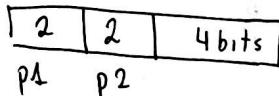


(23)



Tam P1 = 2 bits Offset = 4 bits

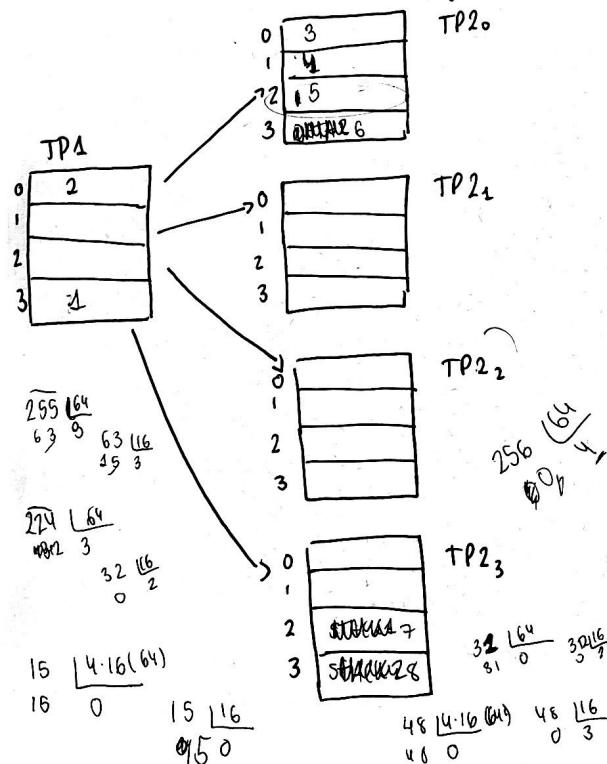
Tam P2 = 2 bits

Tabla páginas 1^{er} nivel = $2^2 = 4$ entradas

Tabla páginas 2º nivel = 4 entradas

$$\text{Tamaño página} = 2^4 = 16 \text{ B}$$

$$\text{Nº marcos de página} = \frac{160 \text{ B} = \text{M.F}}{16 \text{ B/pag}} = 10 \text{ páginas}$$

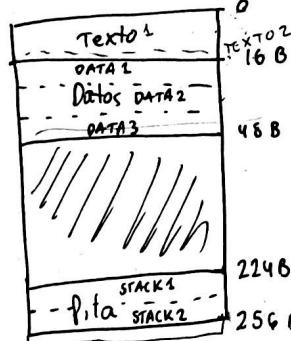


200

also

over
over

Texto 0-15

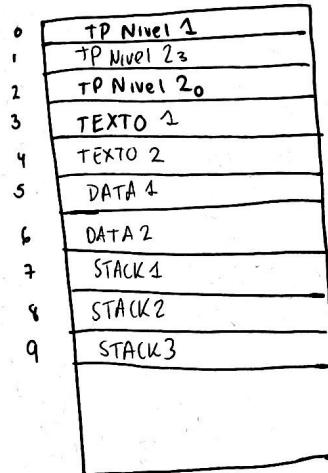


Texto = 1 pág

Datos = 3pôos

Pila = 2 págs

Marcos de Página



$$16 / 4 = 4 = \frac{\text{Contenido (1ª Página)}}{\text{Recto T.P}}$$

offset dentro de la
página

(23)

	kernel									
0	/	/	/	/	/	/	/	/	/	/
1	Pago ₀ P ₁	Pago ₀ P ₂	Pago ₀ P ₃	Pago ₀ P ₄	Pago ₀ P ₅	Pago ₃ P ₁	Pago ₃ P ₂	Pago ₃ P ₃	Pago ₃ P ₄	Pago ₀ P ₁
2	Pago ₀ P ₂	Pago ₀ P ₃	Pago ₀ P ₄	Pago ₀ P ₅	Pago ₀ P ₀	Pago ₁ P ₂	Pago ₁ P ₃	Pago ₁ P ₄	Pago ₂ P ₂	Pago ₂ P ₃
3	*			*	*	Pago ₁ P ₀	Pago ₁ P ₂	Pago ₁ P ₃	Pago ₂ P ₂	Pago ₂ P ₃
	*	*	*		*	*	*	*	*	*

