

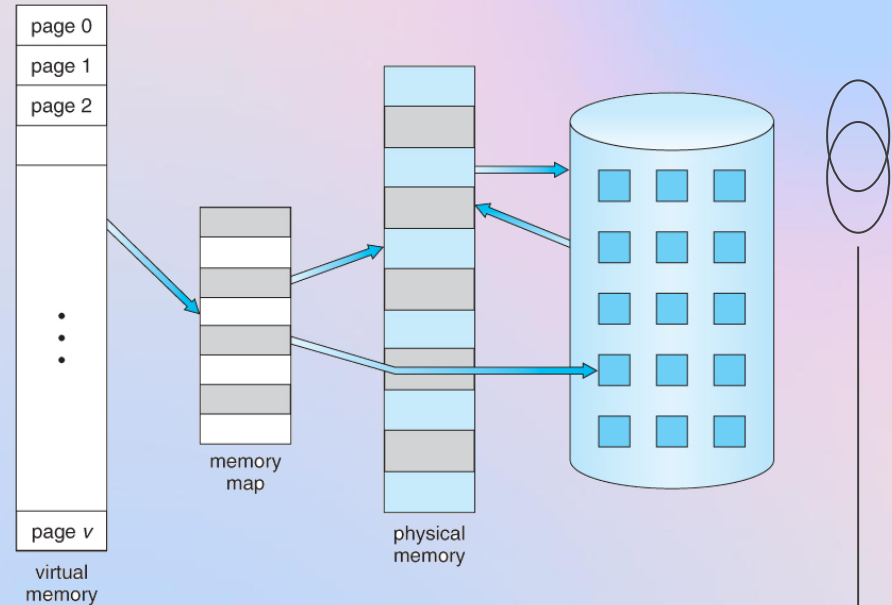


# Unidad III: Administración de memoria

## 3.3 Memoria Virtual

# Memoria virtual

Es una técnica utilizada por el administrador de memoria para generar espacios virtuales de memoria para cada proceso que se ejecuta en el sistema operativo. Esta permite extender la capacidad de ejecutar una mayor cantidad de procesos en el sistema operativo.




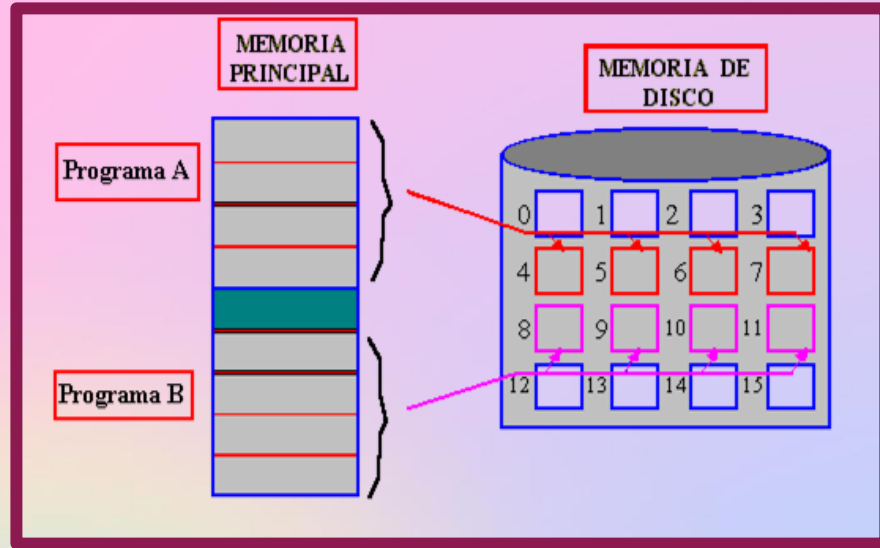


8.4

# Paginación

Esquema de gestión de  
memoria.






La paginación permite que el espacio de direcciones físicas de un proceso no sea contiguo.


- ❑ Evita el problema de encajar fragmentos de memoria en el almacén de respaldo.
- ❑ El soporte para la paginación se gestionaba mediante hardware.

# Método Básico



Descompone la memoria física en una serie de bloques de tamaño fijo, estos son denominados **marcos** y descomponer la memoria lógica en bloques del mismo tamaño que se denominan **página**.

Cuando se ejecuta un proceso son cargadas desde el almacén de respaldo que se encuentran en los marcos de memoria disponibles.



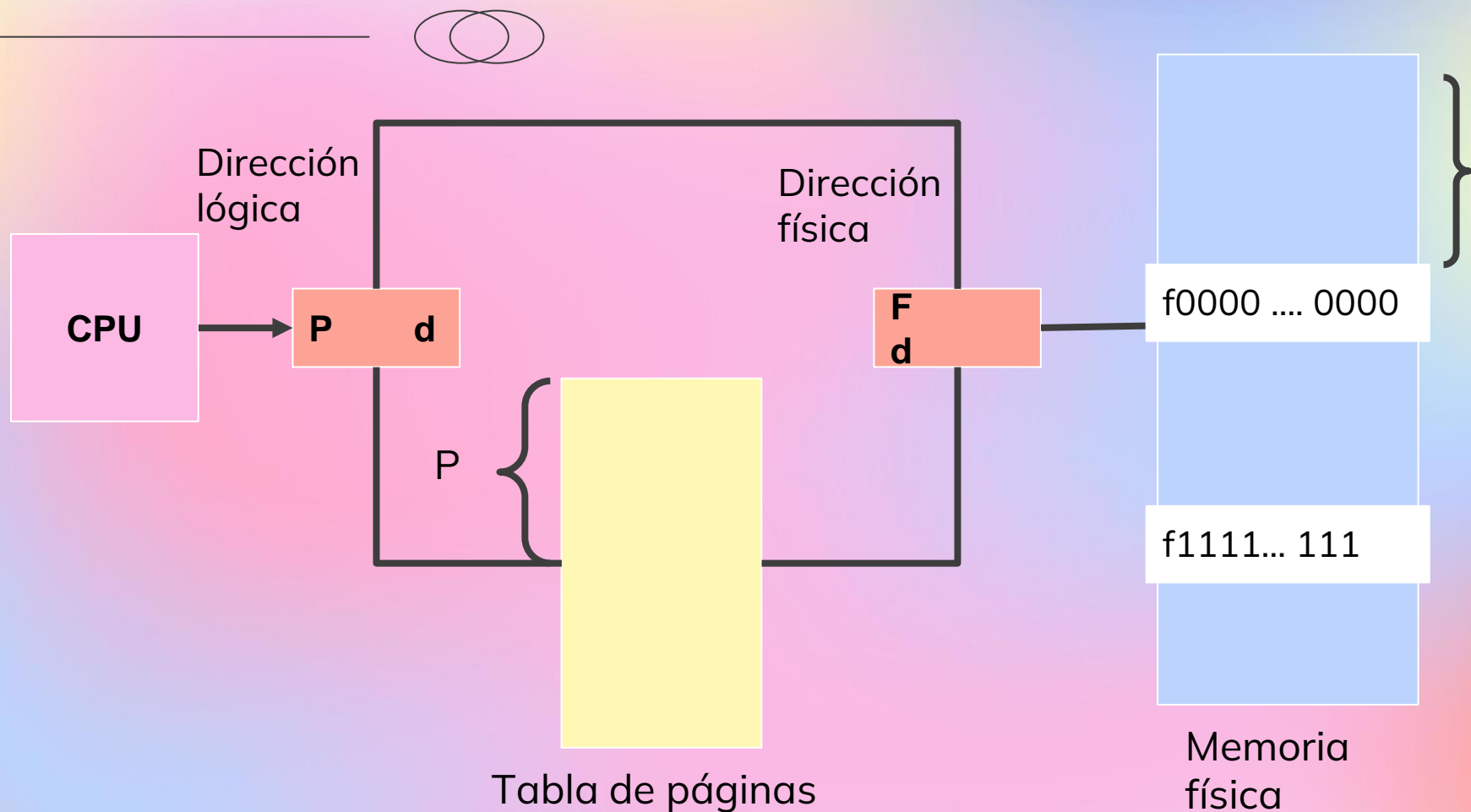


Figura 1. Hardware de paginación.

