PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC 2333 — Sistemas Operativos y Redes
Soluciones Interrogación 2

Lunes 04-Mayo-2015

Duración: 2 horas

1. **[20p]** Responda brevemente las siguientes preguntas. Si considera que falta información indíquelo claramente, o haga un supuesto razonable.

   1.1) **[2p]** Considere un espacio de direcciones lógico para 128 páginas, cada una con espacio para 1024 palabras, mapeadas a una memoria física de 32 frames.

   a) ¿Cuántos bits se requieren para la dirección lógica?
   R. $128 = 2^7$ páginas se pueden direccionar con 7 bit. Cada página tiene espacio para $1024 = 2^{10}$ palabras, por lo tanto el *offset* se describe con 10 bit. Una dirección lógica requiere 17 bit.

   b) ¿Cuántos bits se requieren para la dirección física?
   R. $32 = 2^5$ *frames* se direccionan con 5 bit. Sumado a un *offset* de 10 bit, hace que una dirección física requiere 15 bit.

   1.2) **[1p]** ¿En qué se diferencia el proceso de carga de una librería estática y de una librería dinámica?

   R. Una librería estática se compila como parte de un programa y es parte de la memoria privada de un proceso que se encuentra en memoria. Una librería dinámica, en cambio, se carga en un espacio de memoria separado del proceso y por esto puede ser compartida por diferentes procesos.

   1.3) **[4p]** Considere los métodos de asignación de memoria vistos en clase: asignación contigua, segmentación pura, y paginación pura. ¿Cómo se comportan cada uno de estos métodos en los siguientes aspectos?

   a) Fragmentación externa
   R. Asignación contigua y segmentación: producen fragmentación externa. Es válido mencionar que en el caso de la segmentación, la fragmentación puede ser menor. Paginación: no produce fragmentación externa.

   b) Fragmentación interna
   R. Asignación contigua y segmentación: no producen fragmentación interna. Paginación: sí produce.

   c) Capacidad para compartir código entre procesos
   R. Asignación contigua: es difícil pues requiere definir áreas de memoria compartida dentro de la memoria privada del proceso. Segmentación y paginación: es más fácil pues se pueden definir áreas bien delimitada como segmento o páginas como área compartida.

   d) Capacidad de aumentar la memoria asignada a un proceso
   R. Asignación contigua: es difícil pues requiere que la memoria contigua a la ya asignada se encuentre disponible, y esto no se puede garantizar. Segmentación: menos difícil pues el segmento nuevo puede asignarse en una porción de memoria no necesariamente contigua, sin embargo no está garantizada que un segmento se puede expandir por las mismas razones que el caso de asignación contigua. Paginación: fácil de conseguir, y es una de las ventajas de la paginación

   1.4) **[4p]** Considere un sistema con páginas de tamaño 2 KB. Para las siguientes direcciones de memoria, descrita en base decimal, determine el número de página y el offset

   R. En general, para un dirección d y tamaño de página P, es aceptable decir que el número de página es $p = \lfloor d/P \rfloor$, y que el *offset* es $o = d \mod P$.

   a) 2015
   R. Tamaño de página: 2048. Dirección: $2015 = 0 \times 2048 + 2015$. Página: 0. Offset: 2015.

b) 3049
R. Tamaño de página: 2048. Dirección: $3049 = 1 \times 2048 + 1001$. Página: 1. Offset: 1001.

c) 330104
R. Tamaño de página: 2048. Dirección: $330104 = 161 \times 2048 + 376$. Página: 161. Offset: 376.

d) 650000
R. Tamaño de página: 2048. Dirección: $650000 = 317 \times 2048 + 784$. Página: 317. Offset: 784.

1.5) **[3p]** Considere un sistema con direcciones virtuales de 21-bit, pero solo permite utilizar 16-bit para direccionar memoria física. El sistema utiliza página de 2KB.

a) ¿Cuántas entradas puede tener una tabla de páginas convencional? (un nivel)
R. Páginas de 2KB $= 2^{11}$ requieren 11 bit para *offset*. Quedan 10 bit para el número de página, por lo tanto pueden haber $2^{10} = 1024$ entradas en la tabla de páginas.

b) ¿Cuántas entradas puede tener una tabla de páginas invertida?
R. Si una dirección física ocupa 16 bit y hay 11 bit para *offset* entonces hay 5 bit para el número de *frame*. Una tabla de páginas invertida requere $2^5 = 32$ entradas.

c) ¿Cuánto es la máxima cantidad de memoria física que puede tener este sistema?
R. Si hay 16 bit para direcciones físicas, se pueden direccionar $2^{16} = 2^6 \times 2^{10} = 64$KB.

