Windows CE 6.0 Kernel

Overview of the Windows CE Kernel

Windows CE Overview

- Targeted to embedded devices
  - PPC, Smartphones, STBs, Thin clients, AutoPC, PMC, control panels, robots, etc.
- Benefits
  - Flexible, adaptable, configurable, small
  - Supports ARM, MIPS, SH, x86
  - Real-time
  - Simple driver model
  - Power conscious
  - Shared source
  - Tiered licensing model

Windows CE Overview

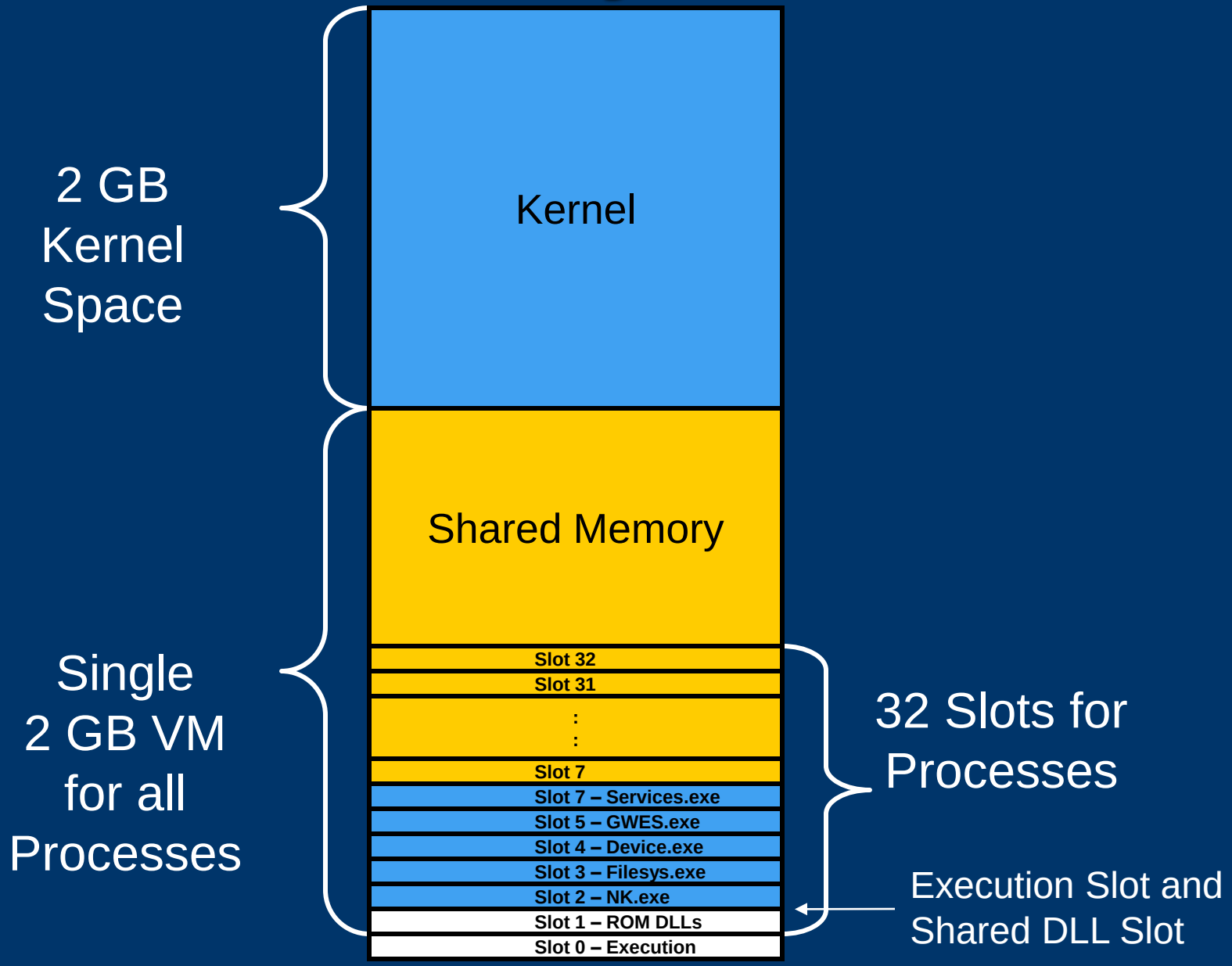
- However, CE 5.0 has a memory model limitation
- It only supports 32 processes and 32 MB per process
- These limitations has now been removed with CE 6.0
  - New Virtual Memory Model

