Sistemas Operativos

Protección

Armando Arce, Escuela de Computación, arce@itcr.ac.cr

Tecnológico de Costa Rica

Protección

Los procesos en un sistema operativo deben protegerse de las actividades realizadas por otros procesos.

- Para proporcionar dicha protección, se puede utilizar diversos mecanismos para garantizar que sólo los procesos que hayan obtenido la adecuada autorización del sistema operativo puedan operar sobre los archivos, los segmentos de memoria, sobre la CPU y sobre otros recursos del sistemas.

Protección

Una razón para proporcionar protección es impedir una violación maliciosa e intencionada de una restricción de acceso por parte de un usuario.

- Una razón más importante es garantizar que cada componente del programa activo en un sistema utilice los recursos del sistema sólo en ciertas formas que sean coherentes con las políticas establecidas.
- El papel de la protección es proporcionar un mecanismo para la imposición de las políticas que gobiernen el uso de recursos.

Principios de protección

El principio de **mínimo privilegio** dicta que a los programas, a los usuarios, e incluso a los sistemas se les concedan únicamente los suficientes privilegios para llevar a cabo sus tareas.

- Un sistema operativo que se ajuste al principio del mínimo privilegio implementará sus características, programas, llamadas al sistema y estructuras de datos de modo que el fallo o el compromiso de un componente no permitan realizar más que un daño mínimo.
- La gestión de los usuarios con el principio del mínimo privilegio implica crear una cuenta separada para cada usuario, con sólo los privilegios que ese usuario necesite.

Dominio de protección

Para poder analizar distintos mecanismos de protección, es conveniente introducir el concepto de dominio.

- Un proceso opera dentro de un *dominio de protección*, que especifica los recursos a los que el proceso puede acceder.
- Cada domino define un conjunto de objetos y los tipos de operaciones que pueden invocarse sobre cada objeto.
- La capacidad de ejecutar una operación sobre un objeto es un *derecho de acceso*.
- Un dominio es una colección de derechos de acceso, cada uno de los cuales es una pareja ordenada (objeto, permisos).
- Cada par especifica un objeto y cierto subconjunto de las operaciones que se pueden realizar en él.

Dominio de protección

Los dominios no tienen por qué ser disjuntos sino que pueden compartir derechos de acceso.

- Por ejemplo, dos dominios pueden tener el derecho de imprimir a una impresora determinada, de esta forma un proceso que se ejecute en cualquiera de estos dos dominios podrá realizar la impresión en dicha impresora.

Dominio de protección

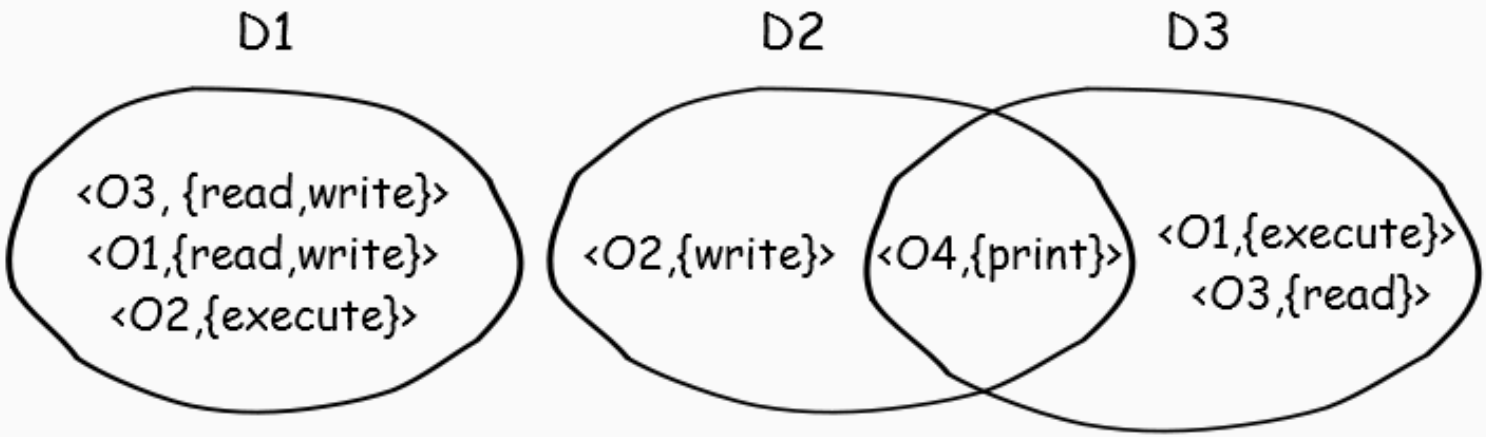

Figure 1:

Dominio de protección

La asociación entre un proceso y un dominio puede ser estática, si el conjunto de recursos disponibles para el proceso está fijo durante la vida del proceso, o dinámica.

