Zephyr® Project
Developer Summit

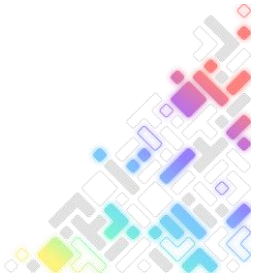
intel®

Userspace in Zephyr: Overview and Practicality

Daniel Leung

Topics

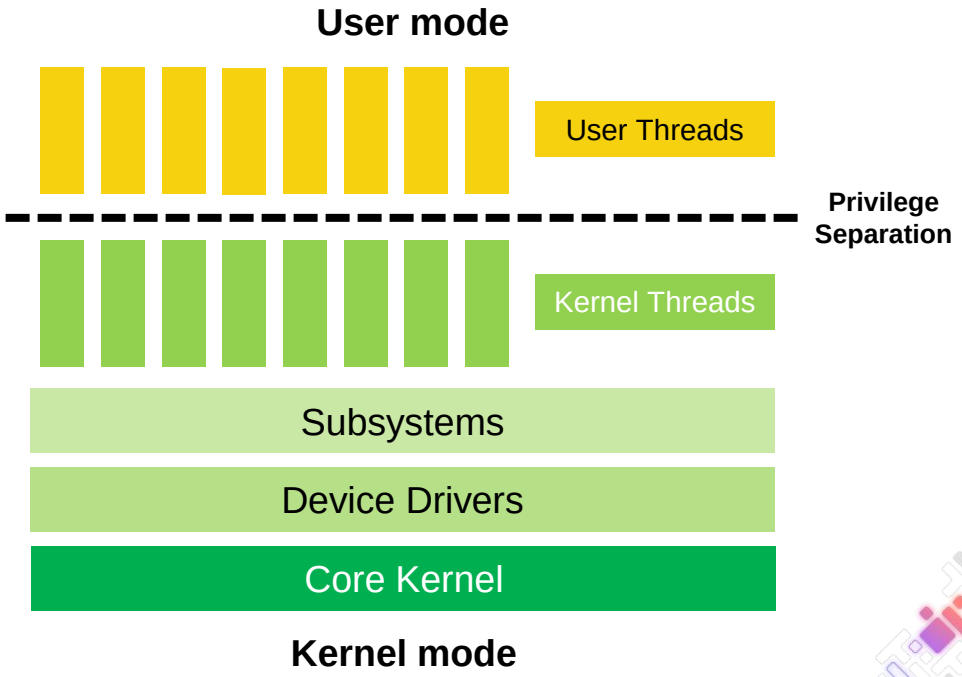
- Overview of User Space in Zephyr
- Sharing Data using Memory Domains
- Syscalls
- Why and Why Not User Space?
- Q&A



