

1. A fundamental aspect of protection in operating systems is rights amplification. Rights amplification enables a more privileged protection domain to perform an operation on behalf of a less privileged protection domain in a controlled fashion without violating protection in the system. For each of the following operating systems, state (a) the protection domain that they support, (b) the mechanism for crossing protection domains, (c) how rights are represented, (d) how rights are amplified crossing domains, and (e) how the OS determines whether to allow the domain crossing.

1. Hydra
2. Multics
3. Unix
4. Pilot

Support your answers with a bit of explanation, such as a concise summary explanation in your own words (a quote of a phrase or sentence from the papers is fine as well). For instance, two possible answers to part (a) for Hydra are:

A protection domain in Hydra is the "local name space" (LNS). An LNS represents the current set of objects and rights to which a process has access, and those objects and rights change when a process moves from one LNS to another.

A protection domain in Hydra is the "local name space" (LNS): "At any instant, the execution environment (domain) of a program is defined by an LNS object associated with it...the rights lists in each capability define the permissible access rights of this program at this instant." (Hydra p. 341).

In other words, we're looking for more than just "local name space" -- but at the same time your answers don't have to be lengthy discussions. The balance in the example above is fine.

Solution for problem 1:

1.1 Hydra

1.2 Multics

1.3 Unix

1.4 Pilot

2. Some of the systems we have read about and discussed use specialized hardware to facilitate their implementation. Choose one such instance, describe the hardware that was used, and what advantage it gave the system implementors and designers. What is one drawback of relying upon specialized hardware? Do we still use hardware of this form today?

Solution for problem 2:

3. Pilot made a strong and persuasive argument for tailoring the design and implementation of operating systems to personal computers. We have also seen commercial operating systems like MS-DOS, Windows before NT, and "classic" MacOS tailored towards personal computers as well. Why do you think we still run multi-user timesharing systems like Unix on our PCs? (Consider, for example, the requirements we have of the systems that we use today.)

Solution for problem 3: