## Memoria virtual

### **El código debe estar en la memoria para ejecutarse,**

Pero el programa completo rara vez se usa (código de error, rutinas inusuales, grandes estructuras de datos). No se necesita el código completo del programa al mismo tiempo.

• El programa ya no está limitado por los límites de la memoria física.

• Cada programa toma menos memoria mientras se ejecuta -> más programas se ejecutan al mismo tiempo

• Se necesita menos E / S para cargar o intercambiar programas en la memoria -> cada programa de usuario se ejecuta más rápido

### **Motivación**

Memoria virtual: separación de la memoria lógica del usuario de la memoria física

• Solo una parte del programa necesita estar en memoria para su ejecución

• Por lo tanto, el espacio de direcciones lógicas puede ser mucho más grande que el espacio de direcciones físicas.

• Permite que los espacios de direcciones sean compartidos por varios procesos

• Permite una creación de procesos más eficiente

• Más programas ejecutándose simultáneamente

• Menos E / S necesaria para cargar o intercambiar procesos

### **Paginación de demanda**

Traiga una página a la memoria solo cuando sea necesario

• Se necesita menos E / S, no se necesitan E / S innecesarias

• Se necesita menos memoria

• Respuesta más rápida

• Se necesita más usuarios Página ⇒ referencia a ella referencia inválida ⇒ abortar sin memoria ⇒ traer a la memoria

### **Bit válido-inválido**

Con cada entrada de la tabla de páginas se asocia un bit válido-inválido:

• v => en memoria - residente en memoria,

• i => no en memoria Inicialmente válido: el bit no válido se establece en i en todas las entradas Durante la traducción de la dirección MMU, si el bit válido no válido en la entrada de la tabla de páginas es i => fallo de página

### **Fallo de página**

Operación que indica que una página requerida no está en la memoria para cargarla. Primero busca la página en almacenamiento, la carga y reinicia las instrucciones que causaron la falla de la página. Un proceso con todas las páginas en la memoria no causa fallas. La paginación de demanda pura causa fallas para cada uno.

### **Pasos en el manejo de fallas de página**

1. Si hay una referencia a una página, la primera referencia a esa página quedará atrapada en el sistema operativo • Error de página

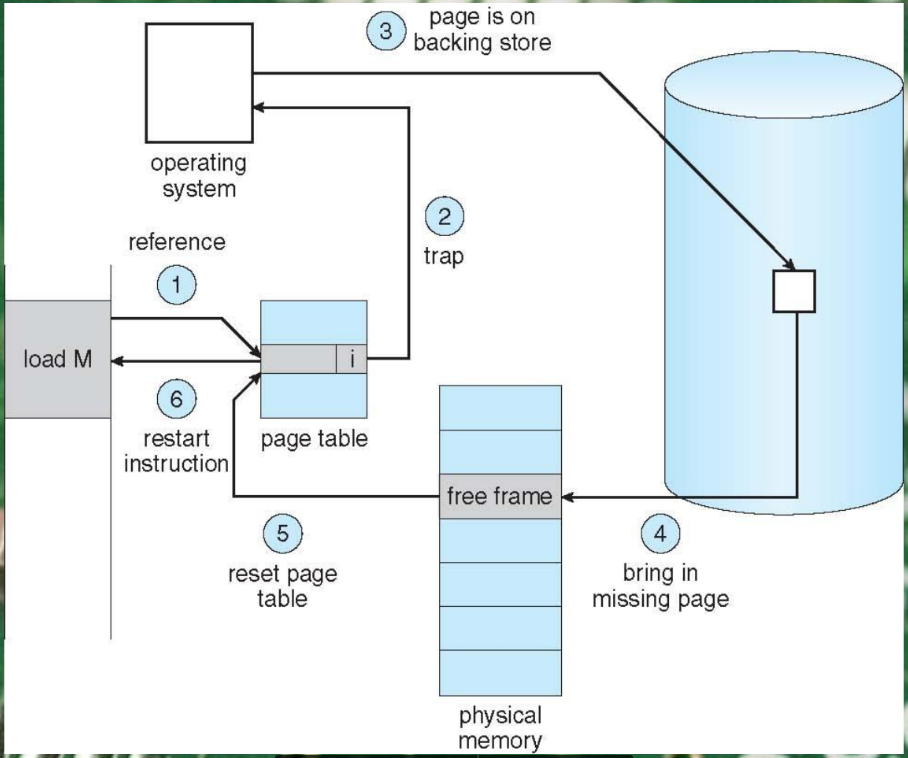
2. El sistema operativo mira otra tabla para decidir: • Referencia inválida ⇒ abortar • Simplemente no está en la memoria

3. Encuentra el cuadro libre

4.Cambie la página en el marco a través de la operación de disco programada

5. Restablezca las tablas para indicar la página ahora en la memoria Establezca el bit de validación = v

6. Reinicie la instrucción que causó la falla de la página



### **Lista de fotogramas libres**

• Cuando ocurre una falla de página, el sistema operativo debe traer la página deseada del almacenamiento secundario a la memoria principal.

