Tema-3-Relacion.pdf

fer__luque

Sistemas Operativos

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

KEEP CALM AND ESTUDIA UN POQUITO

WUOLAH + #QuédateEnCasa

Relación ejercicios Tema 3

Cuestiones sobre memoria virtual

¿Cuáles son las diferencias entre utilizar un mecanismo de paginación sin memoria virtual y con memoria virtual?

Con la paginación, las referencias a memoria dentro de un programa en ejecución se realizan sobre el espacio de direcciones lógico.

En el caso de que tengamos memoria virtual, estas páginas pueden ser retiradas de memoria principal a memoria auxiliar y vueltas a traer a memoria principal en una nueva área de memoria.

Sin embargo, si no disponemos de memoria virtual, durante la ejecución de un programa, todas las páginas que forman dicho programa deben residir continuamente en memoria principal.

¿Qué campos de información se incluyen necesariamente en cada entrada de la tabla de páginas de un sistema de memoria virtual con paginación (memoria virtual paginada)? Explique para que sirve cada uno de ellos.

- **Dirección base del marco de página**: Informa sobre la dirección real que se corresponde con la página virtual
- **Bit de validez/presencia**: Mantiene la información sobre la ubicación de dicha página. Si está en 1, la página está en MP (y MA), si está en 0, no se encuentra en MP y en caso de que se referencie, habría que traerla de MA
- **Protección**: Información sobre el modo de acceso a dicha página (rwx)
- **Modificación**: Informa de si se ha modificado la información que contiene la página para que, en el caso de ser retirada de MP, se "anoten" dichas modificaciones en MA

¿Para qué sirve el TLB de la MMU?

El TLB (*Translation Look-aside Buffer*) (generalmente son vario en el MMU), es el encargado de mantener las correspondencias entre página virtual y física resueltas con anterioridad. De este modo, en el caso de que necesite una correspondencia que ya esté resuelta (***TLB hit***), no necesitará consultar la Tabla de Páginas; solo necesitará acceder a ella en caso de que se necesite una correspondencia no resuelta (***TLB miss***)

¿Por qué los registros del TLB de la MMU (la cual se direcciona mediante direcciones virtuales) pueden producir incoherencias y requieren que el sistema operativo los invalide (flush de TLBs) en cada cambio de contexto y, en cambio, una memoria direccionada mediante direcciones físicas no requiere tal "limpieza" del TLB?

Porque las "traducciones" que pueden guardar los registros TLB de la MMU dependen de cada proceso. Es decir, cada página de memoria virtual de un proceso se corresponde con un determinado marco de página. Esta correspondencia no es la misma para todos los procesos, por tanto al realizar un cambio de contexto (de proceso), se requiere borrar esas traducciones que correspondían al proceso antiguo.

En memoria direccionada mediante direcciones físicas no se requiere de dicha limpieza porque no son necesarias las traducciones de dirección virtual a física.

¿Qué es el *buffering* de páginas y para qué sirve?

El *buffering* de páginas es una pequeña cache entre la memoria principal y la memoria auxiliar. Cuando se produce una falta de página, antes de ir directamente a buscarla en memoria auxiliar, se revisa la caché, pues en el caso de que dicha página se encuentre en esa caché (*buffer*), la transferencia de este *buffer* a memoria principal es más rápida que la transferencia entre memoria auxiliar y memoria principal.

Para la gestión de memoria en un sistema de memoria virtual con paginación, ¿qué estructura/s de datos necesitará mantener el Sistema Operativo para administrar el espacio asignado a los procesos y el espacio libre?

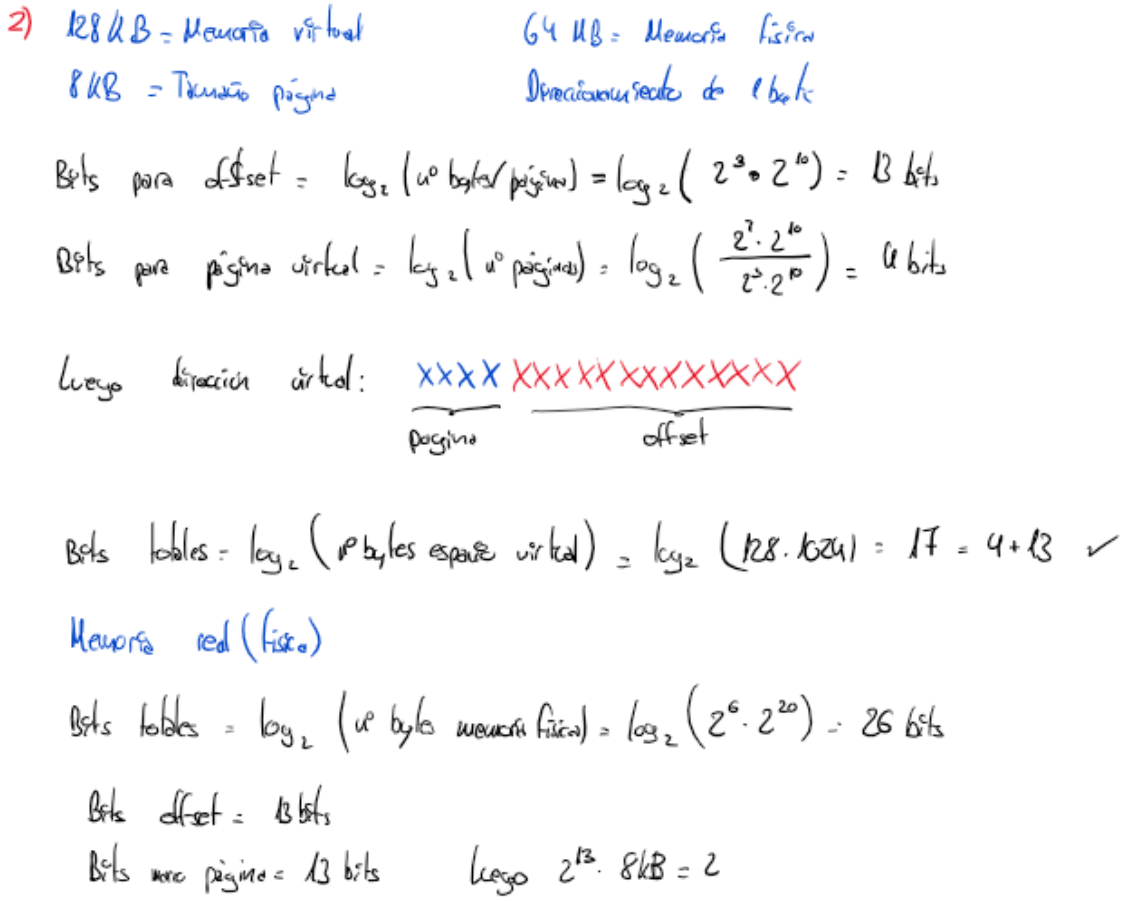
- **Tabla de Páginas**: para la traducción dirección lógica-dirección física
- **Tabla de ubicación en disco**: para la recuperación/almacenamiento de páginas de Memoria Auxiliar (*swapping*)
- **Tabla de Marcos de Página**: mantiene información relativa a cada marco de página en el que se divide la Memoria Principal

¿Cuánto puede avanzar como máximo la "aguja" del algoritmo de reemplazo de páginas del reloj durante la selección de una página?

Tantos como número de páginas del mapa de memoria haya en ese momento cargadas en memoria principal. **Explicar**

Problemas Paginación / Memoria Virtual

Considere un sistema con un espacio de direcciones virtuales de 128K (128 * 1024) páginas con un tamaño de 8KB1 cada una, una memoria física de 64 MB y direccionamiento al nivel de byte. ¿Cuántos bits hay en la dirección virtual? ¿Y en la física? En otras palabras, cuantos bits son necesarios para acceder a una única dirección virtual del espacio virtual y cuantos para acceder a una única dirección física del espacio físico.



