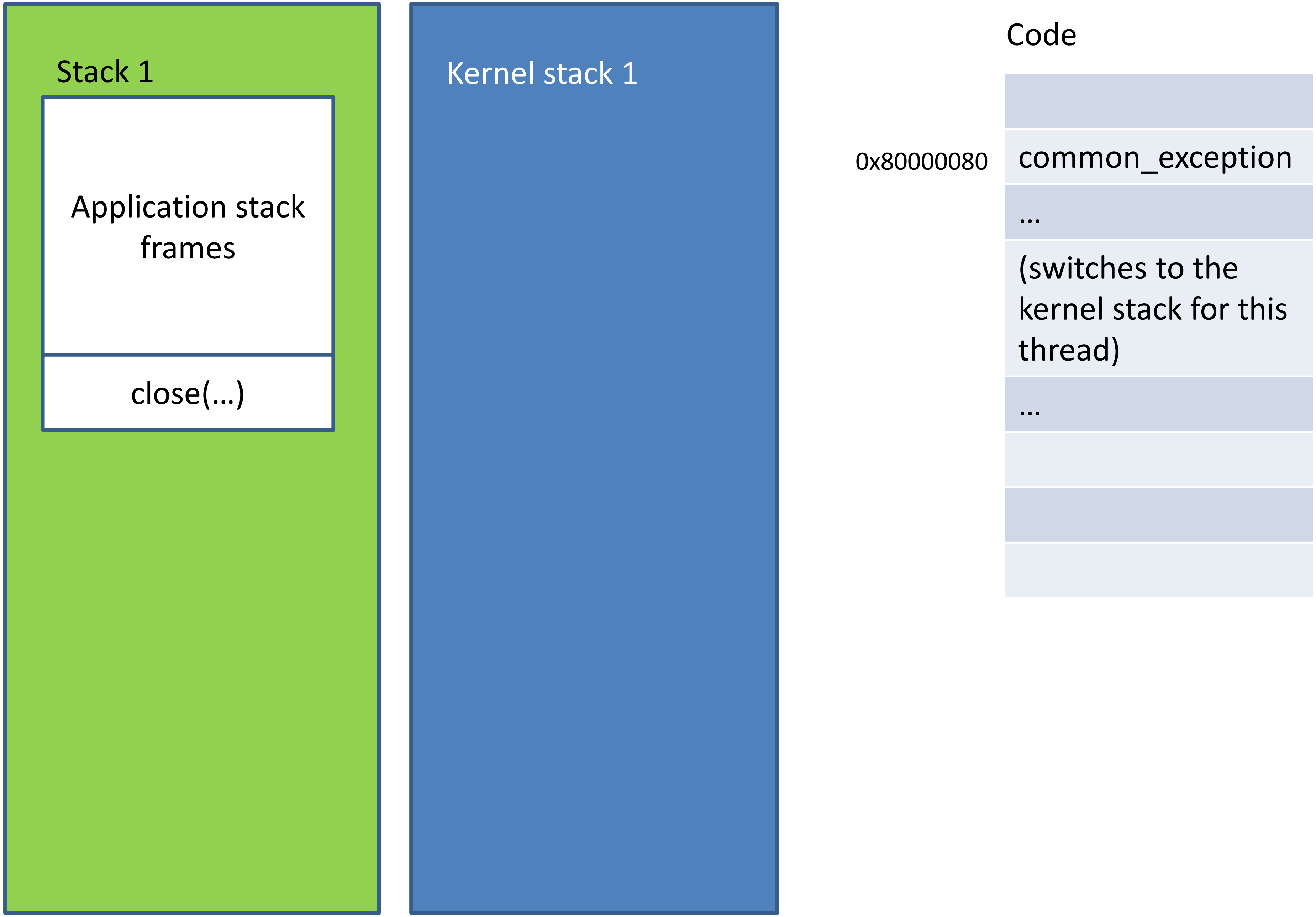
close(…)

…

Load call param li a0,999

Set the syscall # li v0,49

syscall



Stack 1

Application stack frames

close(…)

Kernel stack 1

Code

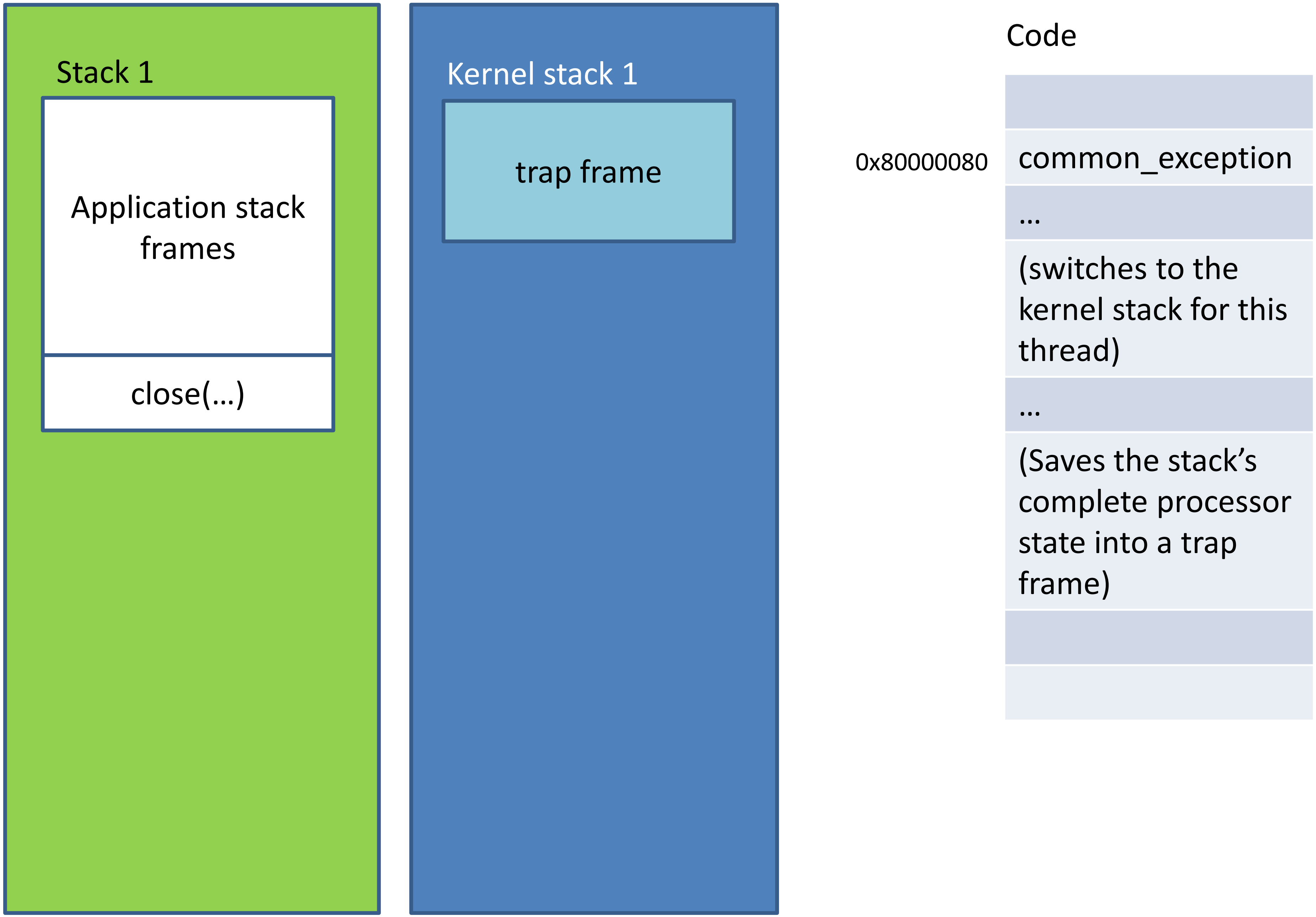
0x80000080

common_exception

…

(switches to the kernel stack for this thread)

…



Stack 1

Application stack frames

close(…)