1.6) **[6p]** Considere un sistema de paginación con una tabla de página almacenada íntegramente en memoria principal. Una referencia a memoria toma 60ns.

a) ¿Cuánto toma una lectura de una dirección virtual utilizando la tabla de páginas?
R. Una lectura a una dirección virtual requiere 2 acceso a memoria (tabla de páginas, y dirección). Por lo tanto, $2 \times 60\text{ns} = 120\text{ns}$.

b) Considere un TLB con tiempo de acceso de 5ns, y una tasa de acierto de 75 %. ¿Cuánto es el tiempo de acceso promedio a una dirección virtual?
R. Una lectura con acierto (*hit*) de TLB requiere un acceso a TLB y un acceso a memoria: $5\text{ns}+60\text{ns} = 65\text{ns}$. Una lectura con falla (*miss*)de TLB requiere un acceso a TLB, un acceso a la tabla de páginas, y un acceso a memoria: $5\text{ns} + 2 \times 60\text{ns} = 125\text{ns}$. Un acceso promedio requiere: $0{,}75 \times 65\text{ns} + 0{,}25 \times 125\text{ns} = 80\text{ns}$

c) Considere un sistema con paginación doble, y tasa de acierto de TLB de 90 %. ¿Cuánto es el tiempo de acceso promedio a una dirección virtual?
R. Similarmente al caso anterior, pero ahora cada lectura de la tabla de páginas requiere dos accesos a memoria. Acceso con *hit* de TLB: $5\text{ns} + 60\text{ns} = 65\text{ns}$. Acceso con *miss* de TLB: $5\text{ns} + 3 \times 60\text{ns} = 185\text{ns}$. Acceso promedio: $0{,}9 \times 65\text{ns} + 0{,}1 \times 185\text{ns} = 77\text{ns}$

2. **[20p]** Responda brevemente las siguientes preguntas. Si considera que falta información indíquelo claramente, o haga un supuesto razonable.

2.1) **[2p]**Considere un espacio de direcciones de un proceso con f frames disponibles e inicialmente vacíos, y una secuencia de accesos a direcciones de memoria (reference string) de largo n.

a) ¿Cuál es la cantidad mínima de Page Faults que pueden ocurrir?
R. Si las n páginas indicadas por las direcciones del memoria del *reference string* son distintas, tendrá que ocurrir al menos una *page fault* para cada páginas distinta referenciada. Por lo tanto, la respuestas es n.
Si se elimina la restricción que todas sean distintas, es posible que las n direcciones se refieren a la misma página, y con eos el mínimo es 1.

b) ¿Cuál es la cantidad máxima de Page Faults que pueden ocurrir?
R. Si las n páginas indicadas por las direccions de memoria del *reference string* son distintas, y por lo tanto no se repiten, ocurrirán a los más n *page faults*.
Si las páginas pueden repetirse, un mal ordenamiento sería acceder consecutivamente a cada página distinta y repetir ei ciclo. De esta manera, se produce una *page fault* por cada acceso, y la respuesta es n.

2.2) **[9p]** Considere la siguiente secuencia de accesos a memoria (reference string).

5,4,6,2,4,5,4,1,5,2,5,4,5,2,8

¿Cuántas page faults ocurren para cada uno de los siguientes algoritmos?

a) LRU con 3, 4, y 5 frames disponibles

R. Con 3 frames disponibles, se producen 9 *page faults*.

Access	5	4	6	2	4	5	4	1	5	2	5	4	5	2	8
Frame 0	5	5	5	2	2	2	2	1	1	1	1	4	4	4	8
Frame 1	-	4	4	4	4	4	4	4	4	2	2	2	2	2	2
Frame 2	-	-	6	6	6	5	5	5	5	5	5	5	5	5	5
PFs	1	2	3	4	-	5	-	6	-	7	-	8	-	-	9

Con 4 frames disponibles, se producen 7 *page faults*.

Access	5	4	6	2	4	5	4	1	5	2	5	4	5	2	8
Frame 0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Frame 1	-	4	4	4	4	4	4	4	4	4	4	4	4	4	4
Frame 2	-	-	6	6	6	6	6	1	1	1	1	1	1	1	8
Frame 3	-	-	-	2	2	2	2	2	2	2	2	2	2	2	2
PFs	1	2	3	4	-	-	-	6	-	-	-	-	-	-	7

Con 5 frames disponibles, se producen 6 *page faults*.