Considere un sistema en el que existe un HDD para el sistema de archivos y otro HDD para el espacio de intercambio de la memoria virtual. Analice y responda las preguntas para cada uno de los siguientes escenarios:

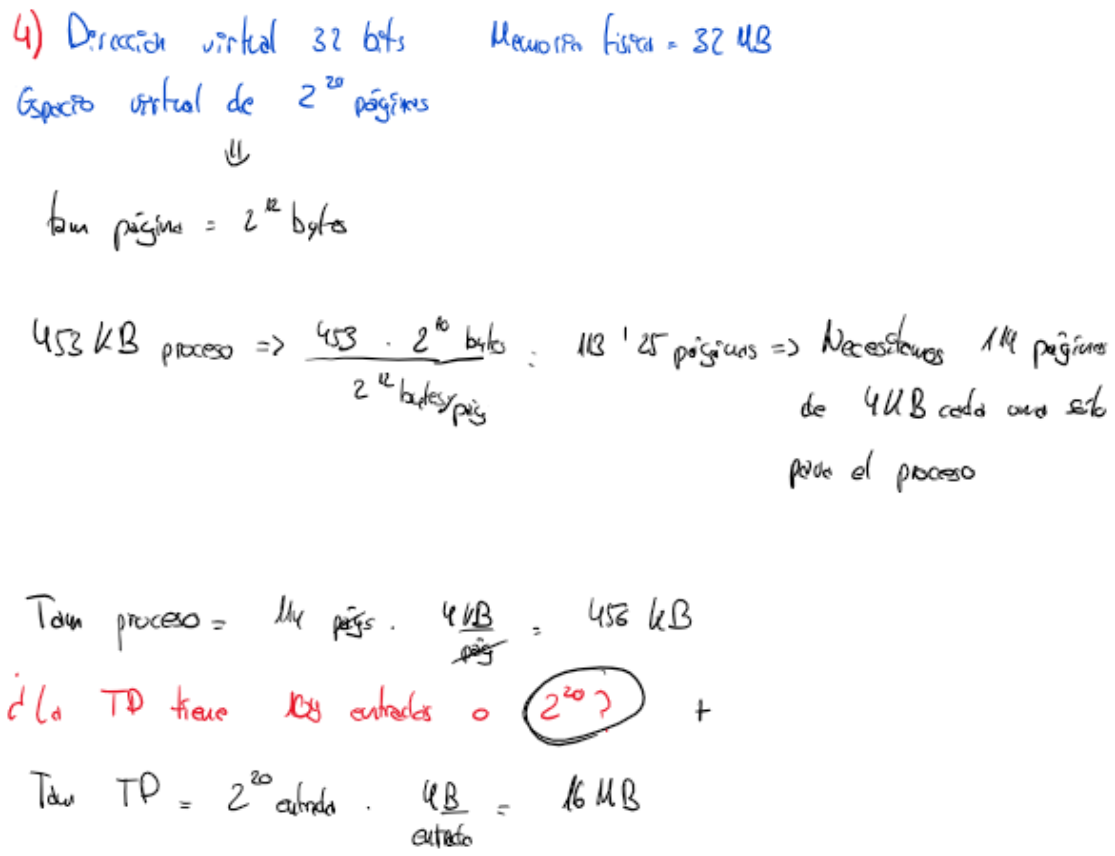
- **Escenario en el cual la tasa de utilización de la CPU es del 15% y el dispositivo de paginación tiene una tasa de utilización del 97%. ¿Qué indican estas medidas de prestaciones del sistema?**

  En este caso, el sistema se encuentra ***thrashing*** (es decir, hiperpaginando). La tasa de utilización de la CPU es tan baja respecto a la tasa de utilización del dispositivo de paginación porque el sistema está más tiempo recuperando páginas de memoria auxiliar (debido al alto grado de multiprogramación), que realizando cómputos.

