

# Administración de Memoria

## Sistemas Operativos

Gonzalo Pablo Fernández

Departamento de Computación  
Facultad de Ciencias Exactas y Naturales  
Universidad de Buenos Aires

1 de octubre de 2015

- ▶ Gestor de Memoria (Memory Manager):  
Componente del sistema operativo encargado de administrar la memoria.

- ▶ Gestor de Memoria (Memory Manager):

Componente del sistema operativo encargado de administrar la memoria.

¿Más específico?

- ▶ Gestor de Memoria (Memory Manager):

Componente del sistema operativo encargado de administrar la memoria.

¿Más específico?

Se encarga de (entre otras cosas):

1. Asegurar la disponibilidad de memoria
2. Asignar y liberar memoria.
3. Organizar la memoria disponible.
4. Asegurar la protección de la memoria.
5. Permitir acceso a memoria compartida.

# Repaso de Orga II

## ► MMU

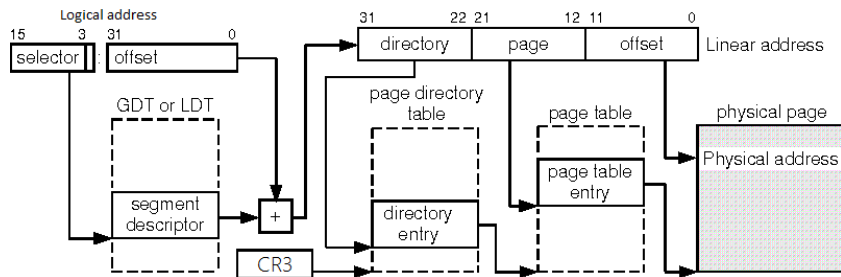
## ▶ MMU

- ▶ Unidad de Gestión de Memoria (Memory Management Unit)
- ▶ Permite traducir direcciones virtuales a físicas.

- ▶ MMU
  - ▶ Unidad de Gestión de Memoria (Memory Management Unit)
  - ▶ Permite traducir direcciones virtuales a físicas.
- ▶ ¿Memoria virtual?

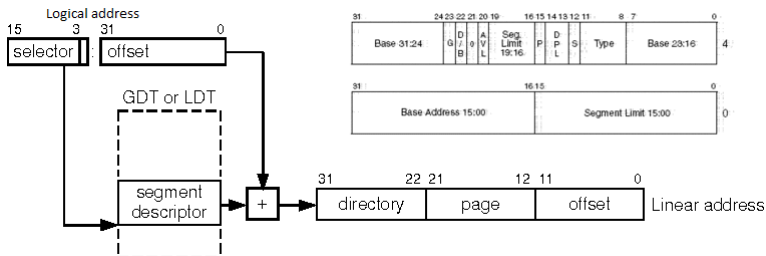


- ▶ MMU
  - ▶ Unidad de Gestión de Memoria (Memory Management Unit)
  - ▶ Permite traducir direcciones virtuales a físicas.
- ▶ ¿Memoria virtual?



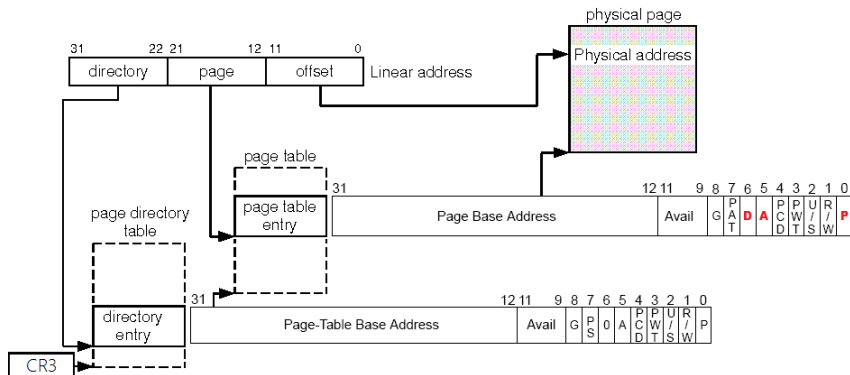
# Segmentación

- ▶ Primeros bits de la dirección indexan el **Descriptor de Segmento** en la tabla de descriptores.
- ▶ El resto es el offset dentro del segmento.
- ▶ La **GDT** (y la **LDT** de cada proceso) contiene los descriptores de los segmentos.
- ▶ Los descriptores almacenan información sobre los segmentos (inicio en memoria física, **tamaño**, atributos).



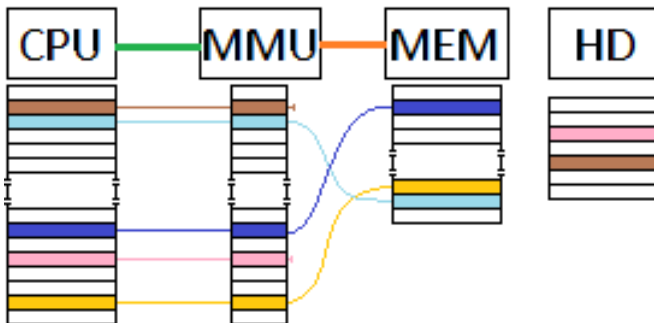
# Paginación

- ▶ Los conjuntos de bits indexan en las sucesivas tablas de páginas. Los últimos son el offset en la página.
- ▶ Cada entrada contiene la dirección base de la página más algunos atributos.



