10.5 Controlled Virtual Address Space

This pattern addresses how to control access by processes to specific areas of their virtual address space (VAS) according to a set of predefined access types. Divide the VAS into segments that correspond to logical units in the programs. Use special words (descriptors) to represent access rights for these segments.

Example

Our operating system improved by using a reference monitor. However, hackers discovered that the unit of access control to memory was coarse. By taking advantage of the lack of precision in controlling access they were able to access other processes' areas.

Context

Multiprogramming systems with a variety of users. Processes executing on behalf of these users must be able to share memory areas in a controlled way. Each process runs in its own address space. The total VAS at a given moment includes the union of the VASs of the individual processes, including user and system processes. Typical allowed accesses are read, write, and execute, although finer typing is possible.

Problem

Processes must be controlled when accessing memory, otherwise they could overwrite each other's memory areas or gain access to private information. While relatively small amounts of data can be directly compromised, illegal access to system areas could allow a process to force a higher execution privilege level and thus access files and other resources.

The solution to this problem must resolve the following forces:

- There is a need for a variety of access rights for each separate logical unit of VAS (segment). In this way security and controlled sharing are possible.
- There is a variety of virtual memory address space structures: some systems use a set of separate address spaces, others a single-level address space. Further, the VAS may be split between the users and the operating system. We would like to control access to all of these types in a uniform manner.

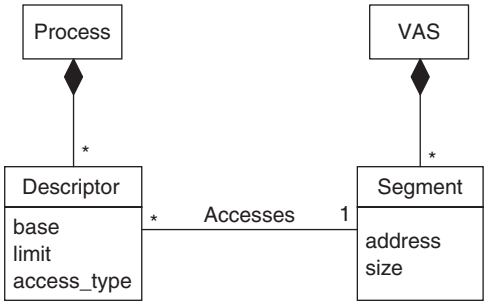
- For any approach to be efficient, hardware assistance is necessary. This implies that an implementation of the solution will require a specific hardware architecture. However, the generic solution must be hardware-independent.

Solution

Divide the VAS into segments that correspond to logical units in the programs. Use special words (descriptors) to indicate access rights that show the starting address of the accessible segment, the limit of the accessible segment, and the type of access permitted (read, write, execute).

Structure

The figure below shows a class diagram for the solution. A process (the `Process` class) must have a descriptor (the `Descriptor` class) to access segments in the VAS.



Class diagram for CONTROLLED VIRTUAL ADDRESS SPACE

Implementation

Some implementation aspects include:

- The limit check when accessing an address must be done by the instruction microcode or the overhead would not be acceptable. This check is part of an instance of REFERENCE MONITOR (256)—see Chapter 8.
- The same idea applies to purely paging systems, except that the limit in the descriptor is defined by the page size. In paged systems pages do not correspond to logical units and cannot perform a fine security control.

- There are two basic ways to implement this pattern:
  - Property descriptor systems. The descriptors are loaded at process creation by the operating system. The descriptors are handled through special registers and disappear at the end of execution.
  - Capability systems. A special trusted portion of the operating system distributes capabilities to programs. Programs own these capabilities. To use them, the operating system loads them into special registers or memory segments.

  In both cases, access to files is derived from their ACLs.

Example Resolved

Descriptors can control areas of memory of any size. A process without a descriptor for an area cannot access it. If sharing is required, several processes can have a descriptor with the same addresses but with different access rights.

Known Uses

The Plessey 250 [Ham73], Multics [Gra68], IBM S/38, IBM S/6000, Intel X86 [Chi84], and Intel Pentium use some type of descriptors for memory access control. The operating systems in these machines must use this approach for memory management. Specific uses include the Choices operating system [Rus89] and AIX [Cam90].

Consequences

The following benefits may be expected from applying this pattern:

- The pattern provides the required segment protection, because a process cannot access a segment without a descriptor for it. Two processes with descriptors with the same memory address base–limit pair[1] can conveniently share a segment.
- The pattern applies to any type of virtual address space: single, segregated, or split.
- If all resources are mapped to the virtual address space, the pattern can control access to any type of resource, including files.

[1] This relates to a simple method of enforcing memory protection by adding two registers to the CPU, a base address and a size limit, which together demarcate a range of memory to which valid references can be made. References outside that range trigger a memory exception. This works well as long as all memory is allocated contiguously, but non-contiguous memory is harder to protect, as is sharing memory between more than two processes.

The following potential liabilities may arise from applying this pattern:

- Segmentation makes storage allocation inefficient because of external fragmentation [Sil03]. In most systems segments are paged for convenient allocation.
- Hardware support is needed, which puts an extra requirement on this solution.
- In systems that use multiple separate address spaces, it is necessary to add an extra identifier to the descriptor registers to indicate the address space number.

See Also

This pattern is a direct application of AUTHORIZATION (245) to the processes' address space.