

19 PROTECCIÓN

- 1 PROPÓSITO DE LA PROTECCIÓN
- 2 DOMINIO DE PROTECCIÓN
- 3 MATRIZ DE ACCESO
- 4 IMPLEMENTACIÓN DE LA MATRIZ DE ACCESO
- 5 REVOCACIÓN DE DERECHOS DE ACCESO
- 6 SISTEMAS BASADOS EN CAPACIDAD
- 7 PROTECCIÓN BASADA EN LENGUAJE

19.1 PROPÓSITO DE LA PROTECCIÓN

- Un Sistema Computacional está constituido por una colección de Objetos de hardware y software.
- Cada objeto tiene un nombre único y puede ser accedido a través de un conjunto bien definido de operaciones.
- Problema de la protección: asegurar que cada objeto sea accedido correctamente y sólo por aquellos procesos a quienes se les ha permitido hacerlo.

19.1 PROPÓSITO DE LA PROTECCIÓN

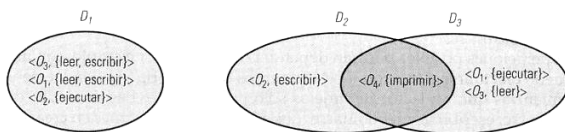
- La **Protección** es un Conjunto de Mecanismos para controlar el acceso de los programas, procesos o usuarios a los recursos definidos en un Sistema Computacional.
- Los mecanismos deben permitir especificar los controles a imponer y deben contar con algún medio de fuerza para hacerlos cumplir.

19.2 DOMINIO DE PROTECCIÓN

- 1 Estructura de Dominios
- 2 Ejemplos

19.2.1 Estructura de Dominio

- Derecho de acceso = $\langle \text{nombre de objeto, conjunto de derechos} \rangle$.
Conjunto de derechos es un subconjunto de todas las operaciones Válidas que pueden ser realizadas sobre el objeto.
- Dominio = conjunto de derechos de acceso.



19.2.1 Estructura de Dominio

- Cada usuario podría ser un dominio: El conjunto de objetos al que se puede acceder depende de la identidad del usuario
- Cada proceso podría ser un dominio: depende la de identidad del proceso. (receptor o emisor)
- Cada procedimiento podría ser un dominio: depende de las variables locales definidas dentro del procedimiento.

19.2.2 Ejemplo

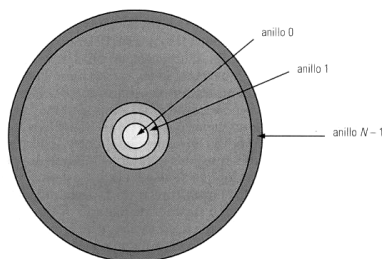
- Modo dual
 - Dominio monitor
 - Dominio usuario
- En un sistema operativo multiprogramado ¿es suficiente este tipo de protección?

19.2.2 Ejemplo: UNIX

- El sistema consiste en 2 dominios:
 - Usuario
 - Supervisor
- Unix
 - Dominio = user-id
 - El cambio de dominio se lleva a cabo via el sistema de archivos
 - Cada archivo tiene asociado con él un bit de dominio (setuid bit)
 - Cuando un archivo es ejecutado y setuid = on, entonces user-id toma el valor del nombre del propietario del archivo ejecutado. Cuando termina la ejecución, user-id retoma su valor original
 - Unix es vulnerable en este aspecto con setuid.

19.2.2 Ejemplo: MULTICS

- Sean D_i y D_j dos anillos de dominio.
- Si $j < i \Rightarrow D_i \supseteq D_j$



19.3 MATRIZ DE ACCESO

Matriz de Acceso: $\text{acceso}(i,j)$, $\rightarrow$ acceso D_i a O_j

dominio \ objeto	objeto			
	F_1	F_2	F_3	impresora
D_1	leer		leer	
D_2				imprimir
D_3		leer	ejecutar	
D_4	leer escribir		leer escribir	

19.3 MATRIZ DE ACCESO

Matriz de Acceso con Dominios como Objetos: conmutación del dominio D_i a D_j si el derecho de acceso conmutar pertenece a $\text{acceso}(i,j)$

dominio \ objeto	F_1	F_2	F_3	impresora láser	D_1	D_2	D_3	D_4
D_1	leer		leer			conmutar		
D_2				imprimir			conmutar	conmutar
D_3		leer	ejecutar					
D_4	leer escribir		leer escribir		conmutar			