Kernel stack 1

trap frame

Code

0x80000080

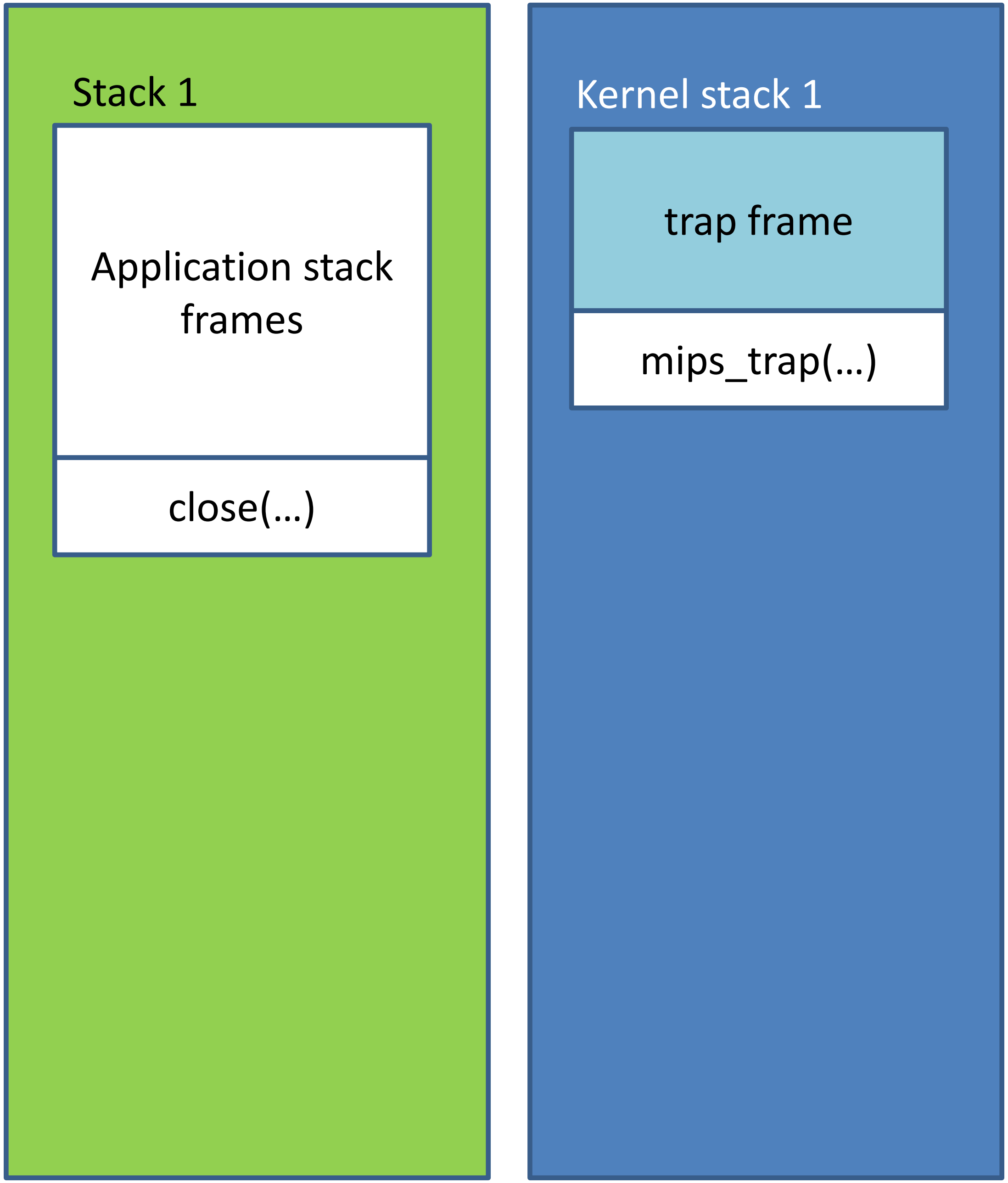
common_exception

…

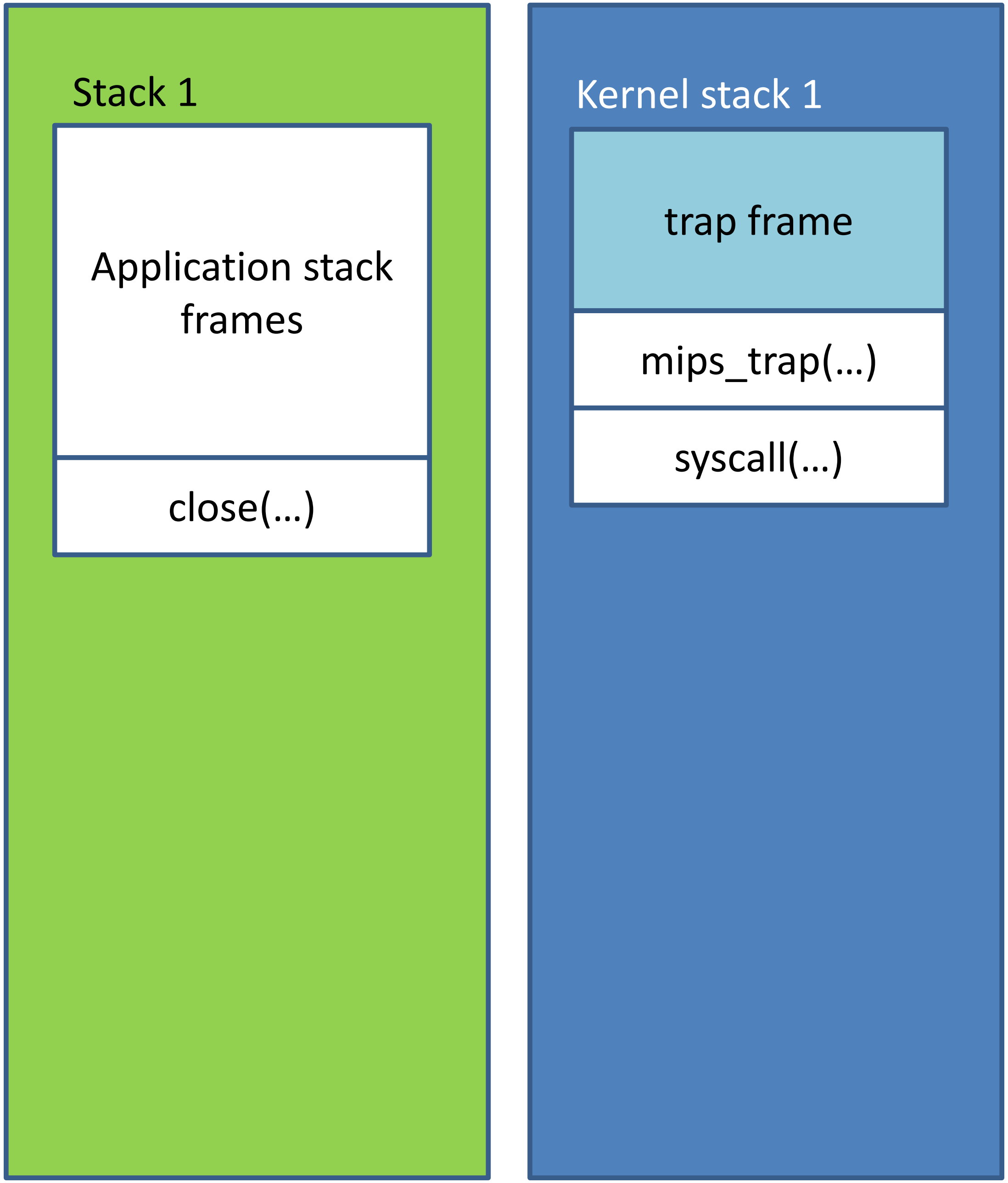
(switches to the kernel stack for this thread)

…

(Saves the stack's complete processor state into a trap frame)



- Check whether this is an exception, interrupt, or system call (all handled by mips_trap).
- If it is not an interrupt, turn interrupts back on.