- Establecer dominios dinámicos es más complicado que establecer dominios estáticos.

Dominio de protección

Si la asociación entre procesos y dominios es fija debe haber un mecanismo para cambiar el contenido de un dominio.

- La razón deriva del hecho que un proceso puede ejecutares en dos fases distintas y puede necesitar, por ejemplo, acceso de lectura en una fase y acceso de escritura en la otra.
- Por tanto, se debe permitir que se modifique el contenido de un dominio para que siempre refleje el mínimo necesario de derechos de acceso.

Dominio de protección

Si la asociación es dinámica, habrá disponible un mecanismo para permitir la **conmutación de dominio**, permitiendo al proceso conmutar de un dominio a otro.

- Si no se puede cambiar el contenido de un dominio, se puede proporcionar el mismo efecto creando un nuevo dominio con el contenido modificado y conmutando a él cuando se requieran nuevos derechos.

Dominio de protección

Un dominio puede llevarse a la práctica de diversas formas:

- Cada usuario puede ser un dominio: la conmutación de dominio tiene lugar cuando se cambia de usuario.
- Cada proceso puede ser un dominio: la conmutación de dominio se da cuando un proceso envíe un mensaje a otro proceso y espere una respuesta.
- Cada procedimiento puede ser un dominio: la conmutación de dominio sucede cuando se lleva a cabo una llamada a un procedimiento.

Matriz de acceso

El modelo de protección puede verse de forma abstracta como una matriz, denominada matriz de acceso.

- Las filas de la matriz de acceso representan dominios y las columnas representan objetos.
- Cada entrada de la matriz está compuesta de un conjunto de derechos de acceso.
- Puesto que la columna define los objetos explícitamente, se puede omitir el nombre del objeto del derecho de acceso.
- La entrada **access(i,j)** define el conjunto de operaciones que un proceso que se ejecute en el dominio Di puede invocar sobre el objeto Oj.

Matriz de acceso

La matriz de acceso puede implementar decisiones de política relativas a la protección.

- Las decisiones de política implican que derechos deben incluirse en la entrada (i,j).
- También se debe definir el dominio en el que cada proceso se habrá de ejecutar.
- Esta última política es usualmente definida por el sistema operativo.

Matriz de acceso

Los usuarios normalmente deciden el contenido de las entradas de la matriz de acceso.

- Cuando un usuario crea un nuevo objeto Oj, se añade la columna Oj a la matriz de acceso, con las apropiadas entradas de inicialización, según determine el creador.
- El usuario puede decidir introducir algunos derechos en determinadas entradas de la columna j y otros derechos en otras entradas, según sea necesario.

Matriz de acceso

Dominio	F1	F2	F3	impresora
D1	read		read	
D2				print
D3		read	execute	
D4	read/write		read/write	

Figure 2:

Matriz de acceso

La matriz de acceso proporciona el mecanismo apropiado para definir e implementar un control estricto de la asociación tanto estática como dinámica entre procesos y dominios.

- Cuando se conmuta un proceso de un dominio a otro, se ejecuta una operación (switch) sobre un objeto (el dominio).
- Se pueden controlar la conmutación de dominios incluyendo los dominios entre los objetos de la matriz de acceso.
- De forma similar, cuando se cambia el contenido de la matriz de acceso, se realiza una operación sobre un objeto: la propia matriz de acceso.
- De nuevo, se pueden controlar estos cambios incluyendo la propia matriz de acceso como un objeto.

Matriz de acceso

Dominio	F1	F2	F3	impr	D1	D2	D3	D4
D1	read		read			switch		
D2				print			switch	switch
D3		read	execute					
D4	read/write		read/write		switch			

Figure 3:

Matriz de acceso

Permitir una modificación controlada del contenido de las entradas de la matriz de acceso; requiere tres operaciones adicionales: **copy**, **owner** y **control**.

- La capacidad de copiar un derecho de acceso de un dominio (o fila) de la matriz de acceso a otro se denota mediante un asterisco (*) añadido al derecho de acceso.
- El derecho de copia permite copiar el derecho de acceso sólo dentro de la columna (es decir, para el objeto) en la que esté definido el derecho.
- Este esquema tiene dos variantes: transferencia del derecho y propagación del derecho (copia limitada).
- En el primer caso al copiar el derecho éste se pierde, en el segundo caso la copia no provoca la pérdida del derecho.

