

SPIN

Objectives

- Enable extensibility – not as an afterthought, but build OS with an integrated extensibility model
- At the same time guarantee safety – have protection model
- Don't sacrifice performance

SPIN

- OS designed to be extended
 - set of core services
 - infrastructure which supports extensions
- Approach
 - use type-safe language and runtime
 - Modula-3
 - interfaces and type-safety and storage management
 - both extensions and OS written in Modula3!

Main techniques

- Co-location – os extensions dynamically linked into kernel address space -> low cost communication
- enforced modularity – compiler enforces interfaces and modular boundaries
- logical protection domains – kernel namespaces which contain code and exported interface
- dynamic call binding – execute extensions in response to system events with procedure call overhead

Protection model

- Address space too coarse grained
- SPIN uses capabilities – unforgeable references
 - Pointers / object references
- Outside of kernel – a level of indirections:
 - externalized via index table
 - (applications need not be written in modula3)
- Built around namespaces
- SPIN includes mechanisms for creating, linking, combine, managing namespaces...

Extension model

- Event – a msg, change of state
- Handler – a procedure that gets the event
 - extension == procedure associated with an event
 - central dispatcher
 - optimized if only single event
 - main module which owns event procedure determines policy for extensions
 - safety issues – compile and runtime checks, preemption, ordering, extra thread, etc...
- Guard
 - predicate on per instance basis
 - evaluation overhead...

Core Services: Memory management

- physical address service
 - allocate event; phy page + access rights; pass to translation service
 - raise deallocate, reclaim
- virtual address service
 - allocate, deallocate
- translation service
 - map phy and virtual address and update MMU
 - create, destroy, add, remove, examine – mapping
 - pagenotpresent, badaddress, protectionfault

Core Service: Thread Management

- Strand – named processor context
- block, unlock, checkpoint, resume

Performance issues

- size – ok
- microbenchmarks
 - protected calls, thread management, vmm
 - direct method calls cheap
 - => support fine grained access control/interfaces
- benefits of extensibility