Stack 1

Application stack frames

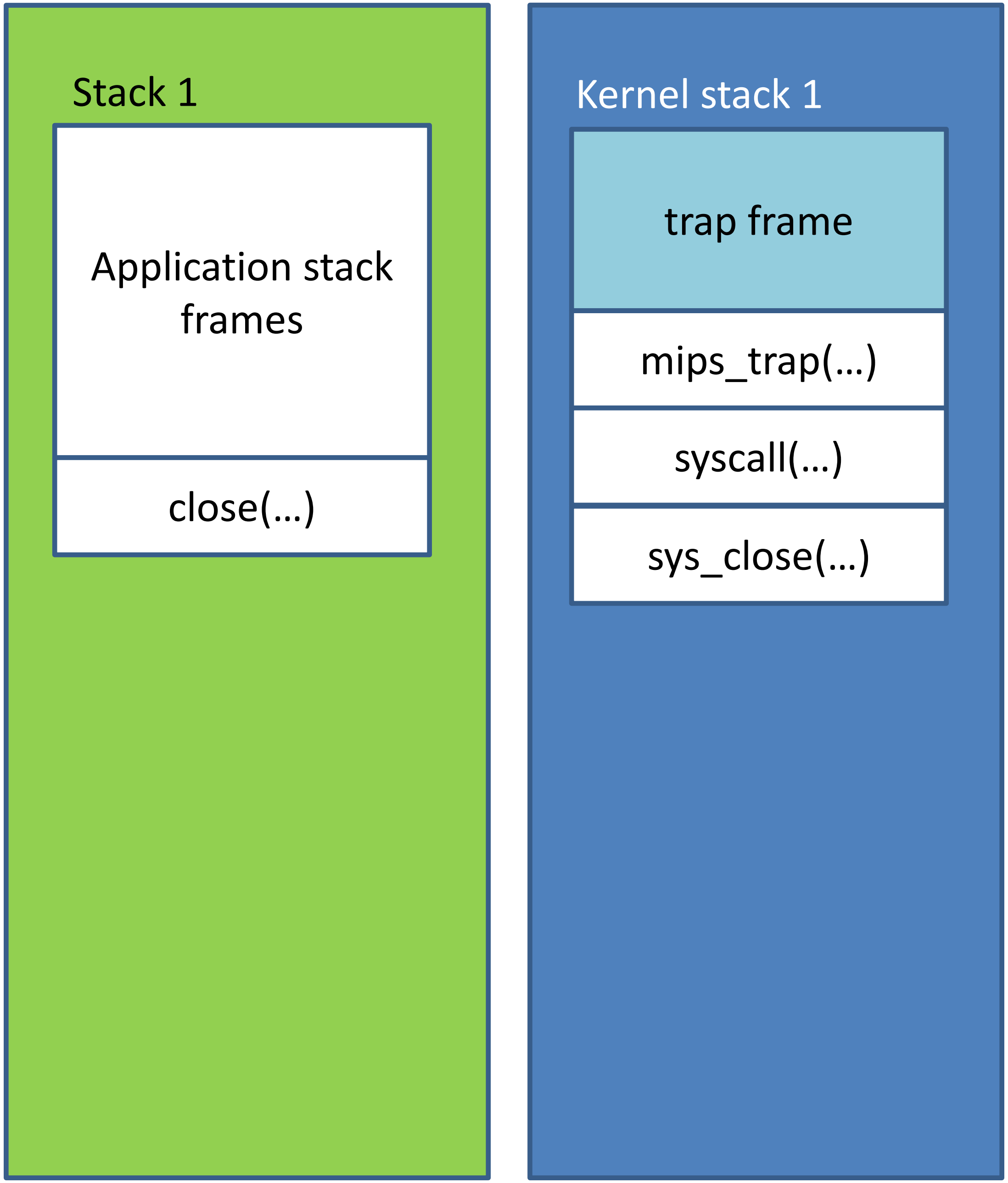
close(…)

Kernel stack 1

trap frame

mips_trap(…)

syscall(…)



Stack 1

Application stack frames

close(…)

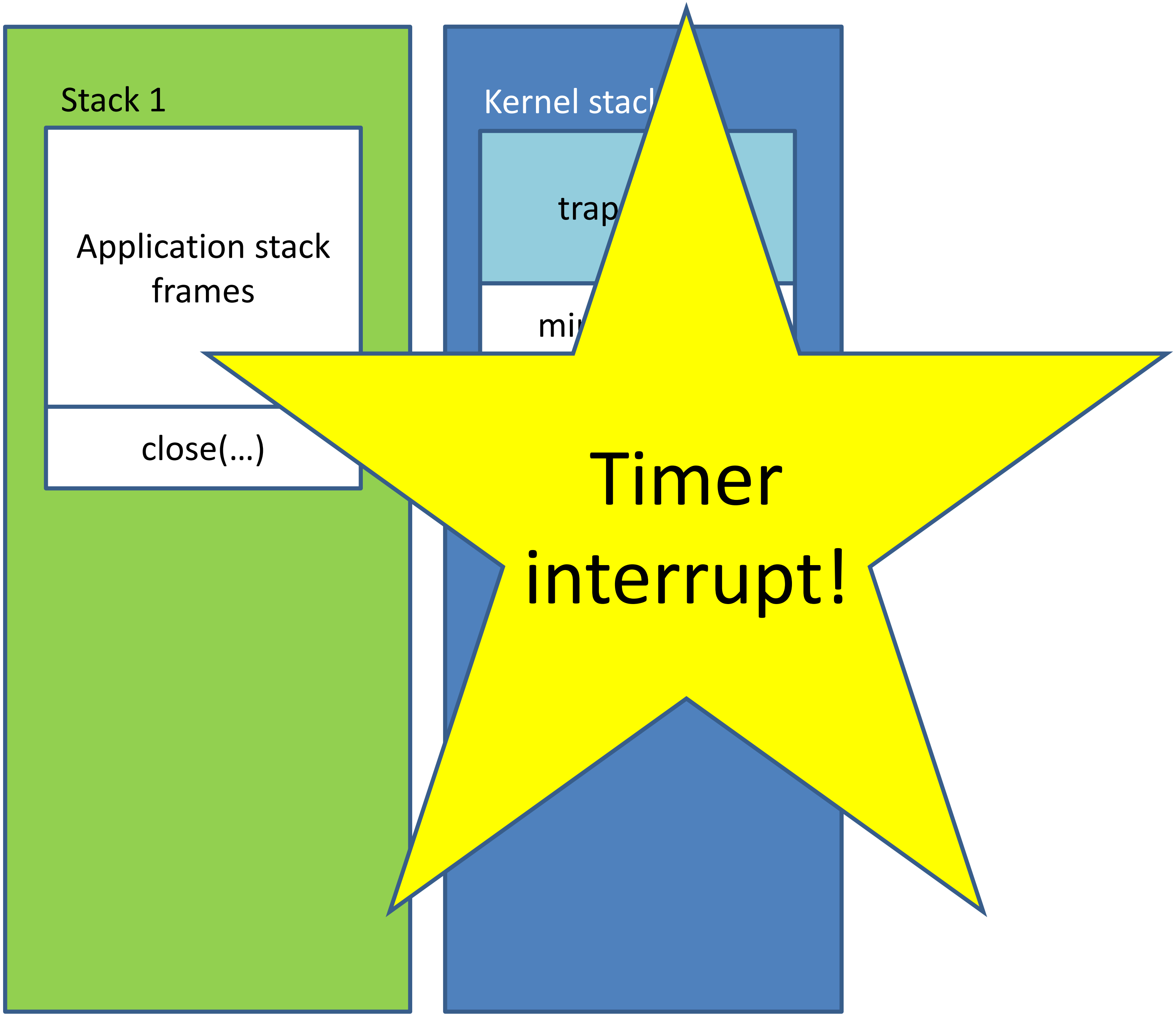
Kernel stack 1

trap frame

mips_trap(…)

syscall(…)

sys_close(…)



Stack 1

Application stack frames

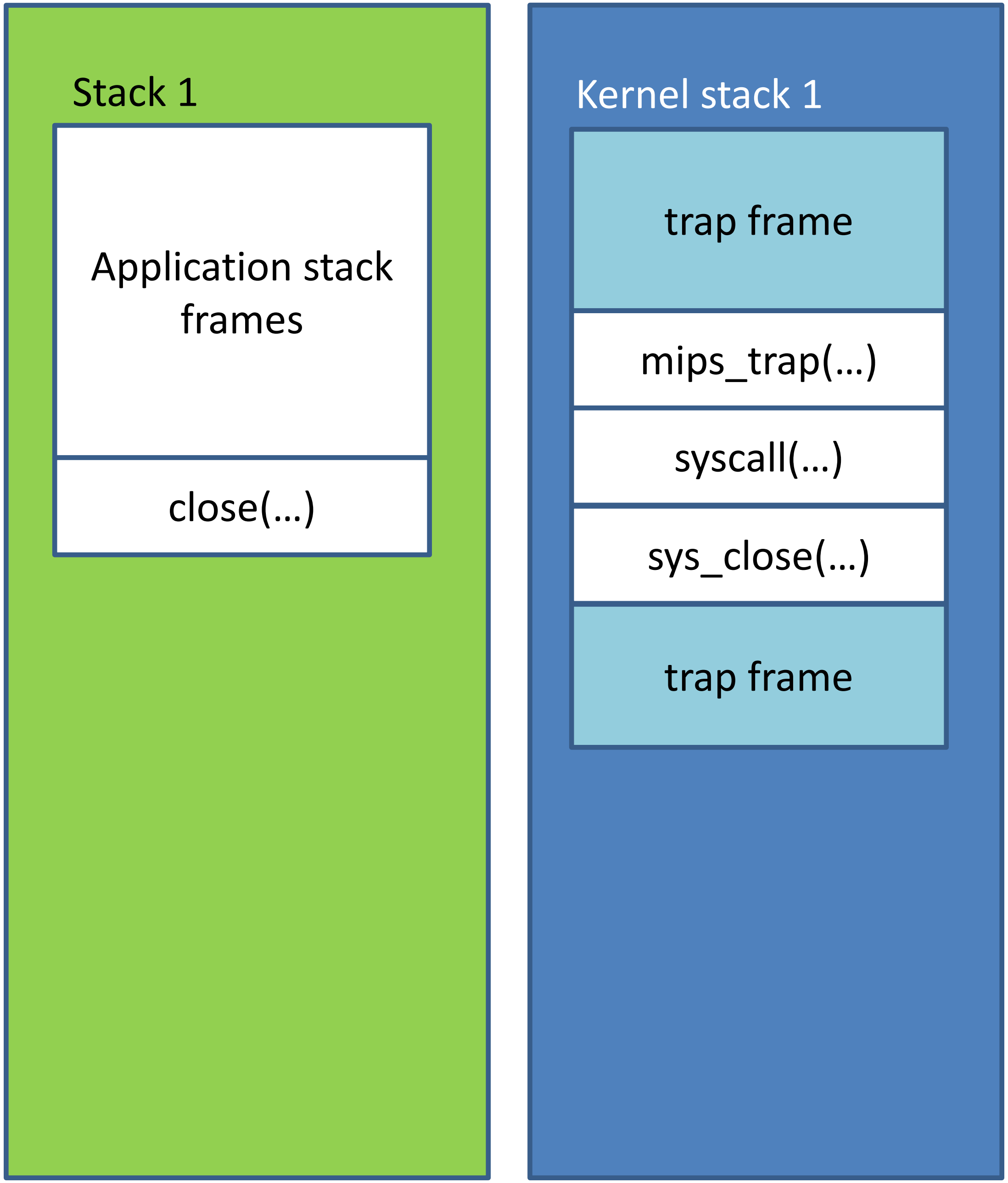
close(…)

