



Sistemas Operativos

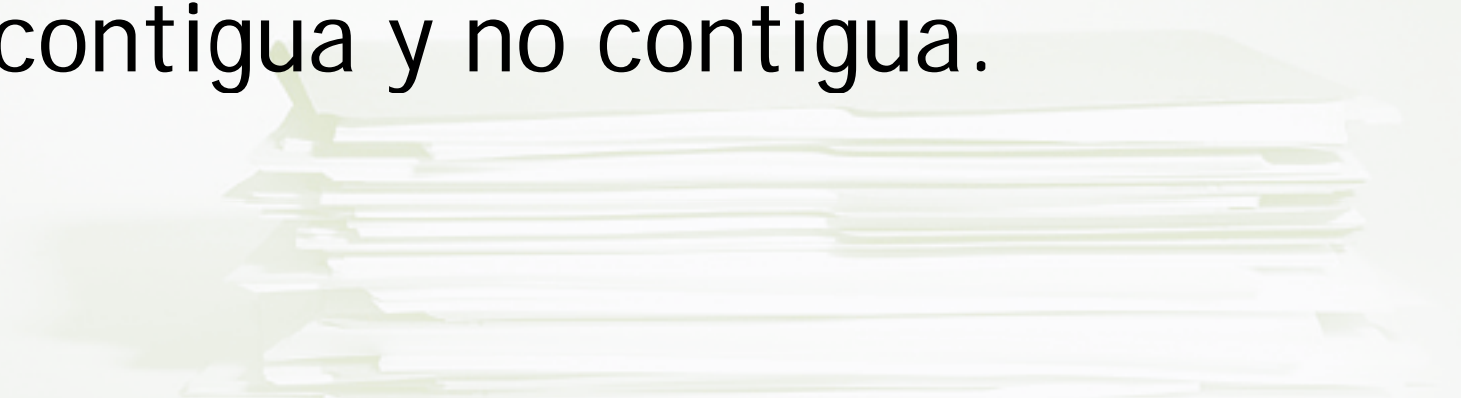


Tema 03
Administración de memoria

Administración de Memoria

Objetivo

El alumno explicará las diferentes técnicas de asignación de memoria contigua y no contigua.



Funciones y operaciones del administrador de memoria

La memoria es uno de los recursos más importantes de la computadora y, en consecuencia la parte responsable del sistema operativo de tratar con este recurso, el gestor de memoria, es un componente básico del mismo.

Funciones y operaciones del administrador de memoria

En un sistema con multiprogramación, el sistema operativo debe encargarse de realizar un reparto transparente, eficiente y seguro de los distintos recursos de la máquina entre los diversos procesos, de forma que cada uno de ellos crea que tiene una máquina para él solo.

Funciones y operaciones del administrador de memoria

El gestor de memoria del sistema operativo debe hacer de puente entre los requisitos de las aplicaciones y los mecanismos que proporcionan el hardware de gestión de memoria.

Funciones y operaciones del administrador de memoria

En el caso de la memoria el sistema operativo con el apoyo del hardware de gestión de memoria del procesador, debe repartir el almacenamiento existente proporcionando un espacio de memoria independiente para cada proceso y evitando la posible interferencia voluntario o involuntaria de cualquier otro proceso.

Funciones y operaciones del administrador de memoria

Se podría considerar que, en el caso del procesador se realiza un reparto en el tiempo, mientras que en la memoria se trata de un reparto en el espacio.

Funciones y operaciones del administrador de memoria

Se pueden destacar las siguientes características como objetivos deseables del sistema de gestión de memoria:



Funciones y operaciones del administrador de memoria

- Ofrecer a cada proceso un espacio lógico propio.
- Proporcionar protección entre procesos.
- Permitir que los procesos compartan memoria.
- Dar soporte a las distintas regiones del proceso.
- Maximizar el rendimiento del sistema.
- Proporcionar a los procesos mapas de memoria muy grandes.

Espacios lógicos independientes

En un sistema multiprogramado de propósito general no se puede conocer a priori la posición de memoria que ocupará un programa cuando se cargue en memoria para proceder a su ejecución puesto que dependerá del estado de ocupación de la memoria.

Espacios lógicos independientes

El código máquina de un programa contenido en un archivo ejecutable incluirá referencias a memoria, tanto para acceder a sus operandos como para realizar bifurcaciones en la secuencia de ejecución por lo que será necesario realizar un

Espacios lógicos independientes

Proceso de traducción (reubicación) de las direcciones de memoria a las que hacen referencia las instrucciones de un programa (direcciones lógicas) para que se correspondan a las direcciones de memoria principal asignadas al mismo (direcciones físicas).

Espacios lógicos independientes

Este proceso de traducción crea un espacio lógico o mapa independiente para cada proceso proyectándolo sobre la parte correspondiente de la memoria principal de acuerdo a una función de traducción.

Espacios lógicos independientes

- **Dirección lógica:** direcciones de memoria a las que hacen referencia las instrucciones de un programa. Son generadas por el CPU. Conocida también por dirección virtual.
- **Dirección física:** Designa la posición real de la memoria principal. Corresponde a aquella cargada en memoria.

Espacios lógicos independientes

- **Espacio de direcciones lógicas:** Conjunto de direcciones creadas por un programa.
- **Espacio de direcciones físicas:** Conjunto de posiciones en memoria principal.

Memory Management Unit

MMU (Unidad de Manejo de Memoria)

- MMU es un módulo específico del procesador que traduce las direcciones virtuales a direcciones de memoria principal.
- En el esquema de MMU el valor en el registro de relocalización se le suma a cada dirección generada por un proceso de usuario al momento en que se manda a la memoria.
- El programa de usuario se comunica a través de la dirección lógica; nunca ve la dirección física

Espacios lógicos independientes



Protección

Cada proceso debe protegerse contra interferencias

- El código de un proceso no puede hacer referencias a posiciones de memoria de otros procesos sin su permiso.
- Es imposible comprobar las direcciones absolutas de los programas, puesto que se desconoce la ubicación de un programa en memoria principal.
- Debe comprobarse durante la ejecución todas las referencias a memoria generadas por un proceso, para asegurar que solo hacen referencia al espacio de memoria destinado a dicho proceso.