Matriz de acceso

Dominio	F1	F2	F3
D1	execute		write*
D2	execute	read*	execute
D3	execute		

Dominio	F1	F2	F3
D1	execute		write*
D2	execute	read*	execute
D3	execute	read	

Figure 4:

Matriz de acceso

También se necesita un mecanismo para permitir la adición de nuevos derechos y la eliminación de otros.

- El derecho **owner** controla estas operaciones.
- Si una columna incluye el derecho de **owner**, entonces un proceso que se ejecute en ese dominio podrá añadir y eliminar cualquier derecho en cualquier entrada de dicha columna.

Matriz de acceso

Dominio	F1	F2	F3
D1	owner execute		write
D2		read* / owner	read* / owner write
D3	execute		

Dominio	F1	F2	F3
D1	owner execute		write
D2		read* / owner write*	read* / owner write
D3	execute	write	write

Figure 5:

Matriz de acceso

Los derechos de copia y propietario permiten a un proceso modificar las entradas de una columna.

- También hace falta un mecanismo para modificar las entradas de un fila.
- El derecho **control** sólo es aplicable a los objetos dominio.
- Si una entrada de la tabla incluye el derecho control entonces un proceso que se ejecute en ese dominio puede eliminar cualquier derecho de acceso en la respectiva fila.

Matriz de acceso

Dominio	F1	F2	F3	impr	D1	D2	D3	D4
D1	read		read			switch		
D2				print			switch	switch control
D3		read	execute					
D4	write		write		switch			

Figure 6:

Implementación de matriz de acceso

Existen diversas formas de implementar la matriz de acceso.

- En general, esta matriz será del tipo “matriz dispersa”.
- Aunque existen técnicas de representar estructuras de datos para matrices dispersas, no resultan particularmente útiles en este caso.

Tabla global

La implementación más simple de la matriz de acceso es una tabla global compuesta de un conjunto de tripletas ordenadas .

- Cada vez que se ejecuta una operación sobre el objeto dentro del dominio particular, se analiza la tabla global en busca de una tripleta .
- Si se encuentra esta tripleta, se permite que la operación continúe; en caso contrario; se genera una condición de excepción (o error).

Tabla global

Dominio	Objeto	Derechos
D1	F1	read
D1	F3	read
D2	F1	read
D2	Impr	print
D3	F2	read
D3	F3	execute
D4	F1	read/write
D4	F3	read/write

Figure 7:

Tabla global

Esta implementación tiene varias desventajas.

- La tabla es usualmente muy grande y no puede, por tanto, ser conservada en memoria principal, por lo que hacen falta operaciones adicionales de E/S.
- A menudo se utilizan técnicas de memoria vitral para gestionar esta tabla.
- Además, resulta difícil aprovechar las agrupaciones especiales de objetos y dominios.
- Por ejemplo, cuando todo el mundo puede leer un objeto concreto, es necesario definir una entrada separada en cada uno de los dominios.

Listas de acceso para los objetos (ACL)

Cada columna de la matriz de acceso puede implementarse como una lista de acceso de un objeto.

- Obviamente, las entradas vacías pueden descartarse.
- La lista resultante para cada objeto está compuesta por una serie de parejas ordenadas , que definen todos los dominios que tengan un conjunto de derechos de acceso no vacío para dicho objeto.

Listas de acceso para los objetos (ACL)

