ISO P5

▼ 1

Una dirección física es una dirección de memorial real. En memoria principal se representa como la dirección absoluta de memoria. Es con la que se accede efectivamente a memoria

La dirección lógica es una abstracción de la dirección física, la cual referencia una localidad en memoria, pero debe ser traducida a una dirección física.

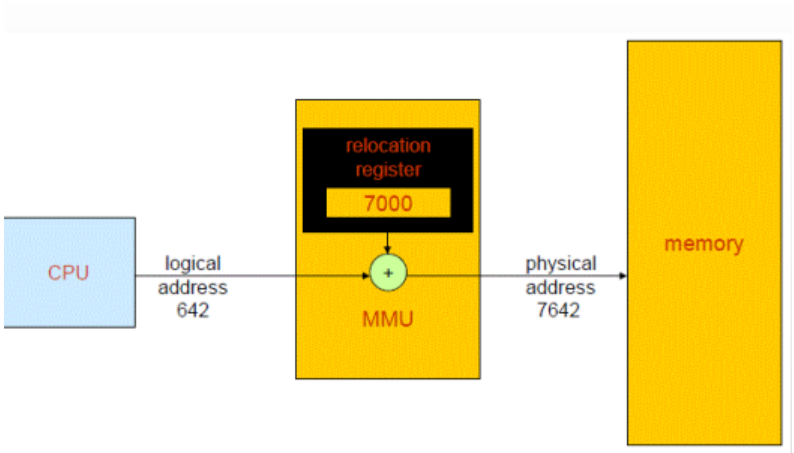
▼ 2

a) Las particiones fijas consisten en la división de la memoria en porciones de un tamaño determinado, el cual puede ser distinto entre las diferentes particiones pero estático. Asigna un espacio para sólo un proceso a través de un mecanismo. Puede generar fragmentación interna y limita la cantidad de procesos.

Las particiones dinámicas consisten en la división de las memoria en porciones de tamaño variable. El tamaño es igual al que el proceso necesita. Asigna un espacio para sólo un proceso. Puede generar fragmentación externa.

b) El SO debe conocer el tamaño del proceso y un registro de las partes de memoria utilizadas y no utilizadas.

c)



(Ja, realice)

▼ 3

Particiones de igual tamaño: búsqueda y asignación más fácil puesto que el algoritmo no debe buscar el inicio/final de cada partición. La desventaja es que es más la fragmentación interna que se genera

Particiones de distinto tamaño: búsqueda y asignación más costosas dado que se debe hallar el inicio/fin de cada partición, pero se puede seleccionar la partición libre que más se ajuste al tamaño del proceso.

▼ 4

a) Fragmentación interna hace referencia a la existencia de espacio de memoria dentro de una partición que el proceso almacenado no usa. En otras palabras, es el sobrante entre la diferencia del tamaño de la partición y el tamaño del proceso almacenado.

La fragmentación externa son aquellos espacios que quedan entre las particiones dinámicas, que posiblemente no sean los suficientemente grandes como para que se establezca una nueva partición en ellos (O sea, alberguen un proceso)

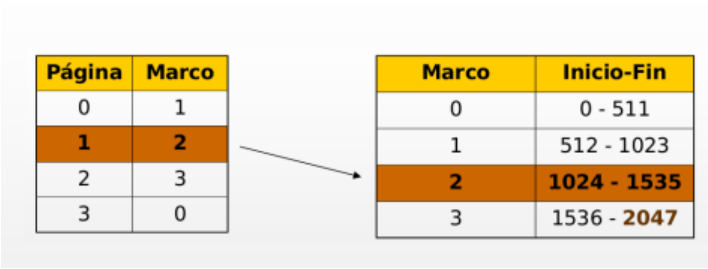
b) La fragmentación externa puede solucionarse haciéndose compactación, que consiste en juntar aquellos espacios entre particiones a través de un algoritmo. Es costoso.