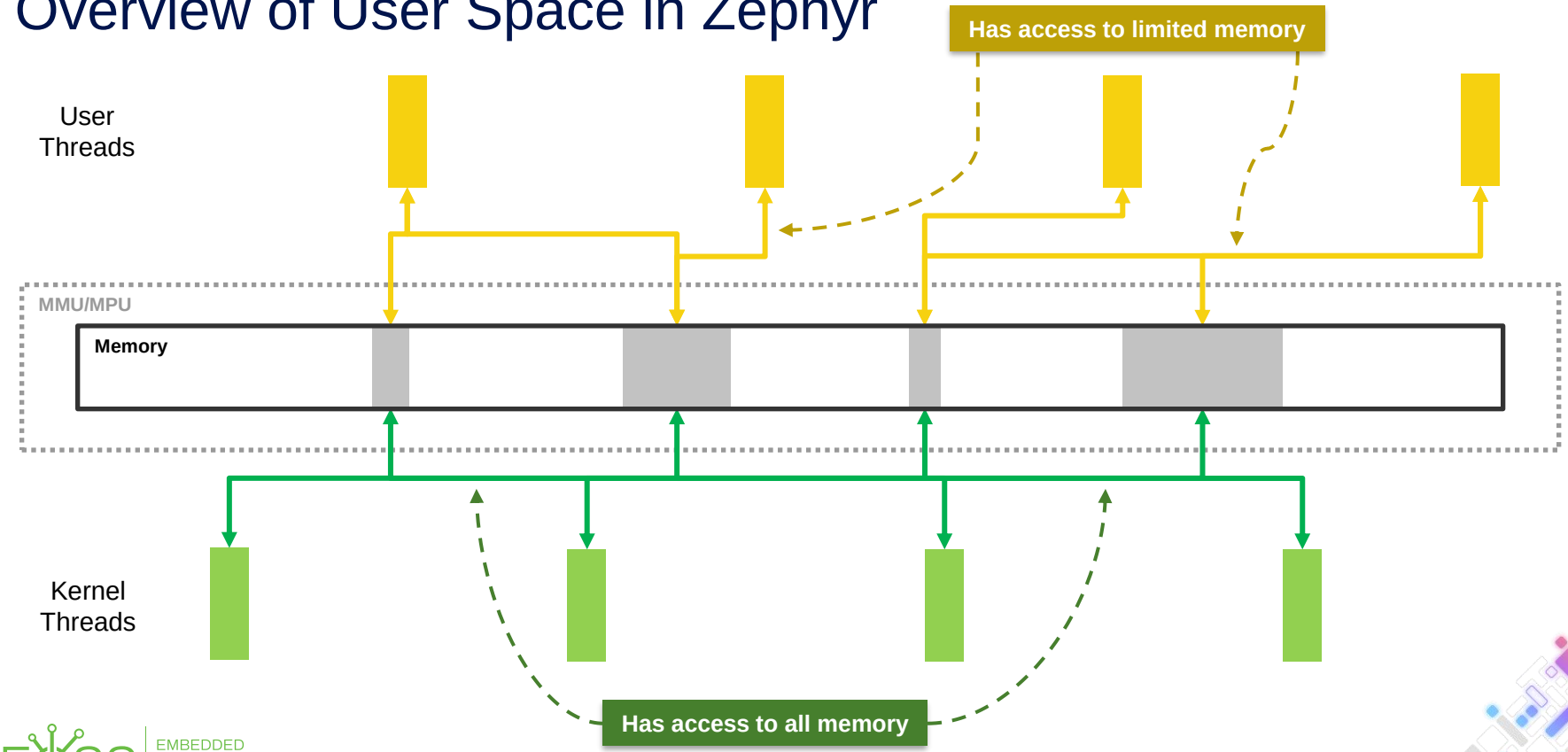
Overview of User Space in Zephyr

- User mode
  - a.k.a. Unprivileged Mode
  - User threads

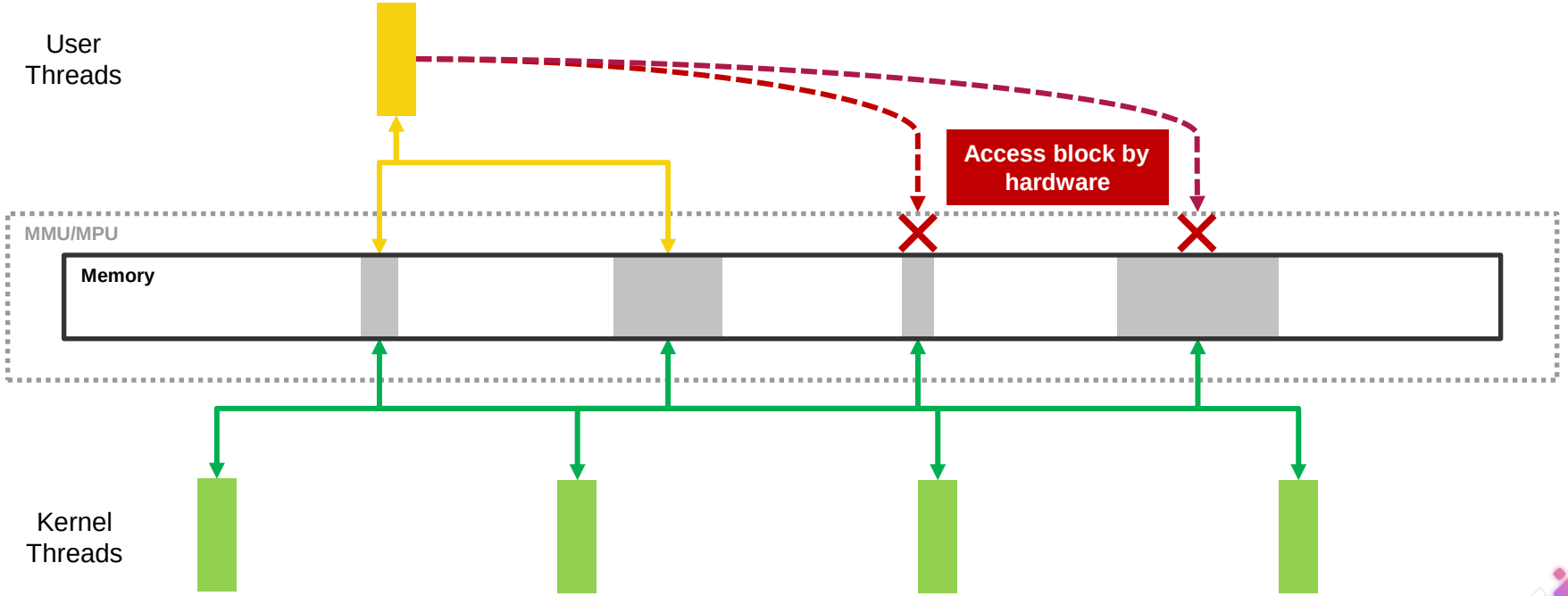
- Kernel mode
  - a.k.a. Supervisor or Privileged Mode
  - Core Kernel
  - Kernel threads
  - Subsystems
  - Device Drivers