Kernel stac

trap

mi

Timer interrupt!



Stack 1

Application stack frames

close(…)

Kernel stack 1

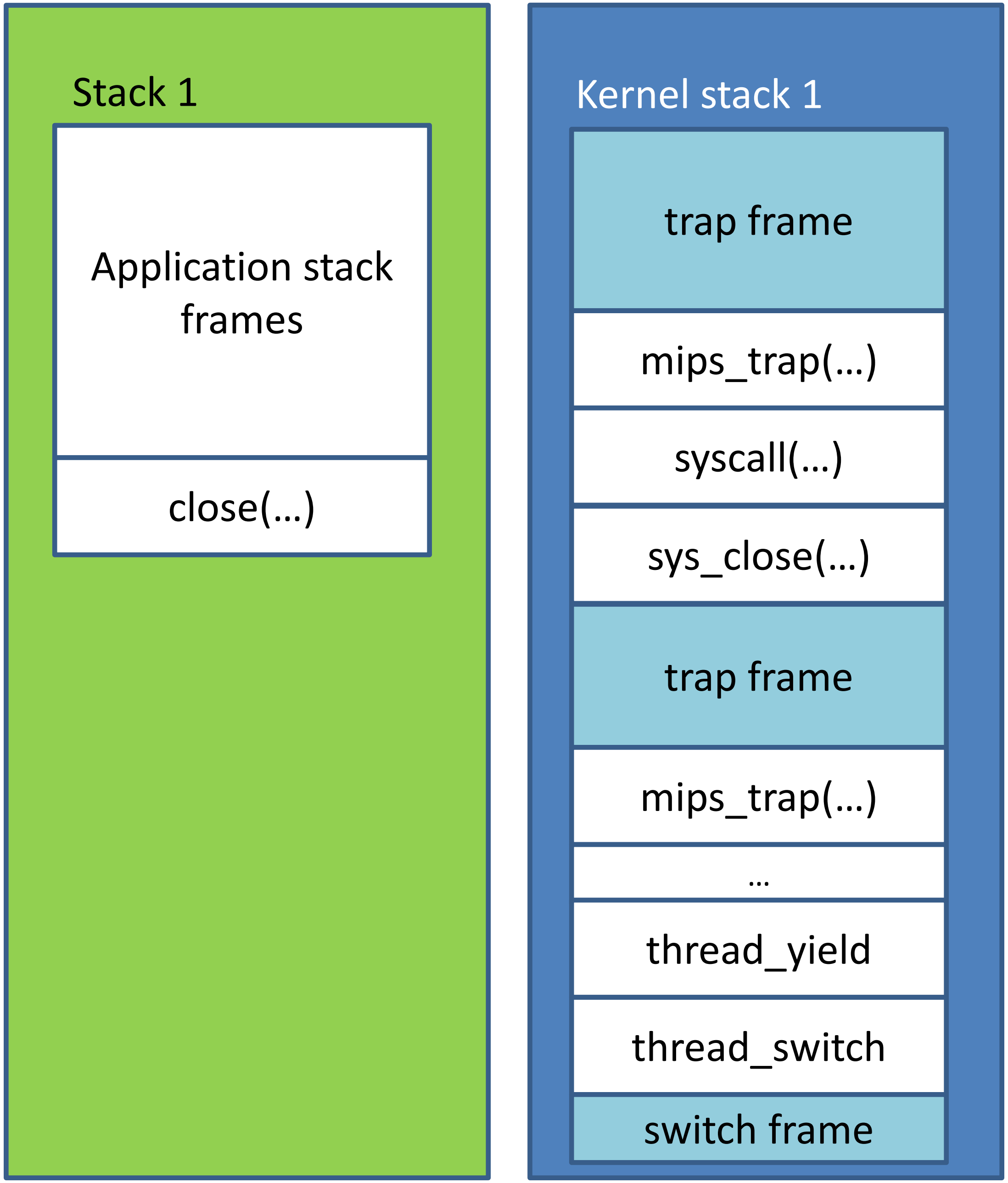
trap frame

mips_trap(…)

syscall(…)

sys_close(…)

trap frame



Stack 1

Application stack frames

close(…)

Kernel stack 1

trap frame

mips_trap(…)

syscall(…)

sys_close(…)

trap frame

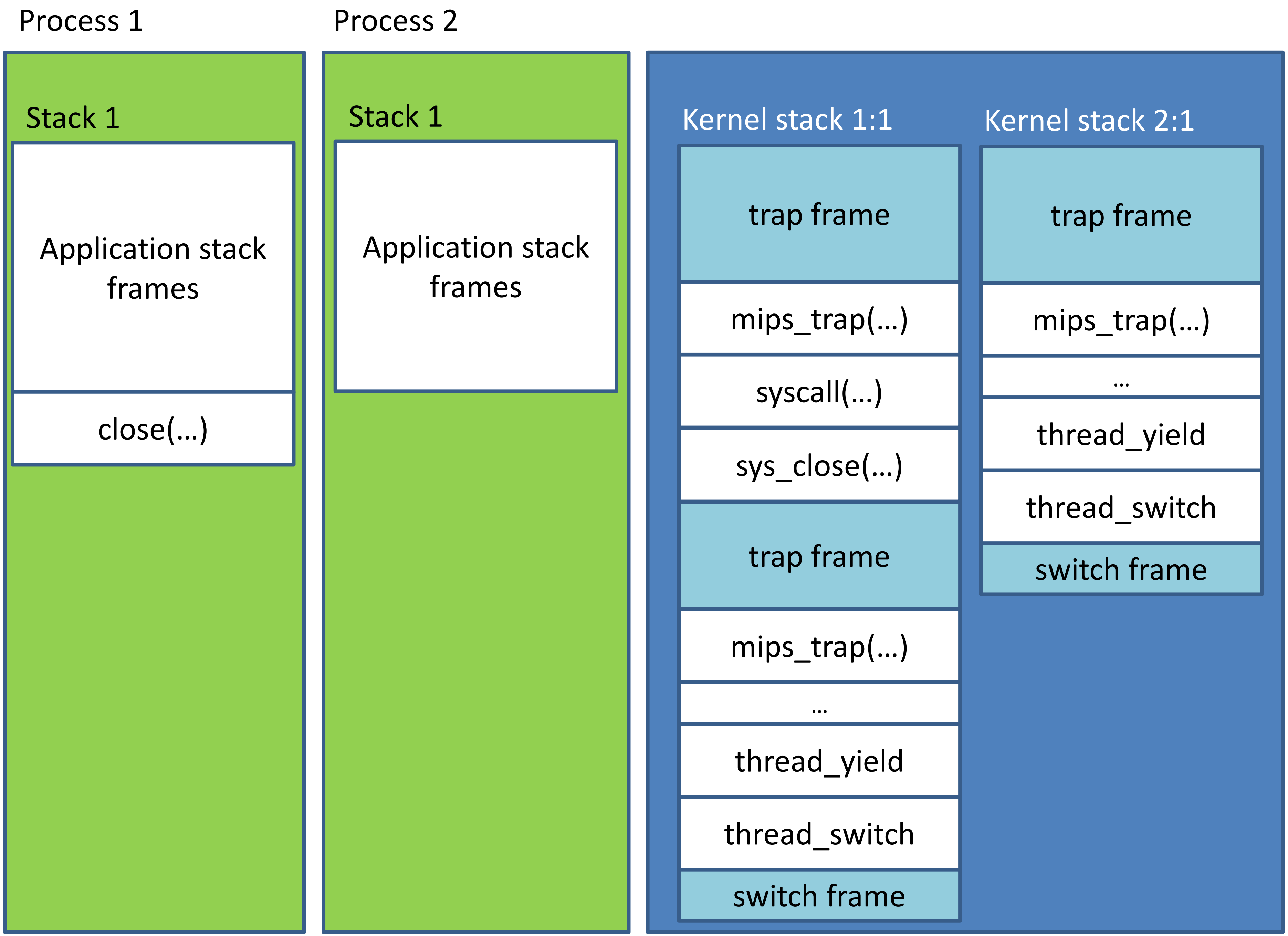
mips_trap(…)

