Architectural Overview

SILOS

Gregory Nutt

Architectural Overview –State Machine

An OS is a *STATE MACHINE*, not a program

- Better thought of as a *library* (but may include *Kernel threads* to monitor and respond to events).
- Manages *tasks*, *threads*, *interrupts*, *resources*, etc.
- Library-like functions respond to *events* from threads and hardware and *requests* from threads.
- Respond with state changes and perhaps thread state transitions (*scheduler*).
- MORE than a scheduler: Complete operating environment.

Thread

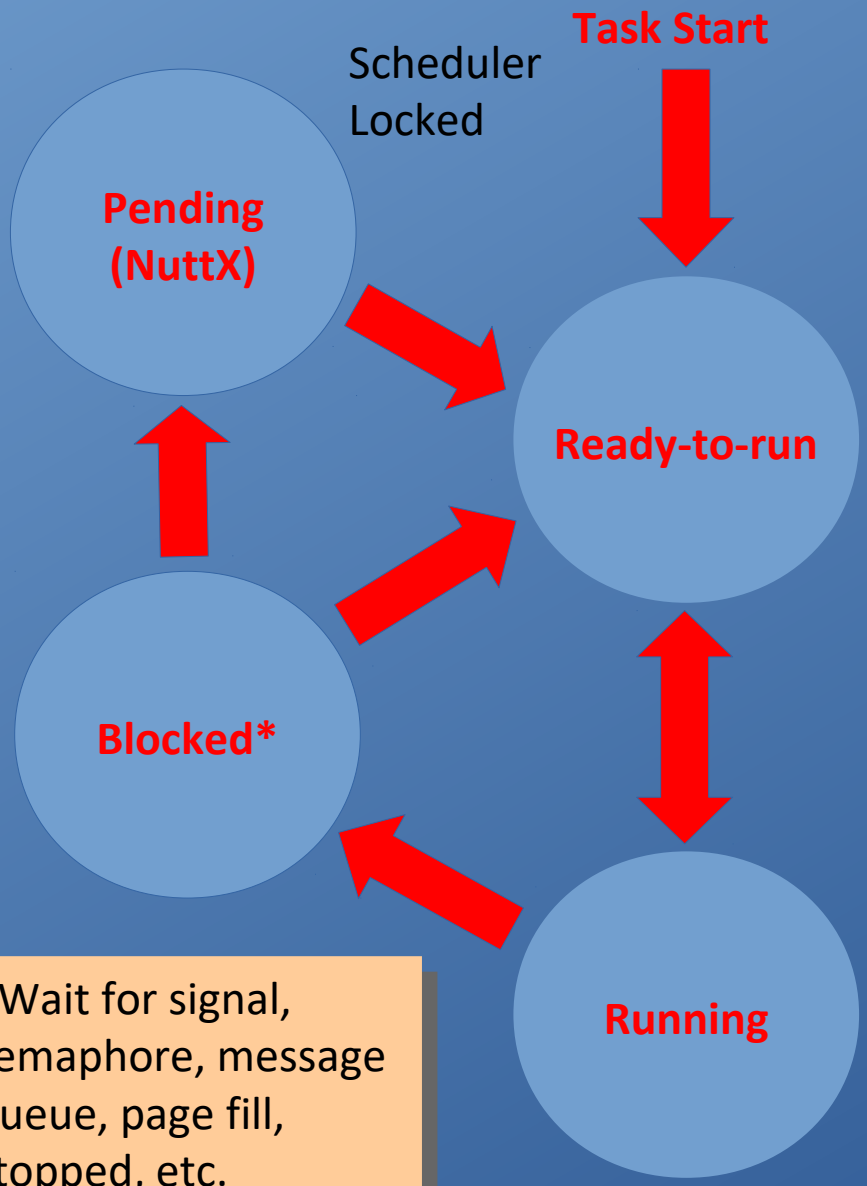
Normal sequential code execution
Separate stack

Task

Main thread of a task group
Members of the group share resources

Architectural Overview – Scheduler

One of MANY components of an Operating System



Fully pre-emptible
Context switch:
Think setjmp/longjmp on steriods

Task / Thread

Sequential code execution
Separate stack

Task Control Block (TCB)

States represented by lists of TCBs

Highest Priority Ready-to-run task is Running

*Wait for signal, semaphore, message queue, page fill, stopped, etc.