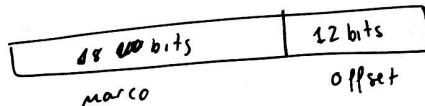
(24)

$$D.L = 4GB = 2^{30+32} B \quad 32 \text{ bits d.L}$$

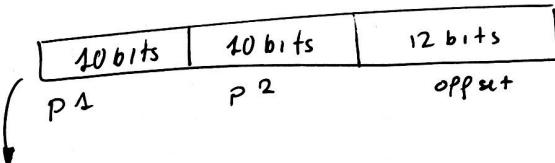
$$T.P = 4KB = 2^{12} B$$

$$D.F = 2^{30} B \quad 30 \text{ bits} = 12 \text{ offset} + 18 \text{ bits marco}$$

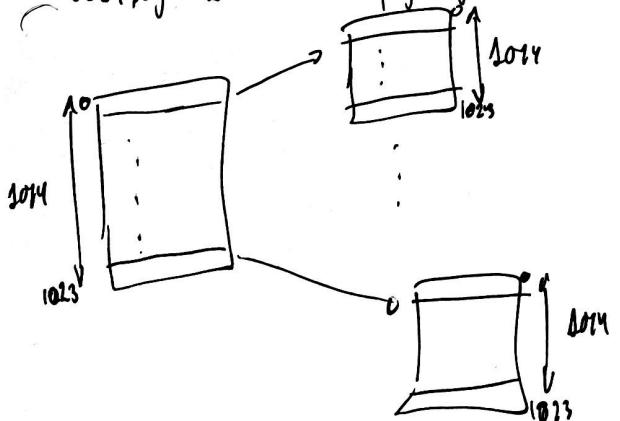
DF tiene 30 bits



D.L 32 bits



$$1024 \text{ pag} = 2^{10} \rightarrow 10 \text{ bits pag 1 y } 10 \text{ bits pag 2.}$$



21. Acceso a pagina $\rightarrow b + M = 1$

limpiar M

(22) $RAM = 4KB = 2^4 \cdot 1024 = 2^{12} B$

Paginación

Paginas = $1KB/\text{pagina} = 2^{10} B$

D.V = 16 bits

Marco 0 \Rightarrow kernel

A) Marcos pagina $\Rightarrow 4KB / 1KB = 4$ marcos

B) $2^2 = \boxed{2 \text{ bits}}$ para identificar marcos

C) LRU

Paginas usadas		
P1	0-99	0
P2	0-500	0
P1	100-500	0
P2	501-1500	0, 1
P1	3500-3700	3
P2	1501-2100	1, 2
P1	561-600	0

1º queremos cargar o
pero no están

ALGORITMO

+ actual - + anterior falta

$1 - 0 = 1 \leq 3 = T \rightarrow$ Mantene págs
referenciadas

2º $2 - 1 = 1 \leq 3 \rightarrow$ Mantenemos

3º $3 - 2 = 1 \leq 3$

4º. $\frac{3}{7} - \frac{4}{4} = \frac{3}{3} > 3 \rightarrow$ Desreferenciamos todas las páginas no referenciadas
entre fo. Ha actual y anterior

5º $\rightarrow 9 - 7 = 2 \leq 3 \rightarrow$ Mantenemos

6º $\rightarrow 12 - 9 = 3 > 3 \rightarrow$ Desreferenciamos NO DESDIFERENCIADOS

(19) 1 4 2 2 2 4 5 5 3 3 5 1 1 1 1 4

1	1	1	1	1	1	1	3	3	3	-	-	-	-	-
2	2	2	2	2	2	2	2	2	2	-	-	-	-	-
	4	4	4	4	4	4	4	4	4	-	-	-	-	-
	5	5	5	5	5	5	-	-	-	-	-	-	-	-

$T = 3$

$t=0 \quad \times \quad \times$

$7 - 2 = 5 > 3$

$12 - 9 = 3 = 3$

Tenemos ahora todos

los marcos ocupados

porque se produciría un
bloqueo en el proceso hasta
que no tuviéramos marcos de
páginas libres