…

thread_yield

thread_switch

switch frame



Process 1

Stack 1

Application stack frames

close(…)

Process 2

Stack 1

Application stack frames

Kernel stack 1:1

trap frame

mips_trap(…)

syscall(…)

sys_close(…)

trap frame

mips_trap(…)

…

thread_yield

thread_switch

switch frame

Kernel stack 2:1

trap frame

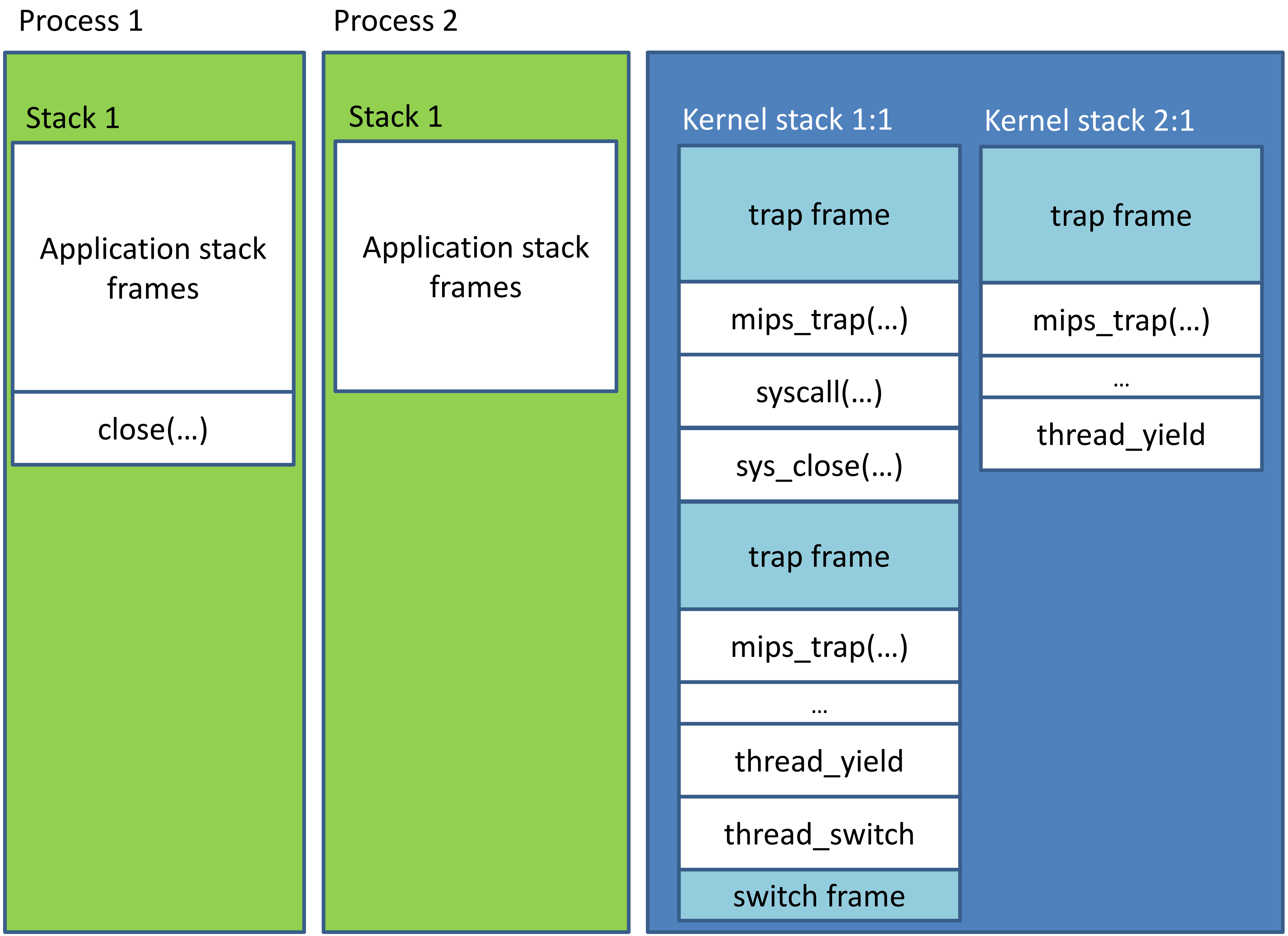
mips_trap(…)

…

thread_yield

thread_switch

switch frame



Process 1

Stack 1

Application stack frames

close(…)

Process 2

Stack 1

Application stack frames

Kernel stack 1:1

trap frame

mips_trap(…)

syscall(…)

sys_close(…)

trap frame

mips_trap(…)

…

thread_yield

thread_switch

switch frame

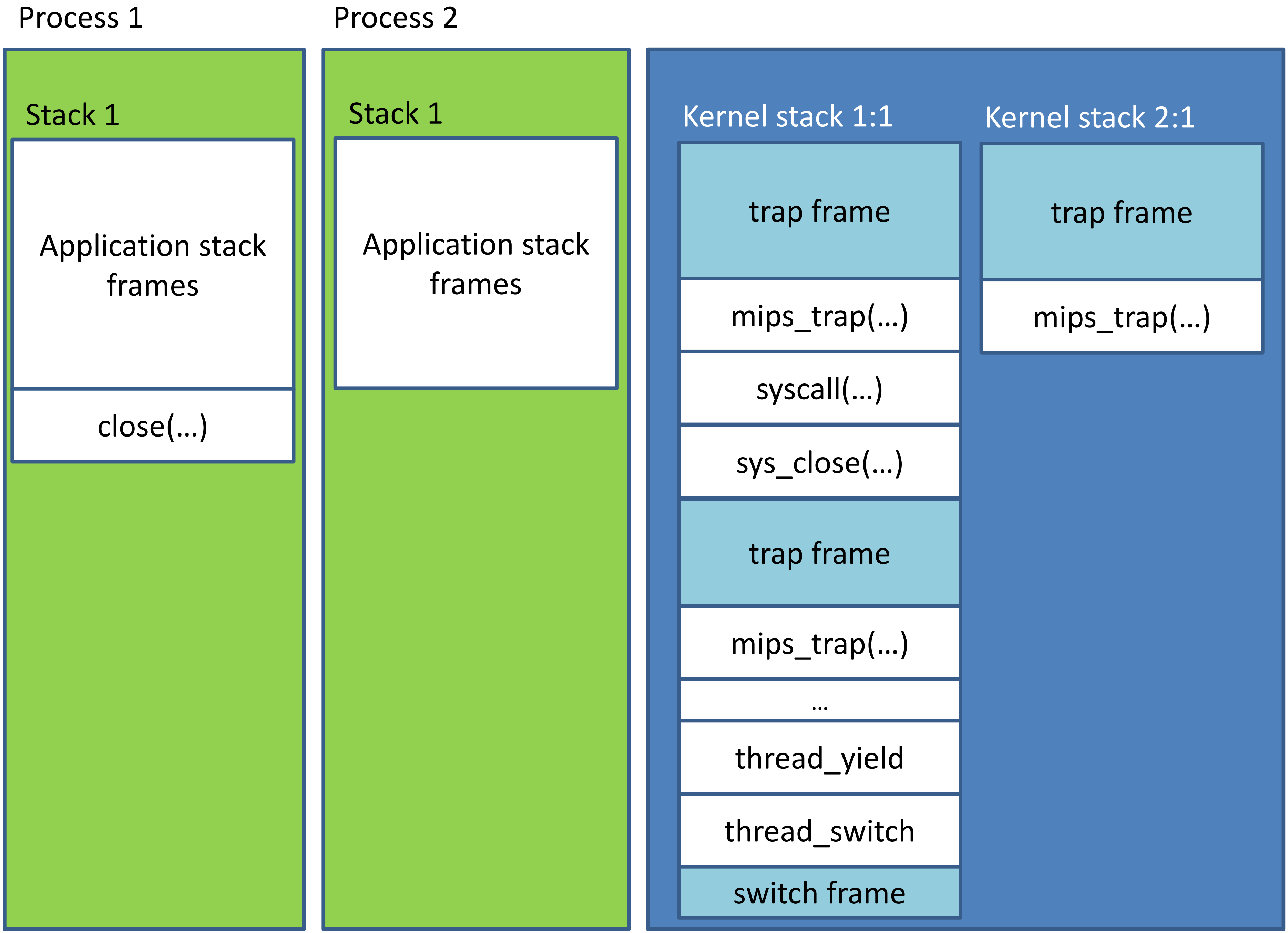
Kernel stack 2:1

trap frame

mips_trap(…)