Overview of User Space in Zephyr

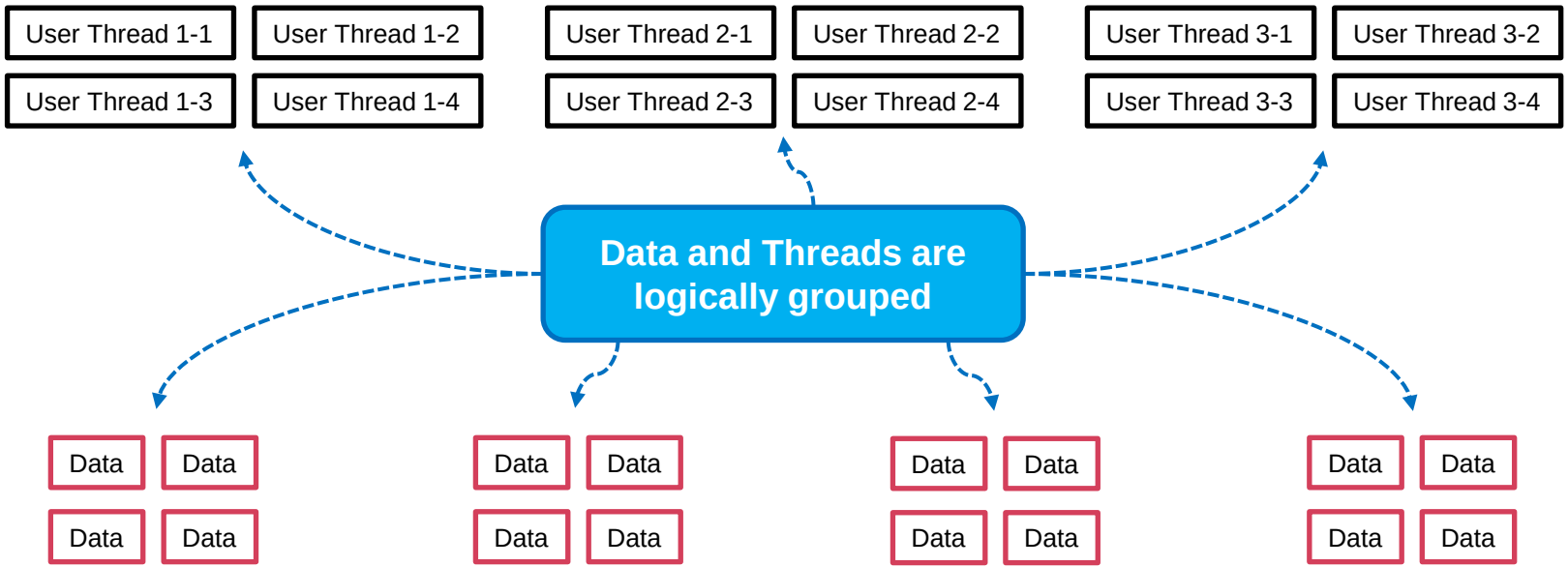


Overview of User Space in Zephyr



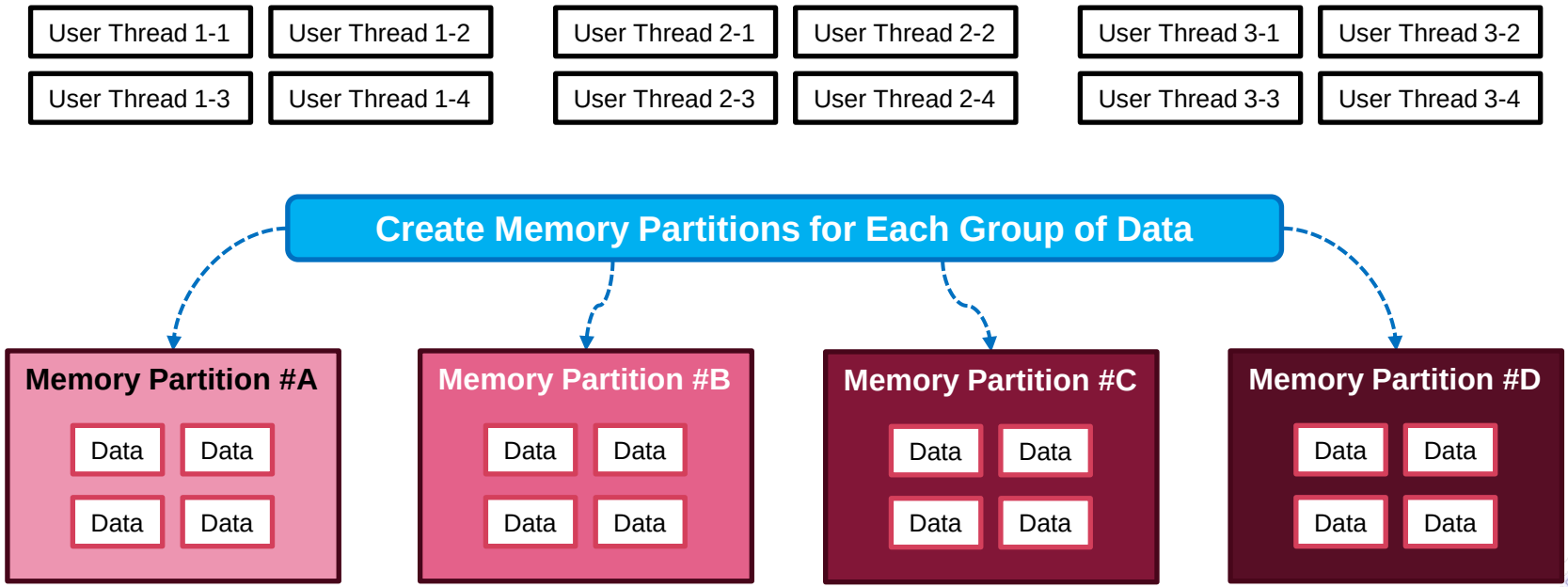
Sharing Data using Memory Domains

… and Memory Partitions



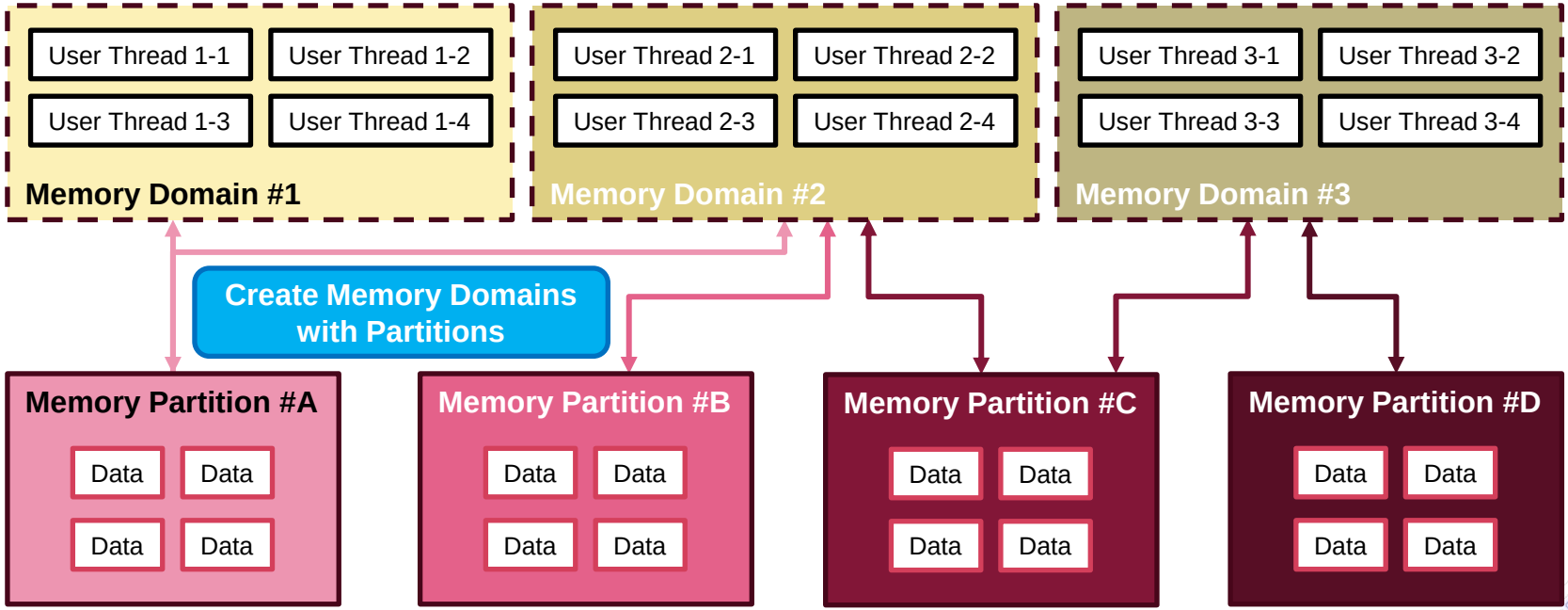
Sharing Data using Memory Domains

… and Memory Partitions



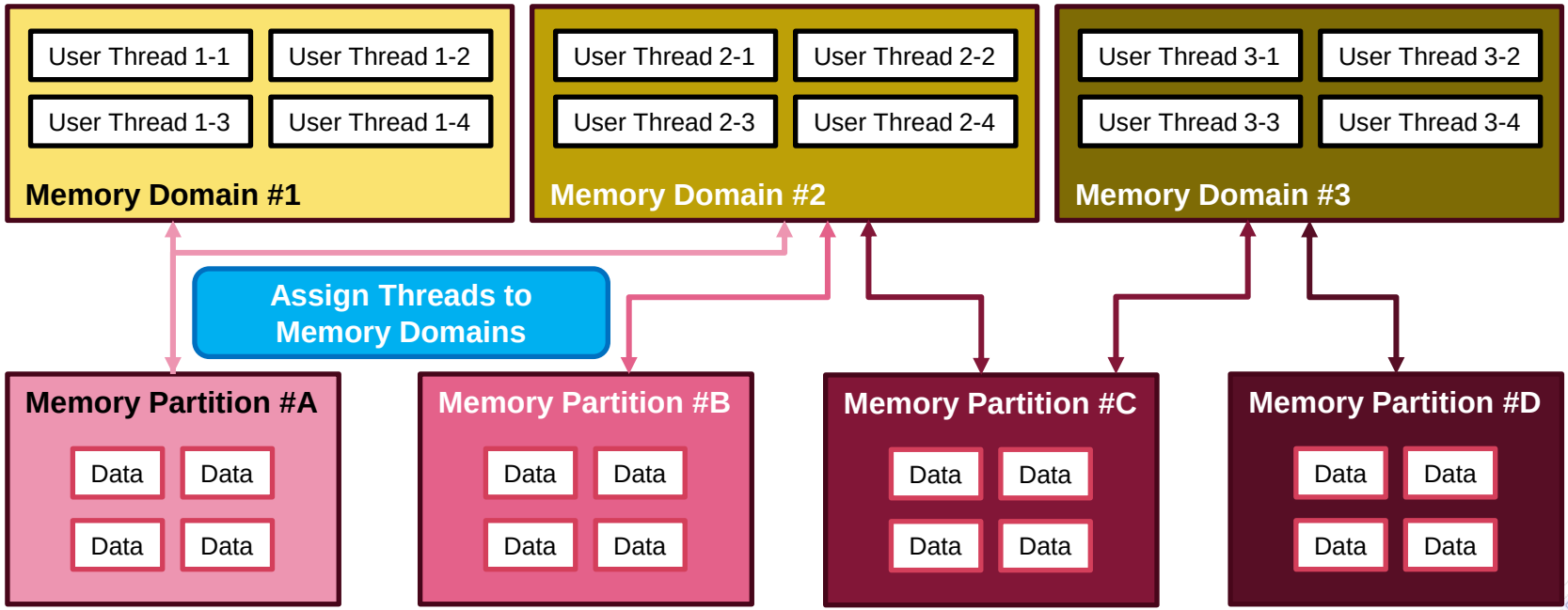
