Sistemas Operativos
1° Parcial 2C2017 - TM - Resolución

Aclaración: La mayoría de las preguntas o ejercicios no tienen una única solución. Por lo tanto, si una solución particular no es similar a la expuesta aquí, no significa necesariamente que la misma sea incorrecta. Ante cualquier duda, consultar con el profesor del curso.

Teoría

1. La TLB permite ahorrar el acceso a memoria de la tabla de páginas ya que nos permite recuperar el frame en el que se encuentra una página a través de una caché. Cada vez que hay un acceso a memoria primero se busca la página en la misma. Cada vez que se carga una página en memoria o se reemplaza se debe actualizar la TLB (escribirla).

    En paginación jerárquica es muy importante ya que este esquema requiere más accesos a memoria para acceder a las TPs intermedias.

2. Ambas técnicas logran que nunca ocurra deadlock. Se diferencian en que prevención actúa sobre la petición, es decir, define una política de cómo pedir los recursos. Evasión por otro lado, deja que se pida de cualquier forma pero luego no asigna los recursos en caso que deje al sistema en estado seguro. Evasión tiene más overhead ya que debe correrse el algoritmo del banquero en cada petición.

    Por ejemplo:En un sistema muy específico como por ejemplo el sistema que maneje un marcapaso, no importa que la forma de realizar las peticiones sea flexible, por lo que se podría definir una política (prevención), y no querríamos tener mucho overhead para no ralentizar el sistema.

3. V o F

a) Falso, si no se encuentra el resultado, vuelve al proceso inmediatamente con un error. Es responsabilidad del desarrollador volver a realizar la llamada posteriormente
b) Verdadero, busca minimizar el tiempo en que tarda en posicionarse en el cilindro indicado, ordenando los pedidos de forma que el brazo del disco se mueva menos.

4. La gestión del conjunto residente está formado por el tamaño de conjunto y la sustitucion de paginas

	Reemplazo Local	***Reemplazo Global***
Asignacion Fija	La cantidad de frames asignados al proceso es fija. La página a reemplazar es elegida de entre los frames asignados al proceso.	No es posible
Asignacion Variable	La cantidad de frames asignados al proceso se modifica cada cierta cantidad de tiempo. La página a reemplazar es elegida de entre los frames.	La página a reemplazar es elegida de entre todos frames de memoria, sin importar que proceso produjo el fallo de página. Esto produce que el tamaño del Resident Set de los procesos varíe constántemente

5. Fragmentación interna: Las estrategias de paginación y buddy system poseen al asignar bloques de tamaños fijos, mayores a lo realmente requerido. En el caso de paginación es únicamente en la última página, en buddy system podría ser de casi medio buddy. Segmentación no presenta ya que asigna en forma dinámica.
Fragmentación externa: Las estrategias de buddy system y segmentación pueden presentar ya que asignan bloques de tamaño variable. Paginación asigna marcos de tamaño fijo, por lo que cualquiera puede ser utilizado.
Asignación contigua. Buddy system es la única que asigna todo el espacio de la imagen en forma contigua, el resto la divide en secciones (páginas o segmentos)Complejidad en la asignación: Paginación es el más sencillo, probablemente tenga un bitmap, cualquier frame libre puede ser usado. En segmentación debe encontrar espacios en los que entren los segmentos utilizando alguna estrategia. En buddy es un poco más fácil que en segmentación (no puede ser cualquier tamaño) aunque puede tener varias iteraciones hasta que se encuentra la partición de tamaño deseado (partiendo los buddies).

Interpretación dirección lógica:
Paginación -> NRO PAG | OFFSET
Segmentación -> NRO SEG | OFFSET
Buddy System -> OFFSET

Práctica

Ejercicio 1

a)
Recursos disponibles = 1 0 1

Necesidad

	R1	R2	R3
P1	2	1	0
P2	1	0	0
P3	0	0	1

Asigno -> disponibles -> 0 0 1
Ejecuto P3 -> 0 1 2
No puedo ejecutar ni P1 ni P2 -> estado inseguro -> no se asigna porque podría generar un deadlock en el futuro si todos piden su máximo

b) Por ejemplo
Asigno -> disponibles -> 0 0 1
Ejecuto P3 -> 0 1 2
P1 pide 1 R2 pero no pide otra instancia de R1 , finaliza y libera
P2 puede ejecutar

c)
Asignados -> 2 2 2
Veo que P1 puede llegar a pedirme 3 de R1, por lo que como mínimo deben haber 3
Totales iniciales -> 3 2 2
Disponibles 1 0 0
Asigno 1 R1
0 0 0
sumo 1 R1 para que pueda ejecutar P2
1 0 0
Ejecuto P2 -> 2 1 0
Ejecuto P1 ->4 1 1
Ejecuto P3 -> 4 2 2 -> totales mínimos

