Espacios de direcciones virtuales

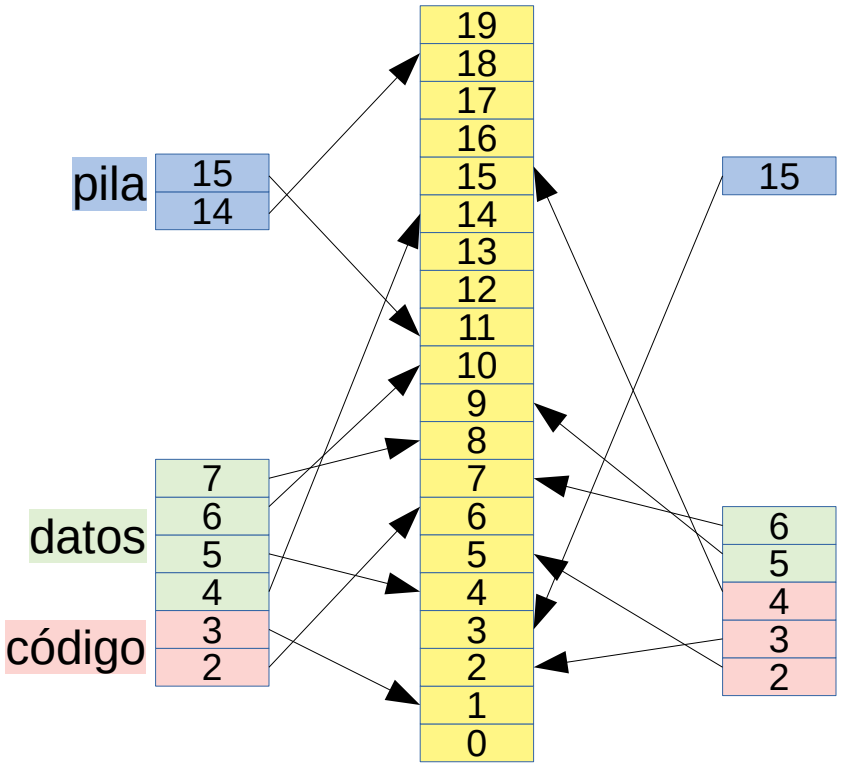
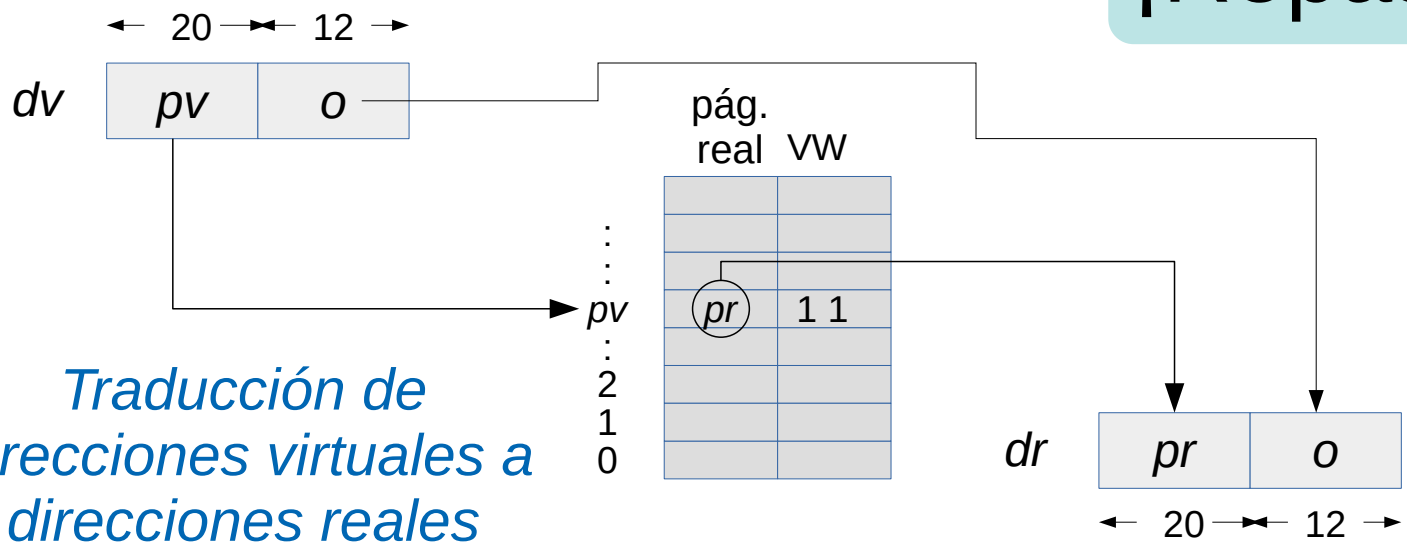