- **Escenario en el cual la tasa de utilización de la CPU sigue siendo del 15% pero la tasa de utilización del dispositivo de paginación es del 20%. ¿Qué indican estas medidas de prestaciones del sistema?**

  En esta situación podemos deducir que todos los procesos activos están limitados por entrada salida. Cabría añadir también que el dispositivo dedicado al sistema de archivos estará a un porcentaje de utilización muy alto.

Sea un sistema de memoria virtual paginada con direcciones virtuales de 32 bits que proporciona un espacio virtual de 220 páginas y con una memoria física de 32 MB ¿cuánta memoria requiere en total un proceso que ejecuta un programa de 453KB, incluida la tabla de páginas necesaria para ejecutarlo y cuyas entradas ocupan un tamaño de 32 bits cada una?



Un ordenador dispone solamente de 4 marcos de página. En la siguiente tabla se muestran: el tiempo de carga, el tiempo del último acceso y los bits R y M para cada página (los tiempos están en tics de reloj). Responda a las siguientes cuestiones justificando su respuesta.

Página	tº de carga	tº última referencia	Bit de referencia	Bit de modificación
0	126	279	1	0
1	230	235	1	0
2	120	272	1	1
3	160	200	1	1

- **¿Qué página se sustituye si se usa el algoritmo FIFO?**

  El algoritmo FIFO sustituirá la página más antigua (la que lleve más tiempo en memoria), en este caso, la que tenga un tº de carga menor (haya sido cargada previamente). La **página 2**.

- **¿Qué página se sustituye si se usa el algoritmo LRU?**

  El algoritmo LRU sustituye la página que se referenció hace más tiempo, es decir, la que tenga menor tº última referencia. La **página 3**

Un ordenador proporciona un espacio de direccionamiento virtual para cada proceso de 65.536 bytes de espacio, dividido en páginas de 4096 bytes. Cierto programa tiene un tamaño de región de texto de 32768 bytes, un tamaño de región de datos de 16386 bytes y tamaño de región de pila de 15878. ¿Cabría este programa en el espacio de direcciones virtual? (Tenga en cuenta que una página no puede ser utilizada por regiones distintas). Si

no es así, ¿cómo podríamos conseguir ubicar el programa en RAM variando los parámetros que permite el esquema de paginación?

Si tenemos $65536\ B$ divididos en páginas de $4096\ B$, tendremos:
$num\_paginas = \frac{65536B}{4096B/pag} = 16\ pags$

Calculemos ahora las páginas que necesitamos para cada región:

Texto: $pag\_texto = \frac{32768B}{4096B/pag} = 8\ pags$

Datos: $pag\_datos = \frac{16386B}{4096B/pag} = 4,0005 \approx 5\ pags$

Pila: $pag\_pila = \frac{15878B}{4096B/pag} = 3,88 \approx 4\ pags$

Total: $8 + 5 + 4 = 17\ pags > num\_paginas$

Como vemos, para direccionar el proceso descrito necesitamos más páginas de las que disponemos. **Cómo ubicarlo?????**

Debido al desperdicio de memoria en la última página de la sección de datos, la idea sería reducir el tamaño de página para reducir la cantidad de memoria desperdiciada por fragmentación.

Suponga que la tabla de páginas para el proceso actual se corresponde con la mostrada en la siguiente figura. Todos los números están en base decimal, la numeración comienza en todos los casos desde cero y todas las direcciones de memoria son direcciones en bytes. El tamaño de página es de 1024 bytes. ¿Qué direcciones físicas, si existen, corresponderán con cada una de las siguientes direcciones virtuales? (no intente manejar ninguna falta de página, si las hubiese):

n.º de página virtual	Bit de validez/presencia	Bit de referencia	Bit de modificación	n.º de marco de página
0	0	1	0	4
1	1	1	1	7
2	1	0	0	1
3	1	0	0	2
4	0	0	0	-
5	1	0	1	0