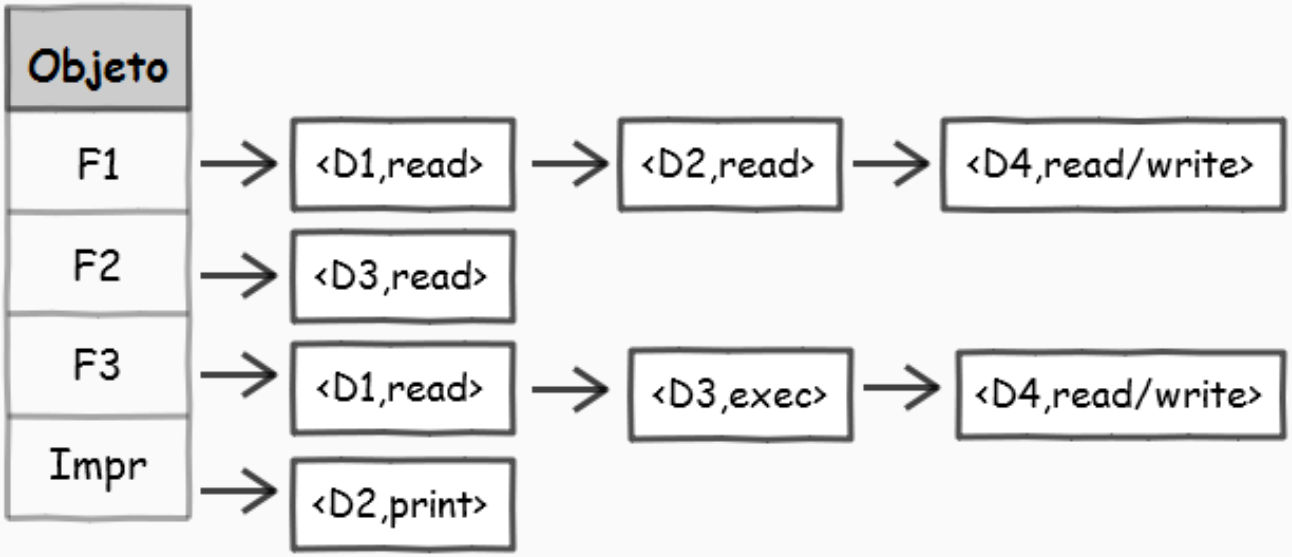

Figure 8:

Listas de acceso para los objetos (ACL)

Esta técnica puede extenderse fácilmente para definir una lista más un conjunto predeterminado de derechos de acceso.

- Cuando se intenta realiza una operación sobre un objeto en un dominio particular, se busca en a lista de acceso del objeto el dominio especificado.
- Si se encuentra la entrada, se permite la operación, en caso contrario, se comprueba el conjunto predeterminado.
- Si la operación está en el conjunto predeterminado, se permite el acceso, en caso contrario se produce una condición de excepción.
- Para aumentar la eficiencia, se puede comprobar primero el conjunto predeterminado y luego buscar en la lista de acceso.

Listas de acceso para los objetos (ACL)

Las entradas en la ACL pueden ser para dominios que representen usuarios individuales o grupos de usuarios.

- Los grupos tienen nombres y se pueden incluir en las ACLs. Hay dos variaciones posibles en la semántica de grupos. En ciertos sistemas, una entrada en la ACL contiene entradas de la forma:

  `UID1,GID1: permisos1; UID2, GID2: permisos2;`

Listas de acceso para los objetos (ACL)

Bajo estas condiciones, cuando se hace una petición para acceder a un objeto, se realiza un comprobación mediante el uso del UID y el GID del proceso que hizo la llamada.

- Si están presentes en la ACL, están disponibles los permisos listados. Si la combinación (UID,GID) no está en la lista, no se permite el acceso.

Listas de acceso para los objetos (ACL)

Ya que un usuario puede pertenecer a varios grupos, en algunos casos, un usuario puede tener acceso a ciertos archivos sin importar qué grupo esté utilizando en ese momento.

Para manejar este caso se introduce el concepto de *comodín* que representa a todos los usuarios, por ejemplo:

```
UID1,*: RW
```

para un archivo dado proporciona acceso a UID1 sin importar en qué grupo se encuentre.

Listas de acceso para los objetos (ACL)

Otra posibilidad es que, si un usuario pertenece a cualquiera de los grupos que tengan ciertos permisos de acceso, se permite el acceso.

- La ventaja aquí es que un usuario que pertenezca a varios grupos no tiene que especificar el grupo que va a usar al momento de iniciar sesión.
- Todos los grupos son válidos en todo momento. Una desventaja de este método es que no hay tanto encapsulamiento.

Listas de acceso para los objetos (ACL)

Al utilizar grupos y comodines se introduce la posibilidad de bloquear de manera selectiva a un usuario específico, para evitar que acceda a un archivo. Por ejemplo, la entrada:

```
UID1, *:(none); *,*": RW
```

proporciona a todos, excepto a UID1, el acceso de lectura y escritura al archivo.

- Esto funciona debido a que las entradas se exploran en orden, y se toma la primera que aplique; las entradas subsiguientes ni siquiera se examinan.

Listas de acceso para los objetos (ACL)

La otra manera de lidiar con los grupos es no tener entradas en la ACL que consistan en pares (UID,GID), sino que cada entrada sea un UID o un GID. Por ejemplo, una entrada para un archivo podría ser:

```
UID1: RW; UID2: RW; GID1: RW
```

Listas de capacidades para los dominios

En lugar de asociar las columnas de la matriz de acceso con los objetos en forma de listas de acceso, se puede asociar cada fila con su dominio.