(16) 7 páginas \rightarrow 5 marcos

LRU	Referencias	2	1	3	4	1	5	6	4	5	7	4	2
Marcos de página	2	2	2	2	2	2	6	6	6	6	6	6	
	1	1	1	1	1	1	1	1	1	1	1	2	
	3	3	3	3	3	5	5	5	3	3	3	3	
	4	4	4	4	4	4	4	4	4	4	4	4	
	5	5	5	5	5	5	5	
	*	*	*	*	*	*	*	*	*	*	*	*	

8 faltas
de página

(18) FFP

Int tiempo entre 2 pag consecutivas

$>$ tiempo > 4 páginas no referenciadas se retrasan MP

$$f_t = \text{inst } t^{\circ} \text{ actual falta}$$

$$Z = \text{páginas referenciadas en } t$$

$$f_{t-1} = \text{inst } t^{\circ} \text{ anterior falta}$$

$$R = \text{conjunto de páginas residentes en MP}$$

5 páginas (0, 1-4)

Inicialmente cargada pag 2 no hay restricciones en cuanto a MP.

$$T = 3 \quad 0 \ 3 \ 1$$

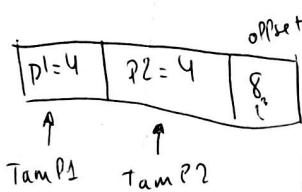
Referencias	0	3	1	1	1	3	4	4	2	2	4	0	0	0	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

1^{er} * \rightarrow que.

Examen.

Paginación:

$$MP = 8000 \quad 8192 \quad B = 8KB = 2^{13}$$



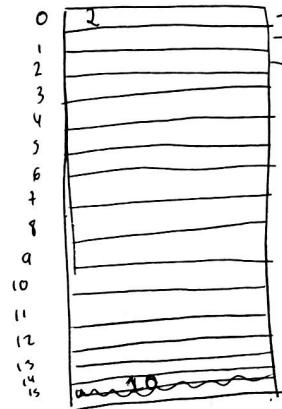
Códigos 0-255

Datos: 256-675

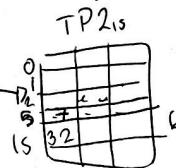
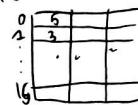
D.la: 256B y base dir: 62976
 $2^8 = 256B \Rightarrow 8192/256 = 32$

TP1 / Página

TP1.

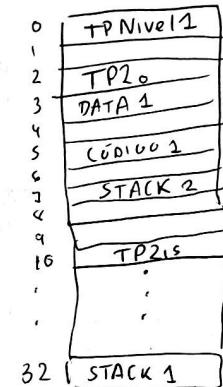


$$TP = 2^8 = 256B \rightarrow P_0$$

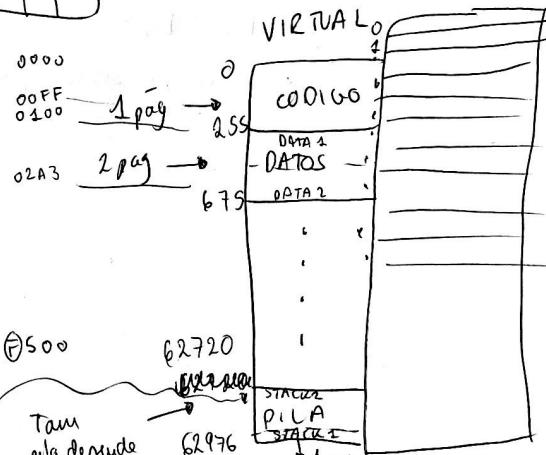


Tenemos que asegurarnos
que pude traducir todo
número virtual (paginación)

Marcor de Páginas



MP



Si Tam Palabra $\neq 1B$ p-ej

Tam Palabra = 2B

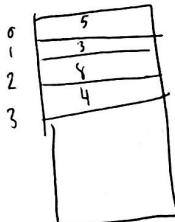
$$\frac{256B}{2B} = 128 \text{ dir}$$

$$\text{y pila } 62976 - 128 = 62848$$

62720

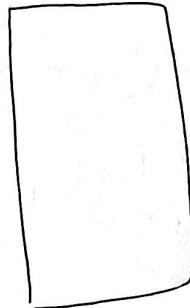
Tam pila depende de tam palabra (Suponemos Tam palabra = 1B)

TP2.