19.3 MATRIZ DE ACCESO

Matriz de Acceso Modificada: $\text{acceso}(i,j)$, incluye control, un proceso que se ejecuta en el dominio D_i puede quitar cualquier derecho de la fila j

dominio \ objeto	F_1	F_2	F_3	impresora láser	D_1	D_2	D_3	D_4
D_1	leer		leer			conmutar		
D_2				imprimir			conmutar	conmutar control
D_3		leer	ejecutar					
D_4	escribir		escribir		conmutar			

19.3 MATRIZ DE ACCESO

Matriz de Acceso con Derechos de Copiar (se denota con un asterisco)

Una entrada que se ejecuta en el dominio D2 puede copiar la operación leer en cualquier entrada del archivo F2

dominio \ objeto	F ₁	F ₂	F ₃
D ₁	ejecutar		escribir*
D ₂	ejecutar	leer*	ejecutar
D ₃	ejecutar		

dominio \ objeto	F ₁	F ₂	F ₃
D ₁	ejecutar		escribir*
D ₂	ejecutar	leer*	ejecutar
D ₃	ejecutar	leer	

19.3 MATRIZ DE ACCESO

• Este esquema tiene variantes:

- Se copia $acceso(i,j)$ a $acceso(k,j)$, se elimina $acceso(i,j)$ -> transferencia
- La copia es limitada:
 - R^* : $acceso(i,j)$ a $acceso(k,j)$
 - En el dominio Dk no se puede copiar pues es R y no R*

19.3 MATRIZ DE ACCESO

Matriz de Acceso con Derechos de Propiedad

Si $acceso(i,j)$ incluye el derecho dueño, un proceso que está en D1 puede añadir y quitar derechos en cualquier entrada de la columna j

dominio \ objeto	F ₁	F ₂	F ₃
D ₁	dueño ejecutar		escribir
D ₂		leer* dueño	leer* dueño escribir*
D ₃	ejecutar		

dominio \ objeto	F ₁	F ₂	F ₃
D ₁	dueño ejecutar		
D ₂		dueño leer* escribir*	leer* dueño escribir*
D ₃		escribir	escribir

19.3 MATRIZ DE ACCESO

Uso de la Matriz de Acceso

- Si un proceso en el dominio D_i trata de hacer la operación "op" sobre objeto O_j , entonces "op" debe estar en la matriz de acceso.
- Se puede expandir a protección dinámica.
 - Operaciones para agregar y eliminar derechos de acceso.
 - Derechos de acceso especiales:
 - Propietarios (owner) de O_j
 - Operaciones de copia de O_j a O_k
 - Control: D_i puede modificar los derechos de acceso de D_j
 - Transferencia: cambiar de dominio D_i a D_j

19.3 MATRIZ DE ACCESO

Uso de la Matriz de Acceso (cont.)

- El diseño de la matriz de acceso separa los mecanismos de las políticas.
 - Mecanismo
 - El sistema operativo provee Matriz de acceso + reglas
 - Asegura que la matriz sólo será manipulada por agentes autorizados y que las reglas serán hechas cumplir de manera estricta.
 - Política
 - El usuario /Administrador dicta las políticas
 - Quien puede acceder qué objeto y de qué manera.

19.4 IMPLEMENT. DE MATRIZ DE ACCESO

- 1 Tabla Global
- 2 Lista de Acceso para Objetos
- 3 Lista de capacidades para Dominios
- 4 Mecanismo de Cerradura y Llave
- 5 Comparación

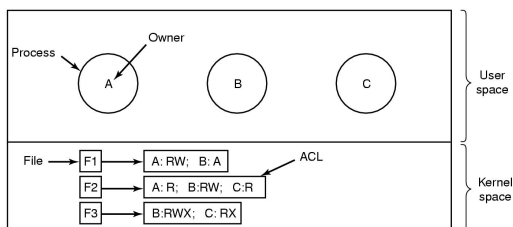
19.4.1 Tabla Global

- Es la implementación más sencilla de la Matriz de Acceso.
- Consiste de un conjunto de tripletas:
<dominio, objeto, conjunto de derechos>
- Es una Matriz Dispersa, puede ocupar mucho espacio, por lo tanto no puede mantenerse por completo en memoria principal.
- No permite manejar grupos con características comunes

19.4.2 Lista de Acceso para Objetos

- Cada columna de la M. de A. puede ser implementada como una Lista de Acceso, asociando cada columna con un Objeto
- Consiste de un conjunto de pares ordenados:
<dominio, conjunto de derechos>
- Se pueden definir conjuntos por omisión (default) de derechos de acceso, para aumentar la eficiencia

19.4.2 Lista de Acceso para Objetos



Uso de listas de control de acceso para administrar el acceso a archivos

19.4.2 Lista de Acceso para Objetos

File	Access control list
Password	tana, sysadm: RW
Pigeon_data	bill, pigfan: RW; tana, pigfan: RW; ...