- $999)_d$

  Expresamos en binario: $999)_{10} = 11\ 1110\ 0111)_2$. Como las páginas son de $1024\ bytes$, necesitaremos $10\ bits$ para indicar el *offset* en cada dirección virtual, por lo que en este caso:

  - $0 =$ número de página virtual
  - $999 =$ *offset*

  Como el número de página virtual es 0 no se puede resolver la dirección física porque se produce una falta de página, ya que el bit de validez para la página nº 0 está a 0

- $2121)_{10}$

  $2121)_{10} = 10\ 00\ 0100\ 1001)_2$

  - $2 =$ número de página
  - $100\ 1001)_2 = 73)_{10} =$ *offset*

  Dirección física : $1 + 73 = 74)_{10}$

- $5400)_{10}$

$1\ 01\ 01\ 0001\ 1000)_2$

- $101)_2 = 5)_{10}$ = número de página
- $1\ 1000)_2 = 24)_{10}$ = *offset*

Dirección física: $0 + 24 = 24)_{10}$

Suponga la siguiente secuencia de referencias a páginas: 1,2,3,4,1,2,5,1,2,3,4,5. Calcule el número de faltas de página que se producen utilizando el algoritmo FIFO y considerando que el número de marcos de página asignados a nuestro proceso es:

- 3 marcos (Suponemos que ninguna de las páginas está cargada en un principio en memoria)

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

  Se producen **9 faltas de página**

- 4 marcos

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

  Se producen **10 faltas de página**

¿Se corresponde esto con el comportamiento intuitivo que esperaríamos ya que, al aumentar el tamaño de memoria de que disfruta el proceso, el número de faltas de página debería disminuir?

No se corresponde porque aumentan el número de faltas de página aún aumentando el tamaño de memoria del que dispone el proceso.

Suponga un sistema de memoria virtual paginada en donde cada proceso tiene asignado un número fijo de marcos de página. Suponga el escenario en el que existe un proceso con 7 páginas y tiene asignados 5 marcos de página. Indique el contenido de la memoria después de cada referencia a una página si el sistema utiliza como algoritmo de sustitución de página el LRU (la página referenciada hace más tiempo). La secuencia de referencias a

memoria es la que se muestra en la siguiente figura. Además, indique cuantas faltas de página se producen.

Marcos\Refs	2	1	3	4	1	5	6	4	5	7	4	2
1	-	**1**	1	1	1	1	1	1	1	**-**	-	-
2	**2**	2	2	2	2	2	**-**	-	-	-	-	**2**
3	-	-	**3**	3	3	3	3	3	3	3	3	**-**
4	-	-	-	**4**	4	4	4	4	4	4	4	4
5	-	-	-	-	-	**5**	5	5	5	5	5	5
6	-	-	-	-	-	-	**6**	6	6	6	6	6
7	-	-	-	-	-	-	-	-	-	**7**	7	7

Se producen **8 faltas de página**.

Un ordenador con soporte hardware para memoria virtual paginada presenta las siguientes características: Memoria RAM de 4KB, páginas de 1KB de tamaño y direcciones virtuales de 16 bits. El SO utiliza solamente el primer marco de página (marco 0) para alojar el kernel y los demás marcos están disponibles para su uso por los procesos que se ejecutan en el sistema. Suponga que tenemos sólo dos procesos, P1 y P2, y que utilizan las direcciones de memoria virtual y en el orden establecido en la siguiente figura. Responda a las siguiente cuestiones:

- **¿Cuántos marcos de página tiene la memoria RAM de este ordenador?**

  $4KB/1KB/marco = 4\ marcos$

- **¿Cuántos bits necesitamos para identificar los marcos de página?**

  $2\ bits$

- **Describa los fallos de página que tendrán lugar para cada intervalo de ejecución de los procesos, si la política de sustitución de páginas utilizada es LRU. Suponga que el algoritmo es de asignación variable y sustitución global.**