CE 5.0 Overview

CE 5.0 Memory Model

- Virtual Memory Map
  - 2 GB for Kernel
  - Single 2 GB mapping for all processes
    - Divided up into 32 MB “slots”
- 32 Process Limit
  - Each process has one 32MB slot
  - 32 slots for processes
- Shared memory
  - Upper half of user space is shared memory
  - Read / Write by all processes

CE 5.0 Memory Model

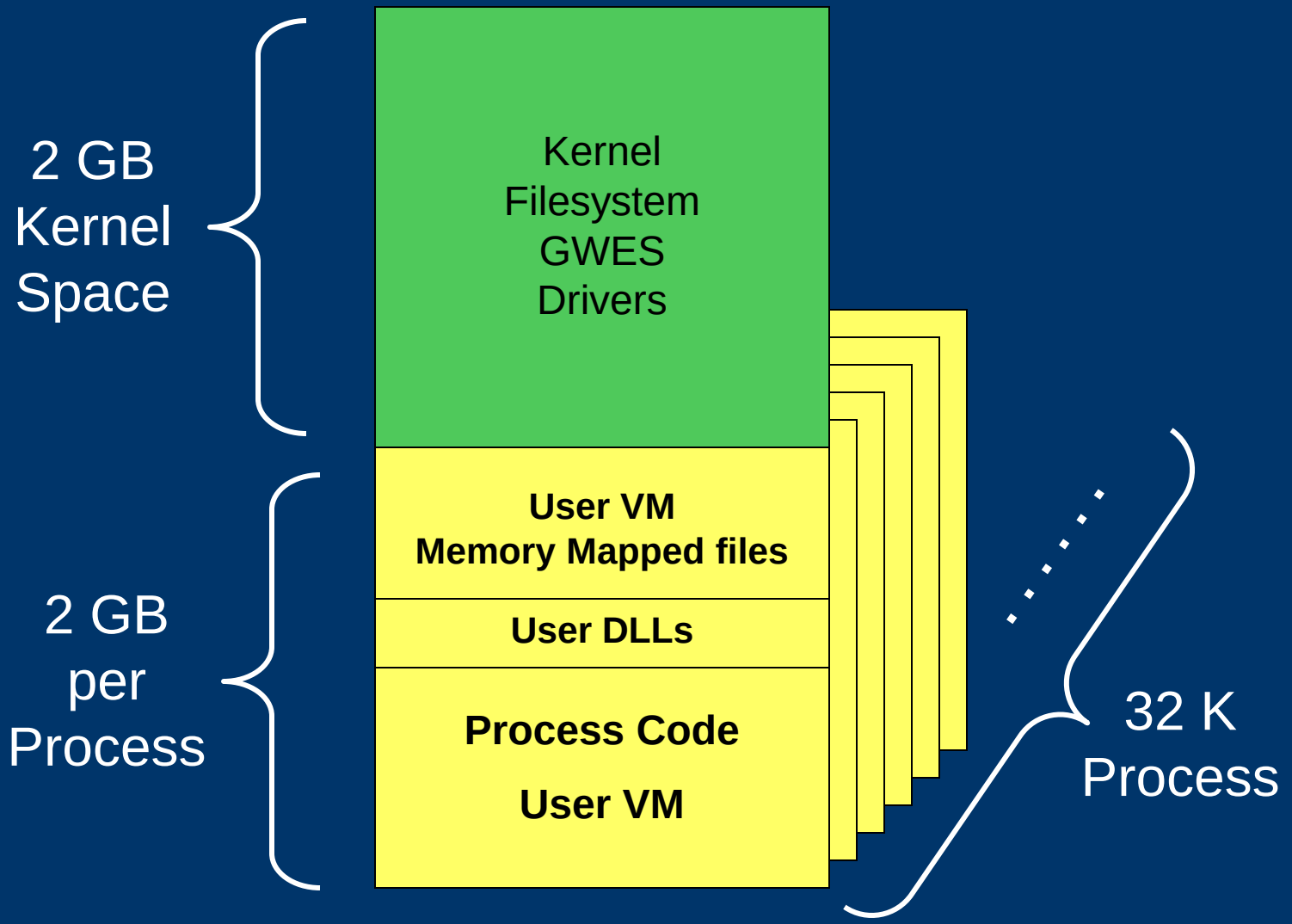


CE 6.0 Overview

The 6.0 kernel

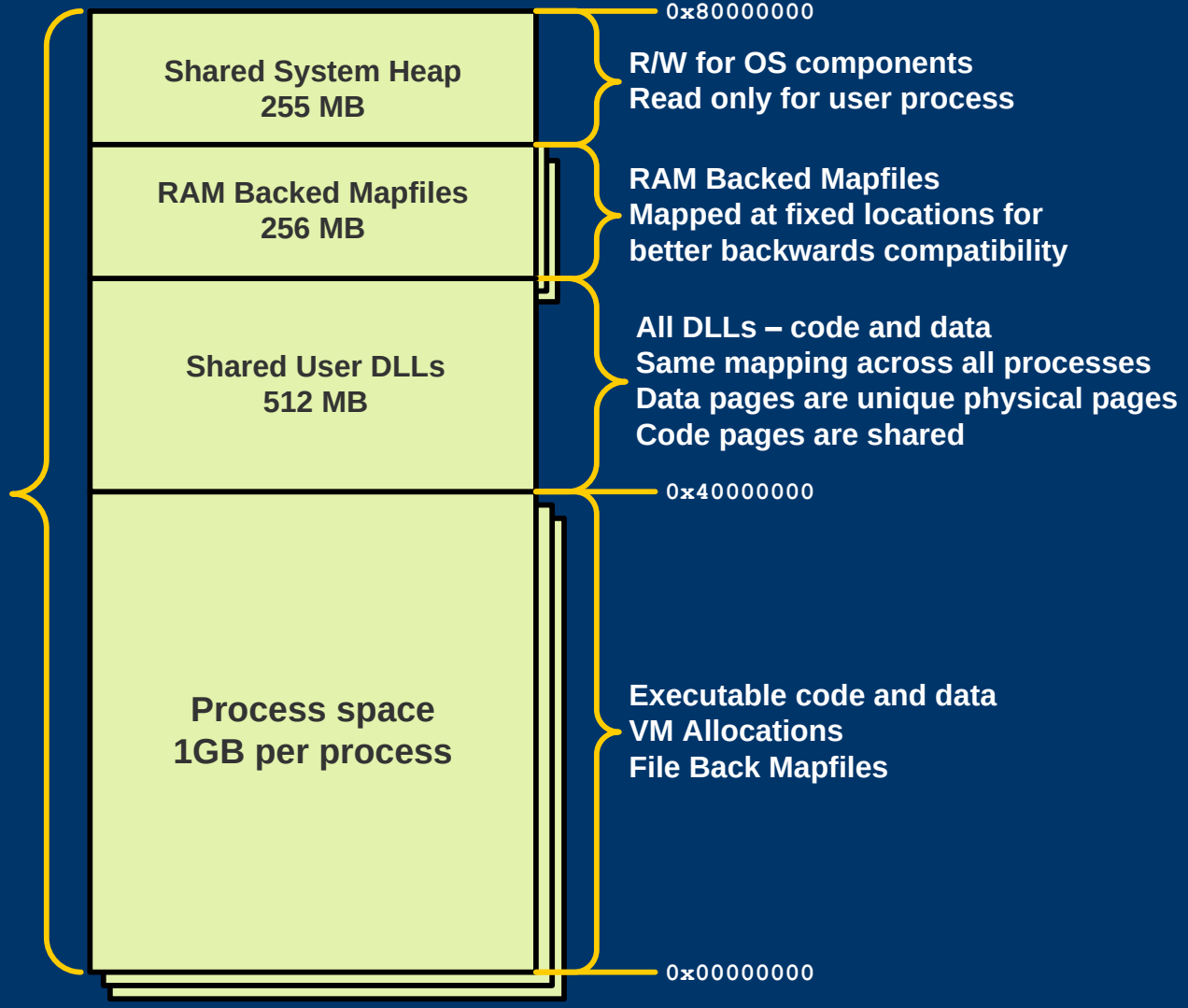
- 2 GB of Virtual Memory per process
- 32,000 processes
- Unified Kernel
  - Critical OS components moved into kernel space
- Improved system performance
- Increased security and robustness
- High degree of backwards compatibility