Architectural Overview – POSIX Interface

Applications

Strict POSIX Interface

OS / Application Interface

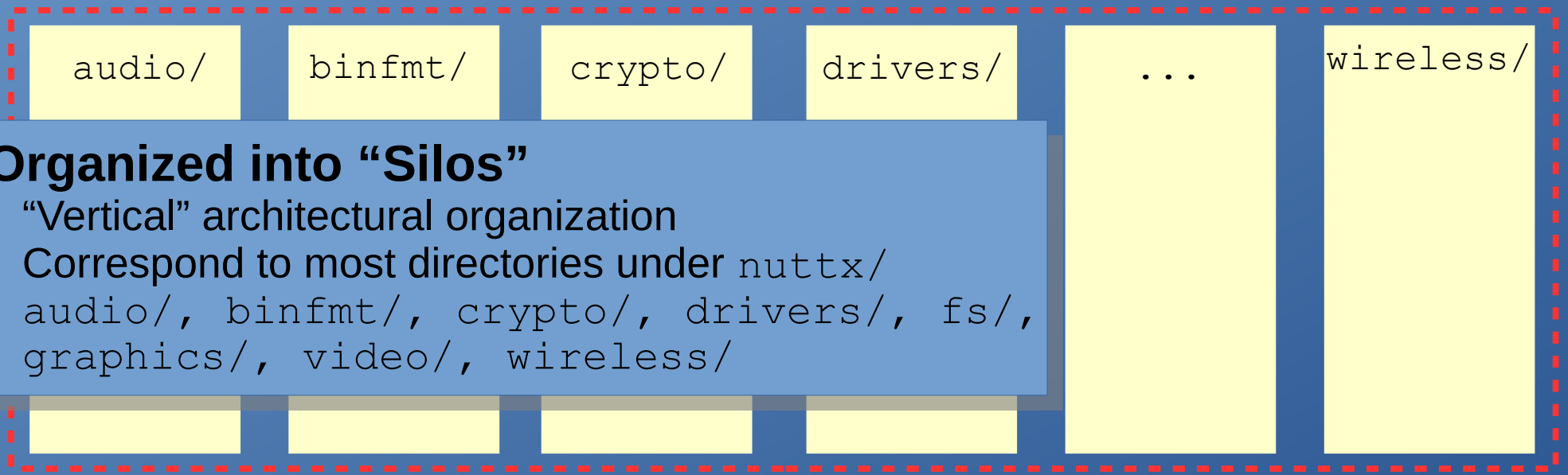
- Strict, standard interface between applications and the OS.
- No C-callable HAL (*Hardware Abstraction Layer*)
- *Ad hoc* calls into OS strictly forbidden

Hardware

Architectural Overview – Silos

Applications

Strict POSIX Interface



Organized into “Silos”

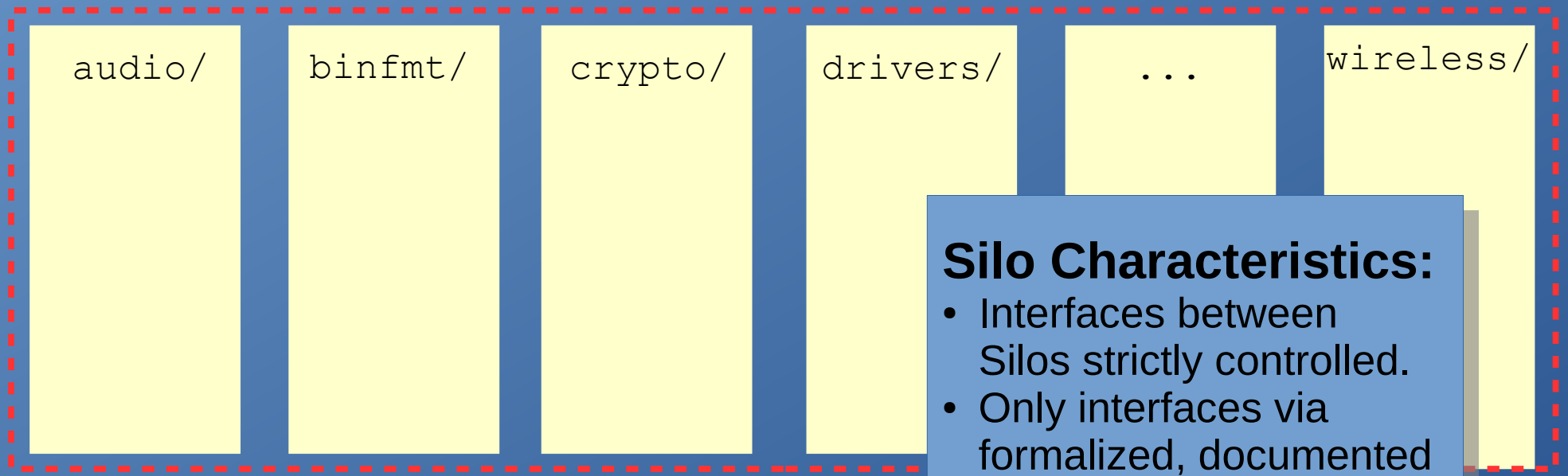
- “Vertical” architectural organization
- Correspond to most directories under `nuttx/` `audio/`, `binfmt/`, `crypto/`, `drivers/`, `fs/`, `graphics/`, `video/`, `wireless/`