Dos listas de control de acceso

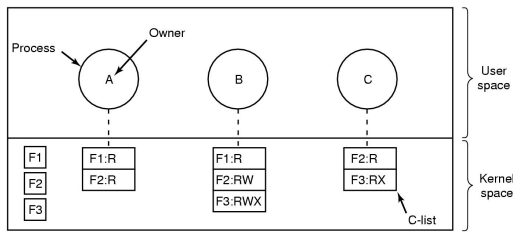
19.4.3 Lista de Capacidades para Dominios

- Cada fila de la M. de A. puede ser implementada como una Lista de Capacidades, asociando cada fila con un Dominio
- Consiste de un conjunto de pares ordenados:
<objeto, conjunto de derechos>
- Para ejecutar la operación M sobre el objeto Q, el proceso ejecuta M, especificando la capacidad (trasponiendo el puntero a Q) como parámetro

19.4.3 Lista de Capacidades para Dominios

- Las capacidades suelen distinguirse de otros datos en una de dos formas:
 - Cada objeto tiene una etiqueta para denotar su tipo
 - El espacio de direcciones asociado a un programa se puede dividir en:
 - Una parte accesible para el programa (datos e instrucciones)
 - Otra parte contiene la lista de capacidades, sólo accesible por el S.O.

19.4.3 Lista de Capacidades para Dominios



Cuando se usan capacidades, cada proceso tiene una lista de capacidades

19.4.4 Mecanismo de Cerradura/Llave

- Es un híbrido entre Lista de Acceso y Lista de Capacidades.
- Cada Objeto tiene un patrón único (cerradura) y cada Dominio tiene una lista de patrones (llaves).
- Un proceso sólo puede acceder a un objeto si dentro de la lista del dominio en el cual se ejecuta está la llave apropiada.

19.4.5 Comparación

- Lista de Acceso
 - Orientada a necesidades del usuario
 - Difícil determinar derechos para un dominio
 - Overhead producido en búsqueda por objeto
- Lista de Capacidades
 - Más orientada al proceso
 - Revocación de capacidades ineficiente
- Mecanismo Cerradura/Llave
 - Mecanismo eficaz y flexible
- Mayoría de los Sistemas usa una combinación de Lista de Acceso y Lista de Capacidades

19.5 REVOCACIÓN DERECHOS DE ACCESO

- En protección dinámica, a veces se podría requerir revocar derechos de accesos a objetos. La revocación sería:
 - Inmediata o diferida
 - Selectiva o general (usuarios)
 - Parcial o total (derechos)
 - Temporal o permanente

19.5 REVOCACIÓN DERECHOS DE ACCESO

- Lista de accesos: eliminar derechos de acceso de la lista de accesos
 - simple
 - inmediata
- Lista de capacidades: Se necesita un esquema para ubicar una capacidad en el sistema antes que pueda ser revocada
 - Readquisición
 - Punteros hacia atrás
 - Indirección (capacidad->tabla global->objeto)
 - Claves

19.6 SISTEMAS BASADOS EN CAPACIDAD

- Hydra
 - Conjunto fijo de derechos de acceso conocidos para el Sistema e interpretables por él.
 - La interpretación de los derechos definidos por el usuario es realizada solamente por el programa de usuario; el sistema provee protección de acceso para el uso de estos derechos.

19.6 SISTEMAS BASADOS EN CAPACIDAD

- El sistema de CAP de Cambridge
 - Capacidad de datos : provee estándares para leer, escribir y ejecutar segmentos individuales de almacenamiento asociados con el objeto
 - Capacidad de Software: la interpretación es dejada al subsistema, a través de procedimientos protegidos.

19.7 PROTECCIÓN BASADA EN LENGUAJE

- La especificación de la protección en un lenguaje de programación permite la descripción de alto nivel de las políticas para la asignación y uso de los recursos.
- La implementación a través de lenguajes puede proveer software para el cumplimiento de la protección cuando el chequeo automático apoyado por hardware no esté disponible.

19.7 PROTECCIÓN BASADA EN LENGUAJE

- La interpretación de especificaciones de protección para generar llamadas en cualquier sistema de protección es provista por el hardware y el sistema operativo.

19.7 PROTECCIÓN BASADA EN LENGUAJE

- Las ventajas del núcleo sobre un compilador (programa)
 - Seguridad
 - Flexibilidad
 - Eficiencia
- La incorporación de conceptos de protección en lenguajes de programación, como herramienta práctica para el diseño de sistemas , está en una etapa muy preliminar