- Una lista de capacidades (lista-C) para un dominio es una lista de objetos junto con las operaciones permitidas sobre esos objetos.
- Cada objeto se suele representar mediante su dirección o nombre físico, denominada capacidad.
- Para ejecutar la operación M sobre el objeto Oj, el proceso ejecuta la operación M especificando la capacidad (puntero) para el objeto Oj como parámetro.
- La simple posesión de la capacidad significa que el acceso está permitido.

Listas de capacidades para los dominios

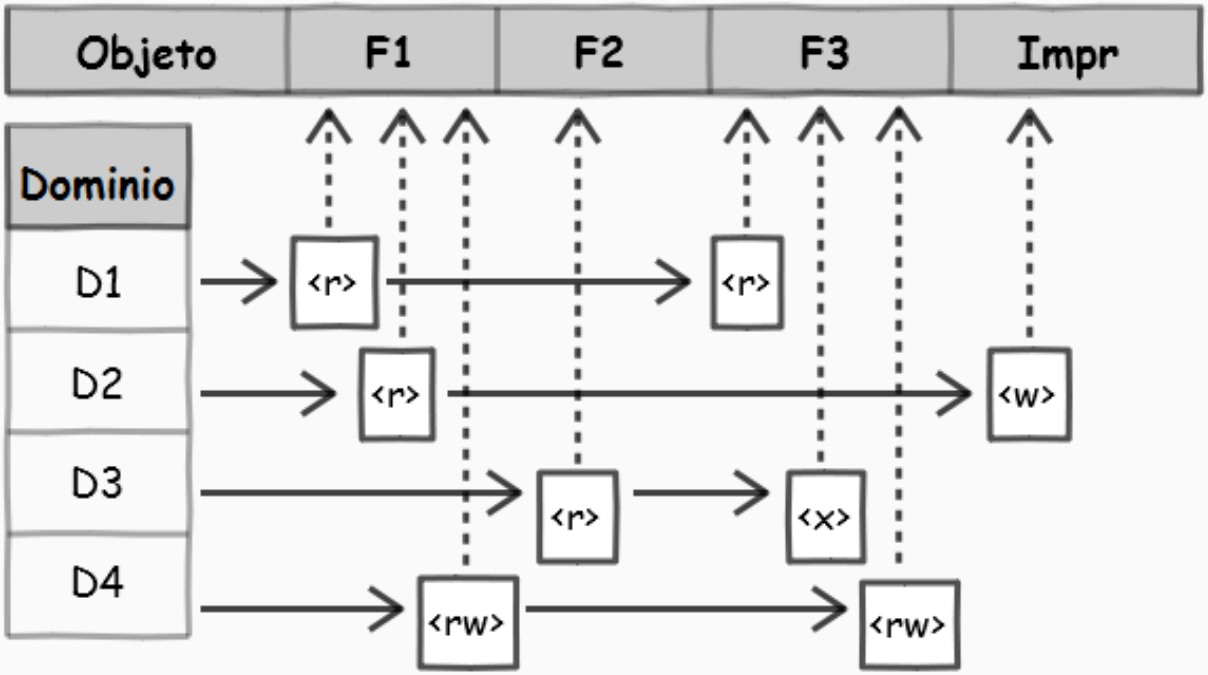

Figure 9:

Listas de capacidades para los dominios

La lista de capacidades está asociada con un dominio, pero un proceso que se ejecute en ese dominio no puede nunca acceder directamente a ella.

- Por el contrario, la lista de capacidades es en sí misma un objeto protegido, mantenido por el sistema operativo y al que el usuario sólo puede acceder indirectamente.
- El mecanismo de protección basado en capacidades descansa en el hecho de que nunca se permite a las capacidades migrar a algún espacio de direcciones que sea directamente accesible por parte de un proceso de usuario (donde podrían ser modificadas).
- Si todas las capacidades son seguras, el objeto al que protegen también estará defendido frente a accesos no autorizados.

Listas de capacidades para los dominios

Existen tres métodos para proteger las listas de capacidades:

- Arquitectura etiquetada
- Lista-C en el espacio del núcleo
- Lista-C en el espacio del usuario

Arquitectura etiquetada

En este método se requiere de un diseño de hardware en el que cada palabra de memoria tiene un bit adicional (o) etiqueta que indica si la palabra contiene o no una capacidad.

- Solo los programas que se ejecutan en modo kernel pueden modificar las palabras que tengan la etiqueta activada.

Lista-C en el espacio del núcleo

En este otro método (descrito anteriormente) se hace referencia a las capacidades con base en su posición en la lista.

- Esta forma de direccionamiento es similar al uso de los descriptores de archivos en UNIX.

