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

IIC 2333 — Sistemas Operativos y Redes — 2/2015
Soluciones Interrogación 2

Martes 29-Septiembre-2015

Duración: 2 horas, 15 minutos

1. **[20p]** Responda las siguientes preguntas:

   1.1) **[8p]** Un proceso de tiempo real, tiene ciertas restricciones. Entre ellas, los procesos deben terminar después de un plazo, de lo contrario es una falta grave del sistema. Considere un sistema que mata a los procesos que no alcanzan a ejecutar antes de su plazo, y un cola con 4 procesos p_0, p_1, p_2, p_3. Cada proceso tiene un plazo $d_0 = 4$, $d_1 = 10$, $d_2 = 8$, $d_3 = 12$, y ráfagas de CPU $b_0 = 5$, $b_1 = 8$, $b_2 = 2$, $b_3 = 5$. Esto significa que cada proceso p_i, al recibir su turno, ejecuta durante b_i ms y luego vuelve a la cola. Si un proceso pasa más de d_i ms (estricto) esperando en la cola, se considera que ha fallado y el sistema lo saca de la tabla de procesos.

   a) Indique cuáles procesos alcanzan a ejecutar luego de haber tenido dos turnos de CPU cada uno, con el algoritmo FCFS (orden p_0, p_1, p_2, p_3).
   R. p_0 ejecuta en $t = [0, 5]$. Tiempo en cola: $p_1 = 5, p_2 = 5, p_3 = 5$
   p_1 ejecuta en $t = [5, 13]$. Tiempo en cola: $p_2 = 13, p_3 = 13, p_0 = 8$. p_0 sobrepasa su tiempo de espera y es eliminado. p_2 sobrepasa su tiempo de espera y es eliminado. p_3 sobrepasa su tiempo en espera y es eliminado.
   p_1 ejecuta en $t = [13, 21]$. No hay más procesos en cola.
   Solo p_1 alcanza a ejecutar dos veces.

   b) Indique cuáles procesos alcanzan a ejecutar luego de dos turnos de CPU cada uno con el algoritmo Round-Robin y *quantum* $q = 3$
   R. p_0 ejecuta en $t = [0, 3]$. Tiempo en cola $p_1 = 3, p_2 = 3, p_3 = 3$.
   p_1 ejecuta en $t = [3, 6]$. Tiempo en cola $p_2 = 6, p_3 = 6, p_0 = 3$.
   p_2 ejecuta en $t = [6, 8]$. Tiempo en cola $p_3 = 8, p_0 = 5, p_1 = 5$. p_0 sobrepasa su tiempo de espera y es eliminado.
   p_3 ejecuta en $t = [8, 11]$. Tiempo en cola $p_1 = 8, p_2 = 3$.
   p_1 ejecuta en $t = [11, 14]$. Tiempo en cola $p_2 = 6, p_3 = 11$.
   p_2 ejecuta en $t = [14, 16]$. Tiempo en cola $p_3 = 13, p_1 = 10$. p_3 sobrepasa su tiempo de espera y es eliminado.
   p_1 ejecuta en $t = [16, 18]$. Tiempo en cola $p_2 = 2$.
   p_1 y p_2 alcanzan a ejecutar dos veces.

   c) Indique cuáles procesos alcanzan a ejecutar con un algoritmo de *scheduling* que escoge siempre el proceso que está más cercano a cumplir su plazo.
   R. p_0 ejecuta en $t = [0, 5]$. Tiempo en cola $p_1 = 5, p_2 = 5, p_3 = 5$.
   p_2 ejecuta en $t = [5, 7]$. Tiempo en cola $p_0 = 2, p_1 = 7, p_3 = 7$.
   p_0 ejecuta en $t = [7, 12]$. Tiempo en cola $p_1 = 12, p_2 = 5, p_3 = 12$. p_1 sobrepasa su tiempo de espera y es eliminado.
   p_3 ejecuta en $t = [12, 17]$. Tiempo en cola $p_0 = 5, p_2 = 10$. p_0 sobrepasa su tiempo de espera y es eliminado. p_2 sobrepasa tu tiempo de espera y es eliminado.
   p_3 ejecuta en $t = [17, 22]$.
   p_3 alcanzan a ejecutar.

   1.2) **[6p]** Considere un sistema donde se ejecuta un $p\,\%$ de procesos del tipo interactivo (alto tiempo de E/S) y un $q\,\%$ de procesos normales (alto tiempo de CPU).

a) Proponga un sistema de *scheduling* para este sistema que permita que los procesos interactivos sean servidos repetidamente y no provoque mucha contención para los procesos de cómputo.
R. Colas con prioridad. Alta prioridad para primera cola, con bajo *quantum*. Si son interactivos, se bloquerán repetidamete. Segunda cola para procesos normales. Solo se ejecutará de aquí cuando la primera cola esté vacía.

b) Justifique cómo se comportaría su sistema para $p >> q$
R. Cola *round-robin*, con bajo *quantum*.

c) Justifique cómo se comportaría su sistema para $q << p$
R. Cola *round-robin* con expropiación.