CE 6.0 Memory Model



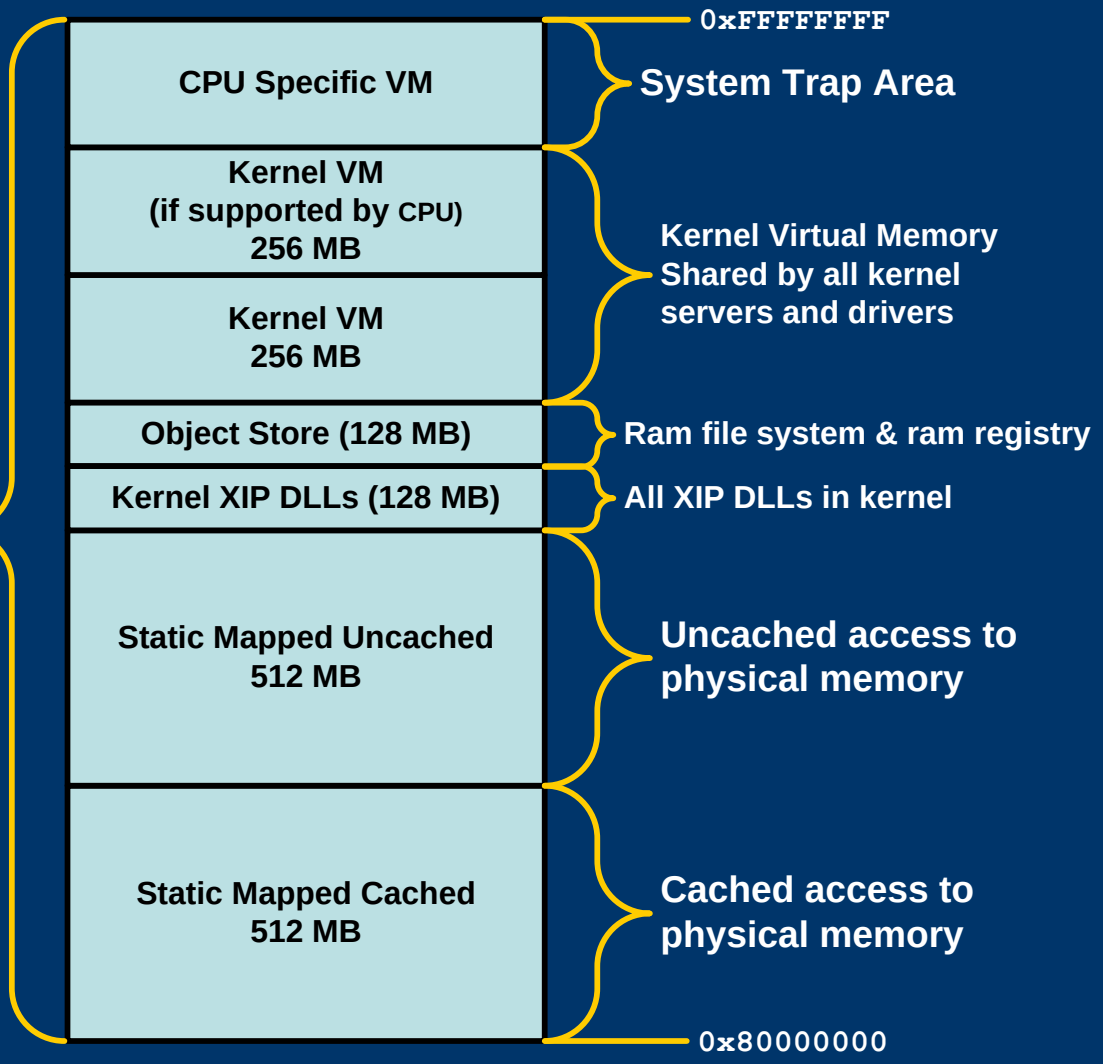
User Space

User Space
2 Gigabytes
Each process has its own mapping