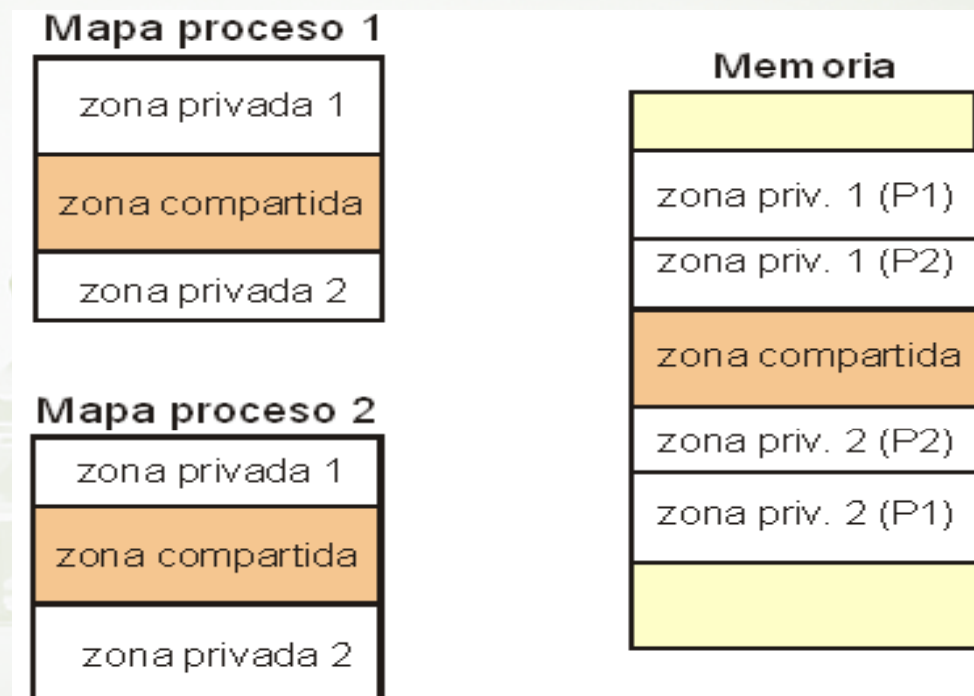
Compartición

Cualquier mecanismo de protección debe permitir el acceso de varios procesos a la misma zona de la memoria principal.

Es mejor permitir a cada proceso que acceda a la misma copia del programa en lugar de tener cada uno su propia copia aparte.

Compartición

Ventajas: Programas que comparten parte de su código, además de acelerar la comunicación entre procesos.



Soporte de las regiones del proceso

El mapa de un proceso no es homogéneo sino que está formado por distintos tipos de regiones con diferentes características y propiedades. Dado que el sistema operativo conoce que regiones incluye el mapa de memoria de cada proceso, el gestor de memoria

Soporte de las regiones del proceso

Con el apoyo del hardware debería dar soporte a las características específicas en cada región.

Mapas de memoria dinámicos:

- Regiones que cambian de tamaño (ejem: pila).
- Se crean y se destruyen regiones.
- Existen zonas sin asignar

Maximizar el rendimiento

El grado de multiprogramación influye directamente en el porcentaje de utilización del procesador y por tanto en el rendimiento del sistema.

Adicionalmente un mayor grado de multiprogramación implica que puede existir un mayor número de usuarios trabajando simultáneamente en el sistema.

Maximizar el rendimiento

El gestor de memoria debe por tanto realizar un reparto de la memoria entre los procesos intentando que quepa el mayor número de ellos en memoria y minimizando el desperdicio inherente del reparto.

Mapas de memoria grandes para los procesos

En los tiempos en que la memoria era cara y en consecuencia los equipos poseían una memoria bastante reducida, se producían habitualmente situaciones en las que las aplicaciones se veían limitadas por el tamaño de la memoria.

Mapas de memoria grandes para los procesos

Para solventar este problema los programadores usaban la técnica de los overlays. Esta técnica consistía en dividir el programa en una serie de fases que se ejecutan sucesivamente pero estando en cada momento residente en memoria una sola fase.

Mapas de memoria grandes para los procesos

Cada fase se programa de manera que después de realizar su labor carga en memoria la siguiente fase y le cede el control.

Mapas de memoria grandes para los procesos

Por tales situaciones se ideó la técnica de **memoria virtual** que permite proporcionar a un proceso de forma transparente un mapa de memoria considerablemente mayor que la memoria física existente en el sistema.

Asignación de memoria contigua

Como hemos visto un sistema multiprogramado la memoria debe ser compartida por varios procesos de cara a obtener una mayor utilización de los recursos de la computadora. Esto provoca que la gestión de la memoria se complique sustancialmente.

Asignación de memoria contigua

En primer lugar, hay que llevar un recuento de las zonas de memoria ocupadas por los procesos, otro problema a resolver viene dado por el hecho de que en el momento de escribir un programa no se sabe en que zona de memoria se ubicará, etc.

Asignación de memoria contigua

En un esquema de asignación de memoria contigua un proceso se ubica en su totalidad en posiciones consecutivas de memoria.

Asignación de memoria contigua

Para sistemas multiprogramados tenemos históricamente se han utilizado dos métodos:

- Particiones estáticas
- Particiones dinámicas

Particiones estáticas

Esta forma de gestión consiste en dividir la memoria en varias zonas, pudiendo ser cada zona de un tamaño diferente.