…

thread_yield



Process 1

Stack 1

Application stack frames

close(…)

Process 2

Stack 1

Application stack frames

Kernel stack 1:1

trap frame

mips_trap(…)

syscall(…)

sys_close(…)

trap frame

mips_trap(…)

…

thread_yield

thread_switch

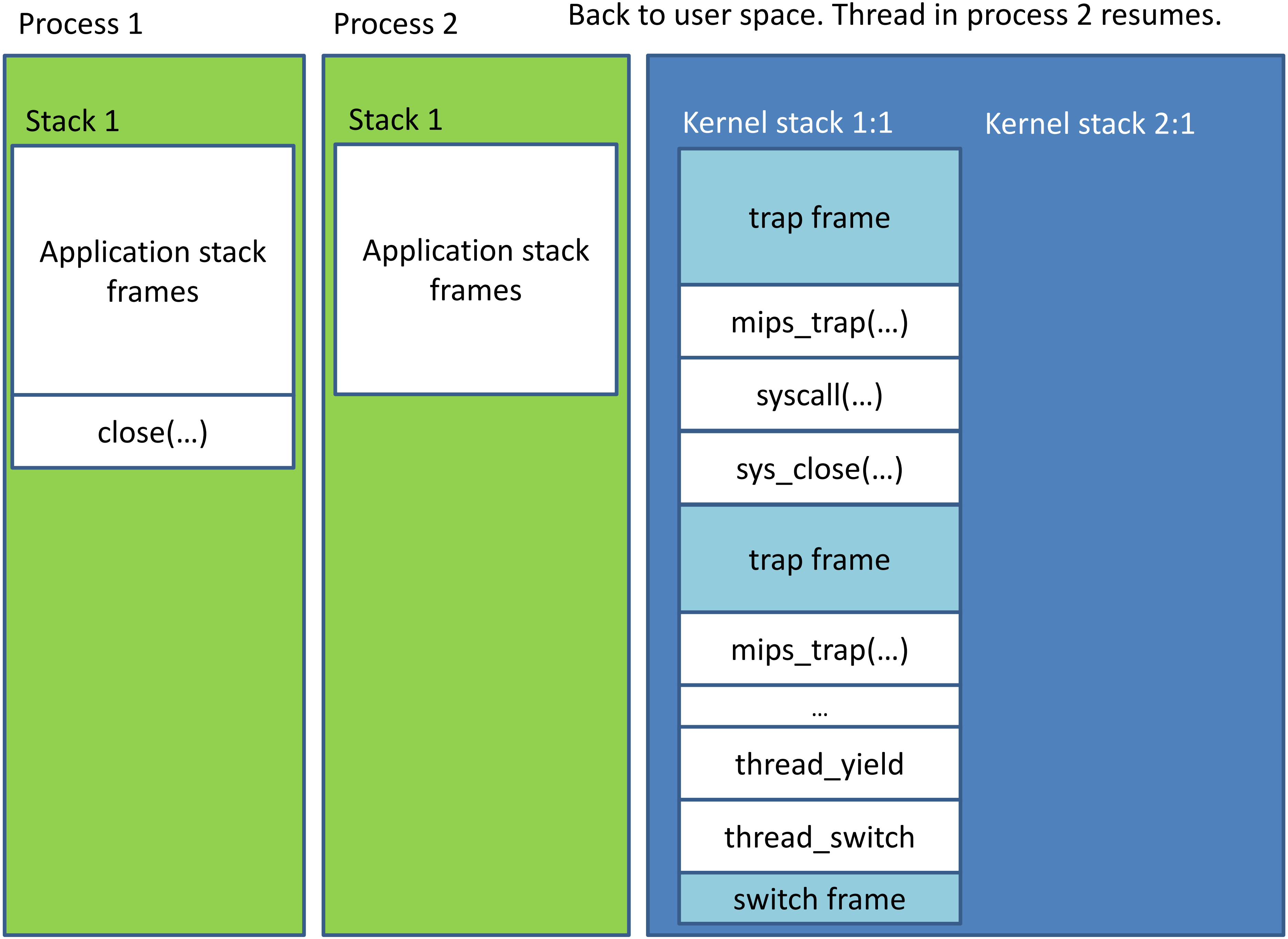
switch frame

Kernel stack 2:1

trap frame

mips_trap(…)

Back to user space. Thread in process 2 resumes.



Process 1

Stack 1

Application stack frames

close(…)

Process 2

Stack 1

Application stack frames

Kernel stack 1:1

trap frame

mips_trap(…)

syscall(…)

sys_close(…)

trap frame

mips_trap(…)