Toda dirección generada por la CPU se divide en dos partes:

- ❑ Número de páginas: Es utilizado como índice para una tabla de páginas.
- ❑ Desplazamiento de página: Definir la dirección de memoria física que se envía a la unidad de memoria.

El tamaño de página esta es definido por el hardware, este es normalmente de potencia 2 y varia de entre 512 bytes y 16 MB por página.

Si el tamaño del espacio de direcciones lógicas es de  $2^m$  y el tamaño de página es de  $2^n$  unidades de direccionamiento entonces los  $m-n$  bits de mayor peso de cada dirección lógica designara el número de páginas.

Por lo que los  $n$  bits de menor peso indicarán el desplazamiento de página, entonces tienen la siguiente estructura:



Un lector puede darse cuenta de que el esquema de paginación es una forma de reubicación dinámica.  
Por lo tanto cada dirección lógica es asignada por el hardware de paginación a alguna dirección física.





## Memoria lógica

0	a
1	b
2	c
3	d
4	e
5	f
6	g
7	h
8	i
9	j
10	k
11	l
12	m
13	n
14	o
15	p

0  
1  
2  
3

5
6
1
2

Tabla de páginas

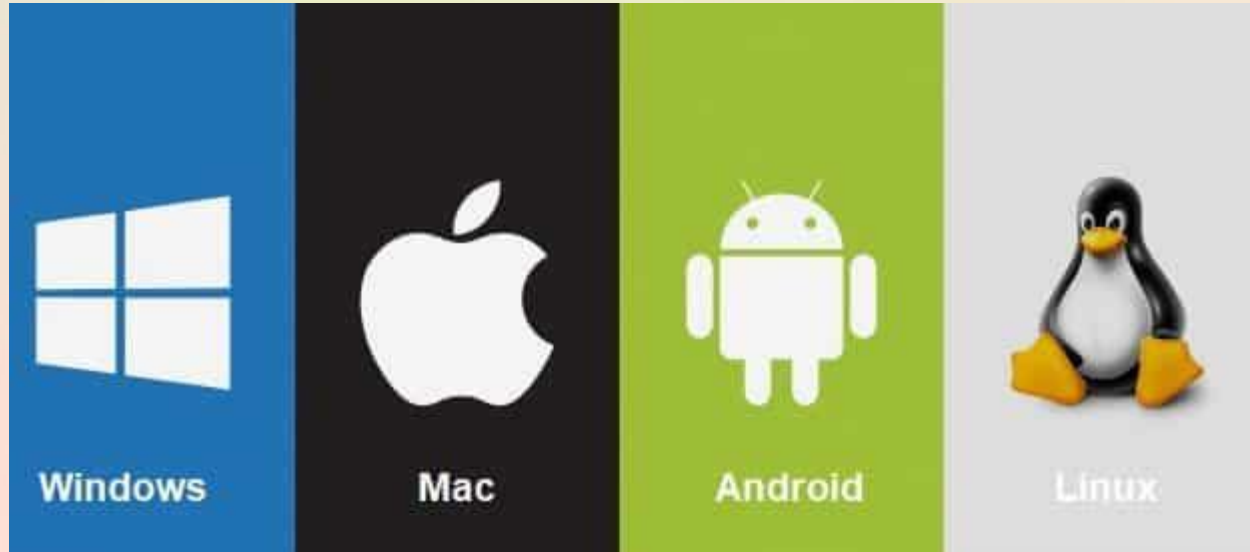
0	
4	I J K I
8	M N O P
12	
16	
20	A B C d
24	E F G h
28	

Memoria física

Figura 2. Ejemplo de paginación para una memoria de 32 bytes con páginas de 4 bytes.

El sistema operativo está gestionando la memoria física, por lo que se entiende que se sabe que marcos está disponibles, cuales han sido asignados, cual es el número total de marcos, etc.

La información está estructurada en una tabla de marcos, esta tabla tiene una entrada por cada marco físico que indica si esta libre o asignado, en caso de estar asignado indica a que pagina de procesos o procesos ha sido asignado.



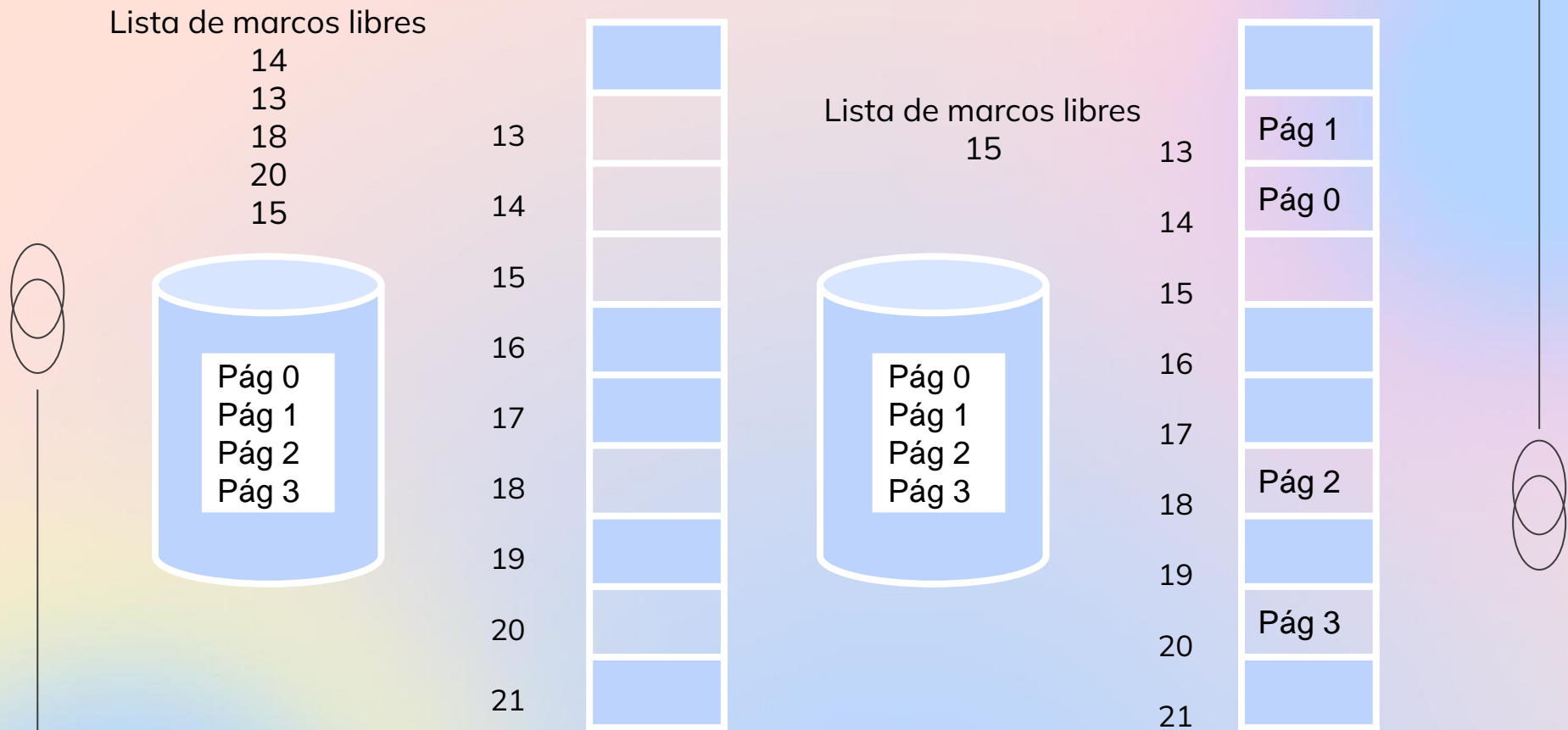




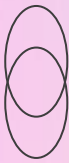
FIGURA 3. MARCOS LIBRES antes de LA ASIGNACIÓN y DESPUÉS DE LA ASIGNACIÓN