Tabla de páginas de P

V: validez
W: escritura

	pág. real	V	W	Atributos
15	11	1	1	
14	18	1	1	
13		0		
12		0		
11		0		
10		0		
9		0		
8		0		
7	8	1	1	
6	10	1	1	
5	4	1	1	
4	14	1	1	
3	1	1	0	
2	6	1	0	
1		0		
0		0		

¡Repaso!



Traducción de direcciones virtuales a direcciones reales

El potencial del paginamiento - I a III

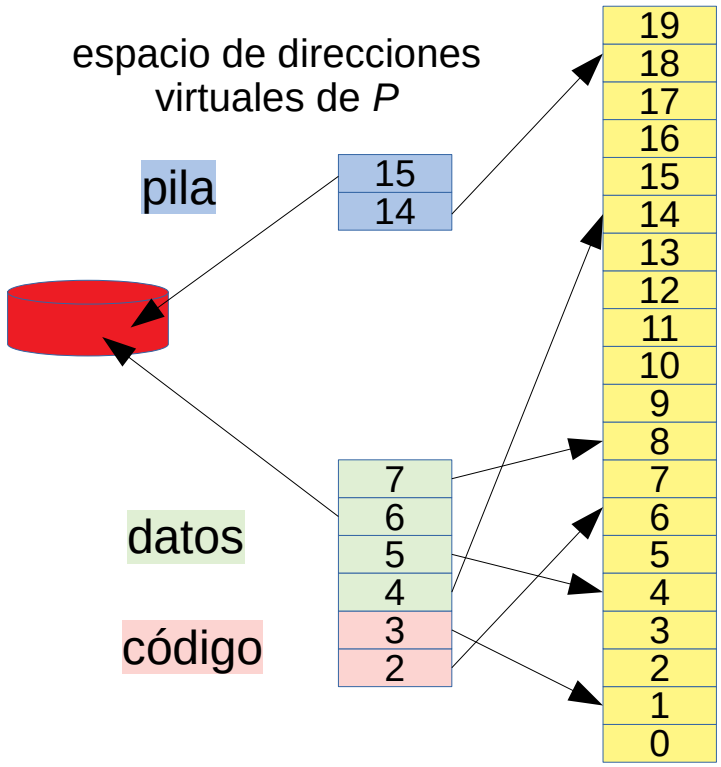
- Espacios de direcciones virtuales
- **Protección entre procesos**
- Extensión explícita de los datos cuando se agota el *heap de malloc*
- Extensión automática de la pila en caso de desborde
- Implementación de fork
  - ineficiente: duplicando espacios de direcciones
  - eficiente: compartiendo páginas pero con *copy-on-write*

El potencial del paginamiento: Swapping

- Cuando la memoria escasea se llevan procesos completos a disco
- Se graba en disco un copia al byte de las páginas de un proceso
- Se cambia el estado del proceso a *swapped*
- Las páginas liberadas se utilizan para los procesos que quedan en memoria
- No se puede ejecutar un proceso mientras está en disco
- Es ingrato para el usuario que está detrás de un proceso interactivo porque podría no haber respuesta hasta por minutos

El potencial del paginamiento: Paginamiento en demanda

- Se llevan a disco las páginas no usadas recientemente por algún proceso
- Se marcan como inválidas pero con un atributo adicional *S* que indica que están grabadas en disco
- El proceso propietario puede continuar ejecutándose
- Si el proceso accede a una página en disco, se gatilla un *page fault* y el núcleo carga transparentemente la página en memoria nuevamente
- Esto se denomina reemplazo de página porque habrá que elegir un página residente en memoria para llevarla a disco



	pág. real	V W S
15	-	0 1 1
14	18	1 1
13		0
12		0
11		0
10		0
9		0
8		0
7	8	1 1
6	-	0 1 1
5	4	1 1
4	14	1 1
3	1	1 0
2	6	1 0
1		0
0		0