Lista-C en el espacio del usuario

En este caso se deben administrar las capacidades de manera criptográfica, de tal forma que los usuarios no puedan alterarlas.

- Este método se adapta en especial a los sistemas distribuidos y funciona de la siguiente manera.
- Cuando un proceso cliente envía un mensaje a un servidor remoto para un archivo, el servidor crea el archivo y genera un número aleatorio grande que le corresponda (diferente del FID).
- Después, el servidor genera y devuelve una capacidad al usuario que contiene cuatro campos: servidor, FID, permisos y por último una concatenación de los campos anteriores pero encriptados.

Lista-C en el espacio del usuario

Cuando el usuario desea acceder al objeto, envía la capacidad al servidor como parte de la petición.

- Después el servidor extrae los campos de la capacidad, recalcula el campo encriptado y lo compara con el campo que envió el cliente.
- Si el resultado coincide con el campo, se respeta la petición; en caso contrario, se rechaza.

Revocación de derechos de acceso

Es posible que se necesite revocar derechos de acceso a objetos compartidos por diferentes usuarios.

- De ser así se deben considerar algunos aspectos sobre la revocación: será inmediata o diferida, se realiza de forma selectiva o general, es parcial o total, y se aplicará de forma temporal o permanente.
- Con un esquema basado en listas de acceso, la revocación resulta sencilla. Se analiza la lista de acceso en busca de los derechos de acceso que hay que revocar y se borra de la lista. La revocación es inmediata y puede ser general o selectiva, total o parcial y permanente o temporal.

Revocación de derechos de acceso

No obstante, si la ACL se revisa sólo cuando se abre un archivo, es muy probable que el cambio tenga efecto únicamente en las futuras llamadas a *open*.

- Cualquier archivo que ya se encuentre abierto seguirá con los derechos que tenía cuando se abrió, aun si el usuario ya no está autorizado para utilizarlo.

Revocación de derechos de acceso

Las capacidades, sin embargo, presentan un problema mucho más difícil para la revocación.

- Puesto que las capacidades están distribuidas por todo el sistema, se deben encontrar antes de poder revocarlas.
- Entre los esquemas que pueden utilizarse para implementar la revocación en un sistema basado en capacidades se pueden citar las siguientes:

Readquisición

Periódicamente, se borran las capacidades de cada dominio.

- Si un proceso quiere usar una capacidad, puede que se encuentre con que esa capacidad ha sido borrada.
- El proceso puede entonces tratar de volver a adquirir la capacidad.
- Si se ha revocado el acceso, el puntero no será capaz de efectuar esa readquisición.

Retropunteros

Con cada objeto se mantiene una lista de punteros, que hace referencia a todas las capacidades asociadas con ese objeto.

- Cuando se necesita realizar una revocación, podemos seguir esos punteros, cambiando las capacidades según sea necesario. Este esquema fue adoptado en el sistema MULTICS.
- Es un sistema muy general, aunque su implementación es costosa.