Sharing Data using Memory Domains

… and Memory Partitions



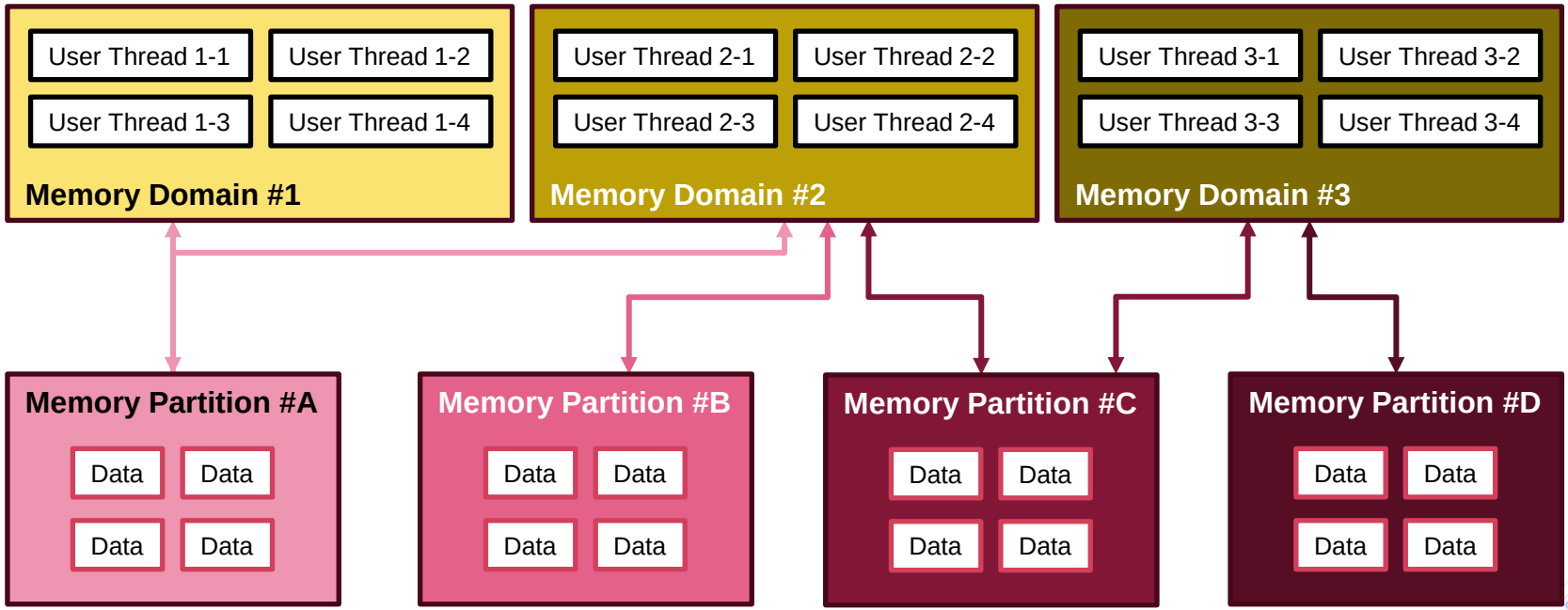
Sharing Data using Memory Domains

… and Memory Partitions



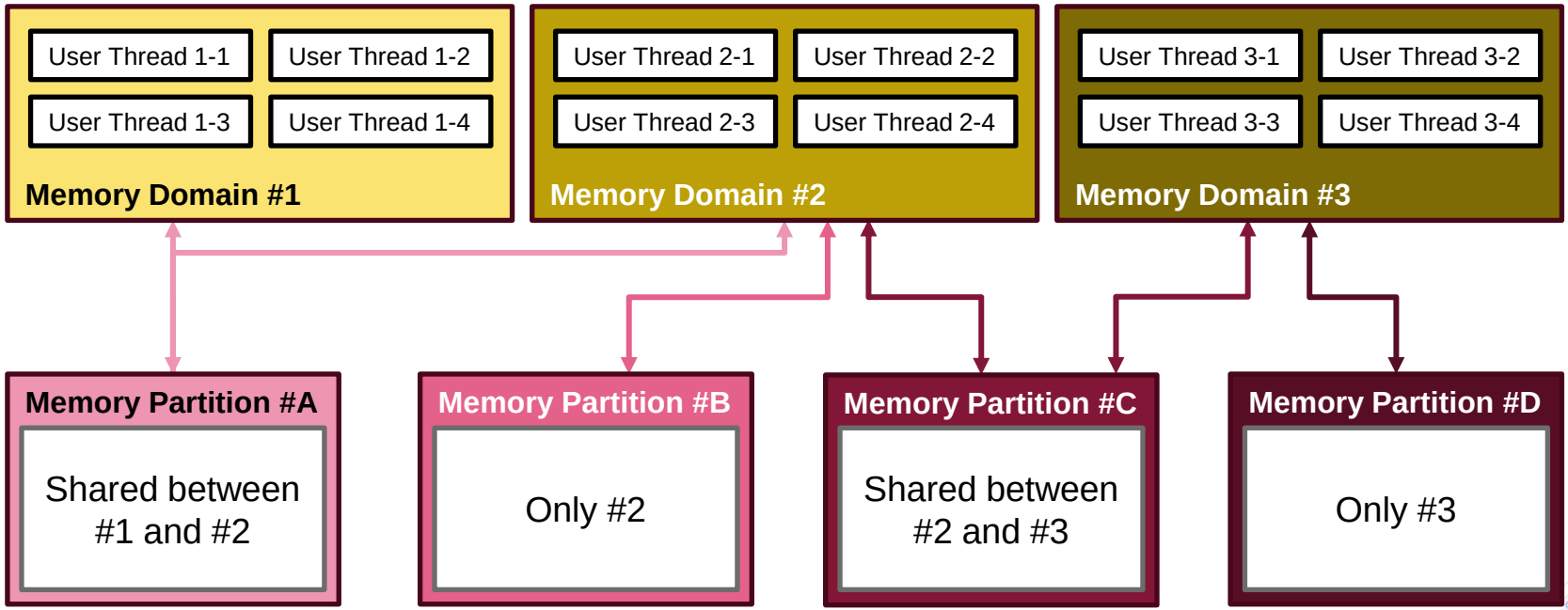
Sharing Data using Memory Domains

… and Memory Partitions



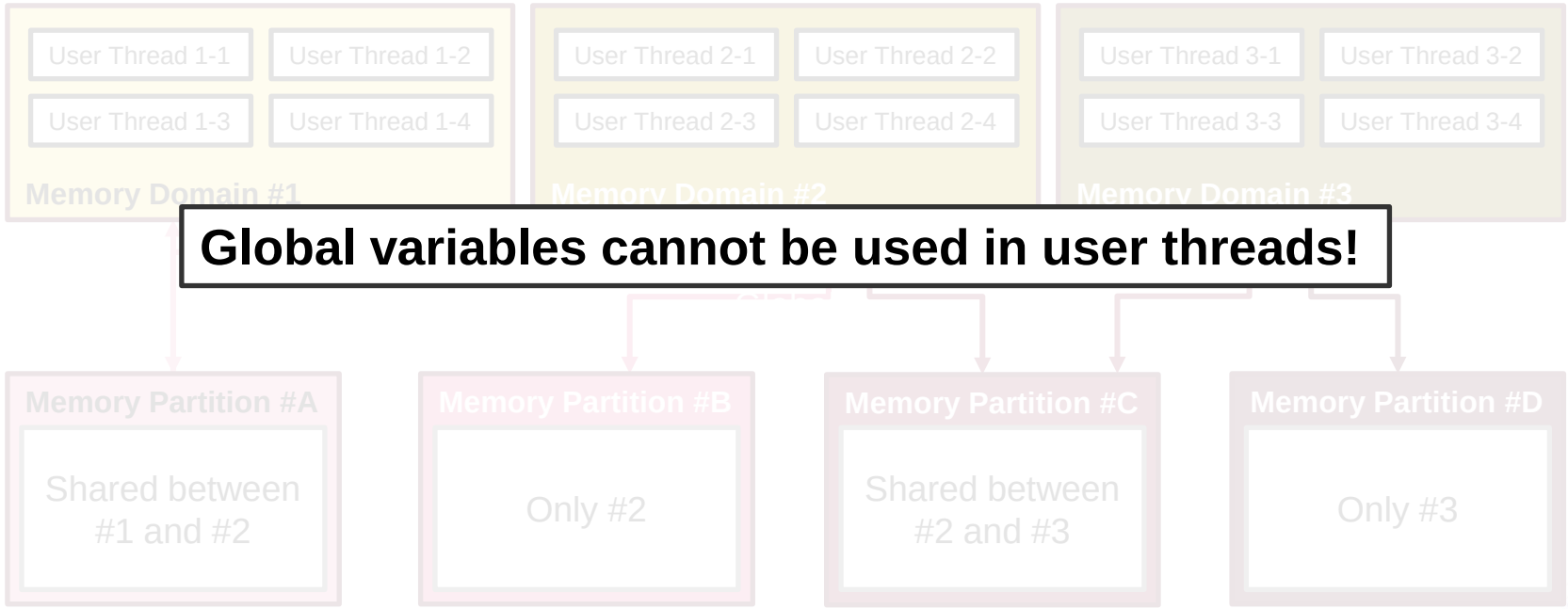
Sharing Data using Memory Domains