• La mayoría de los sistemas operativos mantienen una lista de fotogramas libres: un conjunto de fotogramas libres para satisfacer tales solicitudes.

• Cuando se inicia un sistema, toda la memoria disponible se coloca en la lista de fotogramas libres.



### **¿Actuación?**

Tres acciones necesarias para una falla de página:

• Servicio de la interrupción

• Cargue la página

• Reiniciar el proceso. Tiempo de acceso efectivo: (1-p) tiempo de acceso a la memoria + (p) (sobrecarga de página + intercambio + intercambio) p es la probabilidad de falla de la página

Tomando: p = 1/1000 acceso a memoria = 200 nanosegundos sobrecarga de página = 8,000,000 nanosegundos (8 milisegundos). Tenemos una sobrecarga de (999/1000) \* 200 + (1/1000) \* 8,0 00,000 Eso es 8199.8 nanosegundos (una desaceleración por un factor de 41)

### **Optimizaciones**

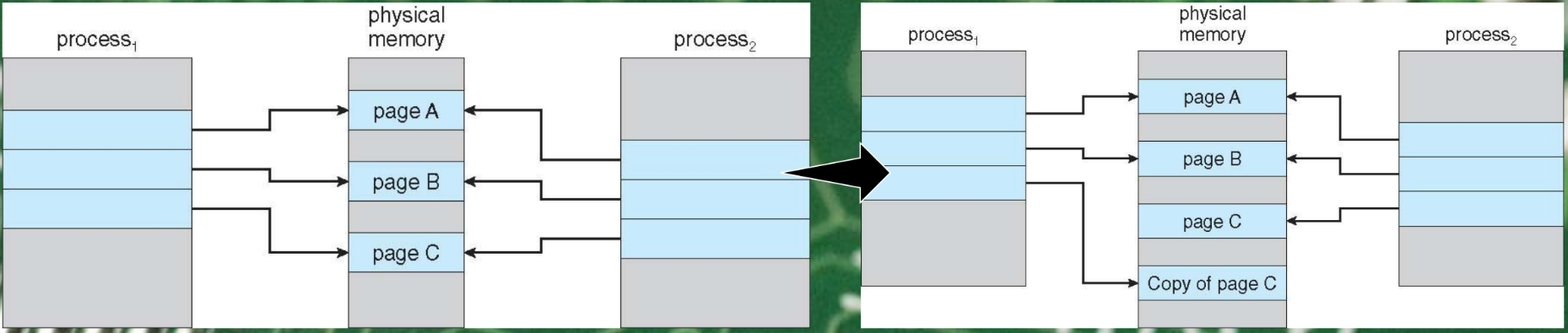
• Use una partición de intercambio dedicada (menos sobrecarga de falla de página). Copie toda la imagen del proceso en el espacio de intercambio, luego ingrese y salga del espacio de intercambio.

• Utilice bits de modificación (sucios).

• Copiar en escritura (después de fork () padre e hijo inicialmente comparten páginas).

• Reclamar páginas de solo lectura.

### **Copiar en escrito**



### **Reemplazo de página**

¿Qué sucede si no hay marco libre? La memoria suele tener una gran demanda Proceso para el reemplazo de la página:

• Si se encuentra un marco libre, úselo

• Si no, encuentre una víctima, quítela de la memoria y cargue la nueva página.

### **Algoritmos de reemplazo de cuadros**

Es necesario tener en cuenta que las páginas se pueden reutilizar. El objetivo es minimizar el número de fallas de página. La evaluación se realiza en cadenas que representan marcos, no en direcciones completas.

0,6,1,2,5,2,1,4,7,7,99,0

### **Algoritmo de primero en entrar, primero en salir (FIFO)**

Las páginas se asignan en el orden en que vienen. Funciona bajo el supuesto de que las páginas más antiguas serán menos necesarias. Cadena de referencia: 7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1 3 cuadros (3 las páginas pueden estar en la memoria a la vez por proceso)

### **Aplicar FIFO:**

Cadena de referencia: 1,2,3,4,1,2,5,1,2,3,4,5

para 3 cuadros para 4 cuadros

Esto se conoce como Anomalía de Belady

Hasta 1969, se creía que más marcos generaban menos fallas de página. La anomalía de Belady indica que esto no es cierto para algunos casos especiales. No es aplicable al reemplazo basado en pila.