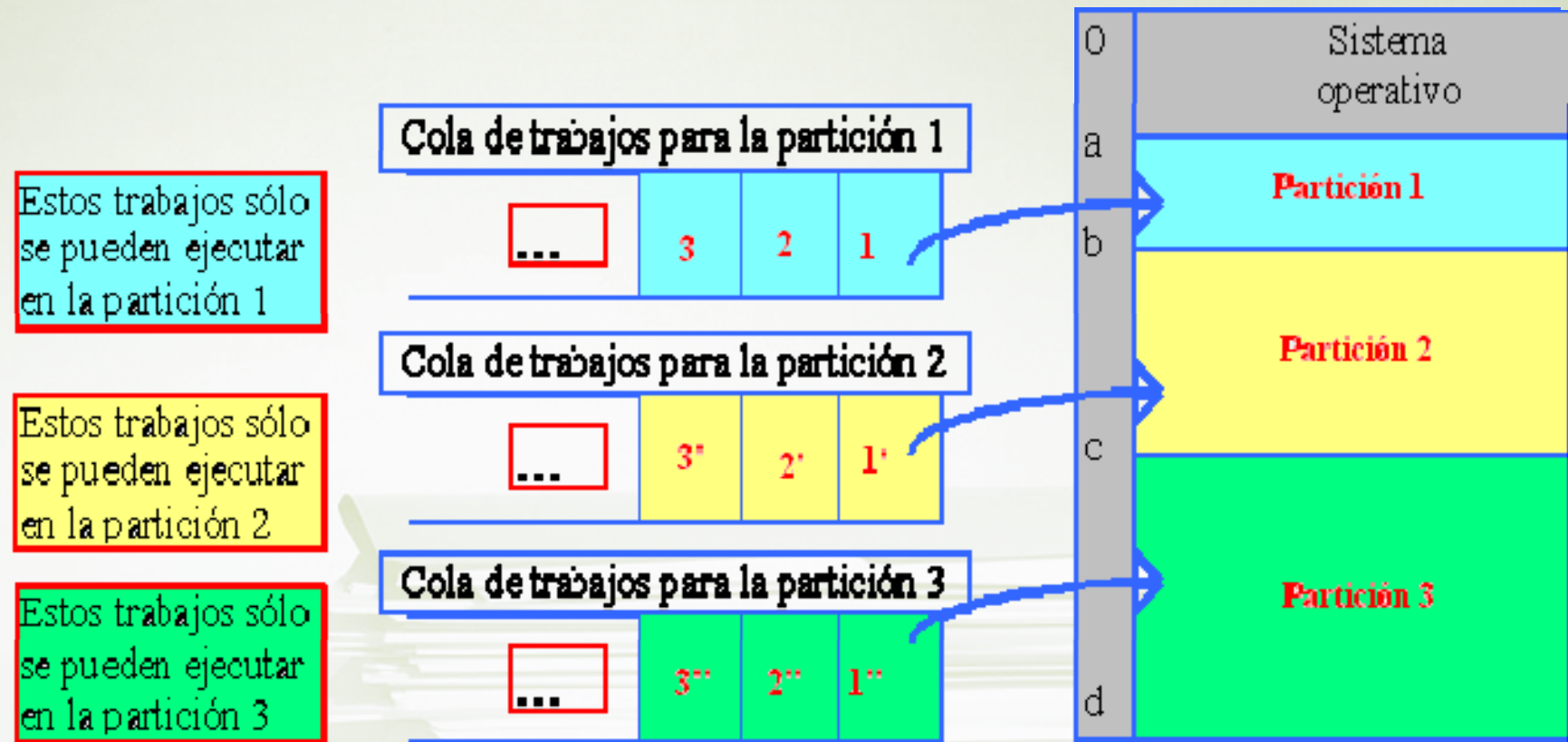
Particiones estáticas

Los trabajos se traducían mediante compiladores y ensambladores para ejecutarse en una partición específica. Una vez introducido un proceso en una partición, permanece en ella hasta su finalización.

Particiones estáticas

Si un trabajo se iniciaba, y la partición para la que estaba compilado estaba ocupada, tenía que esperar aunque estuvieran libres otras particiones. Esto provoca una pérdida de eficiencia.

Particiones estáticas



Multiprogramación con particiones estáticas, con traducción y carga absolutas

Particiones dinámicas

En este método se va asignando la memoria dinámicamente a los procesos, conforme se introducen en la memoria. A cada proceso se le asigna exactamente la memoria que necesita.

Particiones dinámicas

Proceso:

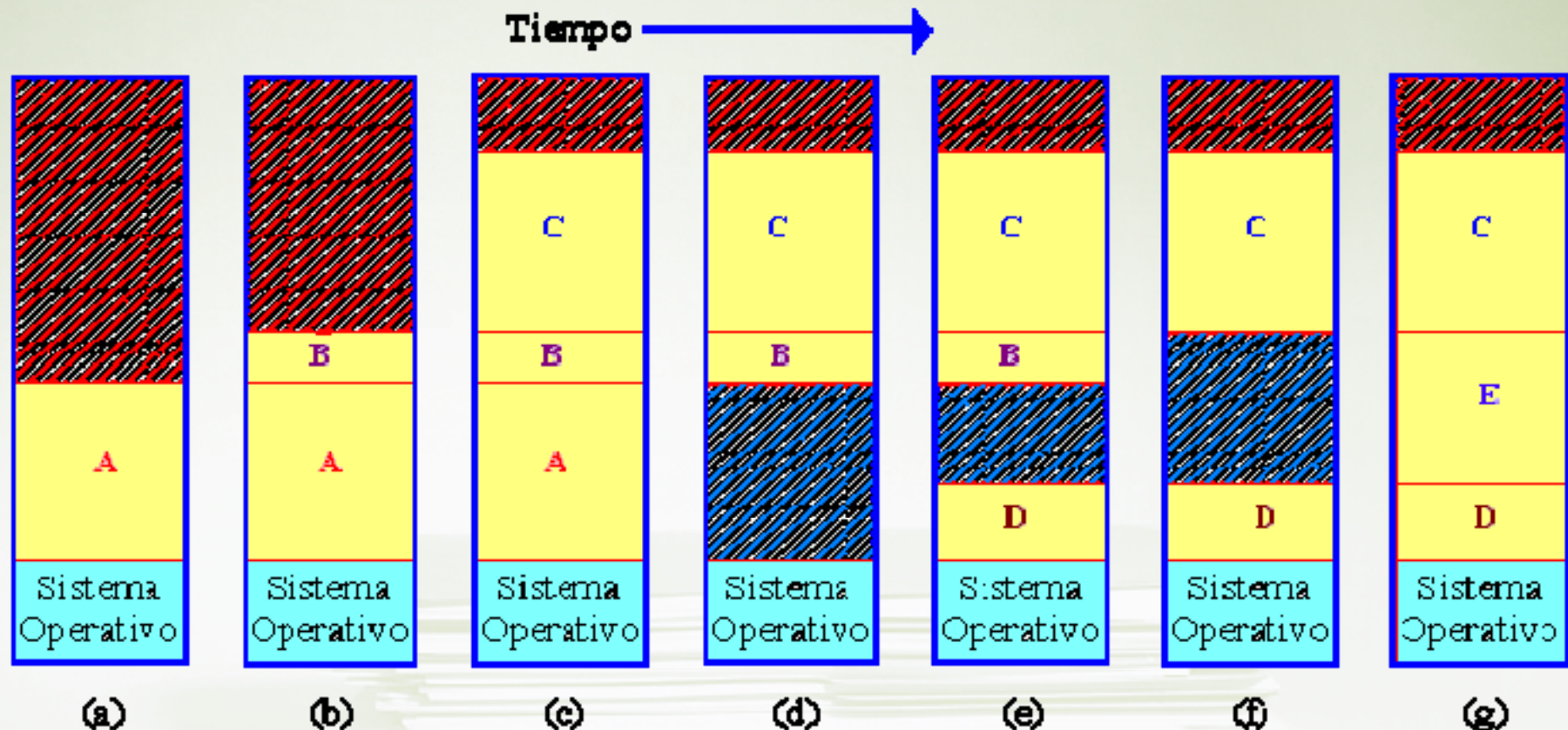
- Busca un hueco de memoria de tamaño suficiente para alojar el mapa de memoria del mismo
- El S.O. reserva la parte del hueco necesaria y crea en ella el mapa inicial del proceso.