… and Memory Partitions



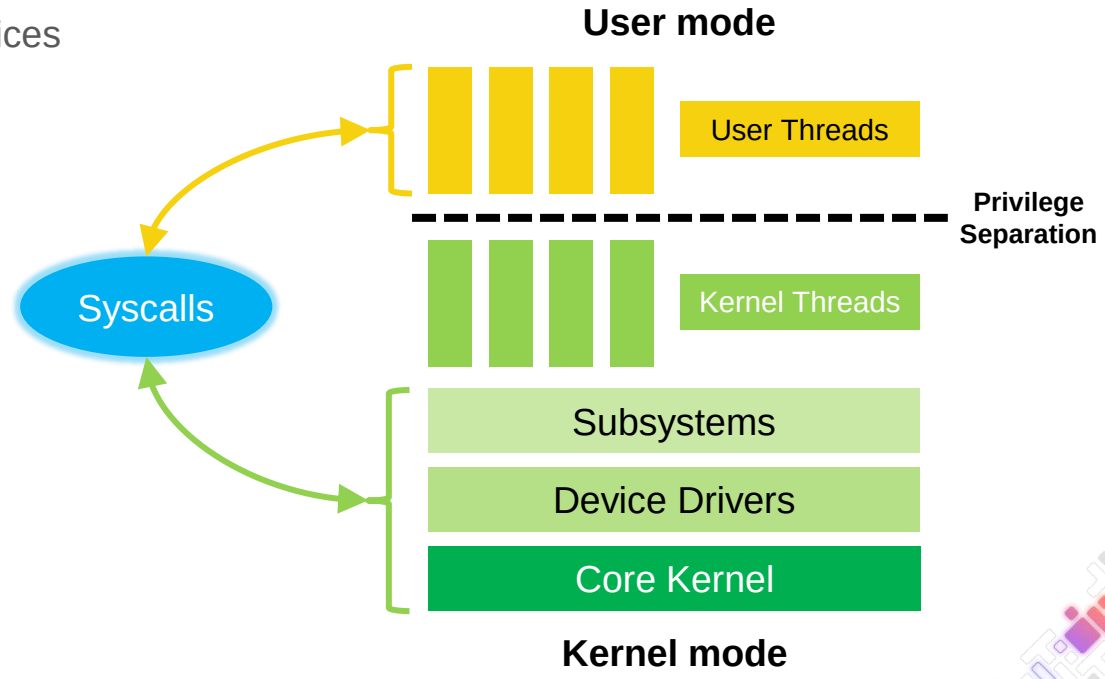
Sharing Data using Memory Domains

… and Memory Partitions



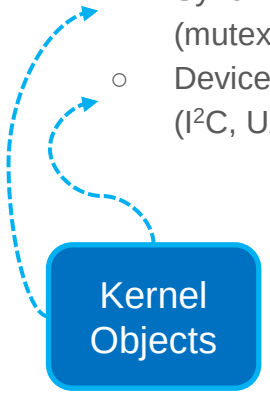
Syscalls

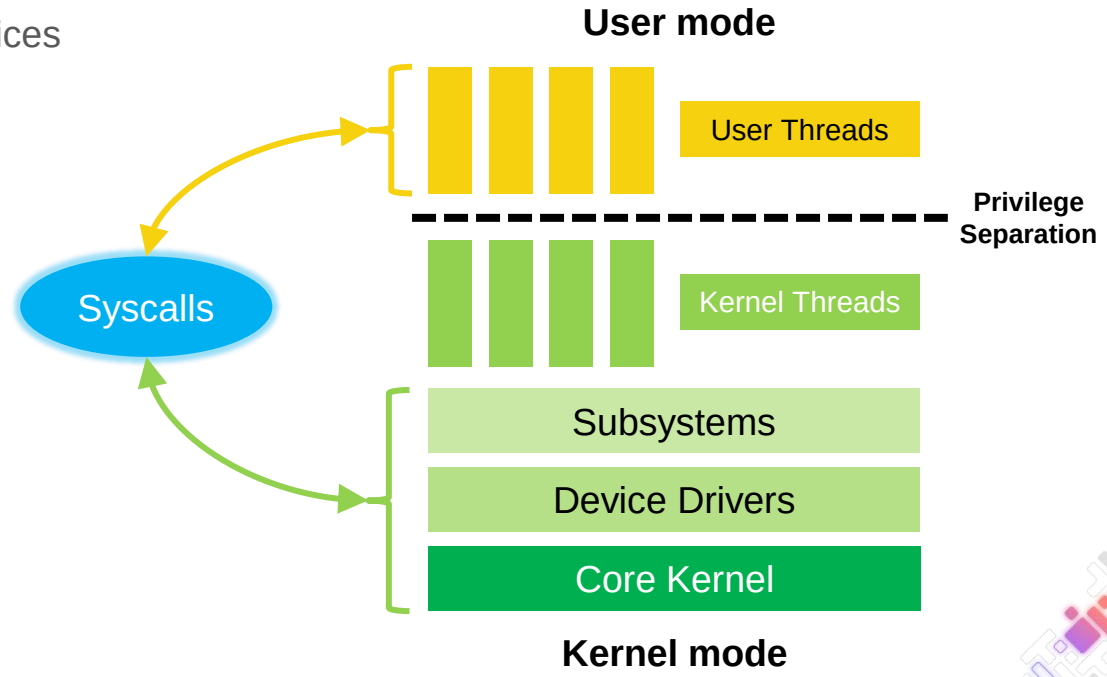
- User threads to access kernel services
  - Synchronization primitives (mutex, semaphore)
  - Device access (I^2C, UART)