▼ 5

La paginación consiste en la división de la memoria física en “marcos” de igual tamaño, y división de los espacios de direcciones de los procesos (memoria lógica) en “páginas”, también de igual tamaño. Los marcos y las páginas tienen el mismo tamaño (generalmente este tamaño es de 512b). Los marcos contienen páginas

b) El SO está encargado de tener una tabla con las páginas de cada proceso, la cual tiene el marco de donde se halla dicha página.

c)



d) No puede producirse fragmentación externa, debido a que la división de la memoria en partes iguales no deja espacio entre los marcos. Puede producirse fragmentación interna en la última página de un proceso, pero este tipo de fragmentación sucede en mucha menor medida.

▼ 6

Son similares en el hecho de que ambas dividen la memoria en partes iguales para asignarles información de procesos. Sin embargo, difieren en que en la paginación el proceso es dividido en partes de igual tamaño que los marcos, por lo tanto se minimizan los casos de fragmentación interna. También en que una partición puede ser de distintos tamaños, aunque no variable. Por último, otra diferencia es que en las particiones fijas se corresponde un proceso por partición, mientras que en la paginación un proceso puede encontrase en muchos marcos.

▼ 7

a) En papel

b)

i) 35 / 512 = 0 → correcta

35 mod 512 = 35 → MF: 1536 + 35 = 1571

ii) 512 / 512 = 1 → correcta

512 mod 512 = 0 → MF: 1560 + 0 = 2560

iii) 2051 / 512 = 4 →error

iv) 0 / 512 = 0 → correcta

0 mod 512 = 0 → MF: 1536 + 0 = 1536

v) 1325 / 512 = 2 → correcta

1325 mod 512 = 301 → MF: 1024 + 301 = 1325

vi) 602 / 512 = 1 → correcta

602 mod 512 = 90 → MF: 2560 + 90 = 2650

c)

i) 509 / 512 = 0

509 mod 512 = 509 → ML: 512 + 509 = 1023

el marco 0 no es usado por el proceso. → error

ii) 1500 / 512 = 2

1500 mod 512 = 476 → ML: 1024 + 476 = 1500

iii) 0 / 512 = 0

0 mod 512 = 0

el marco 0 no es usado por el proceso → error

iv) 3215 div 512 = 6

3215 mod 512 = 143 → ML: 1536 + 143 = 1689

v) 1024 div 512 = 2

1024 mod 512 = 0 → ML: 1024 + 0 = 1024

vi) 2000 div 512 = 3

2000 mod 512 = 464 → ML: 0 + 464 = 464

d) Externa no. Se produce fragmentación interna porque al ser el proceso de 2000kb y en total las páginas 0 a 3 ocupan 2047kb, en la última página (3) habrán 48kb sin utilizar (2047 -1999)

▼ 8

a)

Una página es representada con 3 bits (para las 8 páginas posibles, 8 = 2 ^ 3) y el direccionamiento dentro de la página es representado con 10 bits (para los 1024 bytes posibles, 1024 = 2 ^ 10).

Entonces, una dirección lógica puede representarse con 13 bits

Un marco es representad con 5 bits (para los 32 marcos posibles, 32 = 2 ^ 5) y el direccionamiento dentro de lun marco es representado con 10 bits (para los 1024 bytes posibles, 1024 = 2 ^ 10)

Entonces, una dirección física es representada con 15 bits.

▼ 9

a) La segmentación emplea una tabla de segmentos, la cual posee la dirección de inicio y de fin de un segmento. Los segmentos tienen diferentes tamaños. Las direcciones lógicas equivalen al número de segmento y desplazamiento en el mismo.

Es prácticamente como la paginación, pero como el tamaño de un segmento varía, se tiene una tabla con las direcciones límites de dicho segmento. Entonces una dirección lógica contiene el número de segmento y direccionamiento, busca el segmento en la tabla y a partir de la dirección de inicio contenida en dicho segmento (si es válida el cálculo, es decir, no se pasa de los límites del segmento) se consigue la dirección física efectiva.

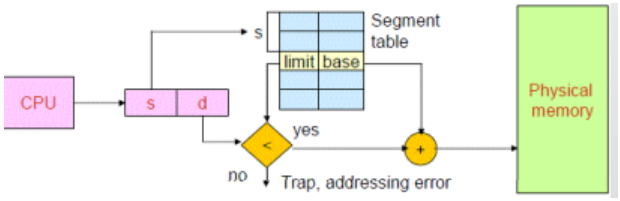
Un proceso se guarda en partes en distintos segmentos

Se hace referencia a la dirección real de memoria física, por lo que no hace falta transformar las direcciones.