Hardware

Architectural Overview – Silos (Continued)

Applications

Strict POSIX Interface



Silo Characteristics:

- Interfaces between Silos strictly controlled.
- Only interfaces via formalized, documented interfaces permitted.

Hardware

Architectural Overview – Common Libraries

Applications

Common Libraries (`libs/`)

Strict POSIX Interface

`audio/` `binfmt/` `crypto/` `drivers/` `...` `wireless/`

Libraries:

- "Horizontal" broad, "layered" software components.
- Also have well-defined interfaces, less strict

User-Facing libraries:

- `libs/` directory contains *share-able* logic that may be used *both* within the OS or by Applications.
- `libc`, `libm` (math), `libnx` (graphics), etc.

Common Libraries (`libs/`)

Hardware

Architectural Overview – User Facing Libraries

Applications

Common Libraries (`libs/`)

Strict POSIX Interface

VFS (`fs/vfs/`), BSD Sockets (net/sockets/), System Calls (`syscall/`)

`audio/` `binfmt/` `crypto/` `drivers/` `...` `wireless/`

User-Facing OS libraries:

- `sched/` provides user OS services
- `syscall/` provides system interface in PROTECTED and KERNEL build modes
- `mm/` memory management
- `fs/vfs/` provides POSIX name space management
- `net/sockets/` provides BSD socket interface

OS (`sched/`), memory Manager (`mm/`), Common Libraries (`libs/`)

Hardware

Architectural Overview – OS Internal Libraries

Applications

Common Libraries (`libs/`)

Strict POSIX Interface

VFS (`fs/vfs/`), BSD Sockets (net/sockets/), System Calls (`syscall/`)

`audio/` `binfmt/` `crypto/` `drivers/` `...` `wireless/`

OS Internal Libraries:

- "Horizontal" broad, "layered" OS components.
- Provide common services to "Silos"
- Also have well-defined interfaces, less strict
- `arch/` provides architecture-specific support
- `boards/` provides board-specific support

OS (`sched/`), memory Manager (`mm/`), Common Libraries (`libs/`)

Architecture-specific support (`arch/`), Board-specific support (`boards/`)

Hardware

Architectural Overview – Layering

Applications

Common Libraries (`libs/`)

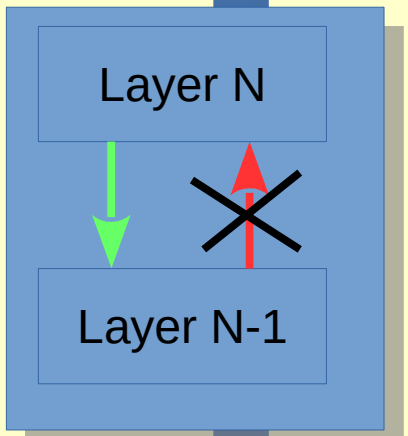
Strict POSIX Interface

VFS (`fs/vfs/`), BSD Sockets (net/sockets/), System Calls (`syscall/`)

`audio/` `binfmt/` `crypto/` `drivers/` `...` `wireless/`

Strict Layering:

- Lowest layers depend on nothing else
- Higher layers depend only on lower layers
- Only downward control. Higher level layers may call into lower levels. Lower levels may not call directly into higher levels.