Reemplazo de páginas

- Considere un proceso *P* que accede a una página *v* residente en disco, se necesita traerla a memoria
- Hay que encontrar una página real *r* en donde colocar la página virtual *v* de *P*
- Como la memoria escasea, hay que elegir una página *w* de un proceso *Q* residente en memoria
- Se lleva *w* a disco
- En la tabla de páginas de *Q*, en la posición *w*:
  - Se marca el atributo *V* en 0 para que cuando *Q* acceda a *w* gatille un page-fault
  - Se rescata el campo página real en *r*: es donde ser colocará *v*
  - También se coloca su atributo *S* (*saved*) en 1 porque hay que distinguir entre páginas grabadas en disco de páginas no atribuidas al proceso *Q* y que deben gatillar la señal *SIGSEGV* (*segmentation violation*)
- Se lee la página virtual *v* de *P* en la página real *r*
- En la tabla de páginas de *P*, en la pocisión de página *v*:
  - Se coloca el atributo de validez *V* de la página *v* en 1
  - Se coloca el campo página real en *r*
- Se invalida la TLB y el cache L1

Estrategias de reemplazo de páginas I

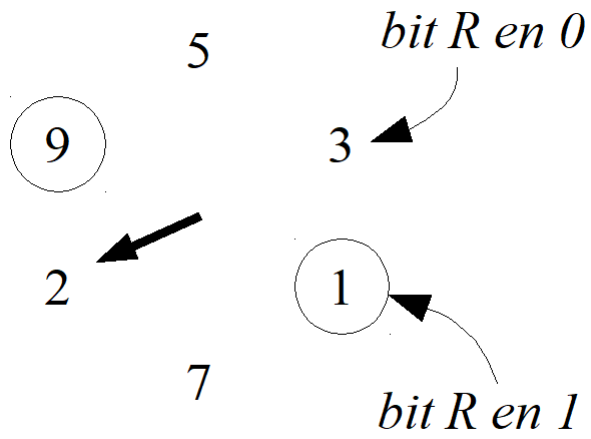
- El desempeño del paginamiento en demanda depende de una buena elección de la página *w* que se irá a disco
- El cómo se elige *w* se conoce como *estrategia de reemplazo de páginas*
- Estrategia ideal: se elige aquella página que será usada en el futuro más lejano
  - minimiza el número de page faults
  - no se puede implementar porque no se conoce el futuro
  - se usa solo como referencia para hacer comparaciones
- Estrategia FIFO: se elige la página que lleva más tiempo en memoria
  - fácil de implementar
  - es inutilizable debido al pésimo desempeño

Estrategias de reemplazo de páginas II

- Estrategia aleatoria: se elige una página al azar
  - pésimo desempeño en paginamiento en demanda
  - se usa en la estrategia de reemplazo de la TLB y de las memorias cache
- Estrategia LRU (least recently used): se elige la página que lleva más tiempo sin ser referenciada
  - se basa en que el pasado es un buen predictor del futuro
  - si una página lleva mucho tiempo sin ser referenciada, probablemente va a pasar otro tanto sin ser referenciada
  - excelente desempeño acercándose a la estrategia ideal
  - prácticamente imposible de implementar
- Estrategias que sí se usan: *estrategia del reloj* y *estrategia del working set*