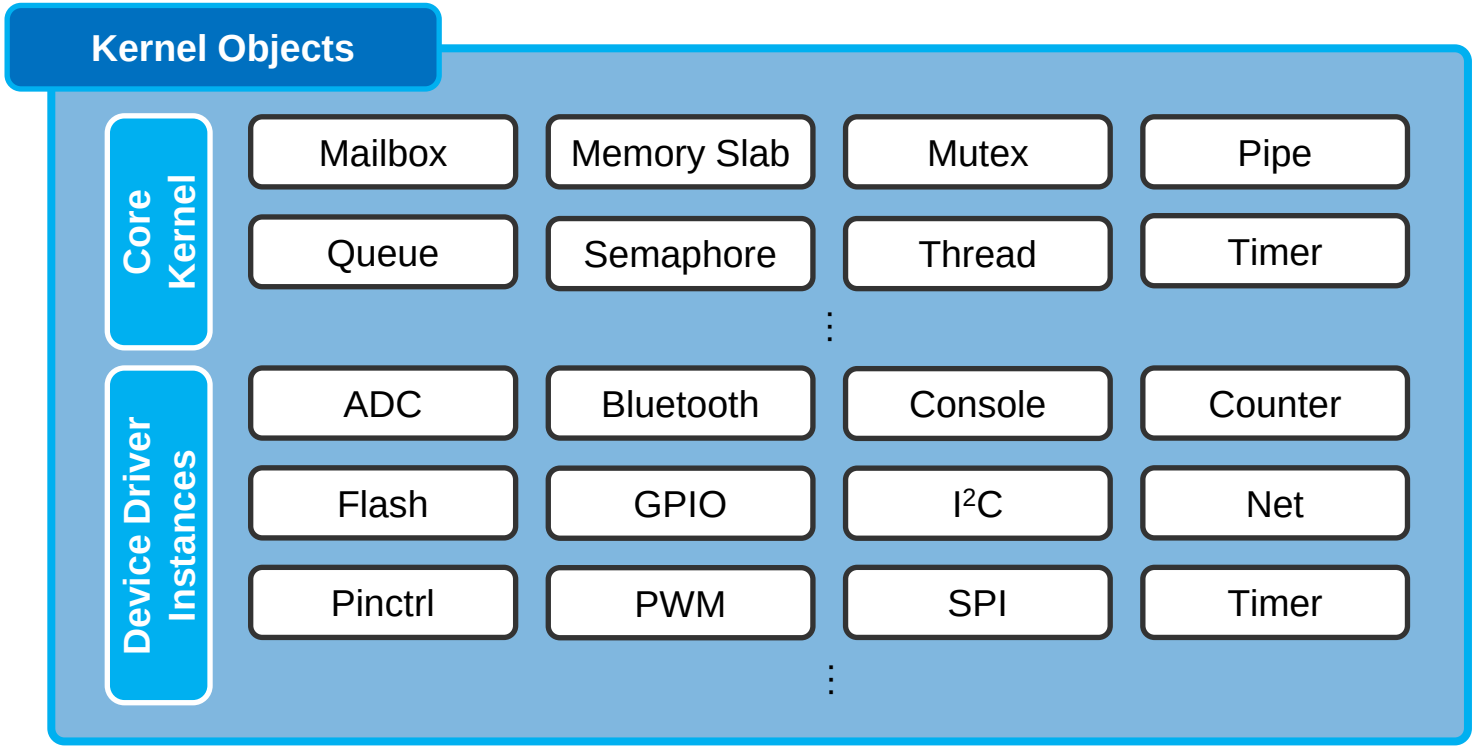
Syscalls

- User threads to access kernel services
  - Synchronization primitives (mutex, semaphore)
  - Device access (I^2C, UART)