OS (`sched/`), memory Manager (`mm/`), Common Libraries (`libs/`)

Architecture-specific support (`arch/`), Board-specific support (`boards/`)

Hardware

Architectural Overview – Layering

Applications

Common Libraries (`libs/`)

Strict POSIX Interface

VFS (`fs/vfs/`), BSD Sockets (net/sockets/), System Calls (`syscall/`)

`audio/` `binfmt/` `crypto/` `drivers/` `...` `wireless/`

Layering within Silos:

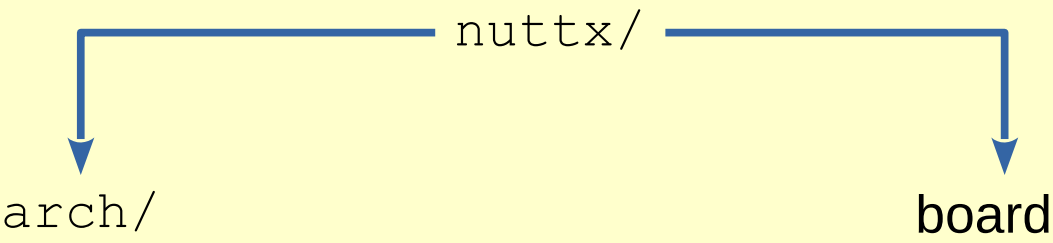
- Internal architecture of each Silo is also layered

OS (`sched/`), memory Manager (`mm/`), Common Libraries (`libs/`)

Architecture-specific support (`arch/`), Board-specific support (`boards/`)

Hardware

Architectural Overview – Platform Directories



arch/**<arch>**

arch/**<arch>**/include/**<chip>**
arch/**<arch>**/src/**<chip>**

arch/**<arch>**/**<chip>**

arch/**<arch>**/**<chip>**/**<board>**

Platform:

- Fully described by CPU architecture **<arch>**,
- By MCU chip architecture **<chip>**, and
- Board design **<board>**

Architectural Overview – Architectures, Boards, and Configuration

Platform Directories:

- Requires specification of CPU architecture, MCU architecture, and board design
- CPU architecture provided by sub-directories of `arch/` and `boards/`: `arm/`, `avr/`, `hc/`, `mips/`, `misoc/`, `or1k/`, `renesas/`, `risc-v/`, `sim/`, `x86/`, `xtensa/`, `z16/`, `z80/`
- MCU architecture provided by sub-directories of `arch/<arch>/src`, `arch/<arch>/include`, and `boards/<arch>/`
- Board design provided by sub-directories of `boards/<arch>/<chip>`.