# direcciones lógicas vs direcciones físicas

- ▶ Memoria física: Una celda dentro de la memoria del sistema. El tamaño está determinado por el hardware.
  - ▶ Un **marco de página** es una porción de memoria física.
- ▶ Memoria virtual: Una representación de la información almacenada. El tamaño depende de la unidad de direccionamiento y la cantidad de bits de direccionamiento.
  - ▶ Una **página** es una porción de memoria virtual.



# 1. Asegurar la disponibilidad de memoria

- ▶ Otra definición de memoria virtual: Hacerle creer al procesador que dispone de más memoria de la que realmente tiene.  
¿Cuánta memoria tiene mi sistema?

# 1. Asegurar la disponibilidad de memoria

- ▶ Otra definición de memoria virtual: Hacerle creer al procesador que dispone de más memoria de la que realmente tiene.  
¿Cuánta memoria tiene mi sistema?
- ▶ Cantidad de bytes de memoria física = MEM\_SIZE.
- ▶ Cantidad de bytes de memoria virtual =  $2^{DIR\_BITS} * DIR\_UNIT$ .
- ▶ Cantidad de celdas de memoria física = MEM\_SIZE / DIR\_UNIT.
- ▶ Cantidad de celdas de memoria virtual =  $2^{DIR\_BITS}$ .

# 1. Asegurar la disponibilidad de memoria

- ▶ Otra definición de memoria virtual: Hacerle creer al procesador que dispone de más memoria de la que realmente tiene.  
¿Cuánta memoria tiene mi sistema?
- ▶ Cantidad de bytes de memoria física = MEM\_SIZE.
- ▶ Cantidad de bytes de memoria virtual =  $2^{DIR\_BITS} * DIR\_UNIT$ .
- ▶ Cantidad de celdas de memoria física = MEM\_SIZE / DIR\_UNIT.
- ▶ Cantidad de celdas de memoria virtual =  $2^{DIR\_BITS}$ .



# 1. Asegurar la disponibilidad de memoria

- ▶ Otra definición de memoria virtual: Hacerle creer al procesador que dispone de más memoria de la que realmente tiene.  
¿Cuánta memoria tiene mi sistema?
- ▶ Cantidad de bytes de memoria física = MEM\_SIZE.
- ▶ Cantidad de bytes de memoria virtual =  $2^{DIR\_BITS} * DIR\_UNIT$ .
- ▶ Cantidad de celdas de memoria física =  $MEM\_SIZE / DIR\_UNIT$ .
- ▶ Cantidad de celdas de memoria virtual =  $2^{DIR\_BITS}$ .

Ejercicio:

- ▶ Tengo 16 bits para direccionar. ¿Cuánta memoria virtual puedo tener direccionando a byte? ¿Cuánta física?



# 1. Asegurar la disponibilidad de memoria

- ▶ Otra definición de memoria virtual: Hacerle creer al procesador que dispone de más memoria de la que realmente tiene.  
¿Cuánta memoria tiene mi sistema?
- ▶ Cantidad de bytes de memoria física = MEM\_SIZE.
- ▶ Cantidad de bytes de memoria virtual =  $2^{DIR\_BITS} * DIR\_UNIT$ .
- ▶ Cantidad de celdas de memoria física = MEM\_SIZE / DIR\_UNIT.
- ▶ Cantidad de celdas de memoria virtual =  $2^{DIR\_BITS}$ .

Ejercicio:

- ▶ Tengo 16 bits para direccionar. ¿Cuánta memoria virtual puedo tener direccionando a byte? ¿Cuánta física?
- ▶ Tengo 8 bits para direccionar 256 ( $2^8$ ) bytes de memoria física. ¿Cómo puedo duplicar la memoria virtual?

# 1. Asegurar la disponibilidad de memoria

- ▶ Otra definición de memoria virtual: Hacerle creer al procesador que dispone de más memoria de la que realmente tiene.  
¿Cuánta memoria tiene mi sistema?
- ▶ Cantidad de bytes de memoria física = MEM\_SIZE.
- ▶ Cantidad de bytes de memoria virtual =  $2^{DIR\_BITS} * DIR\_UNIT$ .
- ▶ Cantidad de celdas de memoria física =  $MEM\_SIZE / DIR\_UNIT$ .
- ▶ Cantidad de celdas de memoria virtual =  $2^{DIR\_BITS}$ .

Ejercicio:

- ▶ Tengo 16 bits para direccionar. ¿Cuánta memoria virtual puedo tener direccionando a byte? ¿Cuánta física?
- ▶ Tengo 8 bits para direccionar 256 ( $2^8$ ) bytes de memoria física. ¿Cómo puedo duplicar la memoria virtual?
- ▶ Tengo 65536 ( $2^{16}$ ) bytes de memoria física dividida en celdas de 16 bits. ¿Cuántos bits necesito para direccionar?

## 2. Asignar y liberar memoria

- ▶ Asignar = reservar una porción de memoria para un proceso.
  - ▶ La Memoria pasa a estar ocupada por el proceso que la solicitó.
  - ▶ Tenemos que saber quién es el dueño de esa porción de memoria.
- ▶ Liberar = la memoria vuelve a estar disponible.

## 2. Asignar y liberar memoria

- ▶ Asignar = reservar una porción de memoria para un proceso.
  - ▶ La Memoria pasa a estar ocupada por el proceso que la solicitó.
  - ▶ Tenemos que saber quién es el dueño de esa porción de memoria.
- ▶ Liberar = la memoria vuelve a estar disponible.