Igual que la paginación, un segmento puede ser compartido por varios procesos.

b) Una tabla de segmentos con el identificador de segmento y su rango de direcciones (inicio y fin)

c)



d) En este esquema no se produce fragmentación interna porque se adapta el tamaño de un segmento al del proceso, pero puede producirse externa debido a que entre segmentos pueden quedar espacios de memoria que no sean lo suficientemente grandes para alocar un proceso

▼ 10

Ambas pueden general fragmentación externa y producen fragmentación externa. Difieren en que el particionamiento dinámico guarda un proceso entero y la segmentación divide el proceso en distintos segmentos.

▼ 11

Ambas dividen a los procesos en partes, ambas dividen la memoria física para albergar partes de los procesos, ambas requieren de una tabla

Difieren en que la paginación genera fragmentación interna y la segmentación externa. La paginación se basa en particiones (marcos) de tamaño fijo, mientras que la segmentación se basa en particiones (segmentos) de tamaño dinámico. La paginación requiere de transformar una dirección lógica para obtener la dirección física correspondiente, en cambio en la segmentación se representa a la dirección física efectiva. La paginación requiere de una tabla con las páginas y sus marcos correspondientes. La segmentación tiene una tabla con el rango de direcciones de cada segmento.

▼ 12

Calculo: dirección fisica del segmento + desplazamiento de página + desplazamiento

i) seg 2, pag 1, direc base 20.

Seg 2, dir base 1500 (La cual es una direccion fisica real)

1500 + 20 + 1 = 1521

ii) 500 + 60 + 15 = 575

iii) 5000 + 120 + 10 = 5130

iv) 1500 + 0 + 5 = 1505

▼ 13

a) La memoria virtual permite ejecutar procesos que requieren más memoria de la que se tiene, de esta manera, un proceso se mantiene en memoria física pero las partes que se necesitan se cargan en memoria principal

La memoria virtual NO es la memoria principal, pero la memoria principal si es virtual.

b) Se debe apoyar en el hardware, el cual debe encargarse que se maneje correctamente el uso de memoria. Debe evitar, entre otras cosas, accesos indebidos a memoria, accesos dentro del rango permitido, etc.

c) Se requiere un bit que indique que dicha página existe en el sistema.

▼ 14

a) Los page fault se producen cuando se requiere una dirección que no se halla cargada en memoria principal. Es una excepción arrojada por el SO. Aunque el término sugiere un mal funcionamiento, se trata de un procedimiento normal y típico.

b) El SO se encarga de detectar los fallos de página

c)

1- Se emite la excepción de fallo de página

2- Se guarda el PC y otros registros en la pila

3- El kernel llama a la rutina para fallos de página

4- Se averigua la dirección virtual que se busca

5- Se comprueba la dirección válida

6- Se selecciona un marco libre (si no lo hay, se libera mediante un algoritmo)

7- Se baja la página. Se marca la página para evitar que se use mientras se lleva a cabo el procedimiento

8- Se sube la página que hay que cargar

9- Se actualiza la tabla de páginas

10- Se devuelven los valores de la pila para continuar con el programa.

▼ 15

a) 2 ^ 32 me da la cantidad de direcciones posibles. Cada una de esas direcciones referencia un byte

⇒ 2 ^ 32 * 1 = cantidad de referencias a direcciones de un byte.

Si cada dirección referenciara 2 bytes, sería * 2, se referenciara 7, sería * 7, y así.

Entonces, la cantidad de direcciones es 4.294.967.296 (numero), y el tamaño del proceso es de 4.294.967.296 bytes (tamaño).

