Administration

• Course newsgroup
  – ubc.courses.cpsc.508
• Paper assignments updated
• Who isn’t registered?
• Who hasn’t sent me paper selections?
Hydra: The Kernel of a Multiprocessor Operating System

Wulf, Cohen, Corwin, Jones, Levin Pierson, Pollack

CMU

Presentation by: norm
Timeline

• In 1971 CMU decided to build a “multi-mini-processor”
  – This became C.mmp

• At the same time, work on the kernel of the OS began
  – HYDRA

• They also worked on language design, debugger design, IDEs, …
The hardware

Switch

M_0
M_1
...
M_m

P_0
P_1
...
P_n
C.mmp

- 16 processors
- 32 MB shared memory
- Each processor is a PDP-11
- Each processor has a memory controller to manage access to memory
Design philosophy

• A collection of facilities of universal applicability and absolute reliability
• … from which an arbitrary set of operating systems can be conveniently, flexibly, efficiently, and reliably constructed
• … the resulting operating systems able to co-exist simultaneously.
Considerations

• Multiprocessor environment
• Separation of mechanism and policy
• Integration of the design with the implementation methodology
• Rejection of strict hierarchy
• Protection
• Reliability
What is an Operating System?

• It defines an abstract (virtual) machine that is more convenient than the bare hardware
• It allocates hardware resources
The architecture

- Objects
- Types
- Procedures
- Local name space (LNS)
- Process
- Capability
Object

- Unique name
- Type part
- Representation
  - Capability part
  - Data part
Type

- An object whose type part is a distinguished object named TYPE
- TYPE itself is a type, by virtue of naming itself in its own type part
Procedure

• An object
• Data part
  – executable instructions
• Capability part
  – Caller independent
  – Caller dependent
    • Holes
    • Parameter positions
    • Templates
Local name space (LNS)

• The environment of an executing procedure
• Consists of
  – Caller-independent capabilities of the procedure
  – Caller-dependent capabilities
    • “Derived” from the parameters
    • Not the same as the parameters
Process

- A stack of LNS’s
- Unit of asynchronous execution
- Creating lots of processes allows you to exploit the multi-processor nature of C.mmp
Capability

- Reference to an object
- Set of access rights
  - Kernel rights
  - Auxiliary rights
- It is impossible to forge a capability
Call and Return

• Call creates a new LNS
  – Caller-independent capabilities
  – Derived caller-dependent capabilities
  – Jump to the code

• Return destroys the current LNS, returning to the instruction following the CALL in the previous LNS
Parameter passing

• A procedure template contains:
  – A type attribute – the required type of the parameter
    • There is a wildcard ANY type attribute
  – Access rights
    • Check field
      – Rights required of the parameter
    • Regular field
      – Rights included in the capability in the new LNS
Paths

• The “walk” primitive returns a capability in the capability list of a target object
• Your capability for the target object must include the “walk” right
Bibliography example
Considerations - review

- Multiprocessor environment
- Separation of mechanism and policy
- Integration of the design with the implementation methodology
- Rejection of strict hierarchy
- Protection
- Reliability
Discussion

• Is there a similarity between HYDRA and today’s micro-kernel operating systems?
Discussion

• Who cares about an operating system meant for a collection of 16 PDP-11? How does that hardware compare to today’s?
Discussion

• In the old days, why do all the operating systems (Hydra, Multics) choose a heavy-weight protection scheme?
• Why doesn't Linux choose to use a heavy-weight protection scheme?
• Are there any contemporary O/Ses that use a heavy-weight protection scheme?
Discussion

• Can type mismatches (part of the parameter checking process) be detected at compile time?
Discussion

• Capabilities in a procedure in Hydra need not be stored in the local capability, but can be (using walk) in the capabilities of an object that is referenced in the current capability.

• Does that mean that for every access to an object, the kernel walks the entire list of capabilities until it finds the match to the object type?
Discussion

• “This implies that a callee may have greater freedom to operate on an object than the caller.”

• Will this allow security breaches if an object amplifies its rights to a global object by way of another object it calls?
Discussion

• Why does the separation of mechanism and policy seem so important?
• What is an example where this separation gains you something?
Discussion

• What rights do you need to create a template within a procedure that amplifies the rights in a given parameter capability?
Discussion

• HYDRA rejects the notion of ownership, Multics embraces it (the owner has absolute rights to a segment it creates).
• Which is better?
• For what situations is one to be preferred over another?
• What happens (in HYDRA) if you accidentally lose critical rights to a critical object?