# SOPOrte HARDwAre

Cada sistema operativo tiene  
sus propios métodos para  
almacenar tablas de páginas.





. La mayoría de los sistemas operativos asignan una tabla de páginas para cada proceso, estos almacenan un puntero a cada tabla que hay de páginas en el bloque de procesos.

La implementación de hardware de la tabla de páginas puede realizarse de varias maneras, la tabla de páginas se implementa como un conjunto de registros dedicados.

Para cada acceso a la memoria debe pasar a través del mapa de paginación.

El uso de registros para la tabla de páginas resulta satisfactorio si la tabla es razonablemente pequeña.

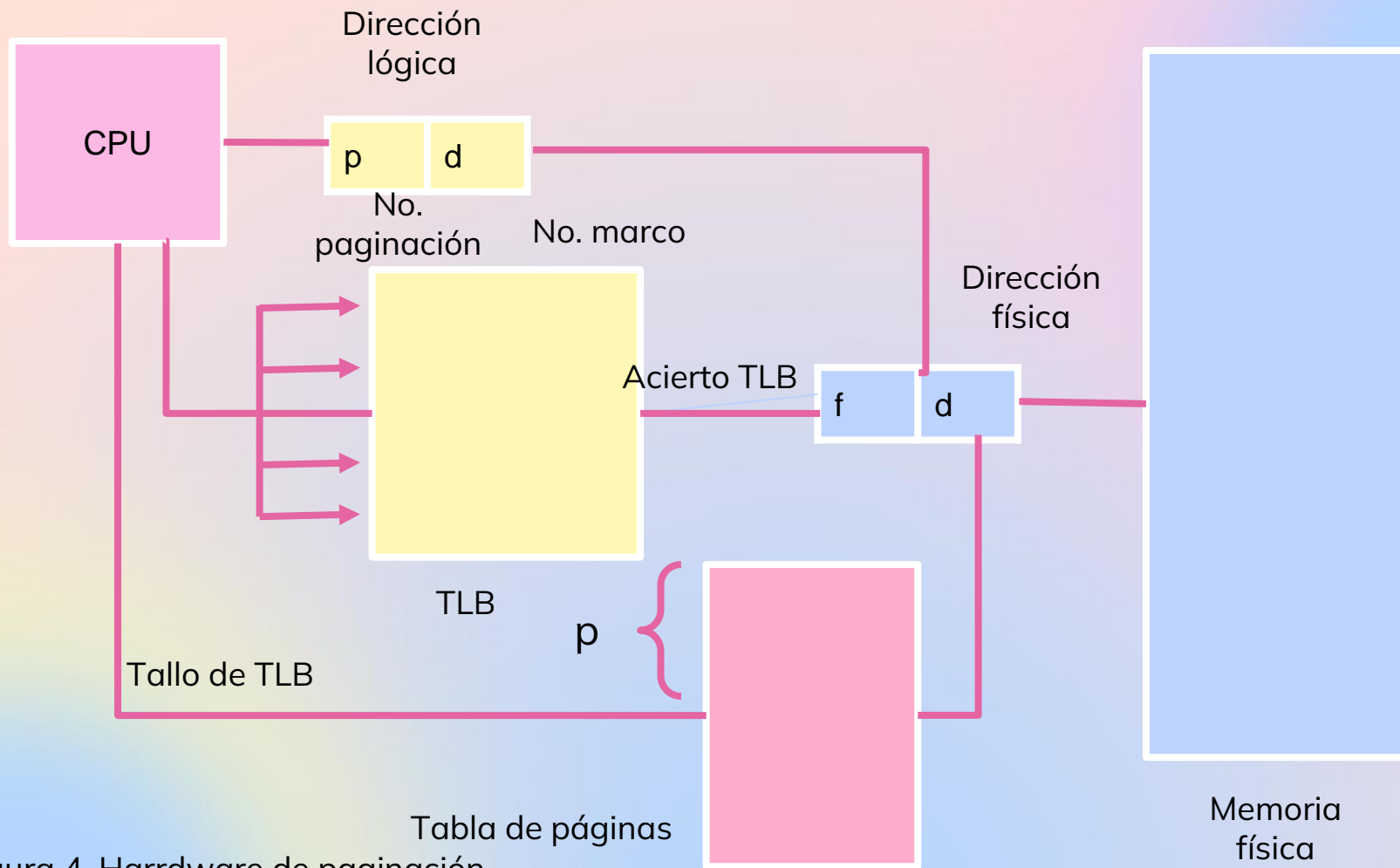



Figura 4. Harrdware de paginación



8.4.3

# Protección

Entorno paginado

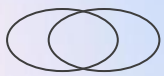


La protección de memoria se consigue mediante una serie de bits asociados con cada marco.

- Estos bits se mantienen en la tabla de páginas
- Uno de los bits puede definir una página como de lectura - escritura.
- Toda referencia a memoria pasa a través de la tabla de páginas.
- Al mismo tiempo que se calcula la dirección física, se pueden comprobar los bits de protección.
- Se asocia un bit adicional a cada entrada de la tabla de páginas.
- Cuando el bit es “válido” la página asociado se encontrara en el espacio de direcciones lógicas del proceso.







000000

Página 0
Página 1
Página 2
Página 3
Página 4
Página 5

10.468

12.287

No. marco

Bit válido inválido

0		
1		
2		
3		
4		
5		
6		
7		


Tabla de páginas

0  
1  
2  
3  
4  
5  
6  
7  
8  
9

Página 0
Página 1
Página 2
Página 3
Página 4
Página 5
Página n




Figura 5. Bit válido (v)-inválido (i) en una tabla de páginas



8.4.4

# Páginas compartidas

Posibilidad de compartir  
código común

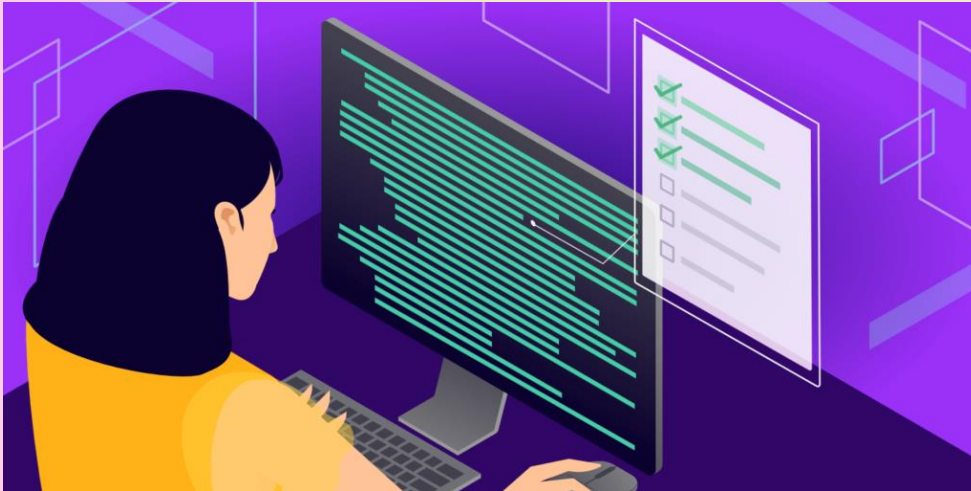


## Ventajas:

- Posibilidad de compartir código común.
  - Tiempo compartido.
  - Código reentrante.
  - Cada proceso tiene su propia página de datos.
  - El código reentrante no se auto-modifica
  - Cada proceso tiene su propia entrada
- almacenamiento de datos.