Proceso	Direcciones virtuales
P1	0-99
P2	0-500
P1	100-500
P2	501-1500
P1	3500-3700
P2	1501-2100
P1	501-600

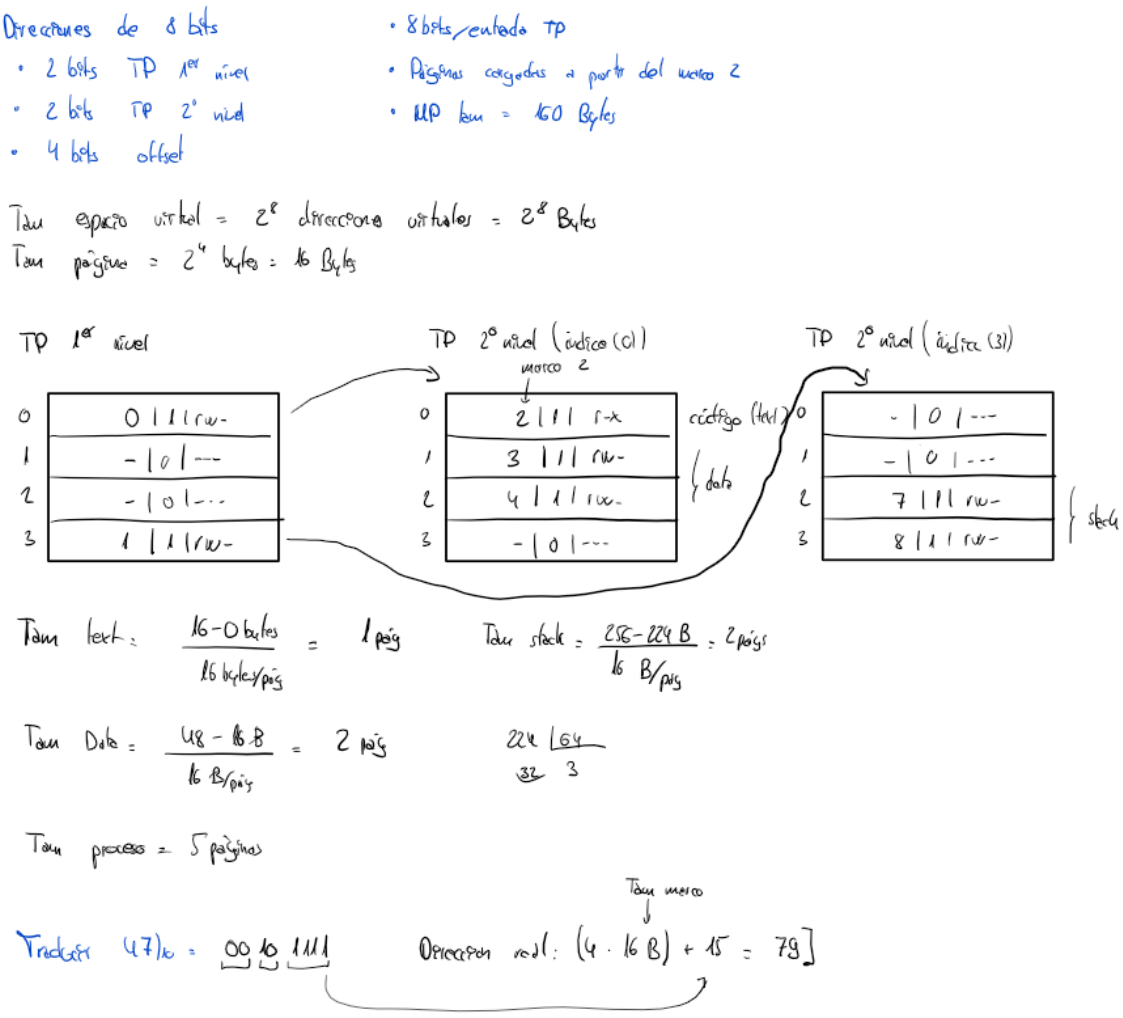
1. **P1**: Referencia a página 0 de P1 => **Falta** (se trae)
2. **P2**: Referencia a página 0 de P2 => **Falta** (se trae)
3. **P1**: Referencia a página 0 de P1 => Ya está en memoria (no falta)
4. **P2**: Referencia a página 0 de P2 => No falta. Referencia a página 1 de P2 => **Falta** (se trae)
5. **P1**: Referencia a página 3 de P1 => **Falta**. Se extrae P1.0 (hace más tiempo), y se trae P1.3