255

virtual



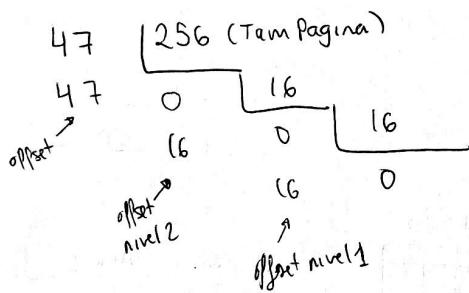
Hay que meter
el del medio (bytes)
porque os contigo
y debes ser
capaz de direccionar
todo lo de en medio

03 03 931

TDA MP

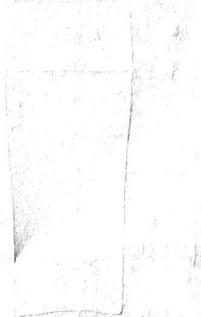
Nivel 1 Nivel 2

$$47 = 602F_{16}$$



$$6523 \rightarrow 197B_{16} \rightarrow \text{SIGFAULT}$$

② Suponiendo que se reservan 512B que se hacen



conjunto residente

(residen las partes del programa necesarias en un momento)

memoria auxiliar (reside el espacio de direcciones completo del

implementación son:

información sobre qué partes del programa se encuentran en MP y en MA, en la memoria en disco, TUD.

solución de un acceso a memoria situado en una parte que no reside en MP. movimiento de partes del espacio de direcciones entre MP y MA.

función de gestión de memoria ~~responsible accesos a memoria por parte del procesador~~

MMU, un dispositivo gestionado por el SO, que traduce direcciones virtuales a direcciones físicas. Se encarga de sumar el valor del registro base a cada dirección generada por la ejecución del programa, siendo éste resultado el que se usa para acceder en el bus de memoria a la dirección física deseada.

MMU, tiene unos registros TLB (Translation Look-aside Buffer) que mantienen la correspondencia entre página virtual/física resueltas con anterioridad.

Las responsabilidades del MMU son:

- Si TLB hit, realizar la traducción.
- Si TLB miss, usar la Tabla de Páginas para traducir, teniendo en cuenta que si la parte del espacio de direcciones que contiene la dirección resultado de la traducción reside en MP, carga el nuevo valor, si no genera una page fault exception.

B.3. Memoria virtual paginada

La organización del espacio no es continua, se asigna la memoria mediante bloques de tamaño fijo (**marcos de página**), cuyo tamaño va de 0.5 a 8 KB.

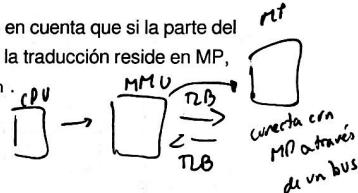
Las direcciones se interpretan a dos niveles:

- Los bits más significativos = página virtual donde está la dirección.
- Bits menos significativos = offset en marco de página.

La correspondencia página virtual y marco de página se almacena en la Tabla de páginas, donde se introduce la dirección física del marco de página.

- **Dirección virtual:** generada por la CPU, contiene un par:

- El número de página, que determina la entrada en la TP (bits + significativos).
- Offset, completa la dirección física (bits menos significativos).



- Dirección física: es la real en memoria principal, se calcula con la dirección de página (almacenada en TP) y el offset (se suma).

Estructuras de MVP:

Se utilizan tres estructuras principales:

- Tabla de páginas, mantiene información necesaria para realizar dicha traducción.
- La Tabla de Ubicación en Disco, mantiene la ubicación de cada página en el almacenamiento auxiliar.
- La Tabla de Marcos de Página, mantiene información relativa a cada marco de página en el que se divide la memoria principal.

Los vamos viendo por orden:

Tabla de páginas:

Contiene una entrada por cada página virtual del proceso que a su vez contiene:

- Dirección base del marco.
- Protección o modo de acceso a la página.
- Bit de validez.
- Bit de modificación.

Nº de "página virtual"	Dirección Base de Marco de Página	Validez/Presencia	Protección	Modificación
	0x00ABC000	1	r-w	0

Tabla de Ubicación en Disco:

Contiene una entrada por cada página virtual con:

- Identificación del dispositivo lógico que tiene la MA.
 - Identificación del bloque que contiene la copia de la página.
- Además, crece hacia abajo.

Memoria virtual mediante paginación por demanda

El modelo de localidad, donde una localidad es un conjunto de páginas que se

en el tiempo, que durante la ejecución de un programa va variando.