Particiones dinámicas

Proceso:

- Se establece la función de traducción, de forma que las direcciones del programa se correspondan con las direcciones existentes en el hueco asignado.

Particiones dinámicas



La asignación de memoria cambia cuando el proceso llega o sale de la memoria.
Las regiones sombreadas son memoria libre.

Particiones dinámicas

Con este método de gestión de memoria se evita el problema de la fragmentación interna. Sin embargo aparece el problema de la fragmentación externa entre particiones.

Particiones dinámicas

El problema consiste en que se crean huecos libres demasiados pequeños como para que quepan procesos, lo que acarrea el desperdicio de la memoria. Una posible solución es la compactación de la memoria que consiste en desplazar todos los procesos

Particiones dinámicas

... hacia la parte inferior de la memoria mientras sea posible. Como la compactación lleva mucho tiempo y además deben detenerse todas las actividades mientras se lleva a cabo, ello puede ocasionar tiempos de respuesta irregulares.

Política de asignación de espacio

El S.O. debe considerar que espacio de los huecos libres se usará intentando encontrar un equilibrio entre buen aprovechamiento de espacio y tiempo de respuesta corto, es decir se aplica un algoritmo de decisión que debe ser eficiente.

Política de asignación de espacio

Este problema es un clásico: ¿Cómo asignar espacio para su aprovechamiento óptimo?

Existen tres enfoques:

Particiones dinámicas

- Mejor ajuste (best-fit): Se elige la zona libre más pequeña donde quepa el proceso.
 - Problema: Conlleva a crear nuevos huecos de tamaño pequeño. Además elegir el hueco más pequeño obliga a mantener ordenados por tamaño los huecos disponibles. No es eficiente.

Particiones dinámicas

- Se busca el hueco más grande intentando evitar la generación de huecos pequeños. Sigue precisando mantener el control de los tamaños.
- El primer ajuste (first-fit): suele ser la mejor política ya que basta con encontrar una zona libre de tamaño suficiente, y ofrece un aprovechamiento aceptable.

Modelo de memoria de un proceso

El mapa inicial de memoria de un proceso esta relacionado con el archivo ejecutable que genera el proceso.

Fases en la generación de un ejecutable

Se realiza en dos fases: la compilación y el enlace.

Compilación:

- Se asignan direcciones a los símbolos definidos.
- Resuelve referencias a los símbolos de cada módulo.
- Genera módulo objeto.

Fases en la generación de un ejecutable

Montaje ó Enlace:

- Resuelve referencias entre módulos objeto.
- Resuelve referencia a símbolos de bibliotecas.
- Genera resultado (archivo ejecutable), que incluye las bibliotecas usadas.

Biblioteca

Colección de módulos objeto relacionados, algunas pueden ser proporcionadas por los usuarios y otras pueden ser dadas por el sistema. Tenemos bibliotecas estáticas y bibliotecas dinámicas.

Biblioteca

Estáticas: El montaje enlaza los módulos objeto del programa y de las bibliotecas dando como resultado un ejecutable auto-contenido.

Desventajas:

- Ejecutables grandes
- Código de función de biblioteca repetido
- Múltiples copias en memoria del código de la función biblioteca
- actualizar la biblioteca implica reenlazar todos los ejecutables.

Biblioteca

Dinámicas: La carga y el montaje se hacen en tiempo de ejecución. El ejecutable contiene únicamente el nombre de la biblioteca y la rutina de carga y montaje en tiempo de ejecución.

Biblioteca

Ventajas:

- Menor tamaño de ejecutables.
- Procesos comparten código de biblioteca.
- Actualización automática de bibliotecas.

Desventajas:

- Mayor tiempo de ejecución debido a carga y montaje.
- Ejecutables no autocontenidos.

Archivos ejecutables

Distintos fabricantes usan diferentes formatos.

En el caso de LINUX tenemos el *Executable and Linkable Format* (ELF). Se estructura con una cabecera y un conjunto de instrucciones:

- Cabecera: Información de control que permite interpretar el contenido del ejecutable y suele incluir:

Archivos ejecutables

-Número mágico: Que identifica a ejecutable. Por ejemplo en el formato ELF el primer byte debe contener el valor hexadecimal 7f, los tres caracteres ELF.

