Object-Oriented Operating Systems

Object-oriented systems are characterised by

encapsulation: objects are abstract data types,
operation are performed by invoking *methods*
or sending *messages*

extensibility: types can be extended, producing derived types
inheriting other types' attributes

late binding: variables can be of generalised type
initialised at run-time with objects of specific type
method actually invoked depends on actual type of object at run-time

What does this mean for operating systems?

OO concepts in operating systems

Encapsulation can be achieved by putting objects into separate *address spaces*

- typically one address space per type (*class*) or class hierarchy
- invocation via RPC according to well-specified interface
- invocation requires *unique identifiers* (as for IPC)
- can restrict callability by "hiding" or protecting entry points ("doors", "portals")

Address space constitutes *server* for objects

OO concepts in operating systems...

Extensibility can be achieved by

- adding to servers, *or*
- providing new (derived type) servers
  which invoke the (base type) servers as appropriate
  ⇒ easy to support *multiple inheritance*

OO concepts in operating systems...

Late binding can be achieved by including an (explicit or implicit) server reference in the object reference

- if implicit need a *naming system* ("name binding") to find server
- which server may depend on actual type
- server reference needs to be protected from forgery to ensure integrity of type system

⇒ *capabilities*

Note: late binding allows extensibility at run-time
by starting up a new server

OO concepts in OS: Summary

- object orientation maps nicely to a server-based approach using message-passing communication
- an efficient μ-kernel is the ideal basis for an OO-OS
- Alternatives: *grafting*
  allow "safe" kernel extensions by run-time loading of kernel modules
  - avoids IPC overhead
  - needs trusted compilers

OO-OS Examples (highly incomplete list)

Hydra (CMU) W. Wulf, E. Cohen, W. Corwin, A. Jones, R. Levin, C. Pierson, and F. Pollack: HYDRA: The kernel of a multiprocessor operating system. *Comm. ACM*, 17:337–345, 1974.

Amoeba (Vrjie U) S.J. Mullender and A.S. Tanenbaum: The design of a capability-based distributed operating system.

Spring (Sun) Y.A. Khalidi and M.N. Nelson: An implementation of UNIX on an object-oriented operating system. Technical Report TR-92-03, Sun, December 1992.

Guide (Bull) D. Hagimont: Protection in the Guide object-oriented distributed system. In *Proc. 8th European Conf. Obj. Orient. Progr. (ECOOP)*, pages 280–298, Bologna, Italy, July 1994. Springer Verlag, Berlin.

We'll encounter some of these again...