4.294.967.296 bytes / 1024 = 4.194.304 kb

4.194.304 kb / 1024 = 4096 mb

4096 mb / 1024 = 4gb

b) 4.194.304 / 512 = 8192 páginas

c) 256 mb = 262.144 kb

262.144 / 512 = 512 marcos

d)

Rta correcta segun la cátedra:

4.294.967.296 (cantidad de direcciones de un proceso)

2 kb

4.294.967.296 * 2 = 8gb

Cada dirección es una dirección lógica, lo que quiere decir que apunta a un bloque de la memoria fisica, la cual probablemente tenga aún más direcciones de memoria.

> 💡 No termino de entender la diferencia entre direcciones lógicas de un proceso y cantidad de páginas de un proceso
>
> La cantidad de direcciones de un proceso = cantidad de páginas de un proceso (??

▼ 16

Las tablas de 1 nivel son tablas que abarcan todo el sistema, en la que hay una entrada por proceso indexadas por PID y una dirección de tabla que contiene las páginas del proceso.

Las tablas de 2 niveles es una estructura de tablas de páginas de dos niveles, en la que una tabla tiene una página que tiene una tabla con las páginas de un proceso. Puede tener más de 2 niveles. Funciona como estructura de árbol.

Las tablas invertidas tienen una entrada por cada página real (o sea, marco) en memoria. El sistema sólo necesita una. Reduce la memoria física ocupada por la tabla de páginas, pero aumenta el tiempo de búsqueda de páginas debido a las verificaciones que debe hacer cada vez que accede a una referencia. Puede utilizarse una tabla de hash para solucionar este problema, pero crea otro que es que se deben hacer dos acceso a memoria.

▼ 17

¿Como sé si es fallo de página?

a)

1052 / 512 = 2 → pagina

1052 mod 512 = 28 → desplazamiento

Página 2 no cargada en memoria principal → fallo de página

b)

2221 / 512 = 4 → página

2221 mod 512 = 173

Página 4 no carga en memoria principal → fallo de página

c)

5499 / 512 = 10 → página

Página 10 pertenece a una dirección de memoria no accesible

d)

3101 / 512 = 6 → página

Página 6 pertenece a una dirección de memoria no accesible

▼ 18

Tamaño de página pequeño: se pueden tener más páginas, lo que puede traer ventajas si los procesos son pequeños porque se reduce la fragmentación interna posible en la última página de dicho proceso. Una desventaja sería si el proceso es grande, puesto que sus partes estarían divididas en muchas páginas, por lo que traer ese proceso a memoria sería más costoso. Otra desventaja es que, al haber más páginas, se deberán reservar más bits para direcciones, y la tabla de de páginas será más grande

Tamaño de página grande: si los procesos son pequeños puede generarse mucha pérdida de memoria con la última página de un proceso (fragmentación interna). Un beneficio es que el espacio para guardar procesos es mayor, debido a que mientras menos páginas haya, menos bits para direcciones se precisará. Tambien es más rápido transferir grandes bloques de datos.

▼ 19

a) La asignación fija determina que cada proceso tiene una cantidad especifica (arbitraria) de marcos para usar. Existen dos tipos de reparto de marcos: equitativo o proporcional. El equitativo reparte el mismo número de marcos a cada proceso, basandose en la ecuación: m / p, siendo m la cantidad total de marcos disponibles en memoria y p la cantidad de procesos. El proporcional reparte un número determinado por la necesidad de cada proceso, basandose en la ecuacion: Vp * m / Vt, siendo Vp las páginas de proceso, m la cantidad de marcos disponibles en memoria y Vt las páginas totales.

La asignación dinámica determina que los procesos se cargan de acuerdo a la cantidad de marcos que necesiten.

b)

i) A cada proceso se le asignan 10 marcos.

ii) Proceso 1 → 15 * 40 / 63 = (9 o) 10 páginas

Proceso 2 y 3 → 20 * 40 / 63 = (12 o) 13 páginas

Proceso 4 → (5 o) 6 páginas

*Los números son con coma, asumo que toman una página de más y que se produce fragmentación interna en ellas.

c) El reparto proporcional es más eficiente debido a que reparte equitativamente a cada proceso y evita que procesos pequeños tomen más páginas de las que necesitan.