copias de los registros y



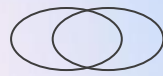
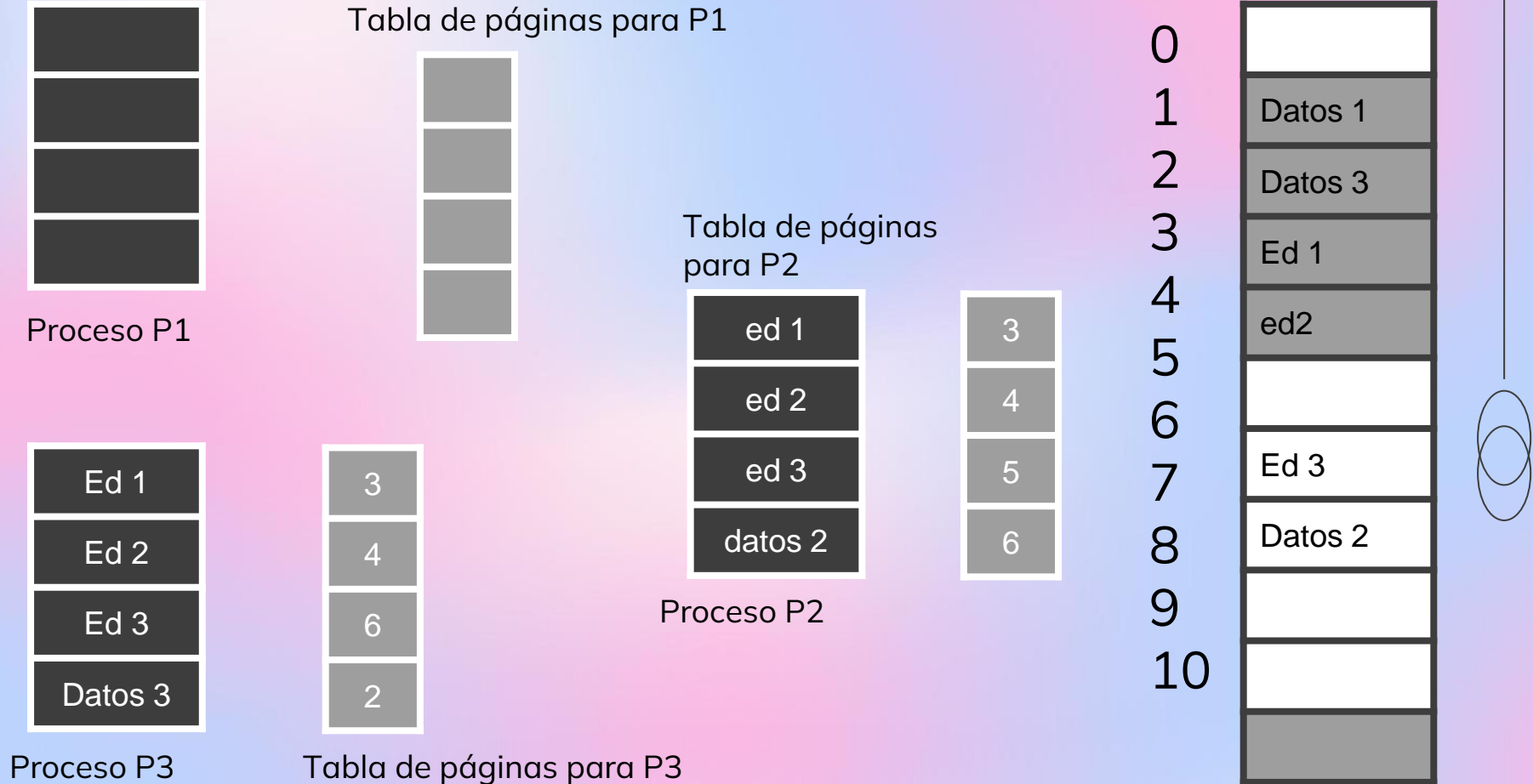


Figura 6. Compartición de código en un entorno paginado





8.5

# ESTRUCTURA DE LA TABLA DE PÁGINAS

Técnicas




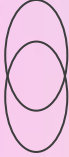


8.5.1

# PAGINACIÓN JERÁRQUICA


Gran espacio de direcciones  
lógicas





Número de página

Desplazamiento de página



10

10

12

Tabla de páginas con correspondencia (mapeo) directa.

