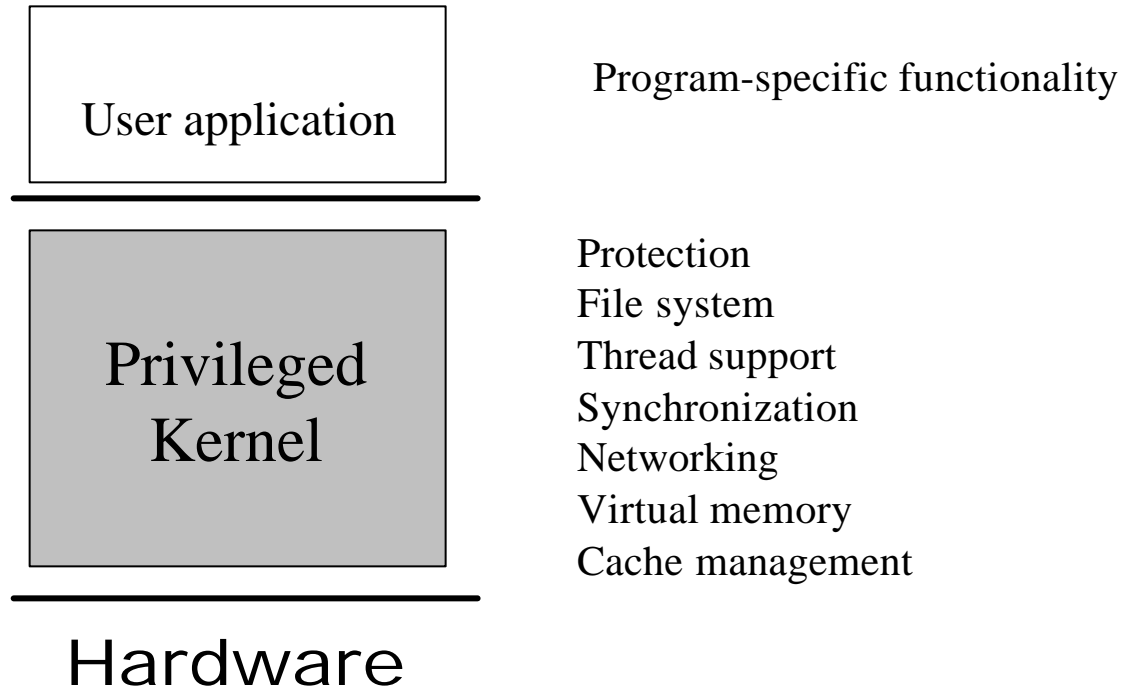


Extensibility, Safety and
Performance in the SPIN
Operating System

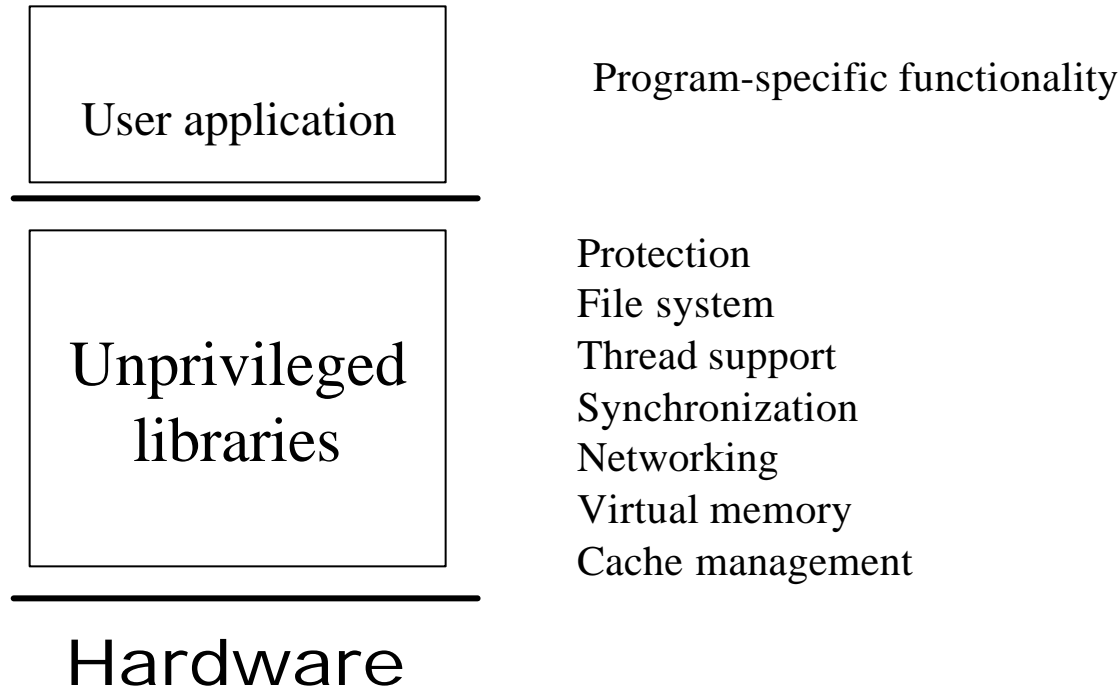
Brian Bershad, et al.

