Protection in General Purpose OS

Adapted from [Pfl96, Chapter 6].

Why protect the OS?

Operating Systems support multiprogramming so they've developed ways to protect the computation of one user from inadvertent or malicious interference from another.

- Executives. Provided linkers and loaders
  - Provided linkers and loaders for relocation.
  - Provided easy access to compilers, assemblers.
  - Provided automatic loading of subprograms from libraries.
- Monitors.
  - Provides scheduling, sharing, and parallel use of resources—memory, I/O devices, sharable programs and data.
  - Oversaw all computing.

Security Methods of OSes

The basis of protection is **separation**.

- Physical Separation. Poor resource utilization.
- Temporal Separation. Poor resource utilization.
- Logical Separation.
- Cryptographic Separation.

Separation is only half the answer. The other half is controlled sharing.

- All or Nothing.
- Access control on objects.
- Capabilities.
- Partial use of objects.
  - E.g., the ability to read but not print as in Adobe reader.
  - Collect statistics from a database but not the actual records.

What is the granularity of sharing?

Protecting Memory

- Using a **fixed fence** with non-relocatable programs that know the fence value at compile time.
  - Protection in only one direction. You can shoot "yourself" in the foot.
  - Cannot sub partition programs into finer granularity of protection.
- Using **variable sized fences** with programs compiled starting at address 0.
  - Programs not relocated, rather indirection used with the fence value stored in a register.
  - Provides "relocation" and protection at the same time.
- Base bounds registers.
  - Provide both lower and upper bounds.
  - Change it for every program at context switch.
  - Use additional base bounds registers for finer granularity partition– say code and data.

Protecting Memory–Segmentation

- Addresses are of the form <seg #, offset>.
- Segments can be separately relocated and protected.
- Each process would normally have its own segment table for address translation.
- Processes that want to share segments map them to the same segment numbers in their segment table.

Pros and Cons:

- Fine granularity of protection, on a per-segment basis.
- Can lead to fragmentation of main memory. Requires compaction.
- Sharing requires same segment numbers in all sharing processes because of inter segment references.

Paging

- Addresses are of the form <page #, offset>.
- All pages are of the *same* size.

Segmentation+Paging

- Break a segment into pages.

References

[Pfl96] Charles P. Pfleeger. *Security in Computing.* Prentice Hall, iind edition, 1996.