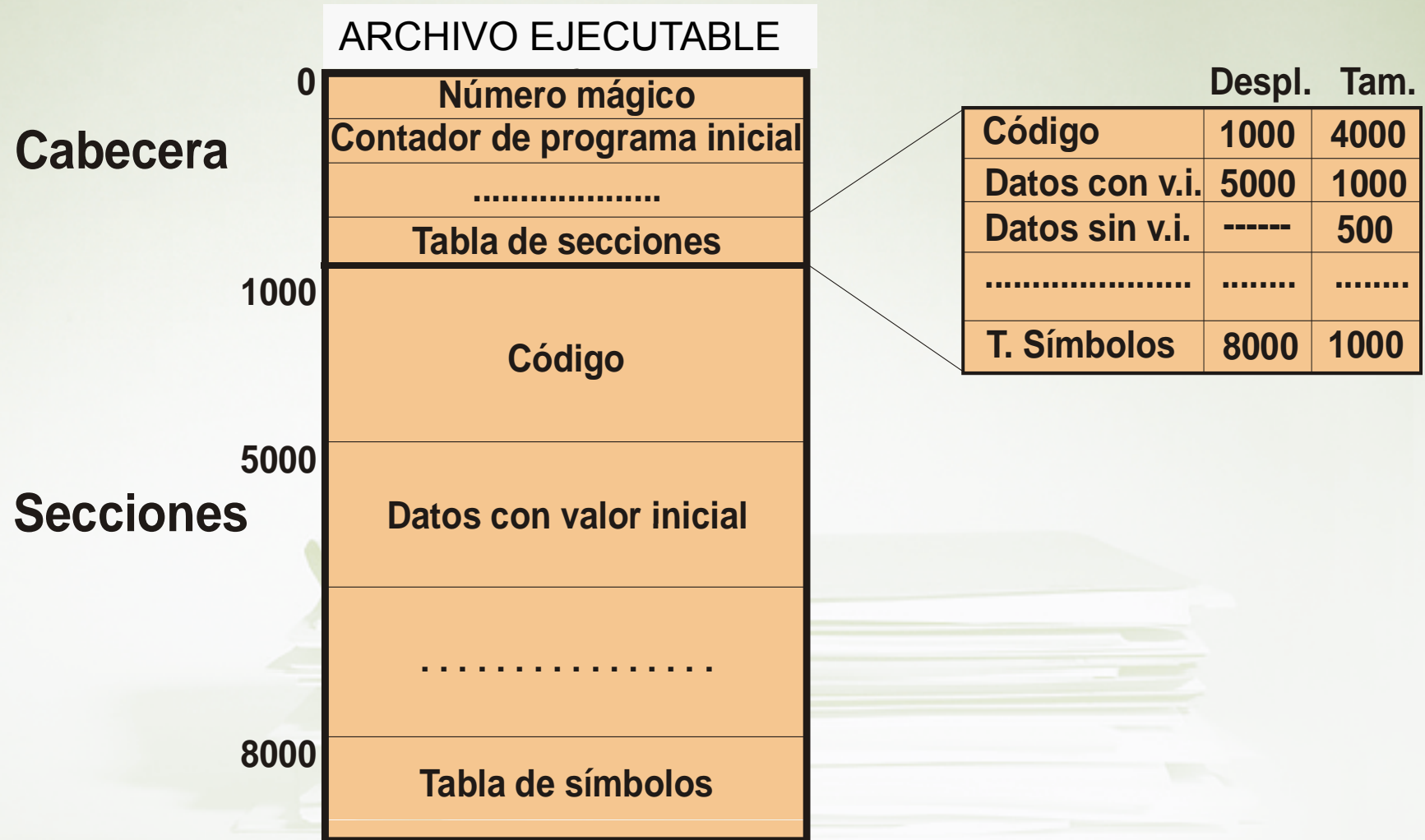
```
: $ ) -- hexdump -C /usr/bin/mbchk
```

```
00000000  7f 45 4c 46 01 01 01 00  00 00 00 00 00 00 00 00 | .ELF..... |
00000010  02 00 03 00 01 00 00 00  10 81 04 08 34 00 00 00 | .....4... |
```


Archivos ejecutables.

- Punto de entrada del programa: es decir el valor que inicialmente contendrá el contador de programa.
- Tabla de secciones: Para cada una de ellas se especifica: tipo, dirección de comienzo en el archivo y tamaño, como ejemplo se tienen las bibliotecas dinámicas usadas.

Archivos ejecutables.



Archivos ejecutables.

Las secciones más relevantes se volcaran en el mapa de memoria del proceso: códigos, datos con valor inicial y datos sin valor inicial.

Archivos ejecutables.

Variables Globales:

- Estáticas.
- Se crean al iniciarse el programa.
- Existen durante toda la ejecución del proceso.
- Dirección fija en memoria y en ejecutable.

Variables Locales y parámetros:

- Dinámicas.
- Se crean al invocar la función.
- Se destruyen al retornar.
- La dirección se calcula en tiempo de ejecución.

Mapa de memoria de un proceso

El mapa de memoria o imagen del proceso estará compuesto por un conjunto de regiones o segmentos; cada una de ellas almacena cierto tipo de información.