Cuando hablamos de *memoria*, ¿nos referimos a memoria virtual o física?

### 3. Organizar la memoria disponible

Pensemos en la memoria como una colección de celdas.  
El MM debe elegir qué porción de memoria (qué celdas) asignar.

### 3. Organizar la memoria disponible

Pensemos en la memoria como una colección de celdas.

El MM debe elegir qué porción de memoria (qué celdas) asignar.

- ▶ ¿Cómo agrupamos la memoria?
  - ▶ De a celdas.
  - ▶ En Bloques de tamaño fijo.
  - ▶ En Bloques de tamaño variable.

### 3. Organizar la memoria disponible

Pensemos en la memoria como una colección de celdas.

El MM debe elegir qué porción de memoria (qué celdas) asignar.

- ▶ ¿Cómo agrupamos la memoria?
  - ▶ De a celdas.
  - ▶ En Bloques de tamaño fijo.
  - ▶ En Bloques de tamaño variable.
- ▶ ¿Cómo organizamos la memoria libre?  
(para determinar si está libre)
  - ▶ Con un mapa de bits.
  - ▶ Con una lista enlazada.

### 3. Organizar la memoria disponible

Pensemos en la memoria como una colección de celdas.

El MM debe elegir qué porción de memoria (qué celdas) asignar.

- ▶ ¿Cómo agrupamos la memoria?
  - ▶ De a celdas.
  - ▶ En Bloques de tamaño fijo.
  - ▶ En Bloques de tamaño variable.
- ▶ ¿Cómo organizamos la memoria libre?  
(para determinar si está libre)
  - ▶ Con un mapa de bits.
  - ▶ Con una lista enlazada.
- ▶ Mecanismos más sofisticados:
  - ▶ Segmentación.
  - ▶ Paginación.
  - ▶ Segmentación + paginación.



# Más sobre paginación

- ▶ Paginación nos permite mapear mucha más memoria de la que realmente tiene el sistema, pero...

Si una página no está cargada en ningún marco de página, el MM se encarga de ir a buscarla al disco y cargarla en memoria.

# Más sobre paginación

- ▶ Paginación nos permite mapear mucha más memoria de la que realmente tiene el sistema, pero...

Si una página no está cargada en ningún marco de página, el MM se encarga de ir a buscarla al disco y cargarla en memoria.

¿Qué pasa si no hay lugar en la memoria? (por lo general,  $\text{MEM\_VIRTUAL} > \text{MEM\_FISICA}$ ).

# Más sobre paginación

- ▶ Paginación nos permite mapear mucha más memoria de la que realmente tiene el sistema, pero...

Si una página no está cargada en ningún marco de página, el MM se encarga de ir a buscarla al disco y cargarla en memoria.

¿Qué pasa si no hay lugar en la memoria? (por lo general,  $\text{MEM\_VIRTUAL} > \text{MEM\_FISICA}$ ).

- ▶ Algoritmos de remoción:

- ▶ FIFO
- ▶ LRU
- ▶ Segunda oportunidad
- ▶ Not Recently Used

# Más sobre paginación

- ▶ Paginación nos permite mapear mucha más memoria de la que realmente tiene el sistema, pero...

Si una página no está cargada en ningún marco de página, el MM se encarga de ir a buscarla al disco y cargarla en memoria.

¿Qué pasa si no hay lugar en la memoria? (por lo general,  $\text{MEM\_VIRTUAL} > \text{MEM\_FISICA}$ ).

- ▶ Algoritmos de remoción:

- ▶ FIFO

La clásica de siempre.

- ▶ LRU

Desalojo la página que hace más tiempo que no se usa.

- ▶ Segunda oportunidad

Si fue referenciada, le doy otra chance.

- ▶ Not Recently Used

Primero desalojo las ni referenciadas ni modificadas. Después las referenciadas y por último las modificadas.

# Ejercicio: Algoritmos de remoción

- ▶ Tengo un sistema con 6 páginas y sólo 4 marcos de página. La memoria comienza vacía.  
Llegan los siguientes pedidos de memoria (número de página) en ese orden:  
1, 2, 1, 3, 4, 3, 5, 6, 2
- ▶ Indique qué página se desaloja tras cada pedido utilizando los algoritmos FIFO, LRU y Second Chance y calcule el *hit-rate* en cada caso.