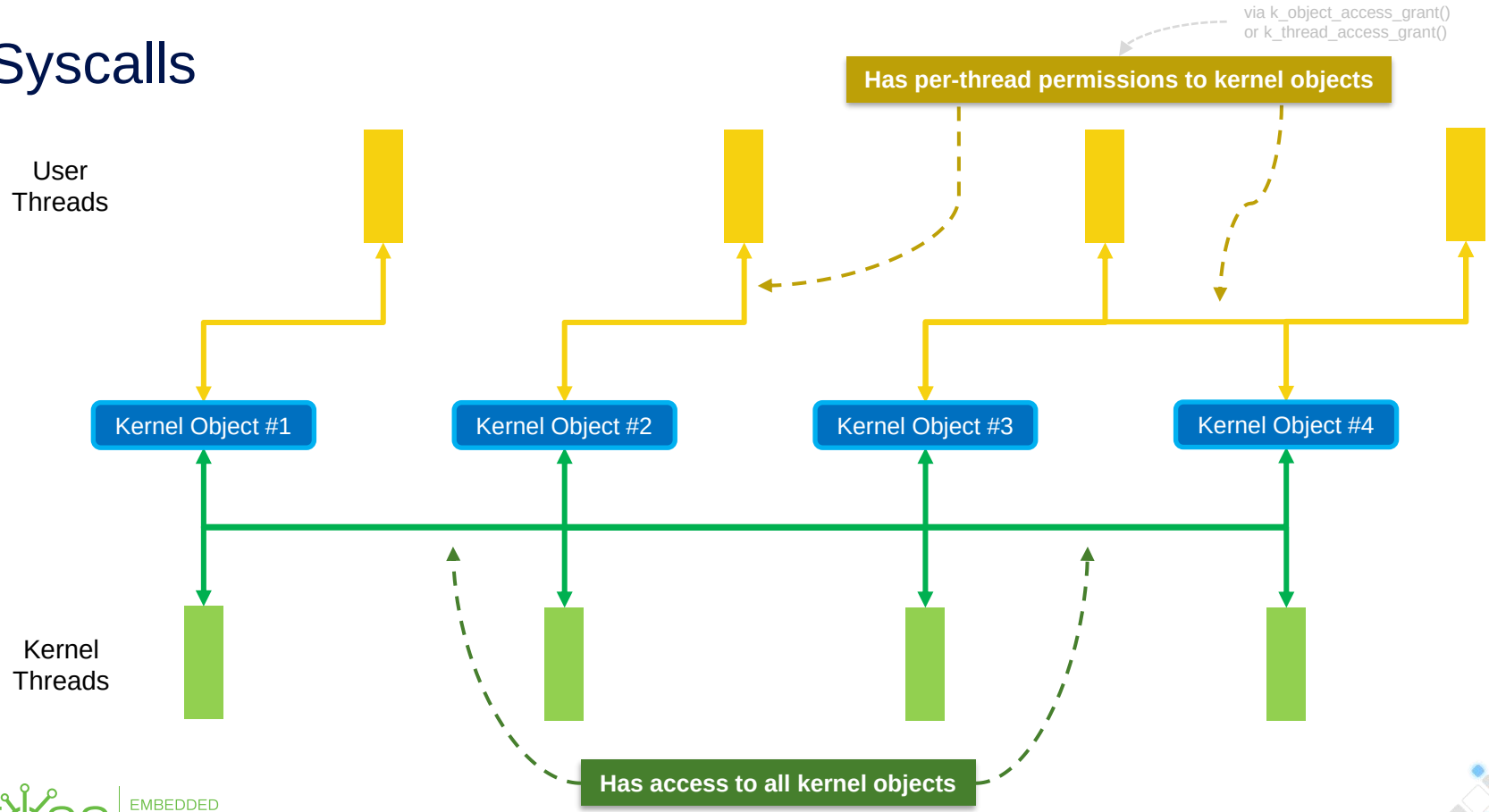




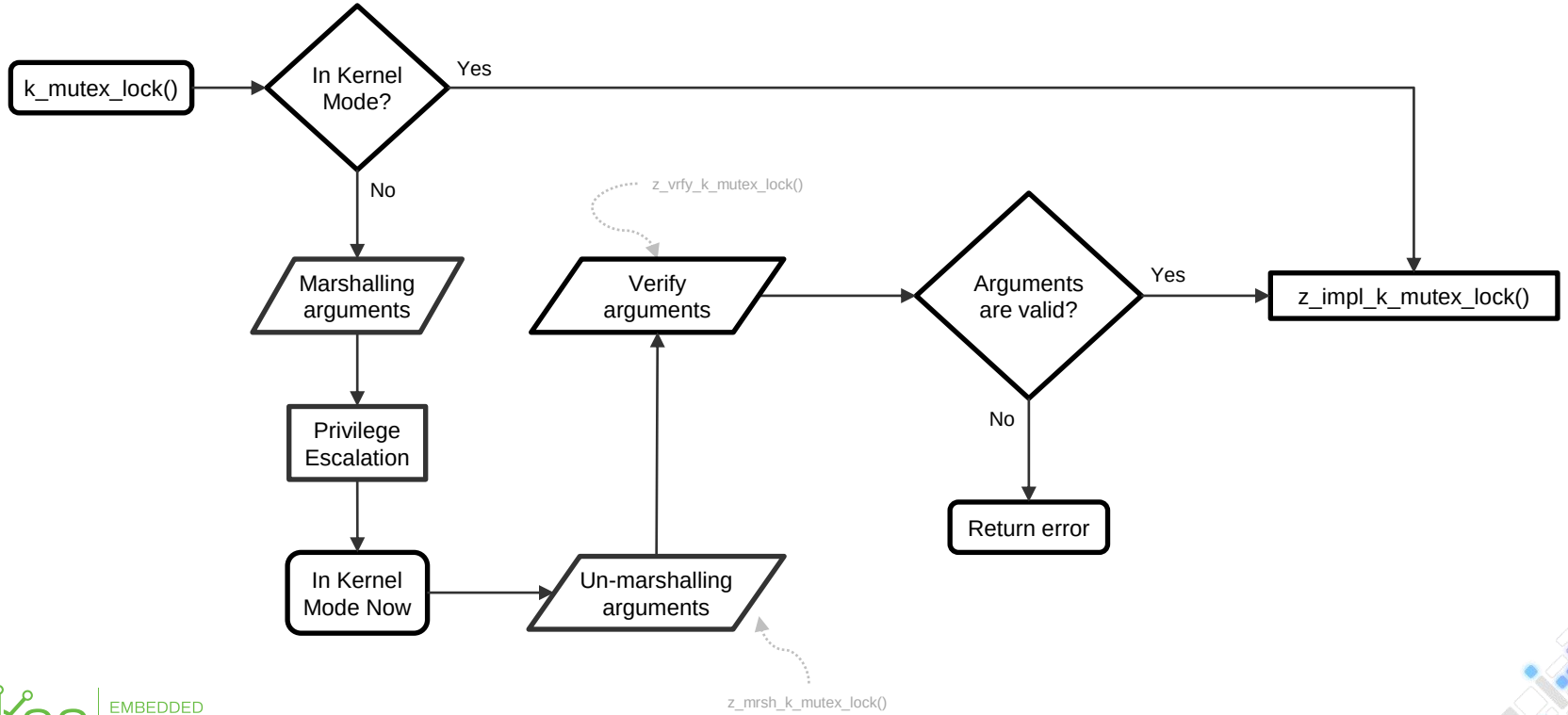
Syscalls


Kernel Objects
Core Kernel
Mailbox
Memory Slab
Mutex
Pipe
Queue
Semaphore
Thread
Timer
Device Driver Instances
ADC
Bluetooth
Console
Counter
Flash
GPIO
I2C
Net
Pinctrl
PWM
SPI
Timer

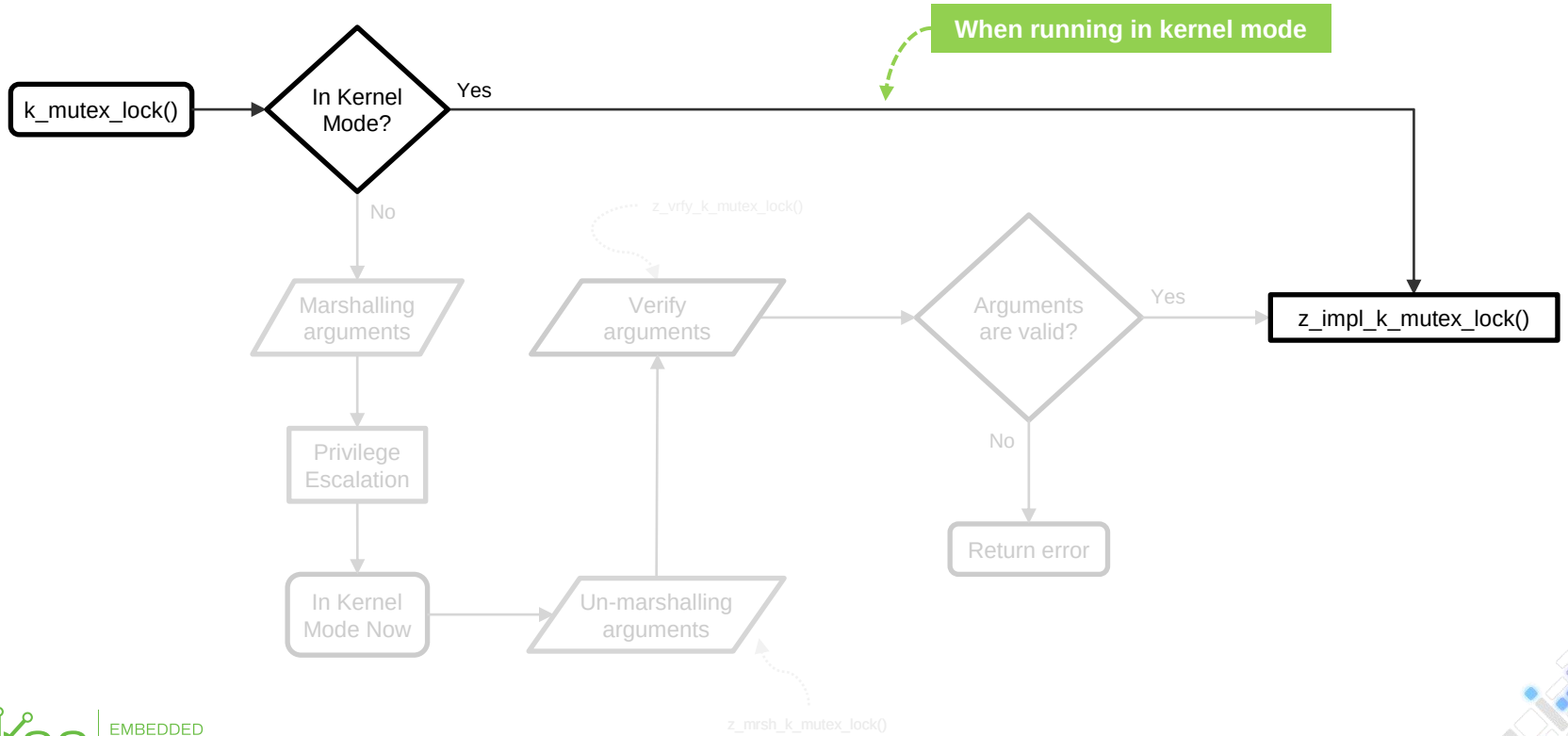
Syscalls


via k_object_access_grant()
or k_thread_access_grant()
Has per-thread permissions to kernel objects
User Threads
Kernel Object #1
Kernel Object #2
Kernel Object #3
Kernel Object #4
Kernel Threads
Has access to all kernel objects