Access	5	4	6	2	4	5	4	1	5	2	5	4	5	2	8
Frame 0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Frame 1	-	4	4	4	4	4	4	4	4	4	4	4	4	4	4
Frame 2	-	-	6	6	6	6	6	6	6	6	6	6	6	6	8
Frame 3	-	-	-	2	2	2	2	2	2	2	2	2	2	2	2
Frame 4	-	-	-	-	-	-	-	1	1	1	1	1	1	1	1
PFs	1	2	3	4	-	-	-	5	-	-	-	-	-	-	6

b) FIFO con 3, 4, y 5 frames disponibles

R. Con 3 frames disponibles, se producen 11 *page faults*.

Access	5	4	6	2	4	5	4	1	5	2	5	4	5	2	8
Frame 0	5	5	5	2	2	2	2	1	1	1	1	4	4	4	4
Frame 1	-	4	4	4	4	5	5	5	5	2	2	2	2	2	8
Frame 2	-	-	6	6	6	6	4	4	4	4	5	5	5	5	5
PFs	1	2	3	4	-	5	6	7	-	8	9	10	-	-	11

Con 4 frames disponibles, se producen 8 *page faults*.

Access	5	4	6	2	4	5	4	1	5	2	5	4	5	2	8
Frame 0	5	5	5	5	5	5	5	1	1	1	1	1	1	1	1
Frame 1	-	4	4	4	4	4	4	4	5	5	5	5	5	5	5
Frame 2	-	-	6	6	6	6	6	6	6	6	6	4	4	4	4
Frame 3	-	-	-	2	2	2	2	2	2	2	2	2	2	2	8
PFs	1	2	3	4	-	-	-	5	6	-	-	7	-	-	8

Con 5 frames disponibles, se producen 6 *page faults*.

Access	5	4	6	2	4	5	4	1	5	2	5	4	5	2	8
Frame 0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	8
Frame 1	-	4	4	4	4	4	4	4	4	4	4	4	4	4	4
Frame 2	-	-	6	6	6	6	6	6	6	6	6	6	6	6	6
Frame 3	-	-	-	2	2	2	2	2	2	2	2	2	2	2	2
Frame 4	-	-	-	-	-	-	-	1	1	1	1	1	1	1	1
PFs	1	2	3	4	-	-	-	5	-	-	-	-	-	-	6

c) Reemplazo óptimo con 3, 4, y 5 frames disponibles

R. Con 3 frames disponibles, se producen 7 *page faults*.

Access	5	4	6	2	4	5	4	1	5	2	5	4	5	2	8
Frame 0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	8
Frame 1	-	4	4	4	4	4	4	1	1	1	1	4	4	4	4
Frame 2	-	-	6	2	2	2	2	2	2	2	2	2	2	2	2
PFs	1	2	3	4	-	-	-	5	-	-	-	6	-	-	7

Con 4 frames disponibles, se producen 6 *page faults*.

Access	5	4	6	2	4	5	4	1	5	2	5	4	5	2	8
Frame 0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	8
Frame 1	-	4	4	4	4	4	4	4	4	4	4	4	4	4	4
Frame 2	-	-	6	6	6	6	6	1	1	1	1	1	1	1	1
Frame 3	-	-	-	2	2	2	2	2	2	2	2	2	2	2	2
PFs	1	2	3	4	-	-	-	5	-	-	-	-	-	-	6

Con 5 frames disponibles, se producen 6 *page faults*.

Access	5	4	6	2	4	5	4	1	5	2	5	4	5	2	8
Frame 0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	8
Frame 1	-	4	4	4	4	4	4	4	4	4	4	4	4	4	4
Frame 2	-	-	6	6	6	6	6	6	6	6	6	6	6	6	6
Frame 3	-	-	-	2	2	2	2	2	2	2	2	2	2	2	2
Frame 4	-	-	-	-	-	-	-	1	1	1	1	1	1	1	1
PFs	1	2	3	4	-	-	-	5	-	-	-	-	-	-	6

2.3) **[2p]** En el algoritmo de reemplazo de páginas de segunda oportunidad, suponga que es posible observar la tasa a la cual avanza el puntero del algoritmo (el puntero indica qué página será reemplazada).

a) ¿Qué se puede decir del sistema si el puntero avanza rápidamente?
R. Si el puntero avanza rápidamente significa que se está demorando en encontrar páginas para reemplazar, ya que cada vez que ocurre un *page fault*, debe avanzar varias posiciones antes de encontrar la página candidata. Esto ocurre cuando el proceso está haciendo referencia a muchas páginas distintas de su espacio de direcciones repetidamente.
Como ejemplo, en un programa en C, al recorrer una matriz grande por columnas se llegará rápidamente a una nueva página. En cambio al recorrer una matriz por filas, se está leyendo permanentemente memoria contigua.