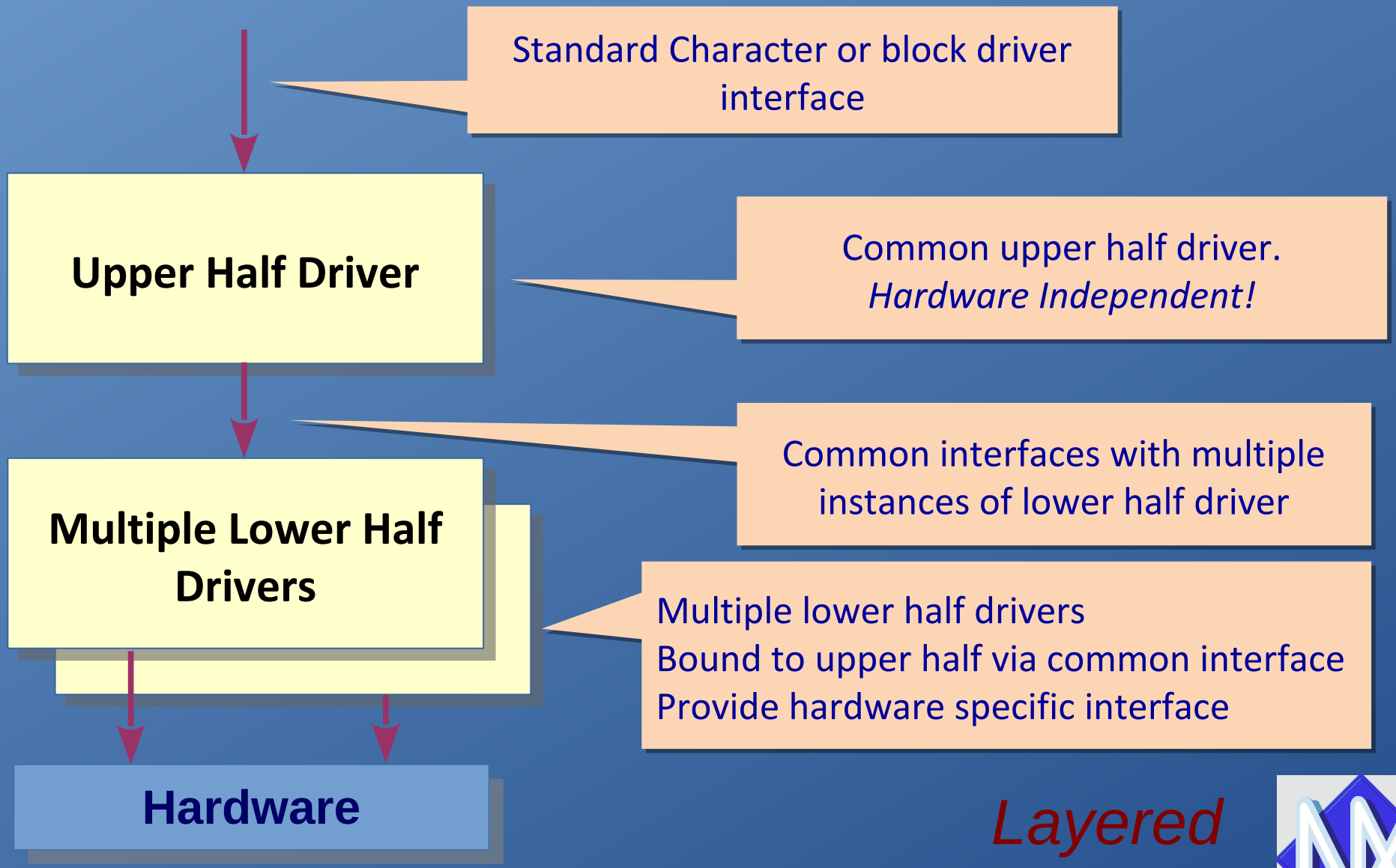
Architectural Overview – Device Drivers

Device Drivers Accessible to Applications via *Name Space*

- Standard *character* and *block* drivers plus *MTD* drivers
- *Special files* in VFS.
- Block drivers support file systems.
- Character drivers accessible via standard, POSIX interfaces like a file (open, close, read, write, etc.)
- MTD drivers support FLASH file systems

- Loop device – Convert file or character device to a block device
- BCH device – Convert a block device to a character device

Architectural Overview – Module Device Driver Design



Architectural Overview – FLAT Build

Applications, OS, and board logic exist in a common FLAT address Environment

FLAT – All addresses have same properties

Applications
Supervisor Privileges

OS + Board Logic
Kernel Space
Supervisor Privileges

Shared, common, heap

Architectural Overview – PROTECTED Build

Privileges mapped onto an otherwise *flat* address space by MPU – Memory Protection Unit