6. **P2**: referencia a página 1 de P2 => No falta. Referencia a página 2 de P2 => **Falta**. Se extrae P2.0 y se trae P2.2
7. **P1**: Referencia a página 0 de P1 => **Falta**. Se extrae P1.3 y se trae P1.0

Se producen **6 faltas de página**.

Problemas sobre paginación a dos niveles

Suponga un sistema que utiliza paginación a dos niveles. Las direcciones son de 8 bits con la siguiente estructura: 2 bits en la tabla de páginas de primer nivel, 2 bits en la tabla de páginas de segundo nivel y 4 bits para el desplazamiento). El espacio de direccionamiento virtual de un proceso tiene la estructura del dibujo. Represente gráficamente las tablas de páginas y sus contenidos, suponiendo que cada entrada de la tabla de páginas ocupa 8 bits y que todas las páginas están cargadas en memoria principal a partir del marco 2. La memoria principal tiene un tamaño de 160 Bytes. Dada la estructura de tabla de páginas realice la traducción de la dirección virtual 47.



Cuestiones sobre el conjunto residente y conjunto de trabajo

¿El tamaño del conjunto residente de un proceso depende directamente del tamaño del programa ejecutable asociado?

El tamaño del conjunto residente de un proceso depende únicamente del algoritmo de cálculo del conjunto residente de páginas, no del tamaño del programa ejecutable

¿El tamaño del conjunto de trabajo de un proceso depende directamente del tamaño del programa ejecutable asociado?

El tamaño del conjunto de trabajo de un proceso tampoco depende del tamaño del programa ejecutable asociado, sino de la cantidad de páginas que el proceso referencia en el intervalo de tiempo $(t-\tau, t)$ (*Working Set*=$W(t,\tau)$)

El tamaño del conjunto residente y el tamaño del conjunto de trabajo de un proceso... ¿es siempre el mismo durante toda la "vida" del proceso? Caso de no ser así, explique que escenarios provocan que el tamaño de dichos conjuntos varíe uno con respecto al otro.

El **tamaño del conjunto residente** dependerá del algoritmo que se utilice para calcularlo, pero generalmente no será el mismo durante toda la vida del proceso, ni para el Algoritmo del Conjunto de Trabajo (el conjunto de trabajo tampoco es siempre igual de grande), ni para el Algoritpo de Frecuencia de Falta de Página (depende de las referencias que haga el programa)

El **tamaño del conjunto de trabajo** tampoco es constante, pues depende de las referencias que haga el programa en el intervalo $(t-\tau, t)$. Si un programa en este intervalo hace referencia únicamente a una página, el tamaño será de 1, mientras que si hace referencia a varias páginas, el tamaño será mayor

Un SO que utiliza un sistema de memoria virtual paginada tiene un mecanismo lock/unlock para proteger marcos de página que funciona de la siguiente forma: lock_page(np), cuyo efecto es proteger contra la sustitución al marco de página en que se ubica la página virtual np; y unlock_page(np) cuyo efecto es suprimir esta protección sobre el marco.