Mapa de memoria de un proceso

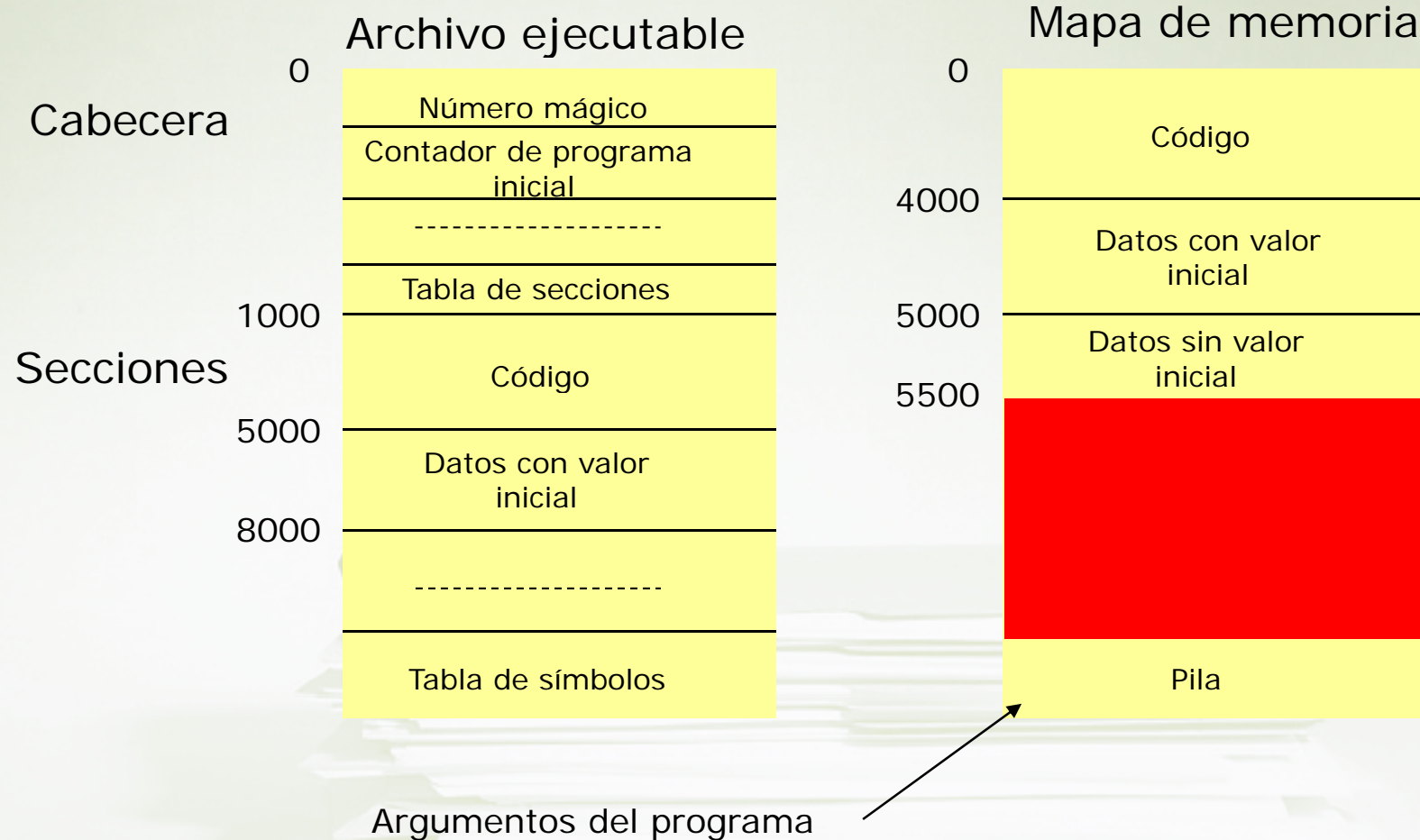
Cada región tiene asociada una información (un objeto de memoria), consiste en una zona contigua de memoria tratada como unidad y que puede estar protegida o compartida.

Mapa de memoria de un proceso

La ejecución de un programa crea un mapa de memoria a partir del archivo ejecutable. Cada sección del ejecutable da lugar a una región del mapa inicial.

- Código (texto)
- Datos con valor inicial
- Datos sin valor inicial
- Pila.

Mapa de memoria de un proceso



Mapa de memoria de un proceso

Durante la ejecución de un proceso se crean nuevas regiones. Es decir el mapa de memoria tiene un carácter dinámico. Las nuevas regiones creadas en tiempo de ejecución pueden ser:

Mapa de memoria de un proceso

- **Región Heap**
 - Soporte de memoria dinámica (malloc en C)
- **Archivo proyectado**
 - Región asociada al archivo proyectado
- **Memoria compartida**
 - Región asociada a la zona de memoria compartida
- **Pilas de threads**
 - Cada pila de hilo corresponde a una región.

Mapa de memoria

Código
Datos con valor inicial
Datos sin valor inicial
Heap
Fichero proyectado F
Zona de memoria compartida
Código biblioteca dinámica B
Datos biblioteca dinámica B
Pila de thread 1
Pila del proceso

Mapa de memoria de un proceso

El mapa de memoria evoluciona durante la ejecución de un proceso y se pueden distinguir las siguientes operaciones:

- Crear región.
- Eliminar región.
- Cambiar tamaño de la región.
- Duplicar región.

Técnica de intercambio (swapping)

La técnica de intercambio o swapping significo en su momento la manera de permitir que en los sistemas de tiempo compartido existieran más procesos de los que caben en memoria.

Es un mecanismo antecesor de la memoria virtual.

Técnica de intercambio (swapping)

El intercambio se basa en usar un disco o parte de un disco (dispositivo de swap) como respaldo de la memoria principal. Cuando no caben en memoria todos los procesos activos se elige un proceso residente y se copia a swap su imagen de memoria.

Técnica de intercambio (swapping)

El criterio de selección puede tener en cuenta aspectos tales como la prioridad del proceso, el tamaño de su mapa de memoria, el tiempo que lleva ejecutando y principalmente su estado.

Técnica de intercambio (swapping)

Expulsar (*swap-out*) a los procesos bloqueados. El proceso de expulsión no implica copiar toda la imagen del proceso .

Un proceso expulsado vuelve a cargarse (*swap-in*) cuando esté listo para ejecutar y haya espacio en memoria.

Técnica de intercambio (swapping)

Políticas de asignación de espacio en swap:

Preasignación: al crear el proceso se reserva espacio de swap.

NO Preasignación: sólo se reserva espacio de swap al expulsar.

Memoria virtual

Introducción

La técnica de la MV se usa prácticamente en todos los SO modernos. Esta técnica se basa en transferir información entre memoria principal y memoria secundaria (por lo que involucra varios niveles de la jerarquía de memoria)

Suele implementarse en un esquema de paginación (es decir, la unidad de información intercambiada entre los diferentes niveles de la jerarquía de memoria es la página)