b) ¿Qué se puede decir del sistema si el puntero avanza lentamente?
R. Si el puntero avanza lentamente, significa que cada vez que ocurre un *page fault*, al puntero le basta con moverse pocas posiciones (probablemente la siguiente página en la cola) para encontrar una página a la cual reemplazar. Esto ocurre cuando el proceso accede repetidamente a la misma página, o a un conjunto pequeño de páginas. El programa puede estar leyendo repetidamente memoria contigua y solo ocasionalmente lee algún valor almacenado en una página que no se encuentra en memoria.

2.4) **[4p]** ¿Es posible que el algoritmo LFU (Lest Frequently Used) genere menos faltas de página que el algoritmo LRU (Least Recently Used)? Describa un caso en que esto ocurre (o diga por qué no es posible), y describa otro caso en que ocurre lo contrario (o diga por qué no es posible).

R. LFU se comporta mejor cuando un grupo de páginas se referencian repetidamente durante ciertos intervalos separados por intervalos más cortos en que no son utilizadas. Durante estos intervalos cortos, las páginas dejan de utilizarse y LRU intentará reemplazarlas por los más recientes. LFU, en cambio mantendrá en memoria las que fueron accedidas repetidamente.

LFU se comporta peor cuando hay muchas páginas que se referencian repetidas veces al inicio de un proceso, acumulando alta cantidad de referencias, y luego dejan de usarse. Si las páginas más recientes no alcanzan a acumular una cantidad similar de referencias, serán escogidas por sobre las más antiguas aun cuando las más antiguas no se utilicen. En estos casos, LRU se comporta mejor.

2.5) **[3p]** Para calcular el *working set* de un proceso, se utiliza un parámetro δ (CORRECCIÓN: debía ser Δ).

a) ¿Qué rol cumple el parámetro δ para calcular el *working set* de un proceso?

R. Δ se utiliza para determinar la longitud de la ventana de tiempo utilizada para determinar el *working set*.

b) Describa cómo se comportaría un sistema si el parámetro δ se fija a un valor muy pequeño. ¿Qué ocurre con la frecuencia de *page fault*s?
R. Si el parámetro Δ es muy pequeño, el *working set* también será pequeño (pues no puede tener más páginas que Δ), y aumentará la cantidad de *page fault*s.

c) Describa cómo se comportaría un sistema si el parámetro δ se fija a un valor muy alto.
R. Si el parámetro Δ es muy grande, cada proceso tendrá un *working set* más grande, y demandará una mayor cantidad de *frames* en memoria. El tamaño del *working set* se ajustará mejor a lo que requiere cada proceso, pero es probable que la memoria se llene más rápidamente y producir *thrashing*.

3. **[20p]** Responda brevemente las siguientes preguntas. Si considera que falta información indíquelo claramente, o haga un supuesto razonable.

3.1) **[6p]** Considere la siguiente secuencia ordenada de solicitudes a cilindros del disco: 44, 32, 75, 6, 40, 12, 15. El disco posee 100 cilindros numerados de 0 a 99. La cabeza lectora se encuentra posicionada en el cilindro 20 y antes de ello venía moviéndose en dirección a los cilindros con mayor número. Cada movimiento de un cilindro a otro contiguo le toma a la cabeza lectora 5 milisegundos. Para los siguientes algoritmos de *scheduling* de disco, determine la cantidad de cilindros que se mueve la cabeza lectora y el tiempo total que demora en atender las solicitudes.

a) FCFS (*First-Come Fist Served*)
R. Desplazamiento: $(44-20)+(44-32)+(75-32)+(75-6)+(40-6)+(40-12)+(15-12) = 213$ cilindros. Tiempo: $213 \times 5 = 1065$ms.

b) SSTF (*Shortest Seek-Time First*)
R. Desplazamiento: $(20-15)+(15-12)+(12-6)+(32-6)+(40-32)+(44-40)+(75-44) = 83$ cilindros. Tiempo: $83 \times 5 = 415$ ms.

c) LOOK
R. Desplazamiento: $(75-20)+(75-6) = 124$ cilindors. Tiempo: $124 \times 5 = 620$ ms.