Características de esta estructura son:

• Las páginas residen en un dispositivo de intercambio, o *backing store*.

• Durante el proceso, el SO sólo carga en memoria un subconjunto de páginas.

La tabla de páginas se inicializa con los valores correctos, páginas válidas y cargadas en RAM; páginas válidas pero no cargadas y páginas no válidas.

- Despues de inicializar, se cambia el proceso a LISTO.

Errores posibles: **PAGE FAULT**

Gestión del error:

1. Bloquear Proceso

1. Encontrar la ubicación en disco de la página solicitada mirando la entrada de la TUD.

2. Encontrar un marco libre. Si no hubiera, se puede optar por reemplazar una página de memoria RAM.

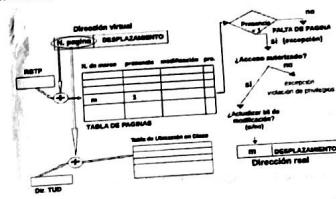
3. Cargar la página desde disco al marco libre de memoria RAM → proceso "BLOQUEADO"

4. FIN E/S (RSI) →

4.1. Actualizar TP(bit presencia=1, nº marco,...)

4.2. Desbloquear proceso → proceso "LISTOS"

5. Planif_CPU() selecciona proceso → Reiniciar la instrucción que originó la falta de página.



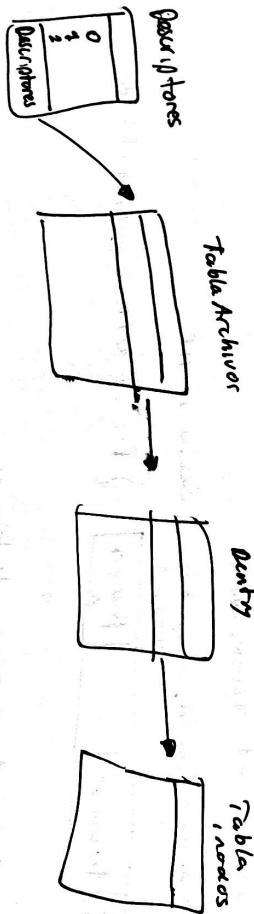
En Linux se distingue entre errores de programación relativos a acceso a memoria y errores debidos a falta de página.

Implementación de la Tabla de Páginas

- La TP se mantiene en memoria principal (kernel).
- El registro base de la tabla de páginas (RBTP) apunta a la TP y forma parte del contexto de registros (PCB).

Monteiro

Mount - mount
Umount -> desconecta



a) FFP 6 páginas (0, 1, 2, 3, 4, 5)

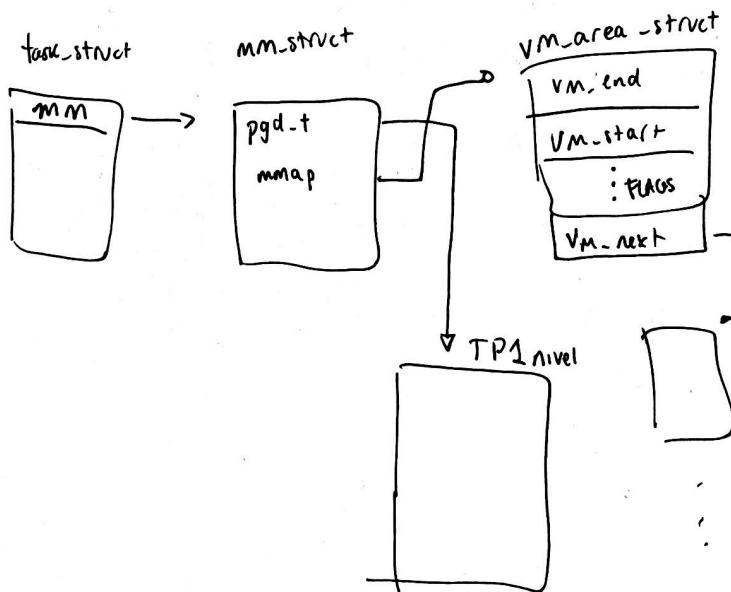
$t=0$ cargada pag = 2

	0	3	2	2	2	3	5	5	1	1	1	2	2	3
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
1														
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4														
5									5	5	5	5	5	5
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	*	*				*	*	*						