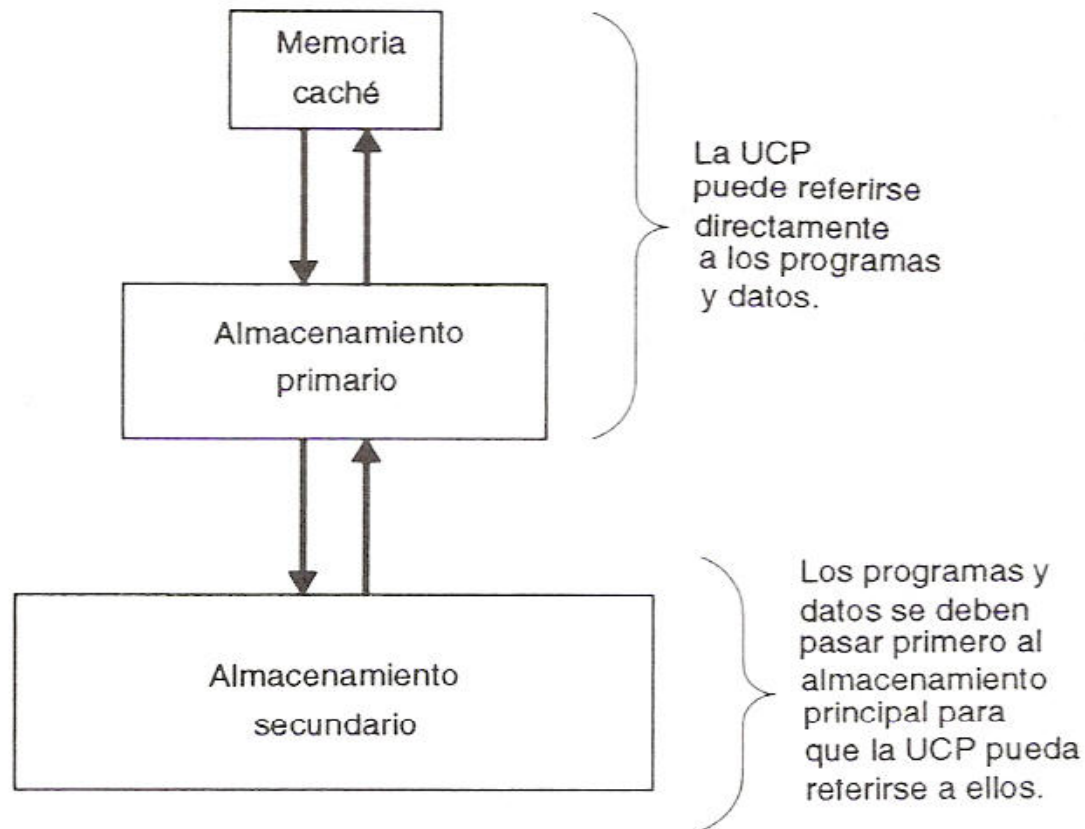
Memoria virtual

Disminuye el tiempo de acceso al almacenamiento.

Aumenta la velocidad de acceso al almacenamiento.

Aumenta el costo de almacenamiento por bit.

Disminuye la capacidad de almacenamiento.



Organización jerárquica del almacenamiento

Memoria virtual

Introducción

Elemento Clave: Proximidad referencial habitual de los procesos. Esta propiedad permite que un proceso puede funcionar disponiendo en memoria de una parte de su imagen de memoria (conjunto residente).

Objetivo final: conseguir que la información necesaria para un proceso (conjunto de trabajo) vaya ocupando la memoria principal según se va necesitando (es decir, conjunto de trabajo === conjunto residente)

Memoria virtual

Ventajas:

- a) Aumento del grado de multiprogramación. Por tanto, aumento en el rendimiento del sistema
- b) Posibilidad de ejecutar programas más grandes que la memoria principal disponible

El uso de la MV no implica que se acelere la ejecución del programa (más bien al contrario, debido a la sobrecarga asociada a los movimientos de información entre niveles de la jerarquía). Este mecanismo no es apropiado para sistemas de tiempo real.

Memoria virtual

Paginación

- **Página:** Zona contigua de memoria de determinado tamaño.
- **Organización:**
 - El mapa de memoria del proceso se considera dividido en páginas.
 - La memoria principal se considera dividida en marcos de página (tamaño de marco = tamaño de página).
 - Los marcos contendrán páginas de los procesos en ejecución

Características

- + La memoria principal está dividida en trozos pequeños de tamaño fijo llamados *marcos*.
- + El compilador o el sistema de administración de memoria dividen los programas en *páginas*.
- + Puede presentarse la Fragmentación Interna en los marcos.
- + No hay fragmentación externa.
- + El sistema operativo (SO) debe mantener una *tabla de páginas* para cada proceso, indicando en qué marco está cada página.

Características

- + El SO debe mantener una lista de marcos libres.
- + El procesador emplea el *número de página* y el *desplazamiento* para calcular las *direcciones absolutas (físicas)*.
- + No todas las páginas de un proceso tienen que estar en marcos de la memoria principal para que el proceso se ejecute. Las páginas se pueden leer cuando se necesiten.
- + La carga de una página en la memoria principal puede exigir descargar otra del disco.

Memoria virtual

Paginación

- La **tabla de páginas** (TP) relaciona cada página con el marco que la contiene. El hardware de traducción (MMU) usa la tabla de páginas para traducir direcciones lógicas a físicas



TCP

Los elementos que debe contener la *Tabla de Correspondencia de Páginas (TCP)* son:

<i>Bit de residencia de página</i>	<i>Dirección en memoria secundaria</i>	<i>Número de marco de página</i>
<i>r</i>	<i>s</i>	<i>p'</i>

r = 0 si la página no está en memoria principal

r = 1 si la página está en memoria principal

Correspondencia Directa

Este método nos ayudará a encontrar la *dirección real (dirección física)* a partir de la dirección virtual utilizando la estructura de la TCP.

La dirección virtual tiene la estructura:

$$V = (p, d)$$

Donde

p = número de página

d = desplazamiento

Errores

- + *Error de Página* : Cuando una página no está en memorial principal, es decir, $r = 0$.
- + *Error por desbordamiento* : Cuando el desplazamiento es mayor o igual que el tamaño de la página.



Ejemplo

Tenemos dos procesos A y B, cada uno con su Tabla de Correspondencia de Páginas. La memoria física inicia a partir de la dirección 500 y el tamaño de la página es de 20. Obtener las direcciones físicas de las siguientes direcciones virtuales, a partir de las TCP's de cada proceso:

a) $P(A)_v = (2, 18)$

b) $P(B)_v = (1, 7)$

c) $P(A)_v = (3, 5)$



Ejemplo (continuación)

TCP
Proceso A

<i>r</i>	<i>s</i>	<i>p'</i>
0	520	5
1	410	20
1	235	11
0	450	8
1	210	5
0	135	14
0	280	10

TCP
Proceso B

<i>r</i>	<i>s</i>	<i>p'</i>
1	320	8
1	100	4
0	70	12
0	300	11
0	720	10

Memoria Principal

500	0
520	1
540	2
560	3
580	4
600	5
620	6
640	7
660	8
680	9
700	10
720	11
740	...