University of Washington

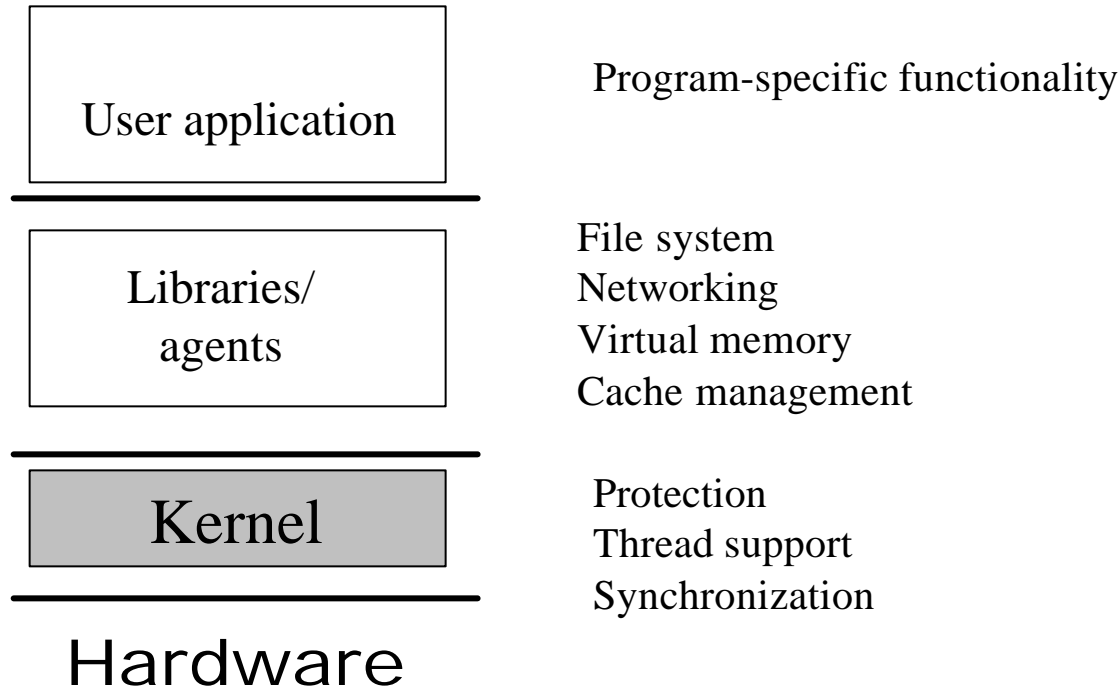
OS Structures - Monolithic



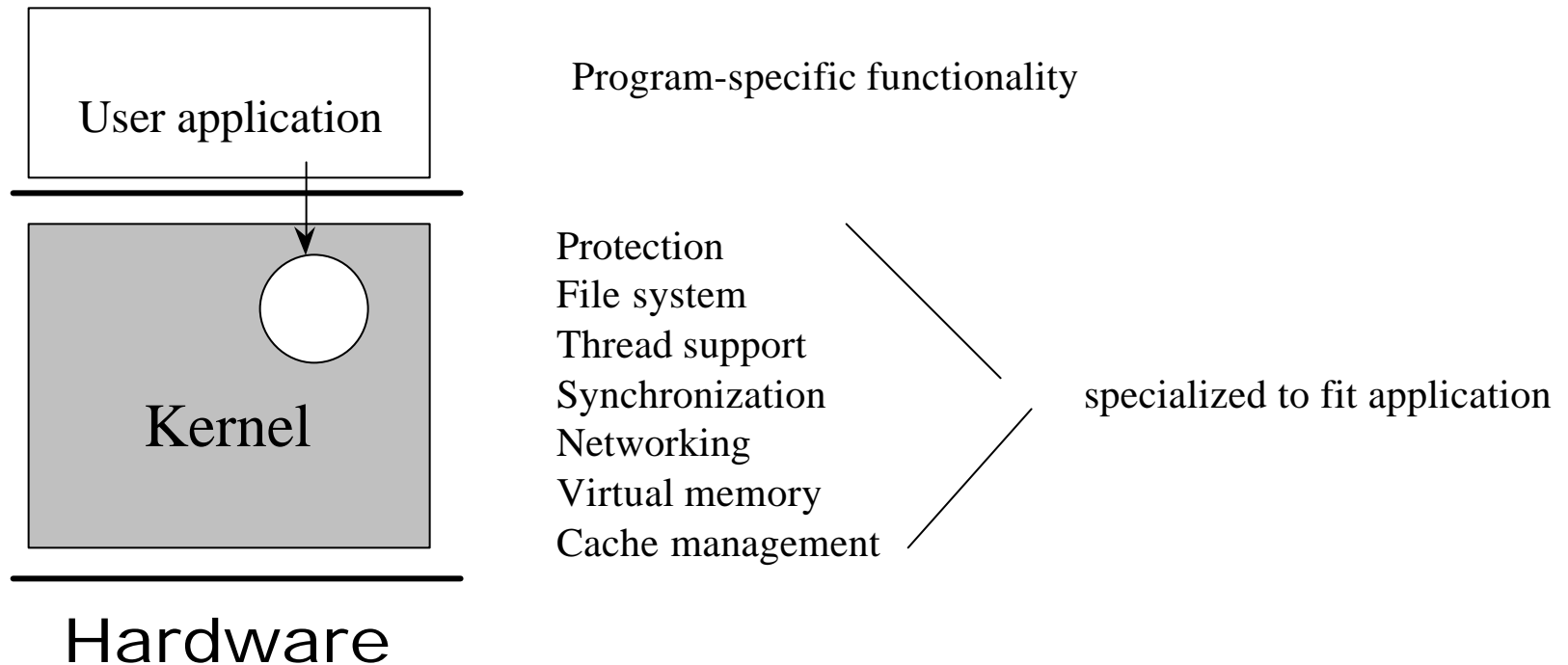
OS Structures – DOS/Windows



OS Structures - Microkernel



OS Structures - Extensible



SPIN goals

- Extensibility
 - Applications add functionality to the kernel (generally with the end goal of higher performance)
- Safety
 - Doing the above without compromising the OS
- Performance
 - Application performance is the end goal
 - Requires low-overhead in the extension mechanisms

SPIN ideas

- “An operating system can be extensible, safe, and fast through the use of *language* and *runtime services* that provide low-cost, fine-grained, protected access to operating system resources”
- Techniques:
 - Co-location – extensions become part of kernel
 - Enforced modularity – language-based isolation
 - Logical protection domains – name-based resolution between extensions, resolved to a procedure call via linking
 - Dynamic call binding – event-based handlers
- Core system services
 - Memory management
 - Scheduling

SPIN Extensions

- Granularity – procedure call
 - Extensions designed to have low run-time overhead
 - Rely heavily on compile-time and load-time checking
- All SPIN extensions are written in Modula-3
 - Compiled with a trusted compiler and executed with a trusted run-time
 - Requirements satisfied by Modula-3
 - Strict type-safety and interface management
 - Automatic memory allocation/deallocation
 - Other features (objects, threads, exceptions) useful but not necessary for SPIN's purposes

SPIN Protection Model

- A protection model controls the set of operations that can be applied to resources
 - e.g. an address space limits access
- SPIN approach
 - Rely on Module-3 type safety and dynamic linking
 - **Capabilities** – a reference to a system object or collection of objects
 - **Protection Domains** – defines what is accessible in an execution context, what can be done with a capability

Capabilities in SPIN

- Capabilities are simply Modula-3 object pointers in the kernel address space
 - Full compile-time checking by compiler for type safety
 - Automatically managed by Modula-3 runtime