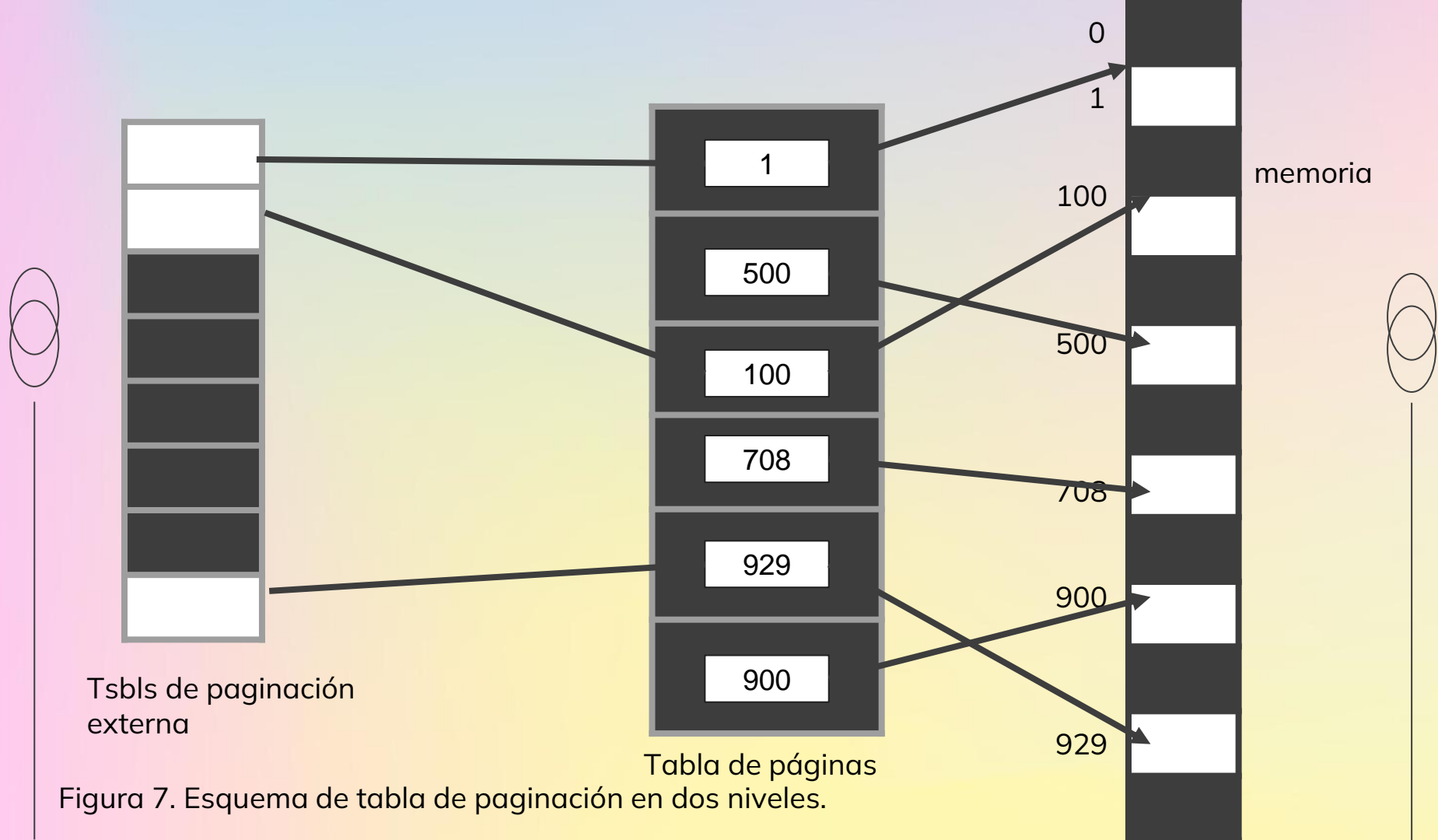


Figura 7. Esquema de tabla de paginación en dos niveles.




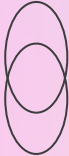



8.5.2

# TABLAS DE PÁGINAS HASH

Gestionar los espacios de  
direcciones superiores a 32  
bits



- 
- 
- El valor hash es el número de página virtual.
  - Cada entrada contiene una lista enlazada de elementos que tienen como valor hash una misma ubicación.
  - Cada elemento esta compuesto de tres campos:
    - Número de páginas virtual.
    - Valor del marco de paginación mapeado
    - Lista enlazada.

El algoritmo funciona de la siguiente forma:

El número de página virtual de la dirección virtual se le aplica una función hash para obtener un valor que se utiliza como índice para la tabla hash.