# Solución

	<b>FIFO</b>		<b>LRU</b>		<b>Second Chance</b>	
	<div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div></div>
1	<div><div>1</div><div></div><div></div><div></div></div>	<div><div>1</div><div></div><div></div><div></div></div>	<div><div>1</div><div></div><div></div><div></div></div>	<div><div>1</div><div></div><div></div><div></div></div>	<div><div>1</div><div></div><div></div><div></div></div>	<div><div>1</div><div></div><div></div><div></div></div>
2	<div><div>1</div><div>2</div><div></div><div></div></div>	<div><div>1</div><div>2</div><div></div><div></div></div>	<div><div>1</div><div>2</div><div></div><div></div></div>	<div><div>1</div><div>2</div><div></div><div></div></div>	<div><div>1</div><div>2</div><div></div><div></div></div>	<div><div>1</div><div>2</div><div></div><div></div></div>
1	<div><div>1</div><div>2</div><div></div><div></div></div>	<div><div>1</div><div>2</div><div></div><div></div></div>	<div><div>1</div><div>2</div><div></div><div></div></div>	<div><div>2</div><div>1</div><div></div><div></div></div>	<div><div>1</div><div>2</div><div></div><div></div></div>	<div><div>1</div><div>2</div><div></div><div></div></div>
3	<div><div>1</div><div>2</div><div>3</div><div></div></div>	<div><div>1</div><div>2</div><div>3</div><div></div></div>	<div><div>1</div><div>2</div><div>3</div><div></div></div>	<div><div>2</div><div>1</div><div>3</div><div></div></div>	<div><div>1</div><div>2</div><div>3</div><div></div></div>	<div><div>1</div><div>2</div><div>3</div><div></div></div>
4	<div><div>1</div><div>2</div><div>3</div><div>4</div></div>	<div><div>1</div><div>2</div><div>3</div><div>4</div></div>	<div><div>1</div><div>2</div><div>3</div><div>4</div></div>	<div><div>2</div><div>1</div><div>3</div><div>4</div></div>	<div><div>1</div><div>2</div><div>3</div><div>4</div></div>	<div><div>1</div><div>2</div><div>3</div><div>4</div></div>
3	<div><div>1</div><div>2</div><div>3</div><div>4</div></div>	<div><div>1</div><div>2</div><div>3</div><div>4</div></div>	<div><div>1</div><div>2</div><div>3</div><div>4</div></div>	<div><div>2</div><div>1</div><div>4</div><div>3</div></div>	<div><div>1</div><div>2</div><div>3</div><div>4</div></div>	<div><div>1</div><div>2</div><div>3</div><div>4</div></div>
5	<div><div>5</div><div>2</div><div>3</div><div>4</div></div>	<div><div>2</div><div>3</div><div>4</div><div>5</div></div>	<div><div>1</div><div>5</div><div>3</div><div>4</div></div>	<div><div>1</div><div>4</div><div>3</div><div>5</div></div>	<div><div>1</div><div>2</div><div>3</div><div>4</div></div>	<div><div>2</div><div>3</div><div>4</div><div>1</div></div>
	<div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div></div>	<div><div>1</div><div>5</div><div>3</div><div>4</div></div>	<div><div>3</div><div>4</div><div>1</div><div>5</div></div>
6	<div><div>5</div><div>6</div><div>3</div><div>4</div></div>	<div><div>3</div><div>4</div><div>5</div><div>6</div></div>	<div><div>6</div><div>5</div><div>3</div><div>4</div></div>	<div><div>4</div><div>3</div><div>5</div><div>6</div></div>	<div><div>1</div><div>5</div><div>3</div><div>4</div></div>	<div><div>4</div><div>1</div><div>5</div><div>3</div></div>
	<div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div><div></div></div>	<div><div>1</div><div>5</div><div>3</div><div>6</div></div>	<div><div>1</div><div>5</div><div>3</div><div>6</div></div>
2	<div><div>5</div><div>6</div><div>2</div><div>4</div></div>	<div><div>4</div><div>5</div><div>6</div><div>2</div></div>	<div><div>6</div><div>5</div><div>3</div><div>2</div></div>	<div><div>3</div><div>5</div><div>6</div><div>2</div></div>	<div><div>2</div><div>5</div><div>3</div><div>6</div></div>	<div><div>5</div><div>3</div><div>6</div><div>2</div></div>

- ▶ Hit-Rate (FIFO) =  $2 / 9$
- ▶ Hit-Rate (LRU) =  $2 / 9$
- ▶ Hit-Rate (SC) =  $2 / 9$

## 4. Asegurar la protección de la memoria

Un proceso no debería poder usar memoria que no reservó.

- ▶ Paginación provee una solución muy elegante para esto:  
Cada proceso tiene su propia tabla de páginas.
- ▶ De forma similar, puede utilizarse segmentación para lo mismo:  
Cada proceso tiene su propia tabla de segmentos.

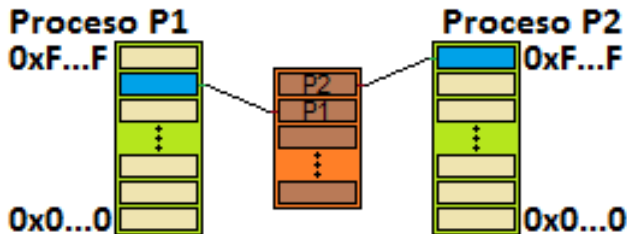


## 4. Asegurar la protección de la memoria

Un proceso no debería poder usar memoria que no reservó.

- ▶ Paginación provee una solución muy elegante para esto:  
Cada proceso tiene su propia tabla de páginas.
- ▶ De forma similar, puede utilizarse segmentación para lo mismo:

Cada proceso tiene su propia tabla de segmentos.



## 5. Permitir acceso a memoria compartida

Queremos que dos procesos lean y escriban sobre una misma variable.

- Una vez más, paginación es la respuesta.

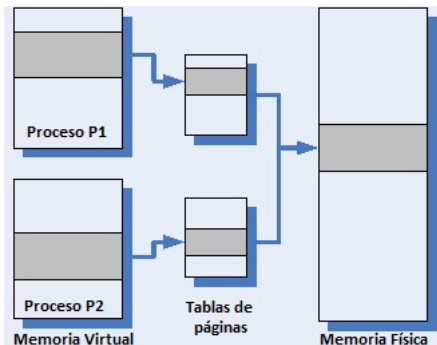
Podemos mapear dos páginas al mismo marco de página.