▼ 20

a) OPT es el que mejor tasa de fallos tiene, sin embargo es un algoritmo teórico (no se puede implementar en la realidad)

LRU es muy bueno pero tambien muy costoso

FIFO Segunda chance es bueno y barato. Es una mejor del FIFO tradicional

FIFO es regular pero muy simple y barato. Presenta un problema que es la anomalía de Belady, la cual explica que a mayor cantidad de marcos en memoria física, mayor tasa de fallos se obtiene.

b) OPT no es implementable

LRU se implementa con una lista enlazada que representa el orden de tiempo que llevan los algoritmos en memoria, siendo el último el más longevo. Es costoso debido a que cada vez que se referencia una página debe ser movida en la lista. Una alternativo es que cada página tenga un contador

FIFO tiene un cola, en la cual el primer elemento es el que es reemplazado cuando se requiere traer una página a memoria.

FIFO de segunda chance actúa igual que el FIFO tradicional, pero cuando se toma el primer elemento de la cola, se chequea un bit que determina si la página fue referenciada luego de llegar a memoria. Si es así, se envía al final de la cola sin dicho bit. Si una página no tiene el bit, se toma como página víctima.

c) El SO pone la página en un frame designado para la descarga asincrónica y envía la orden de descargar mientras se ejecuta otro proceso (por eso asincrónica). El frame de descarga es el mismo que contenía a la página víctima

▼ 21

a) El reemplazo local consiste en que, ante un fallo de página, la página víctima sólo puede seleccionarse entre las páginas propias del proceso. En cambio, en el reemplazo global, la página víctima puede seleccionarse de cualquier proceso.

b) Si es posible. Un proceso puede tomar marcos de otro proceso y asignárselos a sí mismo. También si un proceso es de mayor prioridad a otro (Esto no sucede con el reemplazo local).

▼ 22

Cuando la página victima tiene *, se lo quita y se “reinicia” su contador de longevidad (porque se lo re-encola) y se busca el siguiente más longevo como víctima.

Un elemento se re-encola cuando se le quita la *

En la referencia a 5 (8* referencia) se toma como victima 1, pero al estar referenciada, se quita la referencia, reinserta en la cola y busca la siguiente. Se toma como victima la 2 y pasa lo mismo. Se toma como victima la 3 y como no se halla referenciada se hace el intercambio.

En la referencia a 10 (10* referencia), como los frame 1 y 2 tienen 2 referencias de longevidad (porque se reinsertaron), el que se toma como más longevo es el 4.

En el siguiente instante, o sea la segunda referencia a 4, se toma el frame 1 porque es el primero que se halla (supongo que cuando se encolan 1, 2 y 5 en la referencia a 5 (dos referencias antes) se encola por número de frame a modo de desempate

Cuando se referencia al 12 por primera vez, la página victima en realidad es el frame 5 con el valor 6, pero como está referenciada se toma el siguiente que es el frame 1

En la tercera referencia a 1, se toma como victima la pagina 12 a pesar que llevan el mismo tiempo con la 6. Se hace el intercambio con la 6 porque la 12 estaba referenciada

Referencias en las que hay que desempatar: segunda referencia a 4, primera referencia a 12, tercera referencia a 1

Segunda referencia a 4: parece que se toma 1 por tener el número de página menor. No creo que se tome porque lleva desde el principio porque cuando se quita el * se re-encola y reinicia la longevidad

Primera referencia a 12: parece que se toma la página 6. No tiene número de referencia menor. Se considera la longevidad anterior?

Tercera referencia a 1: 12 y 6 llevan el mismo tiempo. 12 no se toma como victima. Se toma 6 porque se considera la longevidad anterior?

b) FIFO: 2,1 seg

FIFO SC: 2,2 seg

LRU: 2,2 seg

OPT: 1,6 seg

▼ 23

a) 6 fallos para cada técnica

▼ 25

a) La hiperpaginación sucede cuando un sistema pasa más tiempo paginando, o sea cargando o descargando partes de un procesos entre la memoria principal y el disco, que ejecutando procesos.

b) Las razones de que suceda son que a veces un proceso no tiene cargado en memoria todas las páginas que necesita para ejecutarse, por lo que debe cargarlas desde la memoria física a la memoria principal.

c) El SO detecta el thrashing a traves del monitoreo de los procesos.

d) Se elige un proceso para suspender, así reasignar frames

▼ 26

a) Sucede la hiperpaginación. No debería incrementarse el nivel de multiprogramación.