Kernel Space

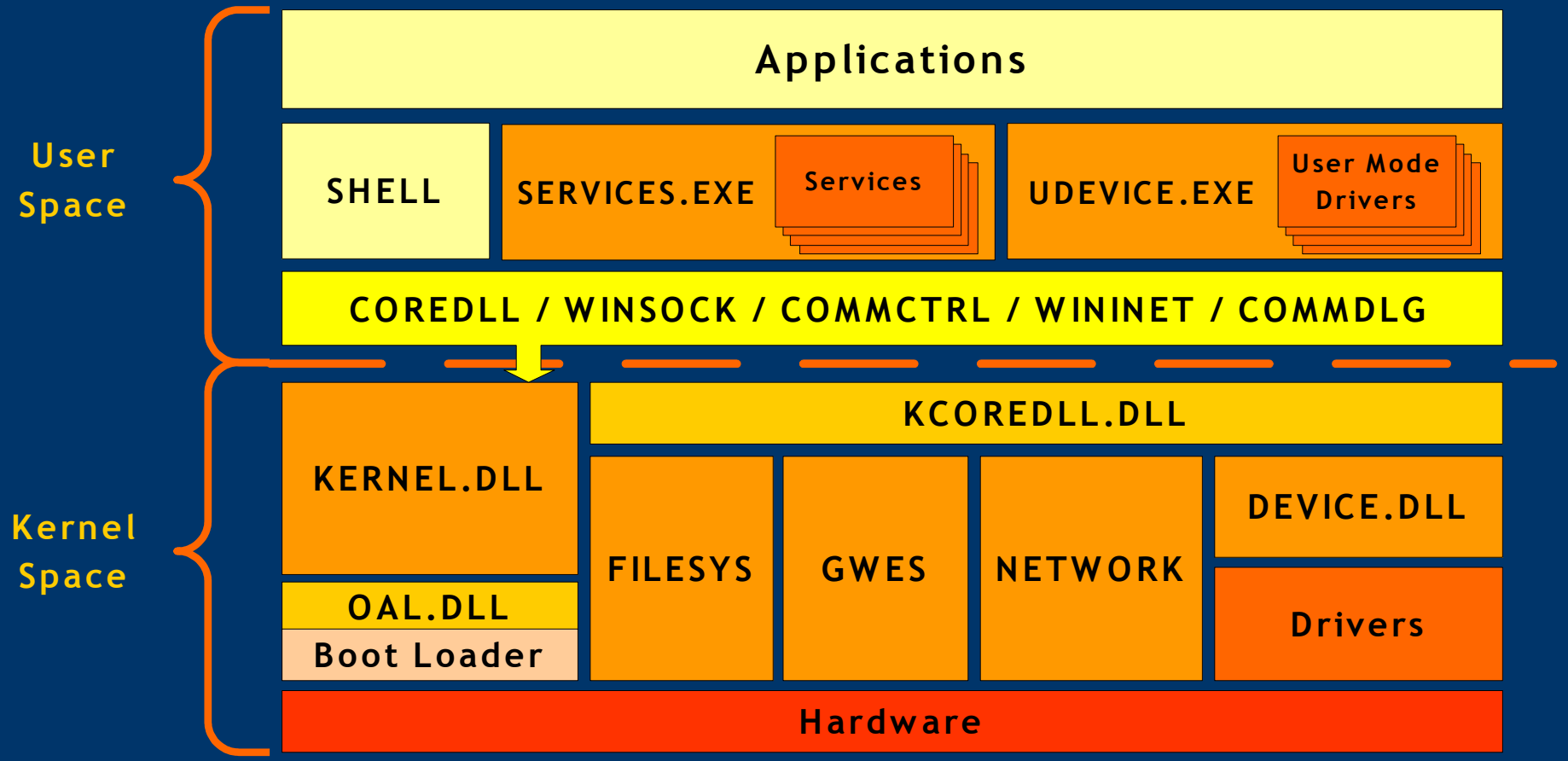
Kernel Space
2 Gigabytes
Fixed mapping
independent of
user space