$T=3$

$7 - 2 = 5 > 3 \Rightarrow$ Desreferenciamos páginas no referenciadas en $t=3$

$2 < 3$



(25)

Grupo 2: Arturo Molina,

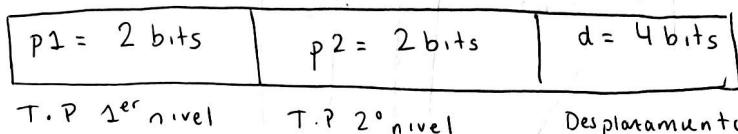
Carlos Pérez, David

Sánchez Pérez, Alberto

Llamas González.

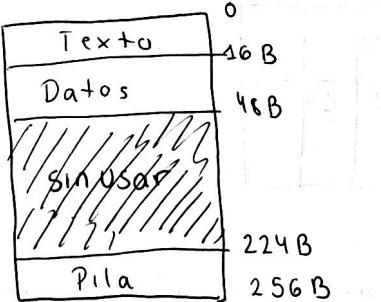
- Páginaación a dos niveles.

Direcciones de 8 bits con la siguiente estructura:



Por lo que podemos saber que la tabla de páginas de 1º nivel tiene 4 entradas ($2^{p_1=2}$) y cada entrada direcciona en la tabla de páginas de 2º nivel ($2^{p_2=2}$) 4 entradas

- Espacio virtual



Del dibujo de la estructura del espacio de direccionamiento virtual de un proceso vemos que cada página ocupa 16 B luego si el total de Bytes del espacio virtual es 256 B tenemos:

$$256 \text{ B} \cdot \frac{1 \text{ página}}{16 \text{ B}} = 16 \text{ páginas}$$

- Ocupa mucha memoria y hay que tener ^{talla} info paginas en MP
- Se requieren 2 accesos
 - tabla páginas
 - posición de memoria } se resuelven con TLB

• Problema adicional → tamaño de la tabla de páginas



Ejemplo

Dirección virtual = 32 bits

Tam pag = 4K bytes (2^{12} bytes)

• tam del campo desplazamiento = 12 bits

• tam nº de página virtual = 20 bits (32 - 12)

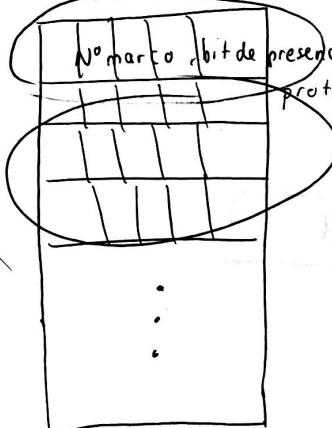
• Nº páginas virtuales = $2^{20} = 1,048,576$ (más de un millón de páginas virtuales)