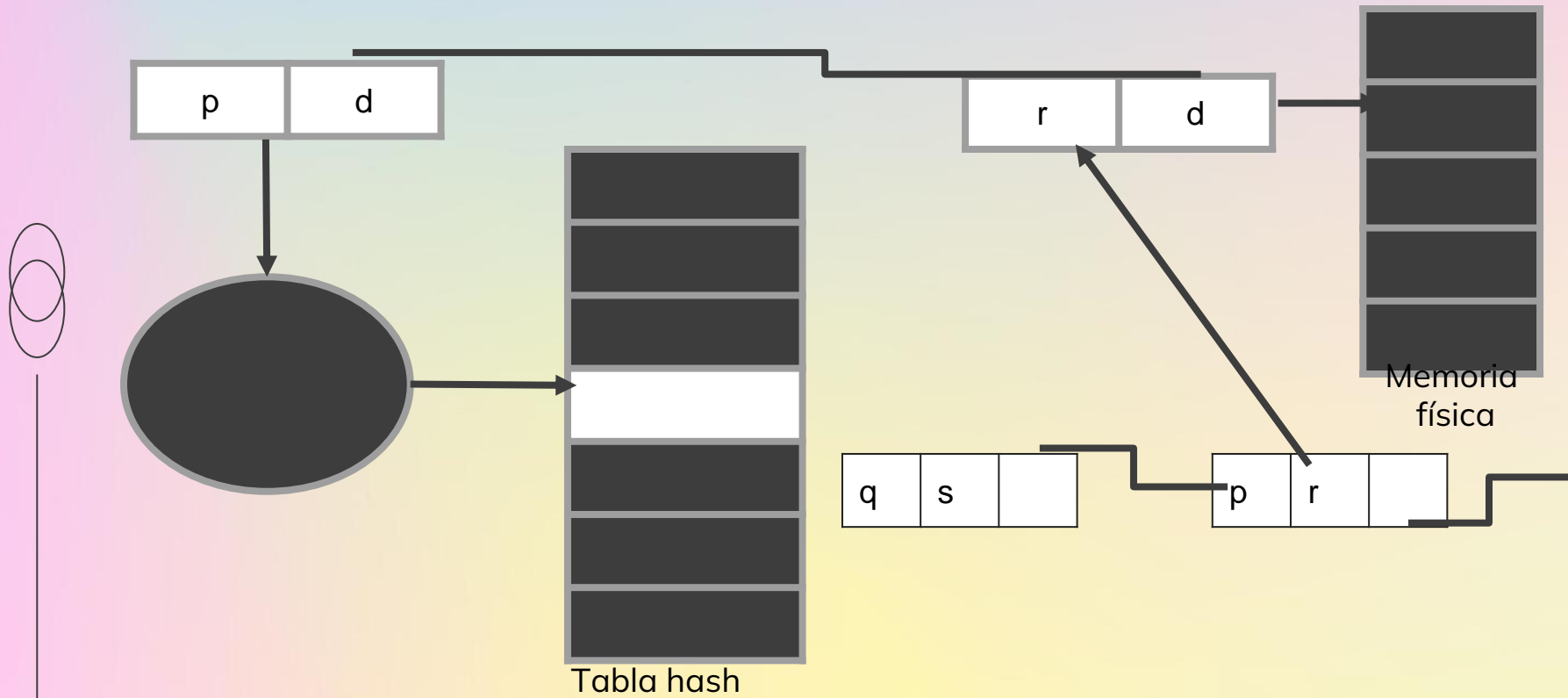



Figura 8. Tablas de páginas hash



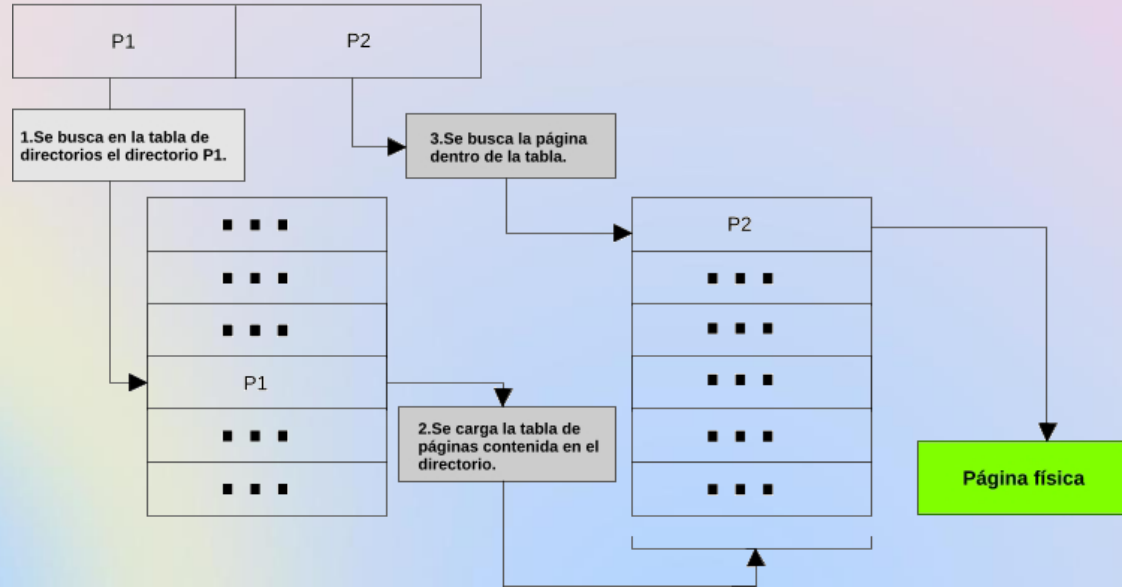
8.5.3

# TABLAS DE PÁGINAS INVERTIDAS

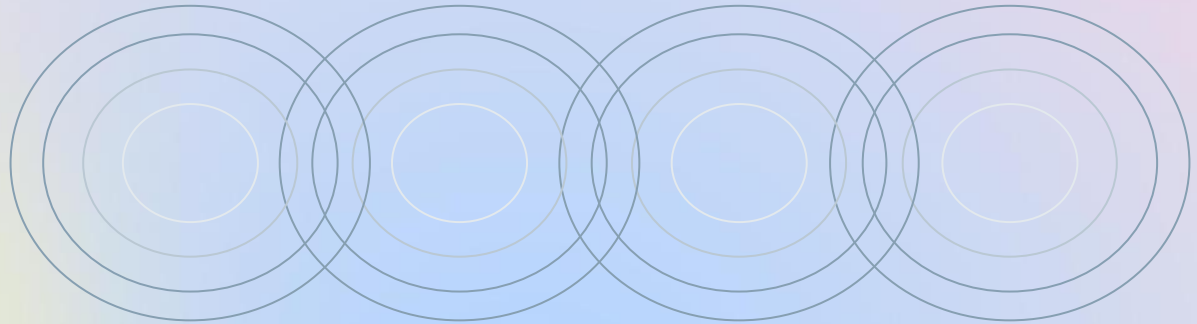
Cada proceso tiene una tabla  
de páginas asociada




Usualmente cada proceso tiene una tabla de paginación asociada. El sistema operativo debe entonces traducir cada referencia a una dirección física de memoria. Estas tablas pueden ocupar una gran cantidad de memoria física, para controlar el modo en que se están utilizando otras partes de memoria física.




Estas tablas pueden ocupar una gran cantidad de memoria física, para controlar el modo en que se están utilizando otras partes de la memoria física podemos utilizar las tablas de páginas invertidas





Cada entrada de la tabla de páginas invertidas es una pareja donde el identificador asume el papel de identificador del espacio de direcciones. Cuando se produce una referencia a memoria, se presenta al subsistema de memoria una parte de paginas invertida en busca de una correspondencia



Los sistemas que utilizan estas tablas tienen dificultades para implementar el concepto de memoria compartida. Se implementa usualmente en forma de múltiples direcciones virtuales, las cuales se hacen corresponder con una misma dirección física. Esto significa que las referencias a las direcciones virtuales que no estén asociadas darán como resultado un fallo de página









02

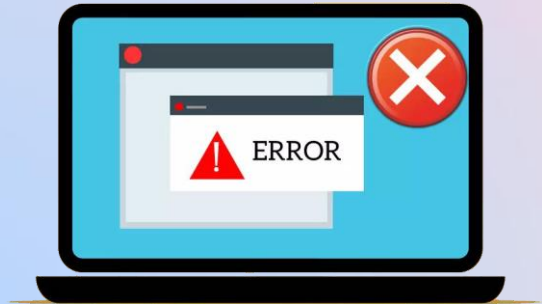
# 9.1 Fundamentos



El requisito de que las instrucciones deban encontrarse en la memoria física para poderlas ejecutar parece, a la vez, tanto necesario como razonable. De hecho, un examen de los programas reales nos muestra que muchos casos, no es necesario tener el programa completo para poderlo ejecutar, considerando lo siguiente

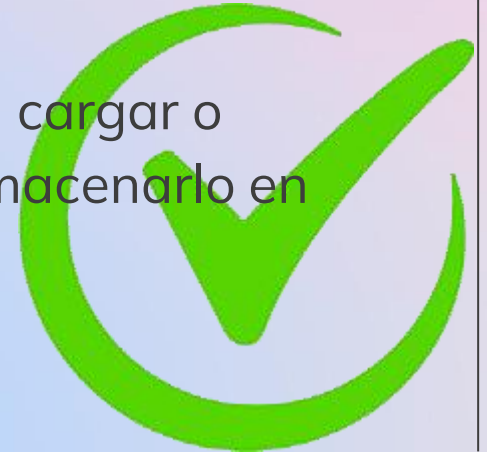


- Los programas incluyen a menudo código para tratar las condiciones de error
- A las matrices, a las listas y a las tablas se las suele asignar más memoria de la que realmente necesitan.
- Puede que ciertas opciones y características de un programa se utilizan raramente




La posibilidad de ejecutar un programa que solo se encontrará parcialmente en la memoria proporciona muchas ventajas:

- Los programas ya no estarían restringidos por la cantidad de memoria física disponible
- Puesto que cada programa de usuario podría ocupar menos memoria física, se podrían ejecutar mas programas al mismo tiempo
- Se necesitan menos operaciones de E/S para cargar o intercambiar cada programa con el fin de almacenarlo en memoria



- Ejecutar programas que no se encuentren completamente en memoria proporciona ventajas para el sistema y el usuario
- La memoria virtual incluye la separación de la memoria lógica, tal como la percibe el usuario, con respecto a la memoria física
- También facilita enormemente la tarea de programación






La memoria virtual incluye la separación de la memoria lógica, tal como la percibe el usuario con respecto a la memoria física. Esta separación permite proporcionar a los programadores una memoria virtual extremadamente grande

El espacio de direcciones virtuales de un proceso hace referencia a la forma lógica de almacenar un proceso en la memoria

Además de separar la memoria lógica de la memoria física, la memoria virtual también permite que dos o mas procesos compartan los archivos y la memoria mediante mecanismos de compartición de paginas





9.2

# PAGINACIÓN BAJO DEMANDA

Podría cargarse un programa ejecutable desde el disco a la memoria, sin embargo, esta técnica presenta el problema de que puede que no necesitemos inicialmente todo el programa en la memoria. Cargar el programa completo en la memoria hace que se cargue el código ejecutable de todas las opciones, independientemente de si el usuario selecciona o no una determinada opción

Loading...

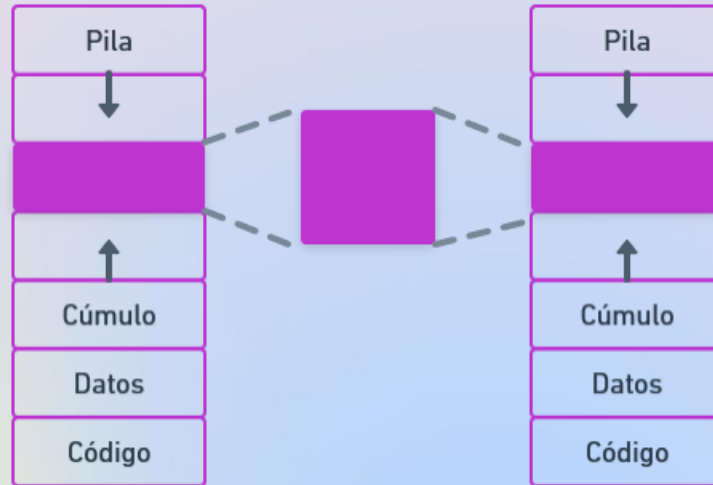




Con la paginación de bajo demanda, solo se cargan las paginas cuando así se solicita durante la ejecución del programa, de este modo, las páginas a las que nunca se acceda no llegan a cargarse en la memoria física