OS Layout Changes

- Moving critical drivers, file system, and graphical window manager into the kernel
  - Kernel version of Coredll.dll
    - Same APIs without the thunks
- Benefit
  - Greatly reduces the overhead of system calls between these components
  - Reduces overhead of all calls from user space to kernel space
  - Increase code sharing between base OS services

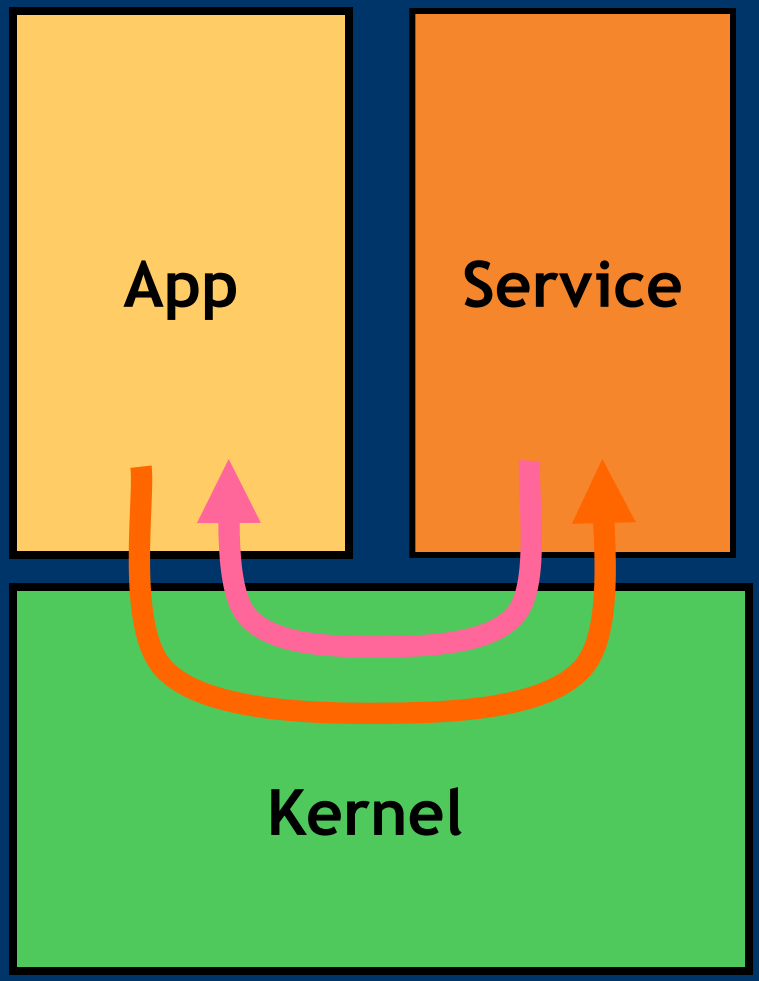
CE 6.0 OS Layout



Performance & Size

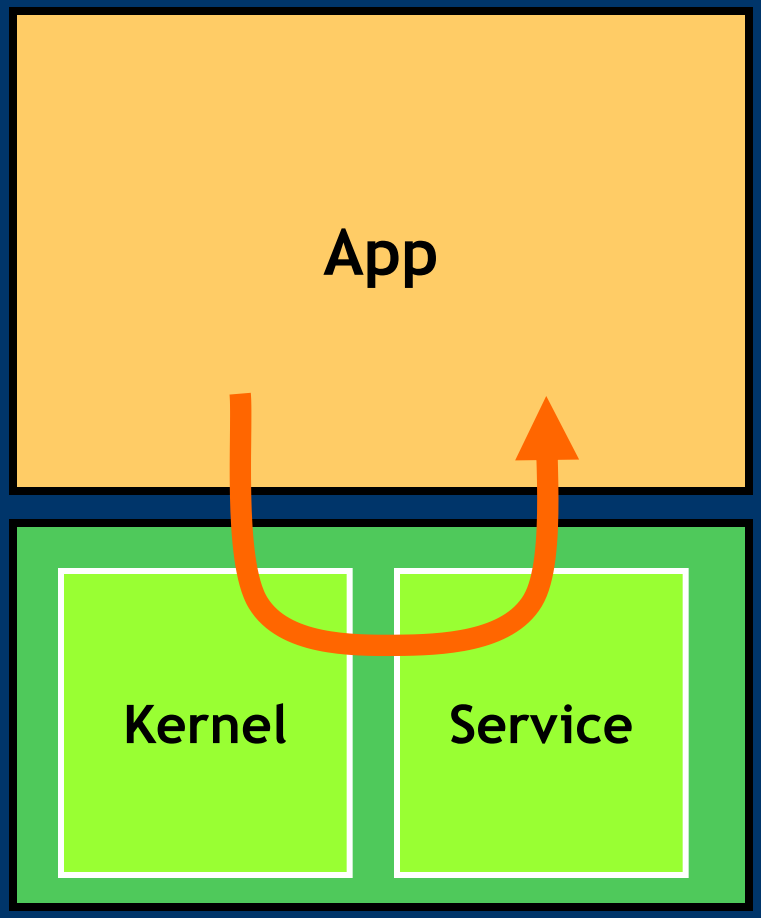
- Improvements expected in process switching
- Same performance
  - Thread Switching
  - Memory Allocation
  - System Calls
- Some slow down with interprocess calls
  - Now involves data marshalling
- Size increase is less then 5%