**Algoritmo Óptimo**

Basado en reemplazar la página que se usará menos en el futuro. Cadena de referencia: 7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1

Mismo problema que SJF, necesita saber el futuro. Se usa para medir qué tan bien funciona su algoritmo

**Menos usado recientemente**

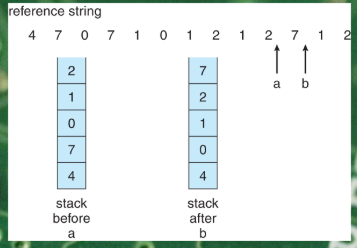
La víctima es la página que menos se ha utilizado. Utiliza el pasado en lugar del futuro. Mejor que FIFO, pero peor que Óptimo. Cadena de referencia: 7,0,1,2,0,3,0,4,2,3,0,3,0,3,2,1,2,0,1,7,0,1

**Implementación de LRU**

• Contador: mantenga la hora en que se utilizó por última vez. Cuando sea necesario, busque la página que no se ha utilizado durante más tiempo.

• Pila: mueva la página utilizada más recientemente a la parte superior de la pila. La víctima está en la parte inferior.

**Implementación de LRU**



**Aproximaciones de LRU**

LRU es lento, por lo que se ha implementado con diferentes aproximaciones

• Bit de referencia: si se usa, marque el bit con 1 y reemplace las páginas con 0.

• Segunda oportunidad: FIFO, pero cuando se selecciona una página como víctima, verifique el bit de referencia. Si es 1, ponga 0. Si es 0, reemplace la página.

**Segunda oportunidad mejorada**

Mejore el algoritmo utilizando el bit de referencia y modifique el bit (si está disponible) Tome el par ordenado (referencia, modificación):

• (0, 0) ninguno usado recientemente no modificado - la mejor página para reemplazar

• (0, 1) no utilizado recientemente pero modificado - no tan bueno, debe escribir antes del reemplazo

• (1, 0) usado recientemente pero limpio - probablemente se usará nuevamente pronto

• (1, 1) recientemente usado y modificado: probablemente se volverá a usar pronto y deberá escribirse antes del reemplazo. Cuando se requiera el reemplazo de la página, use el esquema de reloj pero use las cuatro clases para reemplazar la página en la clase más baja no vacía. Puede que tenga que buscar cola circular varias veces

### **Algoritmos de conteo**

Mantenga un contador del número de referencias que se han hecho a cada página => Algoritmo de uso menos frecuente (LFU) no común:

Reemplaza la página con el recuento más pequeño Algoritmo utilizado con más frecuencia (MFU): basado en el argumento de que la página con el recuento más pequeño probablemente se haya introducido y aún no se haya utilizado

### **Otras Consideraciones**

• Almacenamiento en búfer de páginas: mantenga siempre un búfer de cuadros libres

• Selección de la víctima:

• Global

• local

• Asignación de marcos:

• Fijo (igual o proporcional)

• Prioridad

### **Paliza**

• Si un proceso no tiene páginas "suficientes", la tasa de falla de página es muy alta

• Dado que en el fallo de la página el proceso se bloquea, el programador podría creer que se pueden manejar más procesos. Eventualmente, ningún proceso está haciendo ningún cálculo, todo el tiempo se dedica a reemplazar la página.

### **Localidad**

• Un proceso generalmente usa memoria de ubicaciones cercanas (piense en la siguiente instrucción).

• Esto se conoce como el principio de localidad o localidad de referencia.

• Localidad espacial: la siguiente dirección de memoria a la que se accederá estará cerca de la actual.

• Localidad temporal: en poco tiempo, se accederá al mismo espacio de memoria varias veces.

• Si podemos garantizar una localidad completa en la memoria, reducimos el número de fallas de página.

### **Conjuntos de trabajo**

• Un número fijo de referencias de página, por ejemplo: 10.000 instrucciones.

• Un gran número abarca varias localidades.

• Un número bajo de páginas requiere múltiples fallas de página para la misma localidad.

• Un conjunto de trabajo infinito es lo mismo que cargar todo el proceso.

### **Asignación de memoria del núcleo**

• Tratada de manera diferente a la memoria del usuario

• Asigna usando:

• Sistema de amigos

• losa