1.3) **[4p]** Considere un sistema con m recursos del mismo tipo, y n procesos. Un proceso puede solicitar o liberar solamente un recurso a la vez. Demuestre que si se cumplen las siguientes dos condiciones, este sistema no puede entrar en *deadlock*:

a) La cantidad máxima de recursos que un proceso necesita varía entre 1 y m.

b) La suma de las necesidades máximas de todos los procesos es menor a $m + n$.

R. El objetivo es demostrar que siempre hay al menos un proceso que tenga todos los recursos que necesita. Si un proceso p requiere m recursos, entonces los tendrá. De los restantes procesos, a lo más $n - 1$ pueden estar solicitando recursos, y todos estarán esperando por p. Cada uno de ellos puede pedir a lo más un recursos. Cuando p libere los que tiene, habrán m disponibles. Si $m \geq n$, no hay problema. Si $m < n$, cuando se liberen los m a lo más $n - m$ procesos quedarán con sus recursos asignados y podrán continuar.

1.4) **[2p]** Indique y justifique brevemente si el siguiente código constituye o no una solución al problema de la sección crítica para 2 procesos (no es necesario una demostración completa).

```
1              shared boolean flag[2];
2              flag[0]=flag[1]=false;
3              void proc(int i){
4                while(true){
5                  while(flag[(i+1) mod 2]==true);
6                  flag[i]=true;
7                  // seccion critica
8                  flag[i]=false;
9                }
10             }
```

R. No hay atomicidad entre la ejecución de las líneas 5 y 6. P_0 puede ver que `flag[1]==false` y salir del `while`. Luego es el turno de P_1 y éste ve que `flag[0]==false` y ejecuta directamente hasta la sección crítica. Cuando sea el turno de P_0, la condición del `while` será cierta, pero P_0 ya salió del `while` y entra a la sección crítica junto con P_1.

2. **[20p]** Responda las siguientes preguntas:

2.1) **[4p]** Considere un espacio de direcciones lógico de 128 páginas, cada una con 1024 palabras de 32-bit, y una memoria física de 32 *frames*.

a) Indique cuántos bit se necesitan para la dirección lógica
R.. Para direccionar $1024 = 2^{10}$ posiciones, se necesitan 10 bit. Para direccionar $128 = 2^7$ páginas se necesitan 7 bit. La dirección lógica requiere 17 bit. Dado que la palabra direccionable es de 32 bit, no es necesario multiplicar 1024×4 para obtener el tamaño de la página. Si usan $7 + 12 = 19$ bit, en lugar de 17, reciben sólo 1 pto.

b) Indique cuántos bit se necesitan para la dirección física
R. Como son $32 = 2^5$ *frames*, se necesitan $5 + 10 = 15$ bit. Si usan $5 + 12 = 17$ bit, reciben sólo 1 pto.

2.2) **[10p]** Considere un esquema de direccionamiento de dos niveles. Ambas direcciones, física y lógica son de 32 bit. La dirección lógica utiliza 10 bit para la primera página, y 10 bit para la segunda página. Cada entrada de un tabla de páginas tiene 20 bit para el número de frame, y adicionamente utiliza *dirty bit*, *valid bit*, y un *reference byte* para aproximar LRU usando los últimos 8 accesos a memoria.

a) Indique el tamaño en byte de una página
R. 32 bit para la dirección lógica. 20 se ocupan para las páginas (de ambos niveles), y quedan 12 para el *offset*. Con 12 bit se pueden direccionar $2^{12} = 4096$ byte. La página es de $4KB$.

b) Dibuje la estructura de la tabla de páginas, indican los byte necesarios en cada nivel
R. Primera tabla de página. 2^{10} entradas de 30 bit. (20 bit para el *frame* más 10 bit de control). Segunda tabla de página. $2^10 \times 2^10$ entradas de 30 bit.

c) Indique el tamaño de la tabla de páginas en el primer nivel
R. Primera tabla de página. 2^{10} entradas. Cada entrada de la tabla de páginas usa 20 bit para el número de *frame*, y además 10 bit de control (dirty, valid, y 8 bit para el reference), por lo tanto cada entrada es de 30 bit. Primera tabla de página necesita $2^10 \times 30$ bit $= 2^7 \times 30$ byte $= 3840$ byte. En particular, cabe dentro de una página.

d) Indique la principal ventaja de utilizar un esquema de dos niveles, en lugar de usar direcciones virtuales con 20 bit para el número de página.
R. Ventaja es que la primera tabla de págians cabe dentro de una página. Se necesitan solo dos páginas de memoria para poder efectuar un direccionamiento. Si usamos 20 bit para el número de página, tabla de se hace muy grande y necesitaríamos varias páginas de memoria ($\frac{2^20}{4096}$) para efectuar un direccionamiento.

e) Indique el tamaño máximo de memoria que se puede direccionar con este esquema.
R. Las direcciones son de 32 bit. El tamaño máximo es de $2^32 = 4$GB. Esto no depende de la cantidad de niveles, sino de la cantidad de bit en la dirección física.