Ejercicio 2

1) DF = B31E Con Frame 11 hace referencia a la página 3 de la tabla , dado que B = 1011 en binario = al 11 en decimal . por lo cual de los 16 bits 4 bits son para el número de página y los 12 bits restantes para el desplazamiento de la página y tamaño.
   DL = 331E
2) Los algoritmos que utiliza son LRU y Clock Modificado. El Óptimo termina con las mismas páginas pero en distinto frame.

La justificación son las tablas a continuación:

Frame	Pág	11 L	12 L	10 L	11 E	3 E	5L
11	3 <--	11	11	11 <--	11	11	11
12	5	5 <--	12	12	12 <--	3	3
15	8	8	8 <--	10	10	10 <--	5

LRU (próxima víctima -> menor tiempo de referencia)

Frame	Pág	11 L	12 L	10 L	11 E	3 E	5L
11	3 <--	11	11	11 <--	11 <--	3	3
12	5	5 <--	12	12	12	12 <--	5
15	8	8	8 <--	10	10	10	10 <--

FIFO (uso ptr de próxima víctima)

Frame	Pág	11 L	12 L	10 L	11 E	3 E	5L
11	3 U<--	3 U	3 M <--	10 U	10 U	10	10 <--
12	5 M	11 U	11 U	11 U<--	11 U <--	3 UM	3 UM
15	8	8 <--	12 U	12 U	12 U	12 <--	5 U

Clock

Frame	Pág	11 L	12 L	10 L	11 E	3 E	5L
11	3 U<--	3 U <--	3	10 U	10 U	10	5 U
12	5 M	5 M	12 U	12 U<--	12 U <--	3 UM	3 UM<--
15	8	11 U	11 U <--	11 U	11 UM	11 M <--	11 M

Clock M

Ejercicio 3

sector 15400 / (32 * 4)-> cilindro 120
sector 13100/ (32*4) → cilindro 102
Viene subiendo

SCAN: 120| (hasta el tope) 11 - 11 -(hasta el inicio) - 12 - 100 - 110 - 129
Explicación
T = 0 -> atiende a 11 (único pedido) - va hasta el final del disco (cilindro 199) y cambia de sentido hasta el 100
se recorren (199 - 120) + (199 - 11) = 267 cilindros => 534 ms
Llegaron = 129 - 12 - 100 -11 , se ordenan -> 11 - 12 - 100 -129
Atiende a 11, baja hasta 0, cambia de sentido y atiende a 12.
Una vez que llega a 100, ya se recorrieron (11 + 100) cilindros -> 222 ms -> estaríamos en t = 756, ya llegó el último pedido. Entonces lo atendemos y por último a 129.

N-STEP-SCAN: 120 | (hasta el tope) 11 - (hasta inicio) 12 - 100 -129 -(hasta tope) - 110 - 11
Explicación
T = 0 -> atiende a 11 (único pedido) - va hasta el final del disco (cilindro 199) y cambia de sentido hasta el 100
se recorren (199 - 120) + (199 - 11) = 267 cilindros => 534 ms
Llegaron = 129 - 12 - 100 -11 - los primeros 3 quedan en una cola de atención
cola 1 - se ordenan ->12 - 100 -129
Baja hasta 0, cambia de sentido y atiende a 12.
Una vez que llega a 100, ya se recorrieron (11 + 100) ciliindros -> 222 ms -> estaríamos en t = 756, ya llegó el último pedido, se mete en la otra cola.

cola 2 - se ordena -> 110 - 11
Sube desde 129 hasta el tope, cambia de sentido, atiende a 110 y por último a 11

SSTF: 120| 11 - 12-129-100-11-110
Explicación
T = 0 -> atiende a 11 (único pedido)
se recorren (120 - 11) = 109 cilindros => 218 ms
Llegaron = 129 - 12 , se ordenan -> 12 - 129
Cuando termina de atender a 129 se recorrieron (129 -11) = 118 cilindros = 236 ms -> inst = 454
Llegan 100 y 11 -> se ordenan 100 y 11
Luego de atender a 100, (se recorren 29 cilindros -> inst 512) aún no llegó el último pedido, por lo que se atiende a 11 y por último a 110