HYDRA: The Kernel of a Multiprocessor Operating System
W. Wulf, E. Cohen, W. Corwin, A. Jones, R. Levin, C. Pierson, and F. Pollack
Communications of the ACM 17(6), June 1974, pp. 337-344.

1. Hydra is an example of a kernel or nucleus of an OS. As such, what were its goals?

2. What is the role of HYDRA?

3. What does each object in HYDRA contain?

4. What other options exist instead for protection instead of capabilities? Where are other approaches popular?

5. What does a capability contain? HYDRA capabilities contain generic (kernel) and type-specific (auxiliary) rights. Generic rights include rights such as copying an object, getting, putting data of an object; manipulating capabilities (read, deleting, adding). Why are type-specific rights needed?

6. One of the most important properties of capabilities is that they can't be forged. How can this be ensured? Can you think of where capabilities are used in UNIX?

7. What is in an object of Type Procedure (i.e., Name, Data, and Capability list)? What is a caller-independent capability? What is a caller-dependent capability and how is it implemented?

8. What happens when a procedure is called? What is problematic about this and what is the solution?

9. Why is rights amplification useful? Can you think of an example of where this is used in UNIX? Why can masking away some rights be useful?

10. What is an LNS (Local Name Space)? Why is it not the same as the Procedure Object? How does a Procedure access capabilities that are not local? Is this a correctness or a convenience issue?

11. What is in an object of a Process Type? What types of operations on a Process should the scheduler have rights to? What operations should the scheduler NOT have rights to?

12. What are the strengths of Hydra capabilities? What are their problems?