Retropunteros

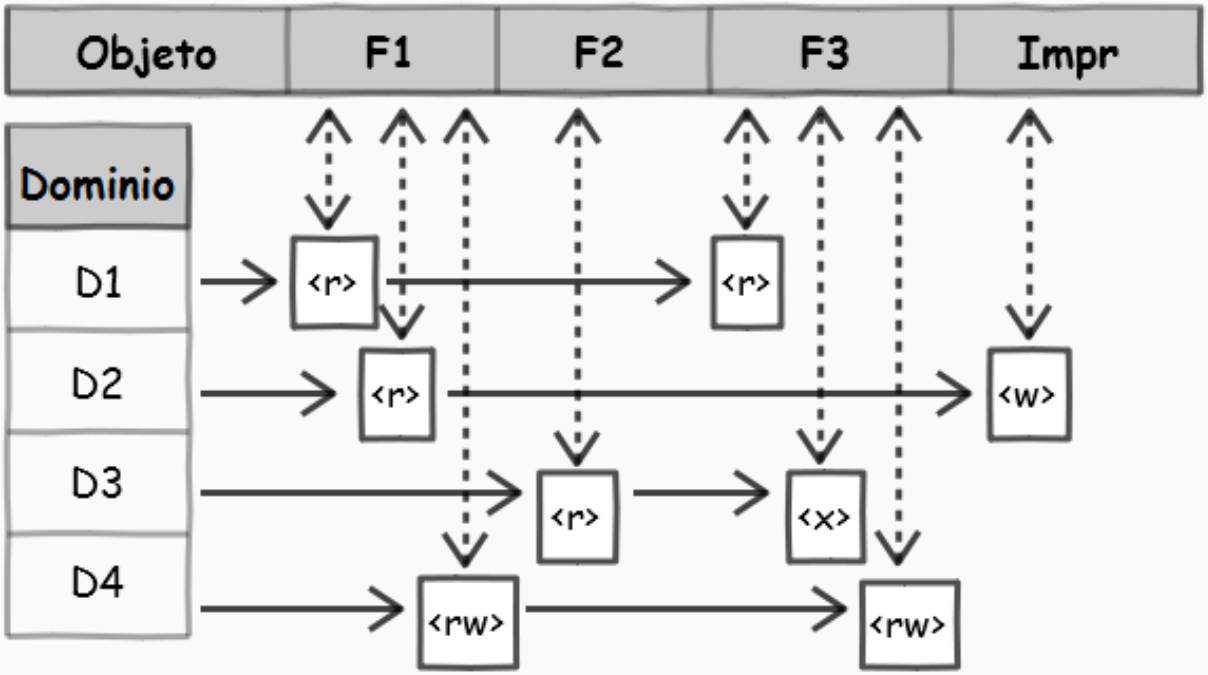

Figure 10:

Indirección

Las capacidades apuntan indirectamente, en lugar de directamente, a los objetos.

- Cada capacidad apunta a una entrada unívoca dentro de una tabla global, que a su vez apunta al objeto.
- La revocación se implementa analizando la tabla global en busca de la entrada deseada y borrándola.
- Entonces, cuando se intenta realizar un acceso, se verá que la capacidad está apuntando a una entrada ilegal de la tabla.
- Las entradas de la tabla pueden reutilizarse para otras capacidades sin ninguna dificultad, ya que tanto la capacidad como la entrada de la tabla contienen el nombre distintivo del objeto.
- Este esquema fue adoptado en el sistema CAL y no permite la revocación selectiva.