ARM7, ARMv7-M, ARMv7-R

Applications
User Space
User Privileges Only

OS + Privileged Board Logic
Kernel Space
Supervisor Privileges

Separate User/Kernel heaps

Architectural Overview – KERNEL Build

Kernel/User Address Environments Mapped With MMU - Memory Management Unit

MIPS, ARM9, ARMv7-A

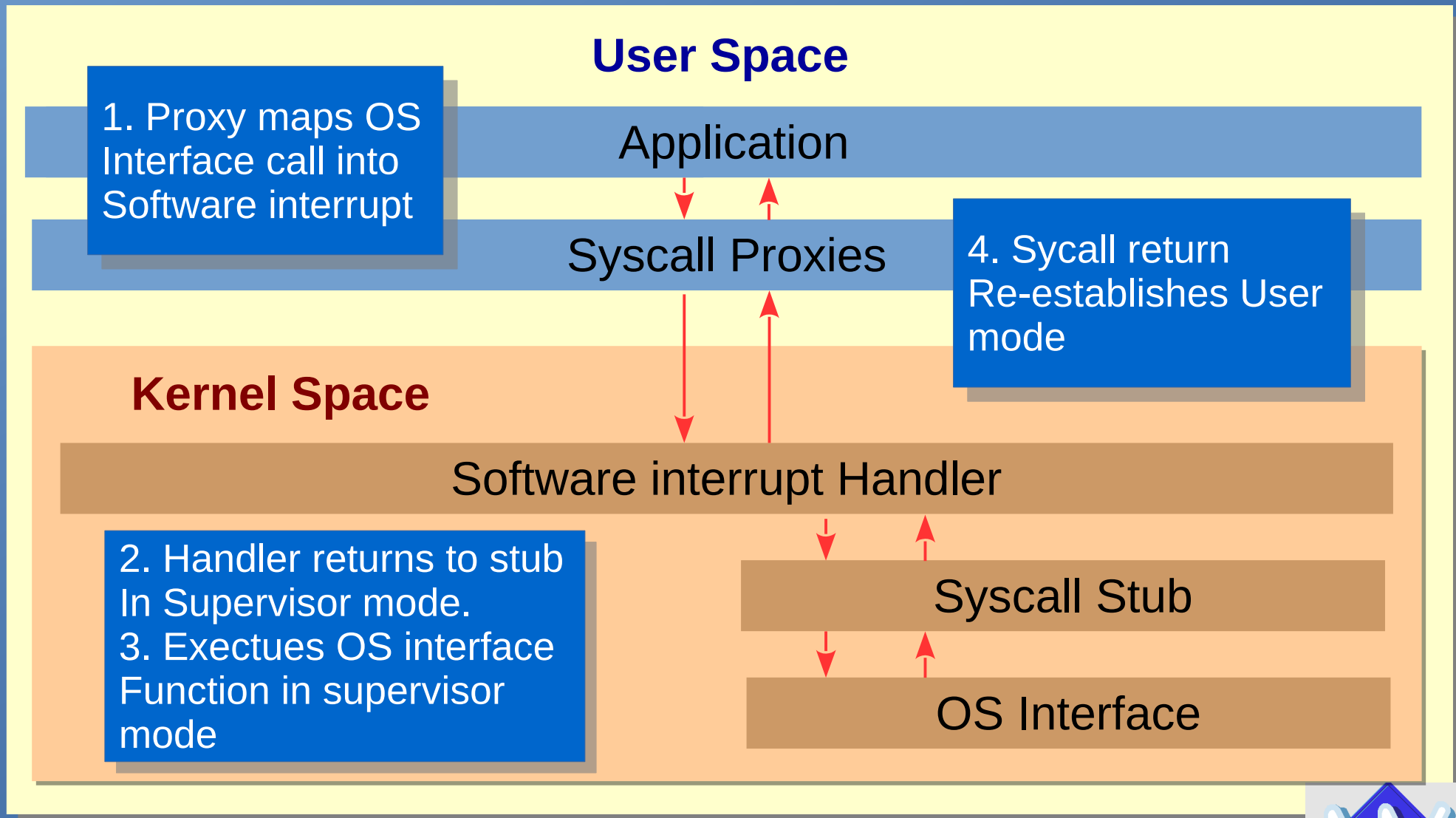
Multiple User Address Environments

Process 1 | Process 2 | Process 3 | … | Process 1

OS + Privileged Board Logic
Kernel Space
Supervisor Privileges

Kernel heap, per process virtual, on-demand heaps.

Architectural Overview – Call Gates



Architectural Overview – More Call Gates

- Implemented as System Calls. Via Software Interrupts
- Requires careful management of OS interfaces: All must be represented with system calls.
- Stubs and proxies for system calls automatically generated via CSV (comma separated value) file

Architectural Overview – *Future* TrustZone Build?

Cortex-A

ARMv7-A

?

ARMv7-R

Cortex-R

Cortex-M33

ARMv8-M

ARMv8-A?

Current Support: Cortex-A in multiple-OS, multi-core environments.