Con la paginación de bajo demanda, solo se cargan las paginas cuando así se solicita durante la ejecución del programa, de este modo, las páginas a las que nunca se acceda no llegan a cargarse en la memoria física






9.2.1



# Conceptos Básicos



Cuando hay que cargar un proceso, el paginador realiza una estimación de que páginas serán utilizadas antes de descargar de nuevo el proceso. En lugar de cargar el proceso completo, el paginador solo carga en la memoria las páginas necesarias

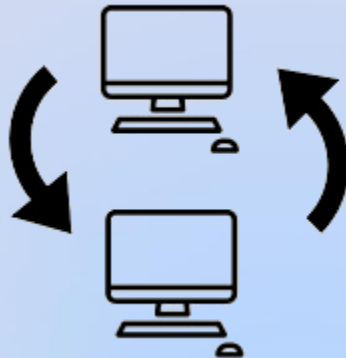



Con este esquema, necesitamos algún tipo de soporte hardware para distinguir entre las páginas que se encuentran en memoria y las páginas que residen en el disco. Sin embargo esta vez cuando se configura este bit como válido, la página asociada será legal y además se encontrará en memoria

- 
- 
1. Comprobamos una tabla interna correspondiente al proceso
  2. Si la referencia era invalida, terminamos el proceso. Si era válida, pero esa página todavía no ha sido cargada, la cargamos en la memoria
  3. Buscamos un marco libre
  4. Ordenamos una operación de disco para leer la pagina deseada en el marco recién asignado
  5. Una vez completada la lectura de disco, modificamos la tabla interna que se mantiene con los datos del proceso y la tabla de paginas
  6. Reiniciamos la instrucción que fue interrumpida


Memoria secundaria almacena aquellas páginas que no están presentes en la memoria principal


Es usualmente un disco de alta velocidad, se conoce como dispositivo de intercambio y la sección de la disco utilizada para este propósito se llama espacio de intercambio






Un requisito fundamental para la paginación bajo demanda es la necesidad de poder reiniciar cualquier instrucción después de un fallo de página, puesto que guardamos el estado del proceso interrumpido en el momento, salvo por que la pagina deseada se encontrara en memoria





Un requisito fundamental para la paginación bajo demanda es la necesidad de poder reiniciar cualquier instrucción después de un fallo de página, puesto que guardamos el estado del proceso interrumpido en el momento, salvo por que la pagina deseada se encontrara en memoria











9.2.2

# Rendimiento de La PAGINACIÓN BAJO DEMANDA


Afecta significativamente al rendimiento de un sistema informático. Para calcular el tiempo de acceso efectivo, debemos conocer cuanto tiempo se requiere para dar servicio a un fallo de página. Cada fallo de página hace que se produzca la siguiente secuencia




- 
1. Interrupción al sistema operativo
  2. Guardar los registros de usuario y el estado del proceso
  3. Determinar que la interrupción se debe a un fallo de pagina
  4. Comprobar que la referencia de pagina era legal y determinar la ubicación de la página en el disco
  5. Ordenar una lectura desde el disco para cargar la pagina en un marco libre
  6. Mientras estamos esperando, asignar la CPU a algún otro usuario
  7. Recibir una interrupción del subsistema de E/S de disco
  8. Guardar los registros y el estado del proceso para el otro usuario
- 

- 
- 
9. Determinar que la interrupción corresponde al disco
  10. Corregir la tabla de páginas y otras tablas para mostrar que la página deseada se encuentre ahora en memoria
  11. Esperar a que se vuelva a asignar la CPU a este proceso
  12. Restaurar los registros de usuario, el estado del proceso y la nueva tabla de páginas y reanudar la ejecución de la instrucción interrumpida

No todos estos pasos son necesarios en todos los casos



En cualquier caso, nos enfrentamos con tres componentes principales del tiempo de servicio de fallo de página


1. Servir la interrupción de fallo de página
  2. Leer la página
  3. Reiniciar el proceso
- 



9.4


# Sustitución de PAGINAGINAS





Si incrementamos maestro grado de multiprogramación, estaremos sobreasignando la memoria. Si ejecutamos seis procesos, cada uno de los cuales tiene diez páginas de tamaño pero utiliza en realidad únicamente cinco páginas, tendremos una tasa de utilización de la CPU y una tasa de procesamiento más altas, quedándonos diez marcos libres.

La sobreasignación de memoria se manifiesta de la forma siguiente. Imagine que, cuando se está ejecutando un proceso de usuario, se produce un fallo de página.






9.4.1


# Sustitución Básica DE PAGINAGINAS










La sustitución de páginas usa la siguiente técnica. Si no hay ningún marco libre, localizamos uno que no esté siendo actualmente utilizado y lo liberamos. Podemos liberar un marco escribiendo su contenido en el espacio de intercambio y modificando la tabla de páginas para indicar que esa página ya no se encuentra en memoria. Ahora podremos utilizar el marco liberado para almacenar la página que provocó el fallo de página en el proceso.






Modifiquemos la rutina de servicio del fallo de página para incluir este mecanismo de de páginas:

1. Hallar la ubicación de la página deseada dentro del disco.
  2. Localizar un marco libre.
  3. Leer la página deseada y cargarla en el marco recién liberado; cambiar las tablas de página y de marcos.
  4. Reiniciar el proceso de usuario
- 




Debemos resolver dos problemas principales a la hora de implementar la paginación bajo demanda: hay que desarrollar un algoritmo de asignación de marcos y un algoritmo de sustitución de páginas. Si tenemos múltiples procesos en memoria, debemos decidir cuántos marcos asignar a cada proceso. Además, cuando se necesita una sustitución de páginas, debemos seleccionar los marcos que hay que sustituir. El diseño de los algoritmos apropiados para resolver estos problemas es una tarea de gran importancia, porque las operaciones del E/S de disco son muy costosas en términos de rendimiento. Cualquier pequeña mejora en los métodos de paginación bajo demanda proporciona un gran beneficio en términos de rendimiento del sistema.






9.4.2


# Sustitución de PAGINAGINAS FIFO






El algoritmo más simple de sustitución de páginas es un algoritmo de tipo FIFO (first-in, out). El algoritmo de sustitución FIFO asocia con cada página el instante en que dicha página cargada en memoria. Cuando hace falta sustituir una página, se elige la página más Observe que no es estrictamente necesario anotar el instante en que se cargó cada página, ya podríamos simplemente crear una cola FIFO para almacenar todas las páginas en memoria y tituir la página situada al principio de la cola. Cuando se carga en memoria una página nueva, la inserta al final de la cola.






Después de sustituir una página activa por otra nueva, se producirá casi inmediatamente un nuevo fallo de página que provocará la recarga de la página activa. Entonces, será necesario sustituir alguna otra página con el fin de volver a cargar en la memoria la página activa.






9.4.3

# Sustitución óptima de PAGINAGINAS






Uno de los resultados del descubrimiento de la anomalía de Belady fue la búsqueda de ritmo óptimo de sustitución de páginas. Un algoritmo óptimo de sustitución de páginas es aquel que tenga la tasa más baja de fallos de página de entre todos los algoritmos y que no sufra de la anomalía de Belady.