2.3) **[6p]** Considere páginas que almacenan 100 entradas, y la siguiente lista de accesos a direcciones virtuales de memoria: $10, 11, 104, 170, 73, 504, 309, 185, 245, 246, 434, 458, 364, 327, 650$

Determine la cantidad de *page faults* que se producen para los siguientes algoritmos de reemplazo de páginas, y 3 frames disponible en memoria física. Suponga que los *frames* están inicialmente vacíos, por lo tanto los primeros accesos siempre provocarán *page faults*.

R. La secuencia de acceso es: $0, 0, 1, 1, 0, 5, 3, 1, 2, 2, 4, 4, 3, 3, 6$. Se pide solo la cantidad de page faults, pero se muestra la secuencia también para mostrar el cálculo.

a) FIFO
R. 7 page fault

#F	0	0	1	1	0	5	3	1	2	2	4	4	3	3	6
0	0	0	0	0	0	0	3	3	3	3	3	3	3	3	6
1	-	-	1	1	1	1	1	1	2	2	2	2	2	2	2
2	-	-	-	-	-	5	5	5	5	5	4	4	4	4	4
	F		F			F	F		F		F				F

b) LRU
R. 9 page fault

#F	0	0	1	1	0	5	3	1	2	2	4	4	3	3	6
0	0	0	0	0	0	0	0	1	1	1	1	1	3	3	3
1	-	-	1	1	1	1	3	3	3	3	4	4	4	4	4
2	-	-	-	-	-	5	5	5	2	2	2	2	2	2	6
	F		F			F	F	F	F		F		F		F

c) ¿De qué tamaño es el *working set* del proceso con $\Delta = 3$?
R. 2

3. **[20p]** Responda las siguientes preguntas:

3.1) **[6p]** Considere un sistema con 6 discos duros, donde los discos fallan con una probabilidad p. Se requiere integridad de los datos ante fallas de los discos, y alta velocidad de escritura. Proponga y argumente una configuración de RAID adecuada. En su descripción incluya la probabilidad de pérdida de datos en función de p y cuántas fallas de disco puede soportar su sistema sin dejar de funcionar. **Solución:**

a) Es posible un número de respuestas. Se propone el nivel **RAID50**, puesto que RAID5 soporta fallas de más de un disco siempre que correspondan a grupos distintos sin pérdida de datos ni degradación notoria de rendimiento y RAID0 permite acelerar el rendimiento de las escrituras y aprovechar al máximo el espacio de los discos. Esta configuración emplea 3 discos en un grupo RAID5 y los dos grupos formados se unen por RAID0. La velocidad de escritura se ve mejorada puesto que es posible escribir varios stripes al mismo tiempo entre los grupos RAID5. En este caso se toleran hasta 2 discos en falla sin pérdida de información. Note que no se consideran los cálculos de paridad puesto que suponen una carga constante para el sistema y se puede asumir sin pérdida de generalidad que se cuenta con una controladora RAID que se encarga de ello.

b) También es admisible **RAID10**, dado que provee tolerancia a fallas y buena utilización de los discos, aunque no provee mejoras de rendimiento de escrituras significativas, más que las entregadas por RAID0. En este caso se formarán 3 grupos RAID1 y se podrán aguantar hasta 3 discos en falla, 1 por grupo respectivamente.

c) En general se evaluará la respuesta en función de los argumentos entregados, por lo que sólo mencionar que RAID50 o RAID10 son solución no otorgará puntaje.

3.2) **[10p]** Considere un sistema de archivos que usa asignación indexada. Cada puntero a disco utiliza 32-bit (4 byte). Para un bloque de disco de tamaño B *bytes*, determine el tamaño máximo de archivo que puede ser almacenado si el bloque índice utiliza 20 punteros directos a bloques de datos, 1 puntero de indirección simple, 1 puntero de indirección doble, y 1 puntero de indirección triple. **Solución:** $20 * B + B^2/4 + B^3/4^2 + B^4/4^3 bytes$

3.3) **[4p]** Considere las siguientes operaciones de creaciones de *links*, en las cuales la sintaxis es:

`ln [-s] destino_link nombre_link`.

El flag `-s` es opcional y su presencia indica que se trata de un *link* simbólico.

```
1 ln -s int1.tex i1-2015.tex
2 ln -s int2.tex i2-2015.tex
3 ln -s tarea2.pdf t2-2015.pdf
4 ln examen.pdf ex-2015.pdf
5 mv examen.pdf examen2015.pdf
6 mv i2-2015.tex i3-2015.tex
7 mv tarea2.pdf tarea3.pdf
8 rm int1.tex
```

Indique qué archivos quedan disponibles en el directorio y el estado de los *links* después de estas operaciones. Si considera que alguna operación produce error, siga con las siguientes.

Solución:

- ex-2015.pdf
- examen2015.pdf
- i1-2015.tex → int1.tex (roto)
- i3-2015.tex → int2.tex (corrrecto)
- int2.tex
- t2-2015.pdf → tarea2.pdf (roto)
- tarea3.pdf