CE 5.0 System Calls



- Application makes call
  - PSL jump
- Kernel
  - Validates parameters
  - Maps Service into Slot 0
    - Possible Cache Flush
  - Calls into to the service
- Service
  - Runs
  - Returns to Kernel
- Kernel
  - Maps App into Slot 0
    - Possible cache flush
  - Returns to App

CE 6.0 Beta System Calls



- Application makes call
  - Same PSL jump
  - App stays mapped during the call
- Kernel
  - Copies call to kernel thread
  - Validates parameters
  - Calls into to the service
- Service
  - Runs
  - Returns directly to the app

Kernel Security Enhancements

Security

- Early Threat Modeling of the kernel
  - Working with MS Secure Windows Team and penetration testers
- Double checked design to tighten up
  - System Calls
  - Handles
  - Exception Handling
  - Memory Allocation
  - Loader
  - and many other components

Security and Robustness Features

- Improved parameter validation for system calls
- Per-Process Page and Handle tables
  - Greatly improves Process isolation
  - Improves code robustness
- Secure Stack
  - System calls run on special kernel side stacks
  - Safe guards system calls from stack tampering
- Robust Heaps
  - Heap control structures separated from heap data
- Safe Remote Heaps for OS components
  - OS servers can open heaps in user process
  - R/W for servers, R/only for user
  - Performance optimization and safe from tampering

CE 6.0 Features

CE 6.0 Features

- Per-Process Page Tables
  - Each process has its own page table
  - Pointers are unique to each process
  - Enables the new virtual memory model
  - Increases security
- Per-Process Handle Tables
  - Each process has its own handle table
  - Handles have reference and usage counts
  - Increases security
  - Increased programming robustness
    - no more stale handles

CE 6.0 Features Continued

- Large Memory Mapped File Support
  - Support for mapping views into very large files
  - Up to 64 bit files
  - Big benefit for in car navigation and multimedia
- Secure Loader
  - Enables of control of which executable and DLLs get loaded by the system
  - Uses encrypted signatures to identify the files
  - Foundation for a code based security model
    - Security is based on knowing what code is running instead of who the user is

CE 6.0 Features Continued

- User Mode UI service
  - Displays UI in user mode for kernel mode drivers
  - Keeps drivers from launching windows from inside the kernel
- Virtual Alloc Ex functions
  - Memory management functions for drivers
    - Just like the Windows XP APIs
  - Enables drivers to allocate memory in user processes

CE 6.0 Features Continued

- Marshalling Helper Functions
  - Helper functions for interprocess data marshalling
  - Services that help drivers to handle user data
- Monotonic clock
  - Always forward going clock independent of user clock
  - Enables services to calculate elapsed time

CE 6.0 Features Continued

- User Mode Services and Drivers
  - Run all services and some drivers in User Mode host
  - Saves kernel resources and increases robustness
- Separate OAL
  - OAL has been split from the kernel
  - Allows independent updates
    - Kernel updates and OEM OAL updates can be done independently
  - Enables easier device updates