## 5. Permitir acceso a memoria compartida

Queremos que dos procesos lean y escriban sobre una misma variable.

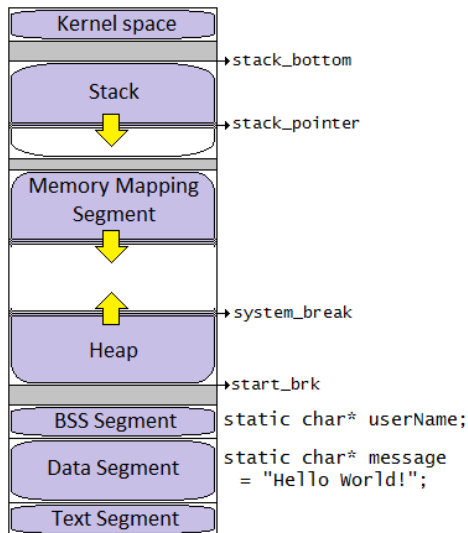
- Una vez más, paginación es la respuesta.

Podemos mapear dos páginas al mismo marco de página.

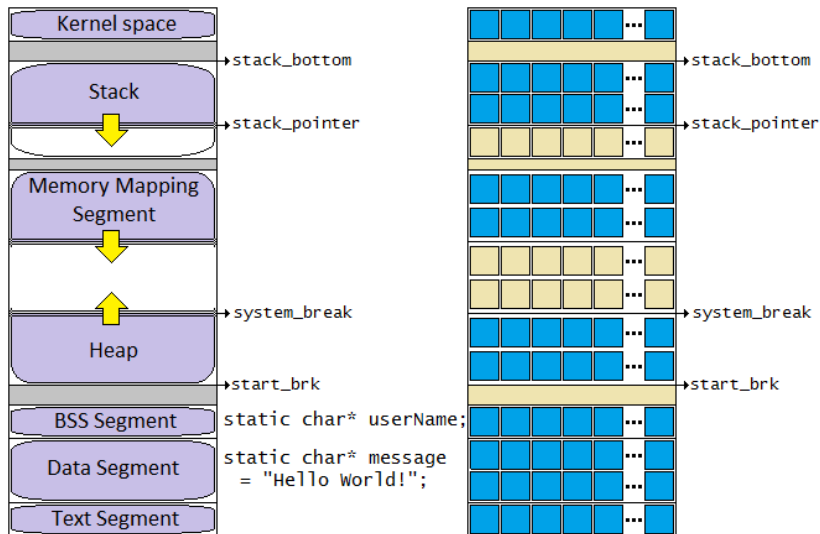


# Malloc, ¿Qué es malloc?

# Malloc, ¿Qué es malloc?



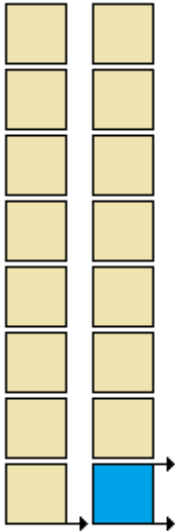
# Malloc, ¿Qué es malloc?