…

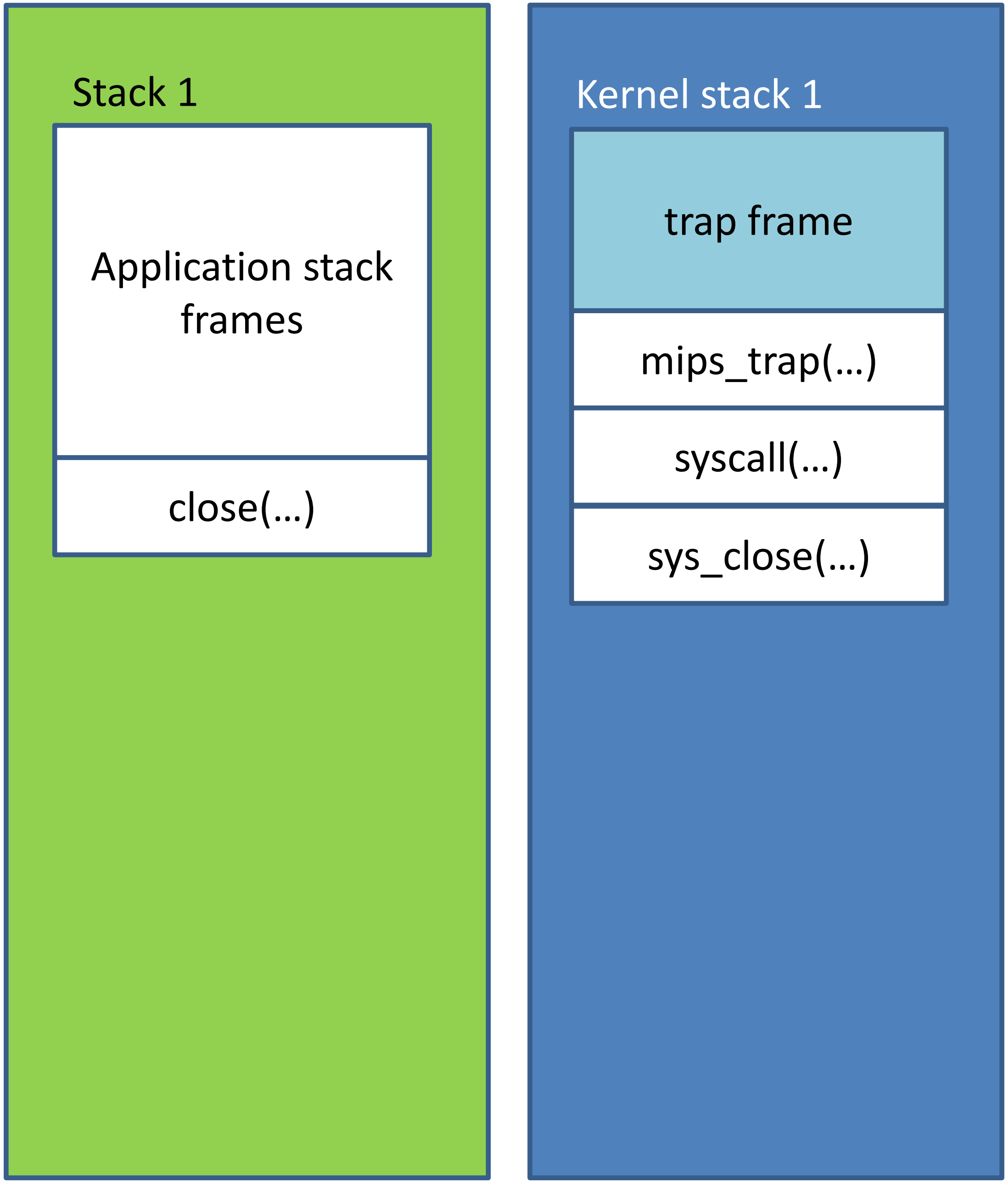
thread_yield

thread_switch

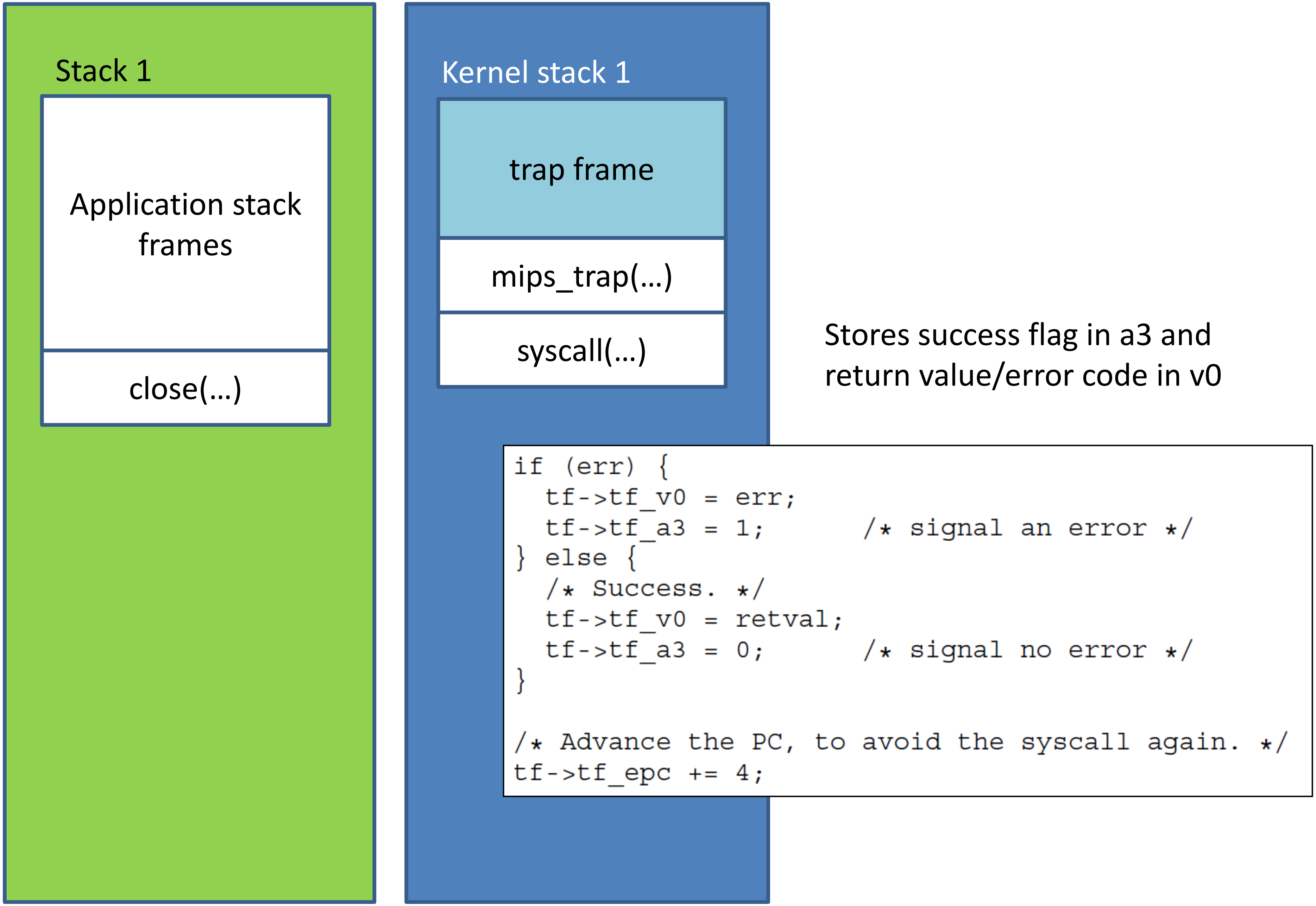
switch frame

Kernel stack 2:1

Let’s go back and assume the interrupt never happened.



When syscall returns, it modifies register values stored in the trap_frame.

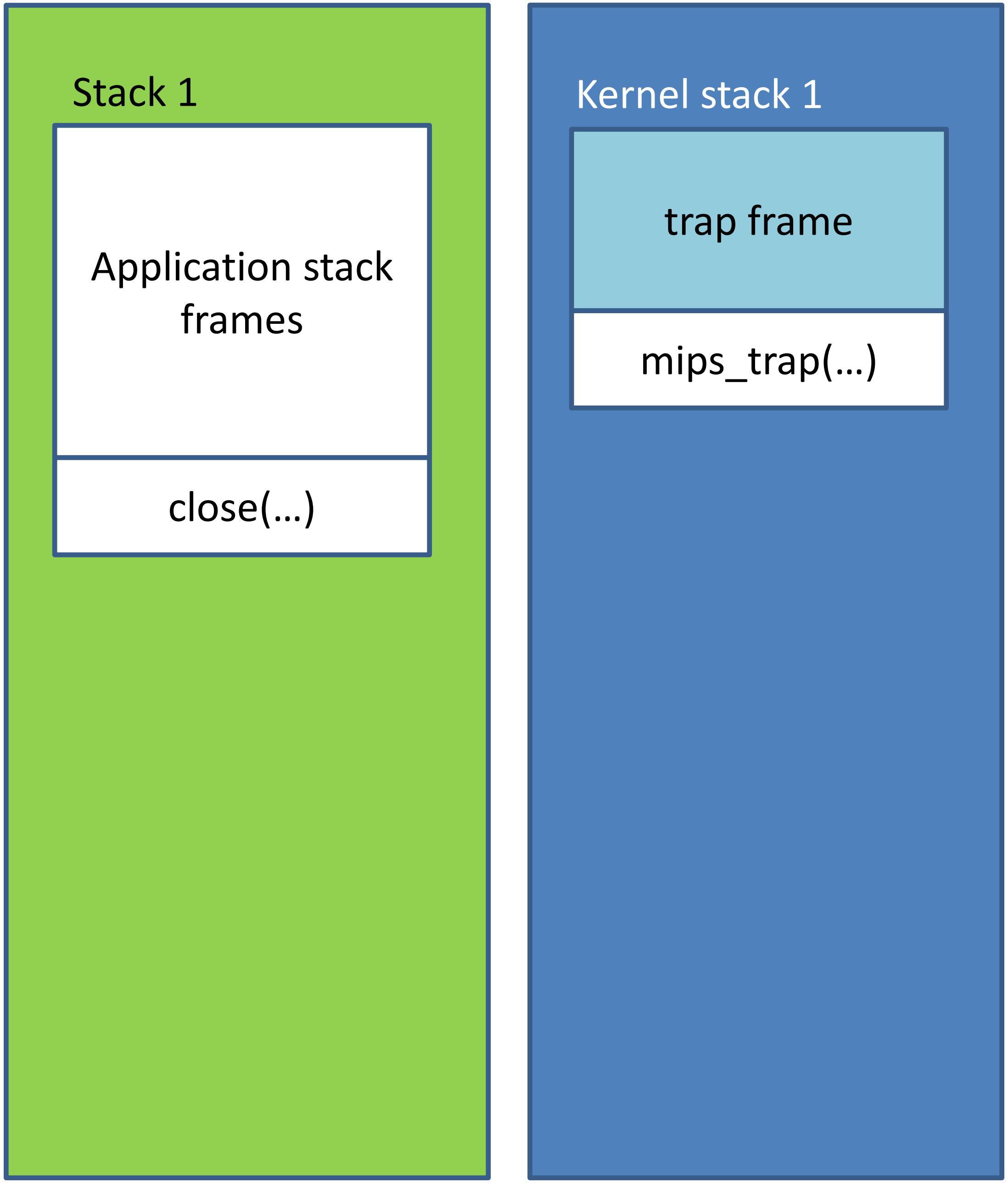


Stores success flag in a3 and return value/error code in v0

```
if (err) {
  tf->tf_v0 = err;
  tf->tf_a3 = 1;      /* signal an error */
} else {
  /* Success. */
  tf->tf_v0 = retval;
  tf->tf_a3 = 0;      /* signal no error */
}

/* Advance the PC, to avoid the syscall again. */
tf->tf_epc += 4;
```

Eventually returns control to the user-space application.