Sustituir la página que no vaya a ser utilizada durante el período de tiempo más largo

Desafortunadamente, el algoritmo óptimo de sustitución de páginas resulta difícil de implementar, porque requiere un conocimiento futuro de la cadena de referencia








9.4.4

# Sustitución de paginaginas LRU







Si el algoritmo óptimo no resulta factible, quizá sí sea posible una aproximación a ese algoritmo' óptimo. La distinción clave entre los algoritmos FIFO y OPT es que el algoritmo FIFO utiliza el instante de tiempo en que se cargó una página en memoria, mientras que el algoritmo OPT utiliza el instante en el que hay que utilizar una página.


El algoritmo de sustitución LRU asocia con cada página el instante correspondiente al último uso de dicha página. Cuando hay que sustituir una página, el algoritmo LRU selecciona la página que no haya sido utilizada durante un período de tiempo más largo.


El problema consiste en determinar un orden para los marcos definido por el instante correspondiente al último uso. Existen dos posibles implementaciones:






**Contadores.** En el caso más simple, asociamos con cada entrada en la tabla de páginas un campo de tiempo de uso y añadimos a la CPU un reloj lógico o contador. El reloj se incrementa con cada referencia a memoria. Cuando se realiza una referencia a una página, se copia el contenido del registro de reloj en el campo de tiempo de uso de la entrada de la tabla de páginas correspondiente a dicha página






**Pila.** Otra técnica para implementar el algoritmo de sustitución LRU consiste en mantener una pila de números de página. Cada vez que se hace referencia a una página, se extrae esa página de la pila y se la coloca en la parte superior. De esta forma, la página más recientemente utilizada se encontrará siempre en la parte superior de la pila y la menos recientemente utilizada en la inferior





Al igual que el algoritmo óptimo de sustitución, el algoritmo de sustitución LRU no sufre anomalía de Belady. Ambos algoritmos pertenecen a una clase de algoritmos de sustitución de páginas, denominada **algoritmos de pila**, que jamás pueden exhibir la anomalía de Belady

Pocos sistemas podrían tolerar este nivel de carga de trabajo adicional debida a la gestión de memoria.







9.4.5


# Sustitución de paginaginas mediante aproximación LRU






Pocos sistemas informáticos proporcionan el suficiente soporte hardware como para implementar un verdadero algoritmo LRL de sustitución de páginas. Algunos sistemas no proporcionan soporte hardware en absoluto, por lo que deben utilizarse otros algoritmos de sustitución de páginas






Podemos disponer de información de ordenación adicional registrando los bits de referencia a intervalos regulares. Podemos mantener un byte de 8 bits para cada página en una tabla de memoria. A intervalos regulares, una interrupción de temporización transfiere el control al sistema operativo. El sistema operativo desplaza el bit de referencia de cada página, transfiriendo al bit de mayor peso de ese byte de 8 bits, desplazando asimismo los otros bits una posición hacia la derecha descartando el bit de menor peso









El número de bits de historial puede, por supuesto, variarse y se selecciona para hacer que la actualización sea lo más rápida posible. En el caso extremo, ese número puede reducirse a cero, dejando sólo el propio bit de referencia. Este algoritmo se denomina algoritmo de segunda oportunidad para la sustitución de páginas.






El algoritmo básico de segunda oportunidad para la sustitución de páginas es un algoritmo de sustitución FIFO. Sin embargo, cuando se selecciona una página, inspeccionamos su bit de referencia. Si el valor es 0, sustitutos dicha página, pero si el bit de referencia tiene el valor 1, damos a esa página una segunda oportunidad y seleccionamos la siguiente página de la FIFO.





Una forma de implementar el algoritmo de segunda oportunidad es una cola circular. Con este método, se utiliza un puntero para indicar cuál es la siguiente página que hay que sustituir. Cuando hace falta un marco, el puntero avanza hasta que encuentra una página con un bit de referencia igual a 0. Al ir avanzando, va borrando los bits de referencia





Cada página pertenece a una de estas cuatro clases. Cuando hace falta sustituir una página, se utiliza el mismo esquema que en el algoritmo del reloj; pero en lugar de examinar si la página a la que estamos apuntando tiene el bit de referencia puesto a 1, examinamos la clase a la que dicha página pertenece. Entonces, sustituimos la primera página que encontremos en la clase no vacía más baja. Observe que podemos tener que recorrer la cola circular varias veces antes de encontrar una página para sustituir.







9.4.6


# Sustitución de paginaginas Basada en contador







Hay muchos otros algoritmos que pueden utilizarse para la sustitución de páginas. Por ejemplo, podemos mantener un contador del número de referencias que se hayan hecho a cada página desarrollar los siguientes dos esquemas





El algoritmo de sustitución de páginas LFU requiere sustituir la página que tenga el valor más pequeño de contador. La razón para esta selección es que las páginas más activamente utilizadas deben tener un valor grande en el contador de referencias. Sin embargo, puede surgir un problema cuando una página se utiliza con gran frecuencia durante la fase inicial de un proceso y luego ya no se la vuelve a utilizar. Puesto que se la emplea con gran frecuencia, tendrá un valor de contador grande y permanecerá en memoria a pesar de que ya no sea necesaria.





El algoritmo de sustitución de páginas MFU (most frequently used, más frecuentemente utilizada) se basa en el argumento de que la página que tenga el valor de contador más pequeño acaba probablemente de ser cargada en memoria y todavía tiene que ser utilizada.









9.4.7


# ALGORITMO DE BÚFER DE PÁGINAS






Además de un algoritmo de sustitución de páginas específico, a menudo se utilizan otros procedimientos. Por ejemplo, los sistemas suelen mantener un conjunto compartido de marcos libres. Cuando se produce un fallo de página, se selecciona un marco víctima como antes. Sin embargo, la página deseada se lee en un marco libre extraído de ese conjunto compartido, antes de escribir en disco la víctima. Este procedimiento permite que el proceso se reinicie lo antes de posible, sin tener que esperar a que se descargue la página víctima. Posteriormente, cuando la víctima se descarga por fin, su marco se añade al conjunto compartido de marcos libre.





Algunas versiones del sistema UNIX utilizan este método en conjunción con el algoritmo de segunda oportunidad y dicho método puede ser una extensión útil de cualquier algoritmo de sustitución de páginas, para reducir el coste en que se incurre si se selecciona la página víctima incorrecta.







9.4.8


# APLICACIONES y sustitución de PÁGINAS






Las aplicaciones que acceden a los datos a través de la memoria virtual del sistema operativo tienen un rendimiento peor que si el sistema operativo no proporcionan ningún mecanismo de búfer.





Algunos sistemas operativos dan a ciertos programas de utilizar una partición del disco como si fuera una gran matriz secuencial de lógicos, sin ningún tipo de estructura de datos propia de los sistemas de archivos. Esta denomina en ocasiones disco sin formato y las operaciones de E/S que se efectúan sobre. matriz se denominan E/S sin formato. La E/S sin formato puentea todos los servicios del archivos, como por ejemplo la paginación bajo demanda para la E/S de archivos, el bloque archivos, la pre extracción, la asignación de espacio, los nombres de archivo y los direct observe que, aunque ciertas aplicaciones son más eficientes a la hora de implementar suspensión de servicios de almacenamiento de propósito especial en una partición sin formato, la mayoría d aplicaciones tienen un mejor rendimiento cuando utilizan los servicios normales del siste archivos






9.5

# ASIGNACIÓN DE marcos





Se puede obligar al sistema operativo a asignar todo su espacio de búfer y de tablas a partir de la lista de marcos libres.

Cuando este espacio no este siend utilizado por el sistema operativo, podra emplearse para soportar las necesidades de paginación del usuario.