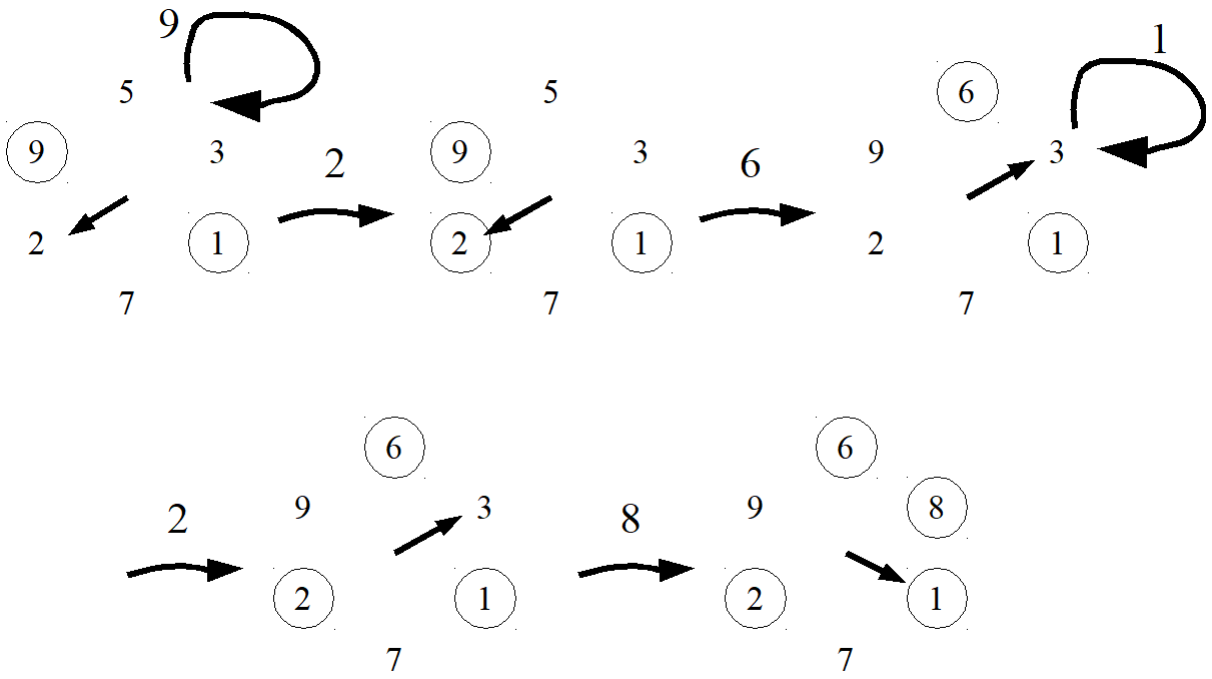
La estrategia del reloj

- Aproximación de LRU
- Escoge una página que lleve bastante tiempo sin ser referenciada
- Usa el bit de referencia R en la tabla de páginas
- La MMU coloca automáticamente el bit R en 1 cuando se referencia la página asociada
- Es mejor explicar la estrategia con un ejemplo
- La figura muestra un computador con 6 páginas reales
- Las páginas virtuales que tienen el bit R en 1 aparecen en un círculo, el resto tiene el bit R en 0
- Por simplicidad consideraremos un solo proceso
- El cursor (la flecha) avanza en el sentido de los punteros del reloj
- Una traza es una secuencia de páginas referenciadas por un proceso. Ejemplo: 9 2 6 1 2 8



La estrategia del reloj

- Ejemplo de traza: 9 2 6 1 2 8



Cuando ocurre un page fault:

```
while ( bitR( cursor( ) ) == 1 )  {
    setBitR( cursor( ) , 0 )
    avanzarCursor( )
}
reemplazarCursor( )
avanzarCursor( )
```

Estrategia en pseudo-código

Propiedades de la estrategia del reloj

- Caso múltiples procesos: se consideran todas las páginas por igual, sin distinguir a qué proceso pertenecen
- Tiempo que toma el cursor en dar una vuelta es el tiempo que se considera suficiente para reemplazar una página si no ha sido referenciada en toda la vuelta
- Depende de la cantidad de page faults, que depende a su vez de la localidad de los accesos
  - buena localidad → mayor tiempo entre vueltas
  - mala localidad → menor tiempo entre vueltas
- Situación anómala: todas las páginas fueron referenciadas
  - se da la vuelta completa
  - no es relevante porque se da cuando hay pocos page faults
- Ventajas:
  - simple de implementar
  - sobrecosto cero cuando la memoria es abundante

El problema de la estrategia del reloj

Thrashing

- Hay demasiados page faults
- Todos los procesos están en espera del reemplazo de una página
- Apenas se retoma un proceso, sufre un page fault y vuelve al estado de espera
- La CPU tiene un porcentaje de ocupación cercano a 0%
- El disco para paginamiento se ocupa al 100%
- La única solución es que el administrador mate algunos procesos pero el shell de comandos de *root* tampoco avanza
- Se produce cuando hay múltiples procesos, porque si bien cada proceso exhibe localidad de accesos, cuando el scheduler los ejecuta en tajadas de tiempo, pierden su localidad
- Solución: el *working set*