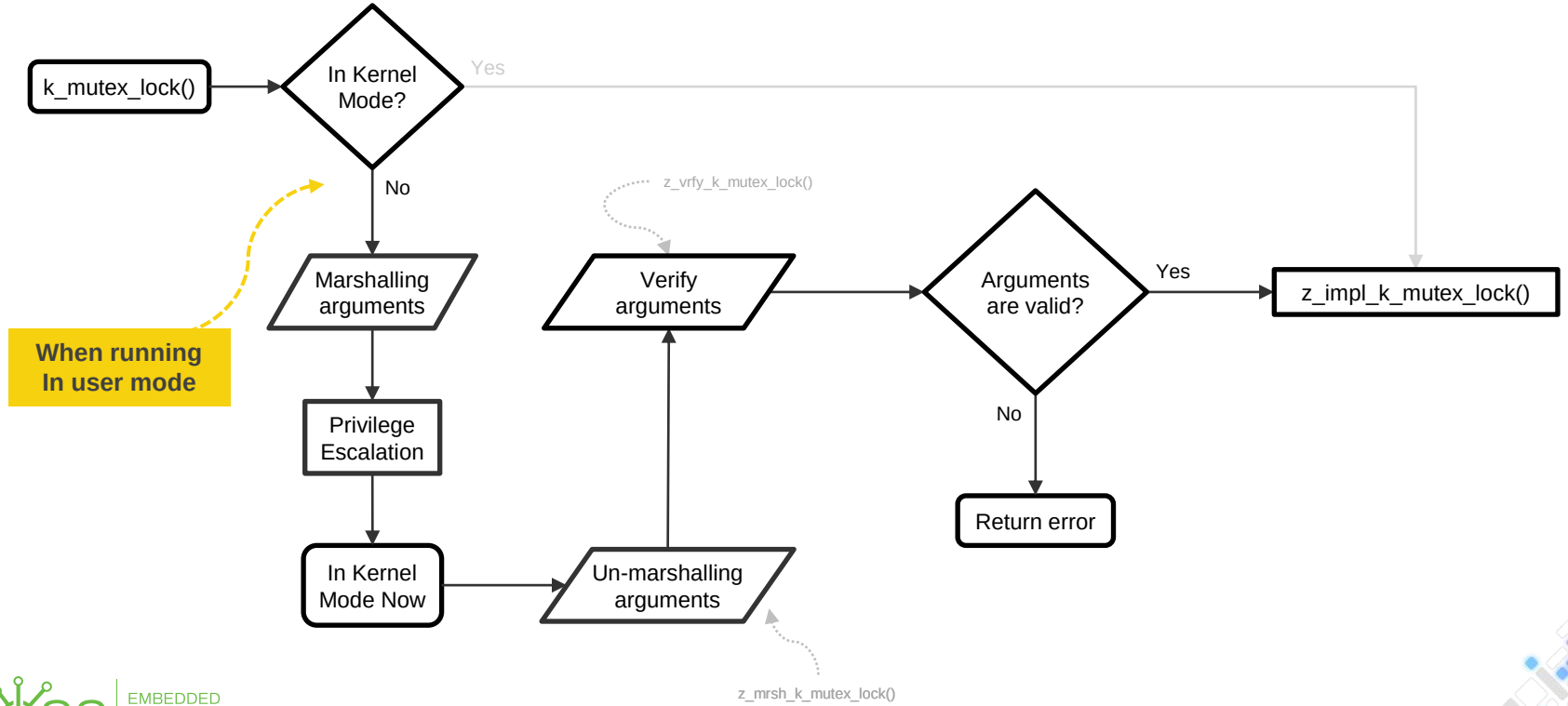
Syscalls


k_mutex_lock()
In Kernel Mode?
Yes
No
Marshalling arguments
Privilege Escalation
In Kernel Mode Now
z_vrfy_k_mutex_lock()
Verify arguments
Un-marshalling arguments
z_mrsh_k_mutex_lock()
Arguments are valid?
Yes
No
z_impl_k_mutex_lock()
Return error

Syscalls

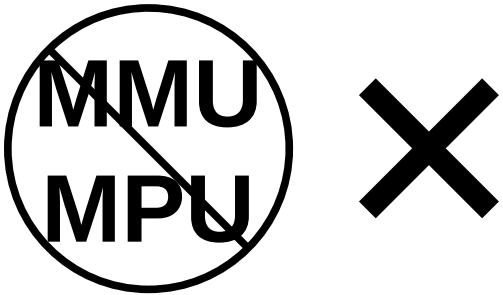

When running in kernel mode
k_mutex_lock()
In Kernel Mode?
Yes
No
z_impl_k_mutex_lock()
z_vrfy_k_mutex_lock()
Marshalling arguments
Verify arguments
Arguments are valid?
Yes
No
Privilege Escalation
Return error
In Kernel Mode Now
Un-marshalling arguments
z_mrsh_k_mutex_lock()

Syscalls

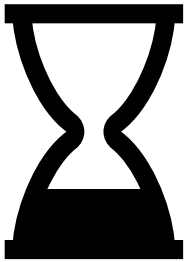

k_mutex_lock()
In Kernel Mode?
Yes
No
When running In user mode
Marshalling arguments
Privilege Escalation
In Kernel Mode Now
z_vrfy_k_mutex_lock()
Verify arguments
Un-marshalling arguments
z_mrsh_k_mutex_lock()
Arguments are valid?
Yes
No
z_impl_k_mutex_lock()
Return error

Why and Why Not User Space?

Hardware Requirements

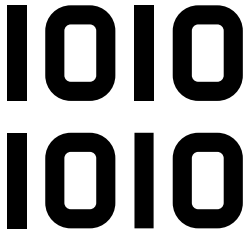


Why and Why Not User Space?



Latency

- **Context Switching**
- **Syscalls**

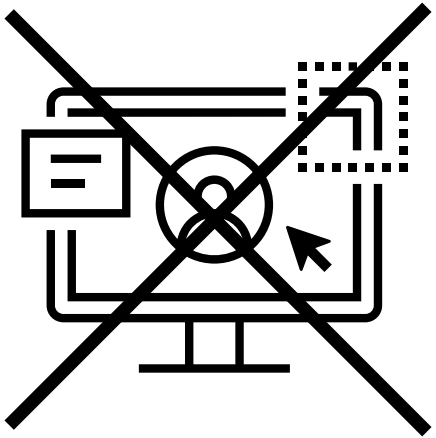


Memory Usage

- **Storage for Page Tables**
- **Stack for Privileged Mode**
- **Page Alignment for Memory Partitions**

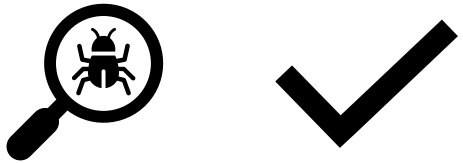
Why and Why Not User Space?

- No interactive users

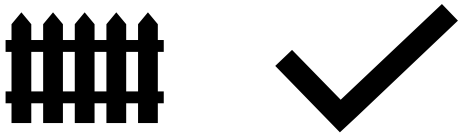


Why and Why Not User Space?

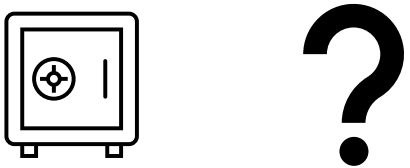
- Catch coding errors



- Isolation between memory domains

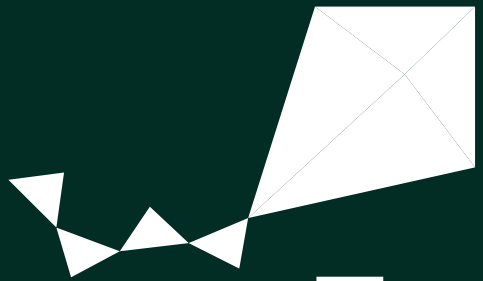


- Not for guarding against malicious code
  - User mode code is not trusted



… gets complicated with LLEXT.

Q&A



Zephyr® Project

Developer Summit