Indirección

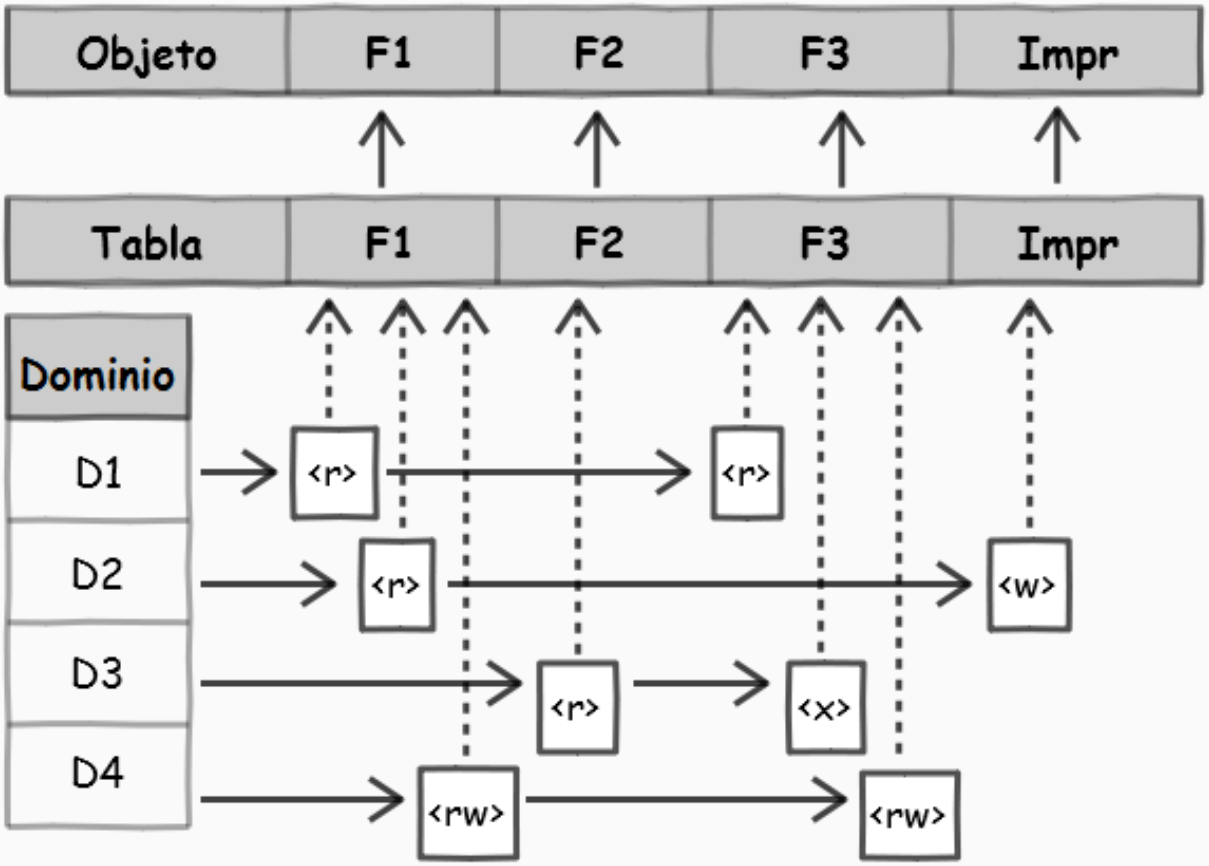

Figure 11:

Claves

Una clave es un patrón distintivo de bits que puede asociarse con una capacidad.

- Esta clave se define en el momento de crear la capacidad y no puede ser nunca modificada ni inspeccionada por el proceso que posee la capacidad.
- Con cada objeto hay asociada una clave maestra. Cuando se crea una capacidad, se asocia con esa capacidad el valor actual de la clave maestra.
- Cuando se ejerce esa capacidad, su clave se compara con la clave maestra. Si las dos claves coinciden, se permite que la operación continúe; en caso contrario, se genera una condición de excepción.

Claves

- El proceso de revocación sustituye la clave maestra con un nuevo valor, invalidando todas las capacidades anteriores para este objeto.
- Este esquema no permite la revocación selectiva, ya que con cada objeto solo hay asociada una clave maestra.
- Si asociamos una lista de claves con cada objeto, entonces podrá implementarse la revocación selectiva.

Claves

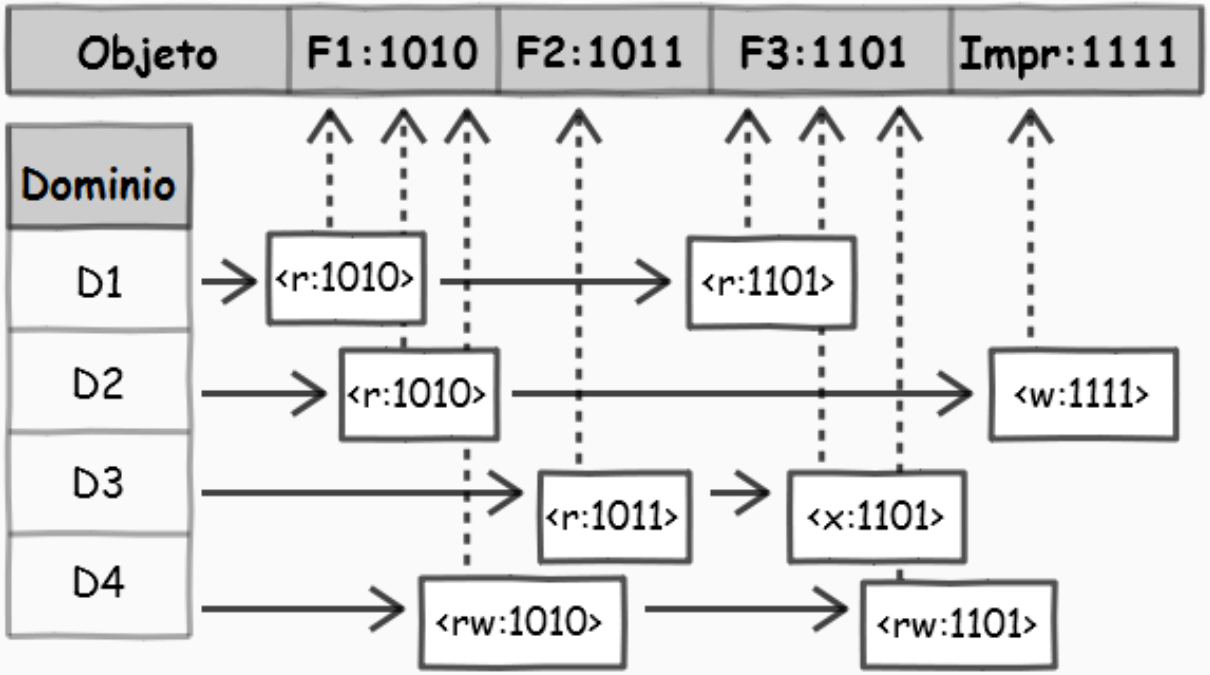

Figure 12:

Referencia bibliográfica

A.SILBERSCHATZ, P. GALVIN, y G. GAGNE, Operating Systems Concepts, Cap.15, 9a Edición, John Wiley, 2013.