Solución

Memoria Principal

500	0
520	1
540	2
560	3
580	4
600	5
620	6
640	7
660	8
680	9
700	10
720	11
740	...

Para encontrar la dirección física ubicamos la dirección en memoria principal en donde está cargado el marco de memoria donde se aloja la página (se observa en la TCP en el campo **p'**) y le sumamos el desplazamiento.

a) $P(A)v = (2,18)$

Observando la TCP del Proceso A, la página 2 está cargada en el marco 11 y éste comienza en la dirección 720. Por lo tanto:

$$\text{Dirección física} = 720 + 18 = 738$$

b) $P(B)v = (1,7)$

Observando la TCP del Proceso B, la página 1 está cargada en el marco 4 y éste comienza en la dirección 580. Por lo tanto:

$$\text{Dirección física} = 580 + 7 = 587$$

c) $P(A)v = (3,5)$

Observando la TCP del Proceso A, la página 3 no está en memoria principal, ya que el valor del campo **r** es cero. Por lo tanto, hay un **error de página**.

Memoria Virtual

Segmentación

Características

- + La memoria principal no está dividida.
- + El programador especifica al compilador los segmentos del programa, es decir, el programador toma la decisión.
- + No hay fragmentación interna.
- + Fragmentación externa.
- + El sistema operativo (SO) debe mantener una *tabla de segmentos* para cada proceso, indicando la dirección de carga y la longitud de cada segmento.

Características

- # El SO debe mantener una lista de huecos libres en la memoria principal.
- # El procesador emplea el *número de segmento* y el *desplazamiento* para calcular las *direcciones absolutas (físicas)*.
- # No todos los segmentos de un proceso tienen que estar en marcos de la memoria principal para que el proceso se ejecute. Los segmentos se pueden leer cuando se necesiten.
- # La carga de un segmento en la memoria principal puede requerir descargar uno o más segmentos al disco.

TCS

Los elementos que debe contener la *Tabla de Correspondencia de Segmentos (TCS)* son:

Bit de residencia del segmento	Dirección en memoria secundaria	Longitud del segmento	Bits de Protección			Dirección base del segmento	
<i>r</i>	<i>s</i>	<i>L</i>	<i>r</i>	<i>w</i>	<i>x</i>	<i>a</i>	<i>s'</i>

r = 0 si el segmento no está en memoria principal

r = 1 si el segmento está en memoria principal

Bits de protección: (1 - si, 0 - no)

r - acceso para lectura

w - acceso para escritura

x - acceso para ejecución

a - acceso para adición

Correspondencia Directa

Este método nos ayudará a encontrar la *dirección real (dirección física)* a partir de la dirección virtual utilizando la estructura de la TCS.

La dirección virtual tiene la estructura:

$$V = (s, d)$$

Donde

s = número de segmento

d = desplazamiento

Errores

- + *Error de Segmento* : Cuando un segmento no está en memorial principal, es decir, $r = 0$.
- + *Error por desbordamiento* : Cuando el desplazamiento es mayor o igual que la longitud del segmento.



Ejemplo

Tenemos la siguiente TCS para los Procesos A, B y C.

El *Proceso A* inicia a partir de la dirección física **100**.

El *Proceso B* inicia a partir de la dirección física **112**.

El *Proceso C* inicia a partir de la dirección física **105**.

	<i>r</i>	<i>s</i>	<i>L</i>	<i>r</i>	<i>w</i>	<i>x</i>	<i>a</i>	<i>s'</i>
100	0	1024	10	1	0	0	1	532
101	1	1035	45	1	1	1	0	1150
102	1	532	28	0	1	1	0	1200
103	0	1150	37	0	1	0	0	45
104	1	1200	36	0	1	0	1	118
105	1	45	20	0	0	1	0	2005
106	0	118	70	1	1	0	1	2310
107	1	2005	55	0	1	0	1	324
108	0	2310	100	1	1	1	1	1501
109	0	324	35	0	0	0	0	1708
110	1	1501	62	0	1	0	0	1810
111	1	1708	48	1	1	0	1	1340
112	1	1810	30	1	0	0	0	1492
113	0	1340	80	1	1	1	1	1232
114	1	1492	27	1	0	0	0	750
115	1	1237	75	0	1	0	1	824
116	0	750	62	0	1	1	0	1024
117	1	824	46	1	1	0	1	1035

Ejemplo (continuación)

Preguntas:

1. ¿Cuántos segmentos tiene cada proceso?

Proceso A \rightarrow 5 segmentos

Proceso B \rightarrow 6 segmentos

Proceso C \rightarrow 7 segmentos

2. ¿De qué tamaño es cada proceso?

Proceso A = 156

Proceso B = 320

Proceso C = 390

3. Obtener las direcciones físicas de las siguientes direcciones virtuales.

a) $P(A)_v = (1, 18) \rightarrow$ Dirección Física = 1168

b) $P(B)_v = (0, 27) \rightarrow$ Dirección Física = 1519

c) $P(C)_v = (2, 34) \rightarrow$ Dirección Física = 358

Ejemplo (continuación)

Preguntas:

4. Proporcioné una dirección virtual por la que ocurra un error de segmento.

a) $P(A)_v = (0, 9)$

b) $P(B)_v = (1, 20)$

c) $P(C)_v = (3, 10)$

5. Proporcioné una dirección virtual donde ocurra un desbordamiento.

a) $P(A)_v = (1, 60)$

b) $P(B)_v = (2, 30)$

c) $P(C)_v = (6, 48)$

Ejemplo (continuación)

Preguntas:

6. Proporcioné una dirección virtual en la que ocurra una inexistencia de segmento.

a) $P(A)_v = (10, 15)$

b) $P(B)_v = (6, 30)$

c) $P(C)_v = (8, 47)$

7. Se desea escribir en la dirección virtual $P(C)_v = (0, 18)$, ¿qué ocurre?

El segmento 0 del proceso C está en la dirección 105. Ocurre un error de escritura, ya que no tiene permiso. En el campo w el valor es cero.