- **¿Qué estructura/s de datos son necesarias para la realización de estos mecanismos?**

  Bastaría con añadir una columna a la Tabla de Páginas (preferiblemente en la estructura que gestione los marcos de página) en la que un bit nos informe de si determinada página se encuentra bloqueada o no, para que el gestor de memoria, a la hora de decidir qué página llevar a MA, revise previamente esa columna, para no llevarla en caso de que se bloquee

- **¿En qué caso puede ser de utilidad estas primitivas?**

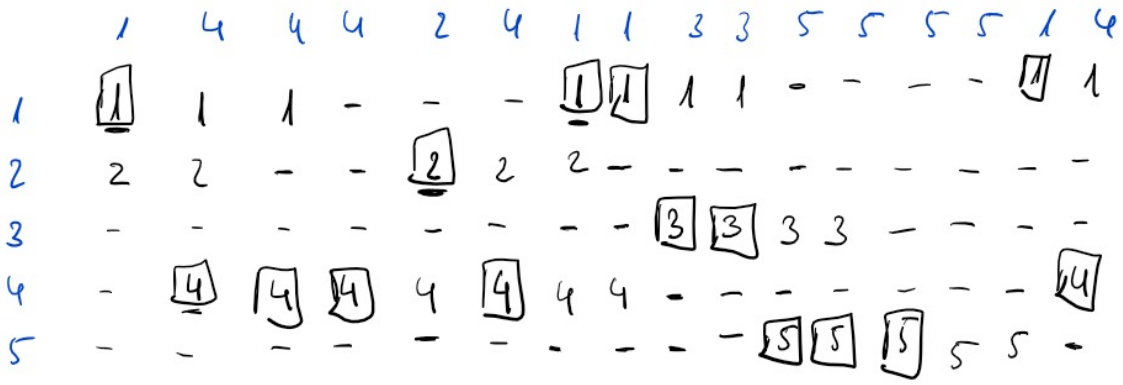
  Para bloquear las páginas que soportan los buffers de los cauces entre procesos por ejemplo

- **¿Qué riesgos presentan y qué restricciones deben aportarse a su empleo?**

  No se puede permitir que el sistema supere un límite de páginas bloqueadas en un momento determinado, pues se podría degradar el rendimiento

Problemas sobre el conjunto residente

Describa el funcionamiento del algoritmo de sustitución global basado en el modelo del conjunto de trabajo utilizando los siguientes datos: 4 marcos de página, en t=0 la memoria contiene la página 2 que se referenció en dicho instante de tiempo, el tamaño de la ventana es τ= 3 y se produce la secuencia de referencias a páginas: 1,4,4,4,2,4,1,1,3,3,5,5,5,5,1,4.



Cuestiones sobre Linux

Dudas

Analice qué puede ocurrir en un sistema que usa paginación por demanda si se recompila un programa mientras se está ejecutando. Proponga soluciones a los problemas que pueden surgir en dicha situación.

No importante

Si se recompila un programa mientras se está ejecutando ocurrirá lo siguiente:

- Las **páginas cargadas en MP** mantendrán los datos del programa anterior a la recompilación
- Las **páginas no cargadas en MP** solo están almacenadas en MA, por lo que serán sobreescritas.

Cuando se produzca una falta de página, se solicitará la lectura de una página de MA, con la información del programa recompilado, lo que puede dar lugar a errores al combinar la información (texto, datos, pila) recompilada que se toma de MA, con la información que se mantenía en MP de antes de recompilar

Soluciones?

Para cada uno de los siguientes campos de la tabla de páginas explique si es la MMU o el kernel del SO quién los lee y escribe (en caso de escritura indique si el bit se activa (1) o desactiva (0)), y en qué momentos:

Poco importante

- **Número de marco**: *kernel*
- **Bit de presencia**: *kernel*
- **Bit de protección**: *kernel*
- **Bit de modificación**: *kernel*
- **Bit de referencia**: *MMU*

Indique las ventajas y desventajas del algoritmo de frecuencia de faltas de página con respecto al algoritmo basado en el conjunto de trabajo con un tamaño de ventana τ y un parámetro τ.

El algoritmo de FFP carga menos el sistema porque solo se ejecuta cuando se produce una falta de página

Ejercicios 16 , 17, 19, 22