# Heap

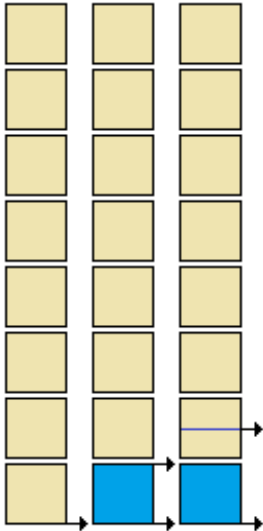


# Heap

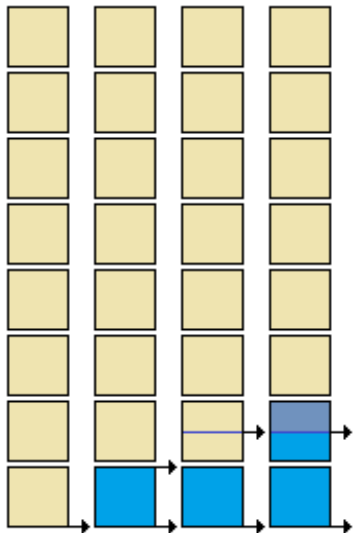




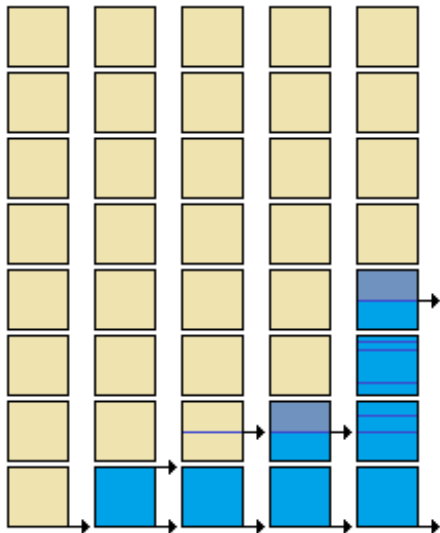
# Heap



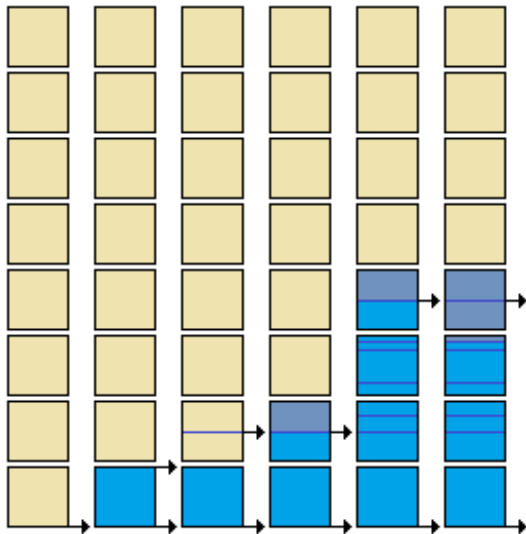
# Heap



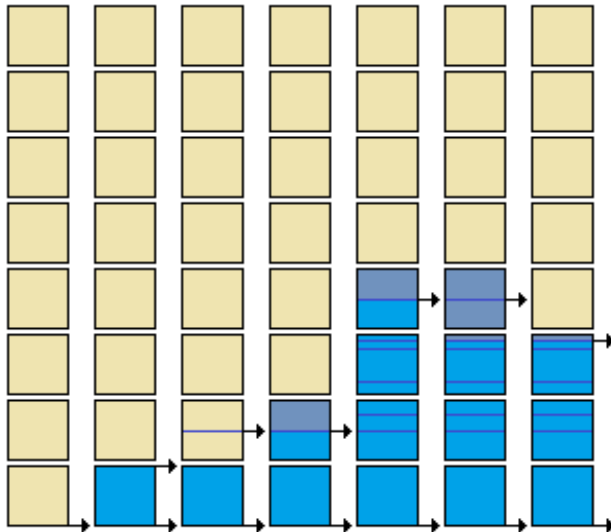
# Heap



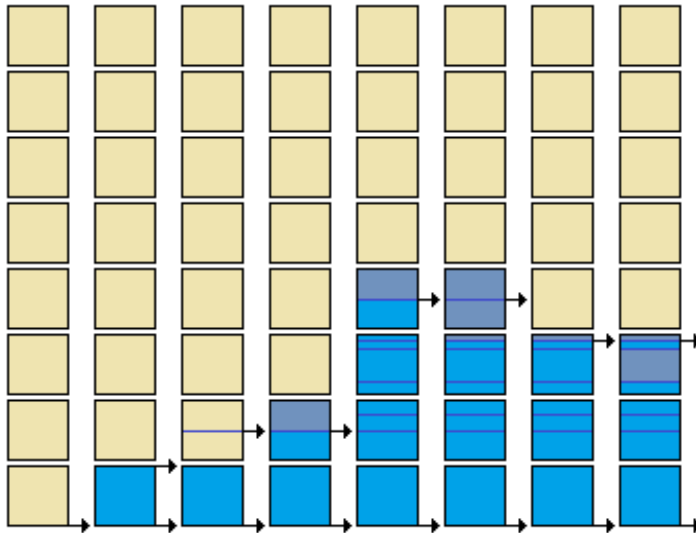
# Heap



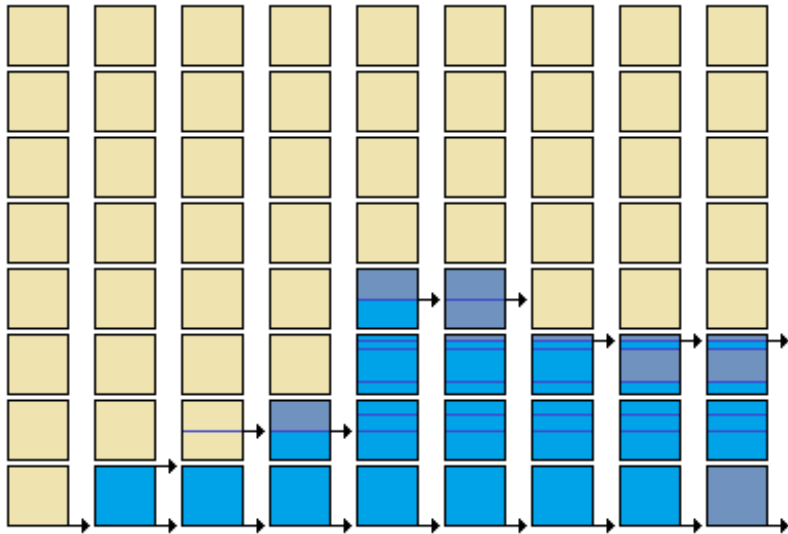
# Heap



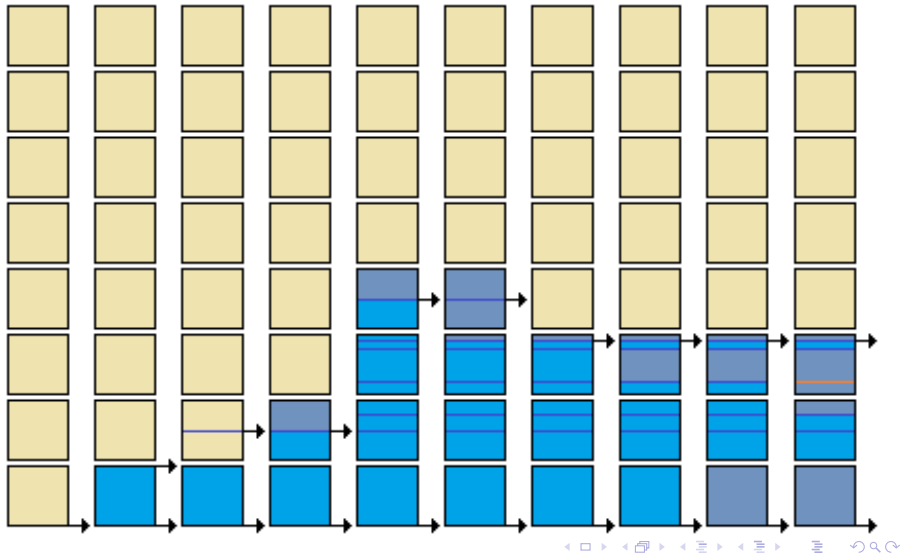
# Heap



# Heap



# Heap





# Yo soy malloc

¿Qué porción de memoria me conviene asignar?

¿Qué porción de memoria me conviene asignar?

¿Cómo agrupamos la memoria?

De a celdas.

En Bloques de tamaño fijo.

En Bloques de tamaño variable.

¿Cómo organizamos la memoria libre?

(para determinar si está libre)

Con un mapa de bits.

Con una lista enlazada.

# Yo soy malloc

¿Qué porción de memoria me conviene asignar?

¿Cómo agrupamos la memoria?

De a celdas.

En Bloques de tamaño fijo.

En Bloques de tamaño variable.

¿Cómo organizamos la memoria libre?

(para determinar si está libre)

Con un mapa de bits.

Con una lista enlazada.

▶ Algoritmos de elección de bloque libre:

▶ First fit

▶ Best fit

▶ Worst fit

▶ Quick fit