Compatibility

CPU Requirements

- Currently the same as Windows 5.0
  - ARMV4I and above
  - MIPSII with sync instructions (ll, sc).
  - x86
  - SH4
  - Best performance on CPU's with Physical tagged caches
    - Virtual-tag-cached CPU have performance penalty and limitation on virtual mappings
- Same hardware as 5.0

Compatibility

- Binary compatibility for applications is the key goal
- The general structure of the OS will be the same
  - Will maintain compatibility in CoreDLL
    - Minimize impact on Win32 APIs
    - Changes hidden in API libraries
  - Continue to shared DLL code
- Well behaved SDK applications
  - Should work with little or no changes
- Apps using undocumented techniques
  - Will likely have to be modified
  - Such as passing handles or pointers between processes
- Main changes will be in how drivers access client memory
  - Some drivers will migrate with little work

Porting incompatible Apps

- Some applications will need work
  - Improper use of handles
  - nonstandard memory usage
  - Use of some CE specific APIs
  - Remove old tricks and workarounds
    - Such as handle sharing and pointer tricks
- Our verification approach
  - Ported Windows Mobile 5.0 to CE 6.0 Beta
  - Running 5.0 commercial applications on 6.0 Beta

Other porting

- Drivers will require some work
  - System calls
  - Use of worker threads
  - Access to caller's memory
- BSP will need some work
  - New memory mappings
  - Changes to OAL to support image updates

Drivers Overview

Drivers

- Two types of drivers will be supported
  - Kernel Mode for performance
  - User Mode for robustness
- The overall structure of the drivers remains the same
  - Main changes are in how the drivers access client memory
  - No SetKMode or SetProcPermissions

Kernel Mode Drivers

- Drivers are loaded in the kernel space by device.dll
- Have full access to the kernel's data structures and memory
- APIs used do not change
- Link to a kernel version of coredll.dll called kcoredll.dll
  - Thin layer for API compatibility
  - Directly links the services together without thunk layer
- Drivers needing the best performance should be kernel mode
  - Such as those with lots of quick calls

User Mode Drivers

- Loaded by udevices.exe
- Mostly the same APIs as Kernel Mode
- No access to kernel structures or memory
- Kernel will marshal parameters during system calls
- Examples
  - Expansion buses like USB and SDIO
  - Keyboard and touch
- Drivers where performance is not a factor should consider moving to user mode
  - Called less often and do more work

Handling Calls

- App memory already mapped correctly
  - Can access it without re-mapping pointers
- Don’t need – These APIs are being removed
  - SetProcPermissions
  - MapPtrToProcess, UnMapPtr
- Accessing callers memory
  - CopyIn / CopyOut
  - ReadProcessMemory / WriteProcessMemory
  - Virtual Alloc Ex APIs
- Marshalling Helper Library
  - Provides APIs for handling user data

OAL changes

- OAL split from kernel
  - Merged into NKLoader
  - Enables separate updates
- Overall OAL structure remains the same
  - Same OEM functions
  - Access kernel through kernel interface
- Changes to the OAL initialization
  - Memory mappings for new memory model

Real-Time

CE is a Real-Time OS

- Real time is being able to respond to an interrupt in a bounded maximum time
  - Analysis by OMAC User Group shows that 95% of real-time applications require between 0.5ms to 10 ms respond time
  - And tolerate 10% variations, or 50µs to 1ms jitter

Typical Real-Time Requirements

Real-Time Kernel

- CE achieves real-time by the design of the kernel and the drivers
  - The majority of the kernel and driver code can be interrupted
  - The uninterruptible parts are small discrete units so interrupts can be handled quickly
  - The length of the largest part is biggest latency
- CE 6.0 Beta kernel and drivers are designed with the same real-time constraints

Windows CE Test Results

- Respond time test using the following configuration
  - Samsung SMDK2410 development board
  - 200 MHz ARM with 16x16 cache
  - Windows CE 5.0 with full UI
  - Running a WMV video

Windows CE Real-Time Test Results

	ISR starts	IST starts
minimum	1.2 µs	31.7 µs
average	3.3 µs	67.2 µs
Maximum	13.3 µs	103.0 µs

Time in microseconds (µs)

CE 6.0 Real-Time

- The new kernel has the same response times as the current kernel
- May even perform slightly better because of reduced system call overhead
  - Performance between app and kernel will be better
  - Drivers and services in services.exe will be slightly worse