- How are they passed outside to non-Modula3 entities (app programs)
 - Object references are stored in a per-application table
 - External representation is simply an index into that table

SPIN Protection Domains

- In SPIN, protection boils down to linking.
- Problem: must handle name conflicts, etc. in such a way that different extensions are unlikely to interfere with each other.
- Solution: explicit name-space creation and management
- A kernel nameserver advertises and authorizes protection domains

SPIN Extension Model

- How do extensions relate to the base system
 - What can they do?
 - How do they get triggered?
- Functionality
 - Passively monitor system activity (e.g. perf monitors)
 - Offer hints to guide the system (e.g. page replacement)
 - Actively replace a system service (e.g. thread scheduler)
- Invocation model
 - Events and handlers

Events and Handlers

- Every procedure call is also implicitly an event with the same name as the procedure
- Handlers
 - The original procedure is the default handler for an event
 - Multiple handlers can exist simultaneously
 - Modules can deny requests to register handlers
 - Handlers may be time-bound, synchronous, asynchronous, or simple calls (default)

SPIN Core Services

- Manage memory and processor resources
 - Implement the base extension services, and
 - Provide extensions something to build upon
- Extensible memory management
 - Handles physical storage, naming and translation
 - Exports concepts which correspond to typical Memory Management Unit constructs
 - Exceptions for “BadAddress”, “Page Fault”, etc.
 - PhysAddr.Reclaim clients may nominate alternative candidates

Core Services - 2

- Extensible thread management
 - Provides scheduling, concurrency and synchronization
 - Not a high-level thread package, but a simpler interface upon which packages can be built
 - Default implementation is trusted and places some restrictions on replacement implementations
 - Kernel-level threads and user-level threads are disjoint (again for safety reasons)
 - Kernel-level threads are actually Modula-3 threads

Evaluation

- What makes up SPIN
 - Module-3 run-time – 25%
 - Extension support – 5%
 - Core services – 20%
 - Device drivers – 50%