# Yo soy malloc

¿Qué porción de memoria me conviene asignar?

¿Cómo agrupamos la memoria?

De a celdas.

En Bloques de tamaño fijo.

En Bloques de tamaño variable.

¿Cómo organizamos la memoria libre?

(para determinar si está libre)

Con un mapa de bits.

Con una lista enlazada.

## ► Algoritmos de elección de bloque libre:

### ► First fit

La primera sección de memoria contigua del tamaño necesario.

### ► Best fit

De todas las secciones de tamaño mayor o igual al tamaño necesario, tomo la más chica.

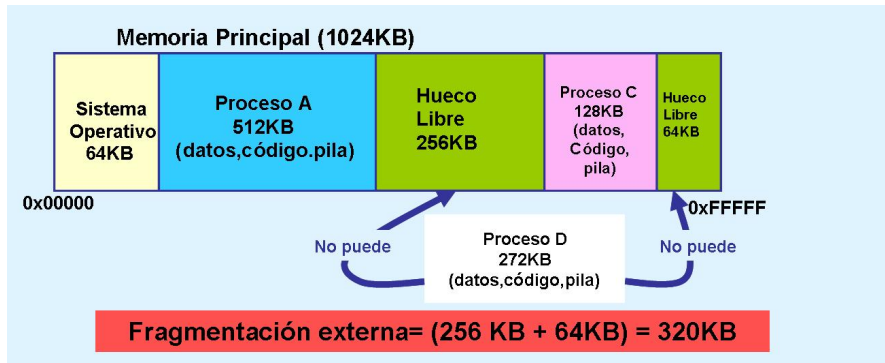
### ► Worst fit

Mejor tomo la más grande.

### ► Quick fit

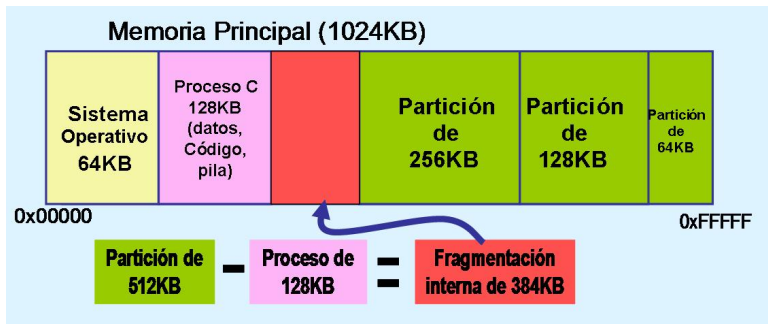
Además me guardo listas de bloques de determinados tamaños.

# Fragmentación Externa



Bloques *pequeños* de memoria no contiguos.

# Fragmentación Interna



Memoria desperdiciada dentro de una partición (un bloque o página).

# Ejercicio: Algoritmos de elección de bloque libre

- ▶ Tengo un sistema con 16 MB de memoria **sin particionar** que direcciona a byte. El estado actual de la memoria es el siguiente:

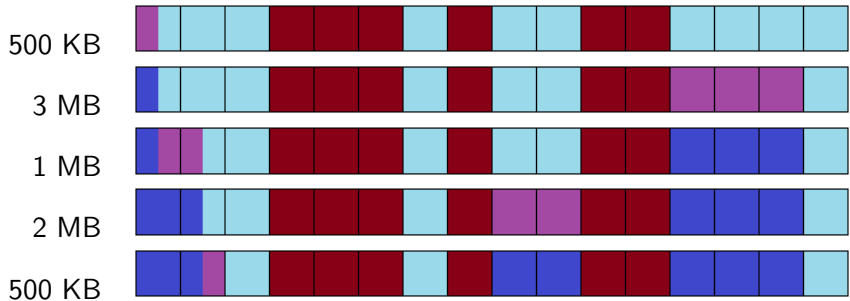


Llegan los siguientes pedidos de memoria en ese orden:

500 KB, 3 MB, 1 MB, 2MB, 500 KB.

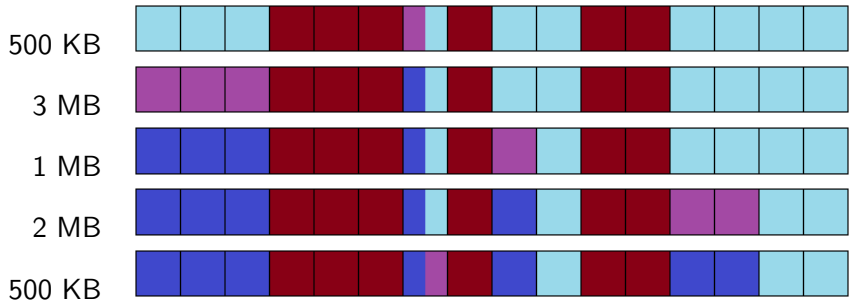
- ▶ Indique qué bloques se asignan para cada pedido utilizando first-fit, best-fit y worst-fit.

# Solución First-Fit

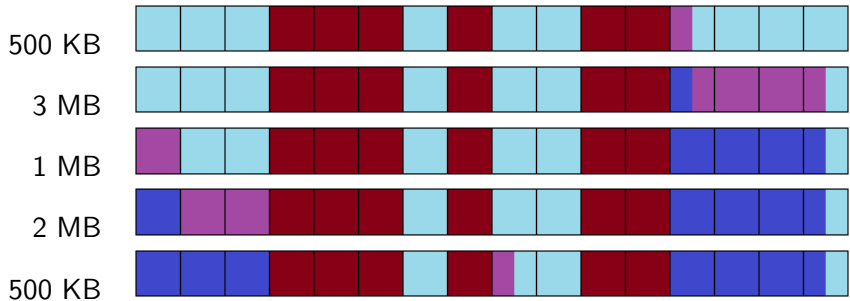




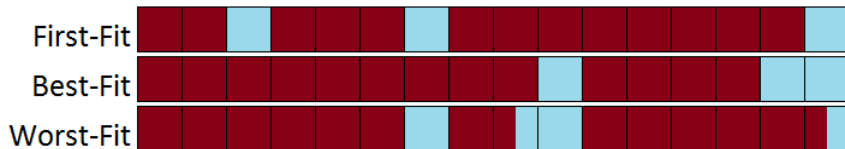
# Solución Best-Fit



# Solución Worst-Fit



# Entonces, ¿Cuál es mejor?



- ▶ ¿Qué pasa cuando una página no está cargada en la memoria?

- ▶ ¿Qué pasa cuando una página no está cargada en la memoria?
1. Un proceso accede a una dirección (virtual) de memoria.
  2. La MMU traduce la dirección virtual a dirección física (accede a la entrada en la última tabla de páginas).
  3. Lee el atributo correspondiente a presencia en la memoria.
  4. Si es negativo, se produce la interrupción Page Fault.
  5. Se ejecuta la **RAI** correspondiente.
  6. Si la memoria está llena, se ejecuta el algoritmo de remoción.
  7. Si la página que se va a desalojar fue modificada, hay que bajarla al disco.
  8. Se carga en el lugar liberado la página solicitada.
  9. Se vuelve a ejecutar la instrucción del proceso que accede a la dirección solicitada.

# Thrashing

- ▶ Situación en la que el SO pasa más tiempo cargando páginas que ejecutando procesos.

Supongan que tenemos 2 procesos y un solo marco de página disponible.

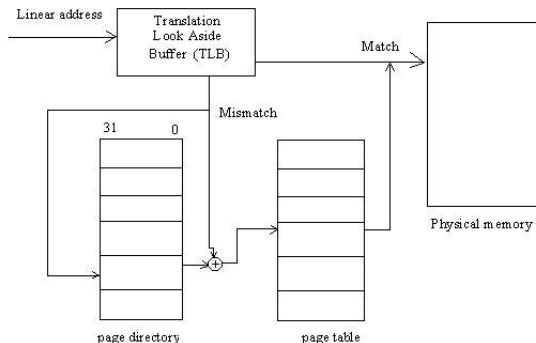
Cada proceso usa una sólo página, pero cada vez que ejecute, su página no va a estar cargada.

- ▶ Siempre va a estar cargada la página del otro.

# TLB (Translation Lookaside Buffer)

Buffer de Traducción Adelantada.

- ▶ Es una caché que guarda 'traducciones'.
- ▶ Paginación de 4 niveles: Cuatro accesos a memoria (1 por cada tabla) más uno para leer la página.



- ▶ ¿Entonces para qué me guardo todas las tablas?

- ▶ Dijimos que al crear un nuevo proceso se duplica toda su memoria.

Sabemos que, en general después de un `fork()` viene un `exec()`. `exec()` inutiliza todas las páginas de memoria, entonces ¿para qué nos gastamos en duplicar todo?

- ▶ Copy-on-Write: Sólo duplico (copy) cuando alguno de los procesos escribe (write).



# FIN

## ¿Preguntas?