SOLUCIÓN ⇒ PAGINACIÓN MULTINIVEL



• Paginar las tablas de páginas

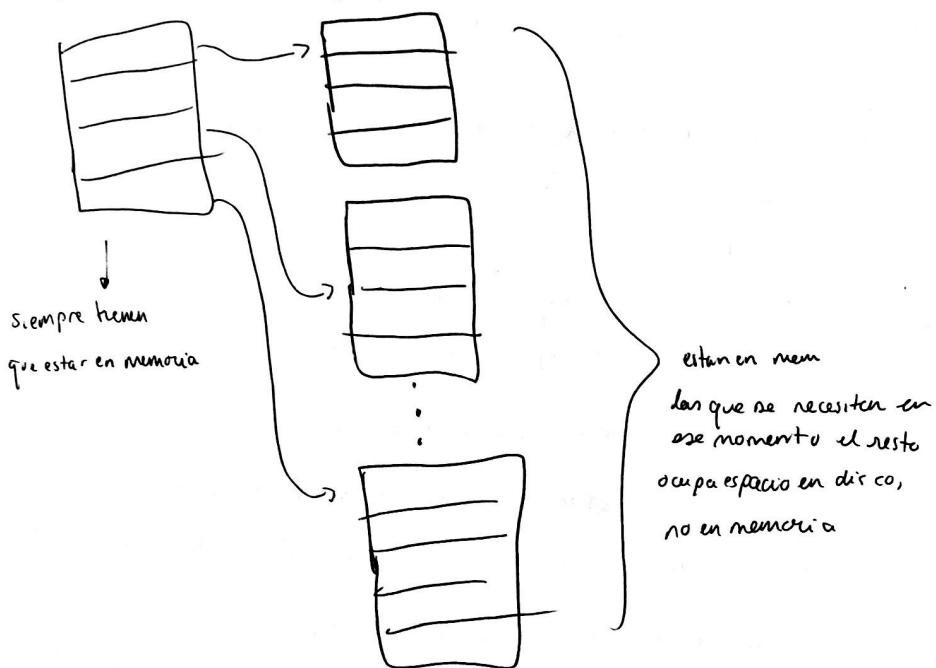
Tabla de páginas · proceso P1



Divide en trozos del mismo tamaño
y se convierten en tablas de páginas de
segundo nivel

Neusito estructura que indexe tablas de páginas de segundo nivel
de son las T.P de primer nivel

T.P 1º nivel \Rightarrow info para acceder un MP a tP de segundo nivel

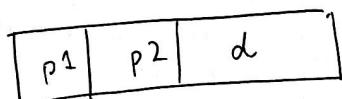


PROBLEMA : muy lento \Rightarrow se resuelve con la TLB

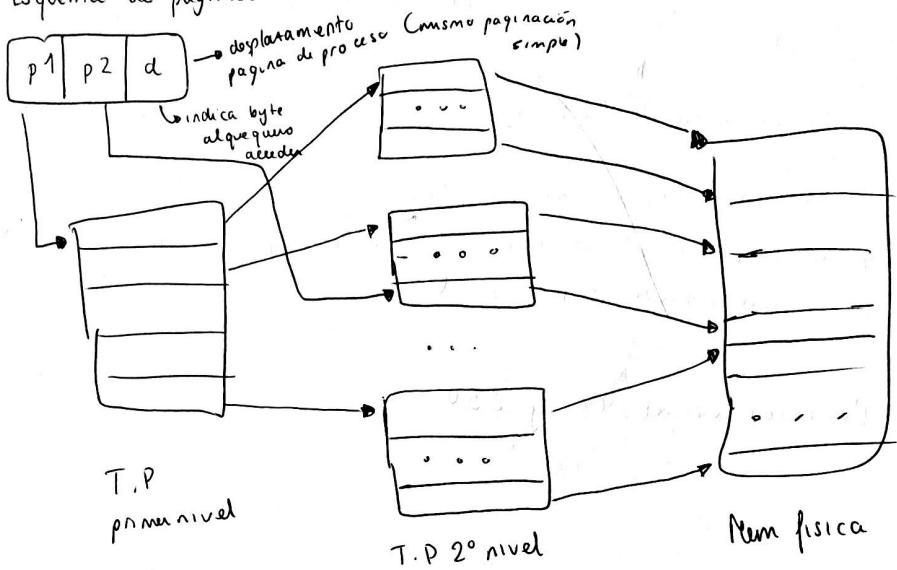
Número página

$$p_1 = k \rightarrow \text{nº página}$$

$$p_2 = n - k \rightarrow \text{desplazamiento de página}$$



Esquema de paginación a dos niveles



Dirección virtual = 20 bits

desplazamiento = d = 10 bits Tam pag = 2^{10} bytes = 1KB

$$p_1 = 3 \text{ bits} \quad p_2 = 7 \text{ bits} \quad (d = p_1 + p_2) \quad 3 + 7 = 10 \Rightarrow 2^{10}$$

T.P primer nivel ($2^{3 \rightarrow p_1^1} = 8$ entradas) paginar

Cada entrada direcciona

$$\text{T.P } 2^{\circ} \text{ nivel} = 128 \text{ entradas } (2^7)$$

Direcciona 128 páginas de 1KB



$$d.l = \frac{129 \text{ KB}}{128 \text{ KB}} = p1 = 1 \text{ KB} \text{ entonces la entrada 1 que es}$$

(dirección lógica)

dónde quiero escribir

p2 en resto de división dividido por tam pag = 1KB

Cada entrada de T.P de 2º nivel corresponde a una página
que estará cargada en un marco de página y T.P entrada tiene
dirección en trama T.P 2º nivel

Dirección virtual = 47 y 230