3.2) **[6p]** Compare los modelos RAID-0, RAID-1 y RAID 5 con N discos de capacidad T cada uno en cuanto a:

a) Cantidad máxima de discos que pueden fallar sin interrumpir el funcionamiento del sistema
R. En RAID-0, si falla un disco el sistema deja de funcionar y los datos almacenados se perderán (por lo tanto no debe fallar ningún disco). En RAID-1, pueden fallar hasta $N/2$ discos (suponiendo que sean todos del mismo *mirror*) y el sistema seguirá funcionando. En RAID-5, puede fallar hasta un disco y el sistema podrá funcionar.

b) Capacidad máxima de almacenamiento.
R. En RAID-0, la capacidad máxima de almacenamiento es NT. En RAID-1, la capacidad máxima de almacenamiento es $NT/2$. En RAID-5, la capacidad máxima de almacenamiento es $(N-1)T$.

c) Velocidad para escrituras
R. La mayor velocidad de escritura se da en RAID-0. En RAID-1 es menor pues se debe escribir en dos discos, y en RAID-5 es menor aún al tener que calcular paridad a través de los $N-1$ discos.

3.3) **[8p]** Considere un sistema de archivos con bloques de 1024 byte, punteros de bloque y de disco de 32-bit. Cada archivo está compuesto por 10 punteros directos a bloque de datos, 2 punteros de indirección simple, 2 punteros de indirección doble, y 1 puntero de indirección triple.

a) **[2p]** ¿Cuál es el tamaño máximo de disco que puede soportar este sistema de archivos?
R. Con punteros de 32 bit se pueden direccionar 2^{32} bloques. Cada bloque es de 2^{10} byte. El tamaño máximo de disco es $2^{32} \times 2^{10}$byte $= 2^{42}$byte $= 4$TB.

b) **[2p]** ¿Cuál es el tamaño máximo de archivo que se puede almacenar en este sistema de archivos?
R. Con 10 punteros directos se almacenan 10×1KB $= 10$KB. Un puntero utiliza 32bit $= 2^2$byte. En un bloque de 2^{10}byte se pueden almacenar $2^{10}/2^2 = 2^8 = 256$ punteros. Un bloque de indirección

simple direcciona $2^8 \times 2^{10} = 2^{18} = 256$KB. Un bloque de indirección doble direcciona $2^8 \times 2^8 \times 2^{10} = 2^26 = 64$MB. Un bloque de indirección triple direcciona $2^8 \times 2^8 \times 2^8 \times 2^{10} = 2^{34} = 16$GB.
El tamaño máximo de archivo es 10KB $+ 512$KB $+ 128$MB $+ 16$GB.

c) **[2p]** ¿Cuánto espacio en disco ocupa un archivo de 5KB?
R. Un archivo de 5KB necesita 1 bloque principal y 5 bloques de datos, por lo tanto ocupa 6KB en disco.

d) **[1p]** ¿Cuánto espacio en disco ocupa un archivo de 1MB?
R. Un archivo de 1MB necesita $2^{20}/2^{10} = 2^{10}$ bloques de datos. Para direccionarlos necesita 1 bloque principal, 2 bloques de indirección simple (512 KB), 1 bloque de indirección doble del cual se utilizan 2 bloques de indirección simple para alcanzar 512 KB más. Esto significa: $1 + 2 + 1 + 2 = 6$ bloques para almacenar punteros.
Por lo tanto, son $2^{10} + 6 = 1030$ bloques de disco, que suman $2^{10} \times (2^{10} + 6) = 1$MB $+ 6$KB.

e) **[1p]** ¿Cuánto espacio en disco ocupa un archivo de 2GB?
R. Un archivo de 2GB necesita $2^{31}/2^{10} = 2^{21}$ bloques de datos. Para direccionarlos necesita 1 bloque principal, 2 bloques de indirección simple (512 KB), 2 bloques de indirección doble que además agregan 512 bloques de indirección simple (128 MB). Un bloque de indirección triple. Faltan 1024MB $- 128$MB $= 896$MB. Cada bloque de indirección doble agrega 64MB, por lo tanto faltan 14 bloques de indirección doble. Los 13 primeros bloques dobles utilizan todos sus bloques simples, pero al número 14 hay que quitarle 2 bloques simples (por los dos bloques simple direccionados desde el bloque principal. En suma, los bloques de punteros son: $1+2+2+2\times256+1+14+13\times256+1\times254 = 4114$ bloques para almacenar punteros. Es válido dejarlo expresado si los cálculos están bien hechos. Por lo tanto son $2^{31} + 2^{10} \times (4114) = 2^{31} + 2^{10} \times (2^{12} + 18) = 2$GB $+ 4$MB $+ 18$KB en disco. Es válido dejarlo expresado si los cálculos son consistentes.