La paginación no está afectando el rendimiento de la CPU (No sé si significa que está siendo útil o no).

b)

c)

No entiendo bien como resolver este ejercicio.

▼ 27

La d) es seguro, porque una buena solución a largo plazo a la paginación es aumentar el tamaño de la memoria principal. (Puede ser la b) tambien pero no sé fundamentarla)

▼ 28

a)

([1 + (1-0,3) * 20 + 0,3 * (300+20)] +

[20 + (1-0,3) * 20 + 0,3 * (300+20)])

/ 2

= 120,5

b)

(1 * 0,6 + 20 * 0,4) + (1-0,3) * 20 + 0,3 * [(500 * 0,7 + 300 * 0,3) +20]

= 160,6

c)

200 > 1 + (1-p) * 20 + p * [(500 * 0,6 + 300 * 0,4) + 20]

⇒ p debe ser menor a 0,42, o sea, la tasa de fallas de página máxima aceptable es 0,42

▼ 29

a) La anomalía de Belady es una situación que se presenta en el algoritmo de reemplazo de páginas de FIFO, en la que se explica que a mayor cantidad de marcos en memoria física, mayor tasa de fallos se obtiene, lo cual se creía que pasaba lo contrario.

b)

i) Con tres marcos suceden 9 fallos

ii) Con cuatro marcos sucede 10 fallos

▼ 30

Según la diapo de teoría:

Habiendo una matriz X [i, j]

Dos for anidados, que ambos iteran la misma cantidad de veces

Si el valor que se incrementa en el segundo for es el de j, la cantidad de page faults es igual al tamaño de iteración del for.

Si el valor que se incrementa en el segundo for es el de i, la cantidad de page faults es igual al tamaño de iteración del for x 2.

```
#define Size 64
int A[Size; Size], B[Size; Size], C[Size; Size];
int register i, j;
for (j = 0; j < Size; j ++)
    for (i = 0; i < Size; i++)
        C[i; j] = A[i; j] + B[i; j];
```

PF = 64 x 64

```
#define Size 64
int A[Size; Size], B[Size; Size], C[Size; Size];
int register i, j;
for (i = 0; j < Size; j ++)
    for (j = 0; i < Size; i++)
        C[i; j] = A[i; j] + B[i; j];
```

PF = 64

a)

Se almacenan 4 filas por página en 16 páginas

Primera página: [0,0] → [0,63]; [1,0] → [0,63]; [2,0] → [2,63]; [3,0] → [3,63];

Segunda página: [4,0] → [4,63]; [5,0] → [5,63]; …

…

¿Hay PF cada 3 referencias? Si se utilizan 3 marcos para las matrices

b) Se puede solucionar haciendo que primero se itere sobre i y luego sobre j, en vez de primero sobre j y segundo sobre i

▼ 31

a) El working set indica la cantidad de referencias previas que debo tomar. Son las “cercanas” al proceso actual. Son las referencias más actuales

Segun la ventana de tamaño 5, WSA(24) tiene hacia atras WSA(20) = 4; WSA(21) = 5; WSA(22) = 6; WSA(23) = 6; WSA(24) = 6.

⇒ el cto. de trabajo del proceso A en el instante 24 es {4; 5; 6}

⇒ y del proceso B es {2; 3; 5; 6}

No se si el actual se toma, o se toman los 5 anteriores.

b)

WSA(60) = {1; 2; 3; 4; 5}

WSB(60) = {3; 4; 5}

c) m (marcos) = 8

WSS24 = 7 ({4; 5; 6; 2; 3; 5; 6}, todos los procesos del working set de A y B en el instante 24)

D = WSS24 = 7

m > D ⇒ no hay thrashing con 8 marcos

Pero con m = 6, m < D ⇒ hay thrashing con 6 marcos

d)

6 PF

e)

Se le asignan más marcos al proceso. Si no hay más frames disponibles se puede suspender el proceso.