Stack 1

Application stack frames

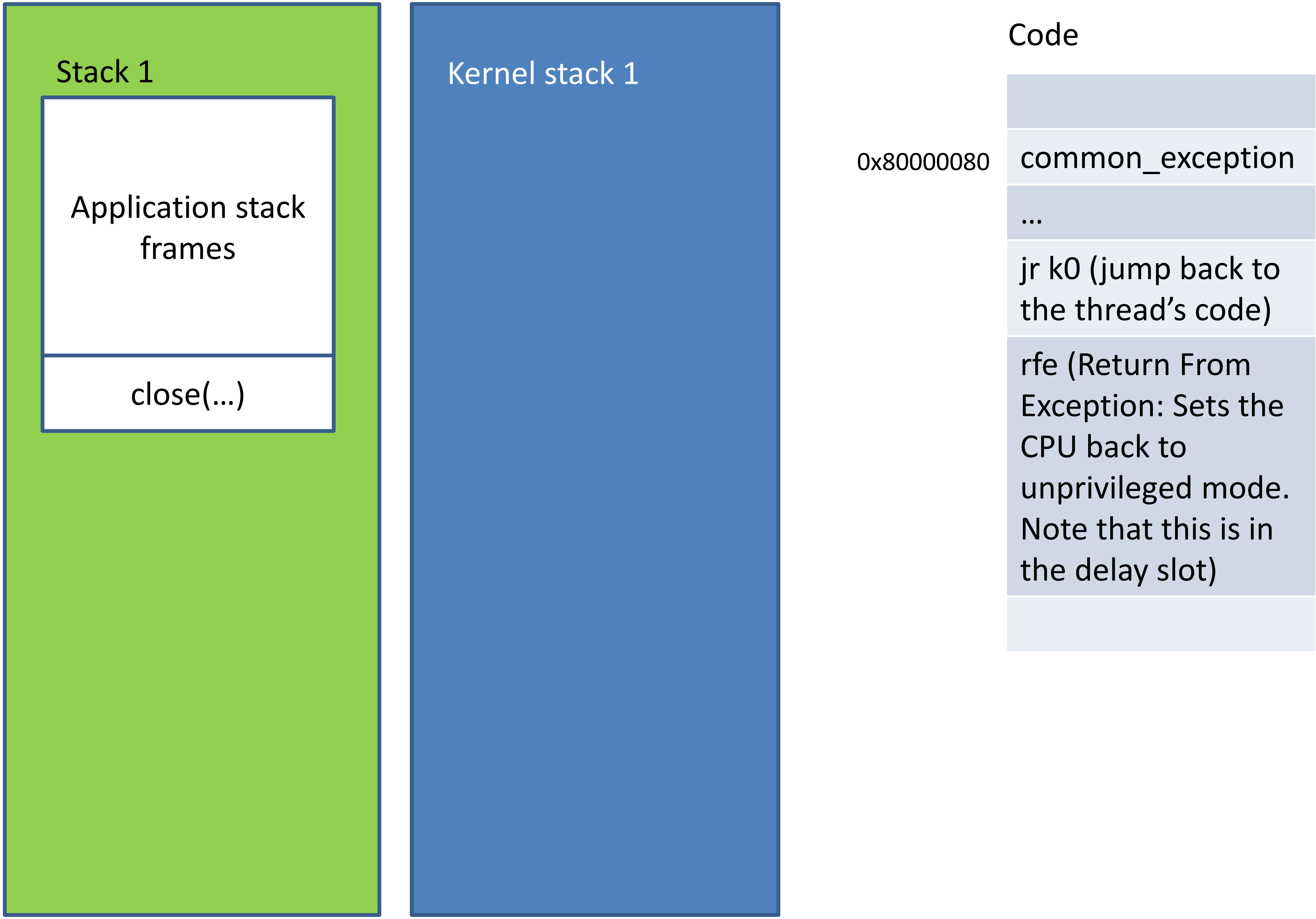
close(…)

Kernel stack 1

trap frame

mips_trap(…)

Eventually returns control to the user-space application.



Stack 1

Application stack frames

close(…)

Kernel stack 1

Code

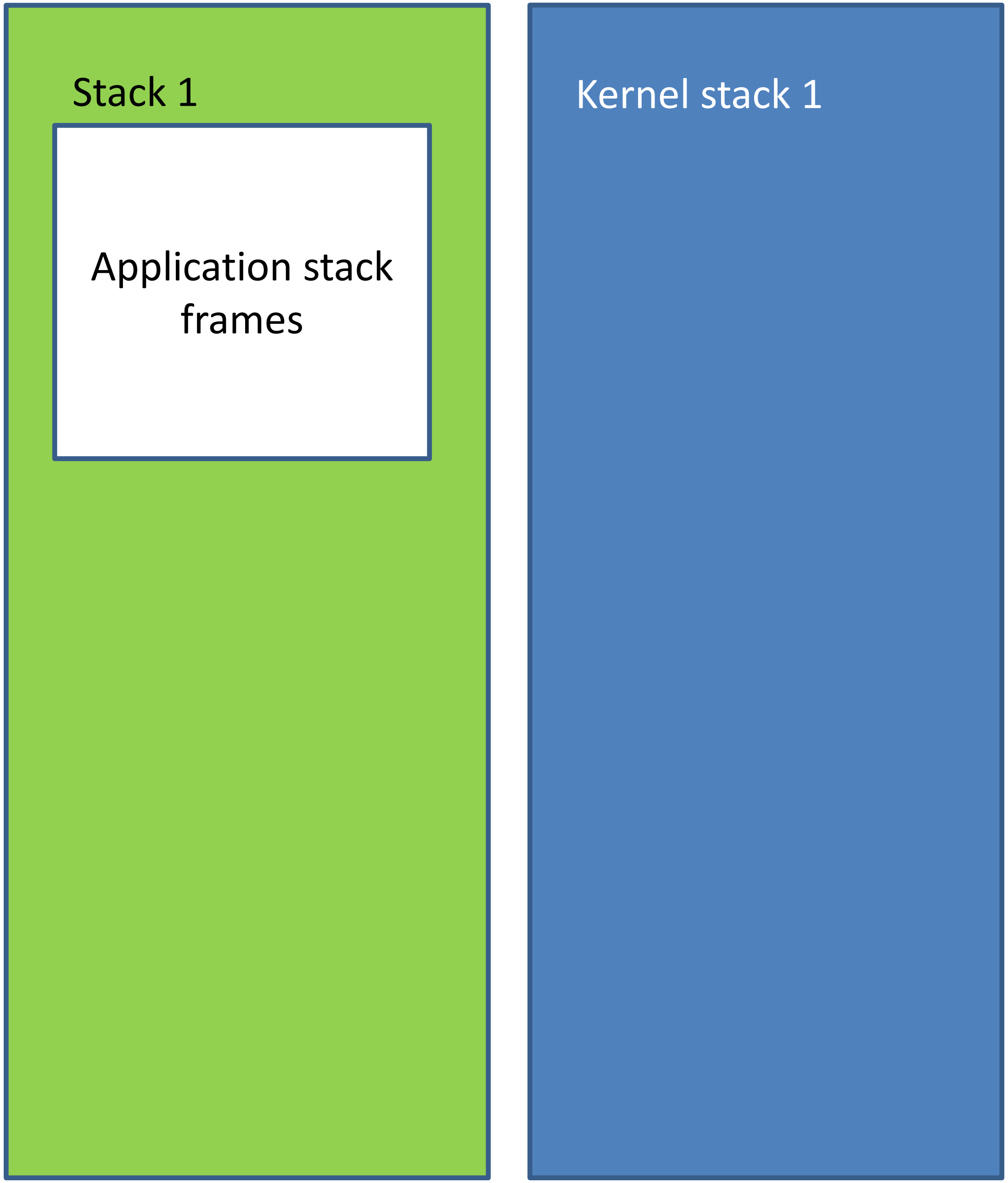
0x80000080 common_exception

…

jr k0 (jump back to the thread's code)

rfe (Return From Exception: Sets the CPU back to unprivileged mode. Note that this is in the delay slot)

Returns from the user-space system call library back to the application code.



Stack 1

Application stack frames

Kernel stack 1