Sistemas Operativos

I. T. Informática de Sistemas

Córdoba, 11 de febrero de 2003

Cuestiones (3 puntos)

1. En UNIX un proceso en modo núcleo no puede ser apropiado, ¿puede este hecho permitir que un proceso en modo núcleo acapare el procesador de forma indefinida?

2. La instrucción TSL (*Test and set lock*) está presente en muchos sistemas de cómputo. Su formato es `boolean TSL(boolean b)`. TSL actúa sobre una variable lógica fijándola a TRUE y devolviendo el valor antes de su modificación. Ambas operaciones se ejecutan de manera indivisible. En seudocódigo sería:

```
Boolean TSL (Boolean lock) {
  Boolean old;

  old = lock;
  lock = TRUE;
  return old;
}
```

   Indique como se puede implementar la exclusión mutua para dos procesos utilizando la instrucción TSL. ¿La implementación indicada, tiene espera activa?

3. Considera la siguiente modificación del algoritmo de Peterson para 2 procesos para que pueda implementar la exclusión mutua de N procesos:

```
int Turno = 0;
BOOLEAN QuiereEntrar[N] = {FALSE, ..., FALSE};

void Proceso (int i) {
  extern BOOLEAN QuiereEntrar[N];
  extern int Turno;
  int k;

  while (TRUE) {                                                    10
      QuiereEntrar[i] = TRUE;
      Turno = (i+1) % N;
      for (k=0; k < N; k++)
        while (QuiereEntrar[k] && Turno == k);

      /* Seccion critica. */

      QuiereEntrar[i] = FALSE;

      /* Seccion residual. */                                       20
  }
      }
```

   Indique si la solución propuesta es una solución correcta al problema de exclusión mutua con N procesos. En caso contrario proponga, si es posible, las modificaciones necesarias para que lo sea.

4. La mayoría de los sistemas implementan las llamadas al sistema mediante interrupciones, ¿cuál es la principal razón de ello?

5. Supóngase un sistema informático que planifica el procesador mediante un algoritmo Round-Robin puro, ¿qué inconvenientes presentaría para el funcionamiento eficiente del sistema?

6. Suponga que ha de administrar un sistema informático, ¿en qué situaciones consideraría la implementación de un sistema de paginación anticipada para mejorar el rendimiento del sistema?

Problemas (7 puntos)

1. Supóngase un proceso en UNIX que realiza las operaciones que se indican a continuación:

   a) El proceso es creado en un instante en el cual la carga del sistema es normal. Posteriormente es planificado para ejecutar por primera vez.

   b) El proceso abre un fichero binario para lectura y lee una matriz desde dicho fichero.

   c) El proceso realiza la inversión de la matriz leída. Para completar este proceso, que es muy largo, se necesitan varios cuantos de CPU.

   d) El proceso realiza el producto de la matriz invertida por un vector. Este vector está contenido completamente en una página pero esta página no se encuentra en memoria real.

   e) Mientras el proceso ejecuta en modo usuario, otro proceso realiza un intento de escritura en una posición de memoria sobre la que no tiene acceso.

   f) El proceso crea un proceso hijo.

   g) El proceso finaliza mediante una llamada `exit()`.

   Realice los siguientes ejercicios:

   a) Indique cuál o cuáles de los pasos anteriores no son posibles en un sistema UNIX monoprocesador.

   b) Indique por los estados que pasaría el proceso a medida que va realizando los pasos anteriores (sin tener en cuenta los señalados como imposibles en el apartado anterior).

   c) Indique cómo se verían afectadas las estructuras de información del sistema, especialmente el PCB, a medida que se realizan las acciones siguientes.

2. *Problema del lavabo unisex.* Una oficina tiene en su quinta planta un único lavabo que es unisex. Para el funcionamiento correcto de este lavabo los empleados han de cumplir las siguientes reglas:

   a) En el lavabo no puede haber hombres y mujeres a la vez.

   b) La capacidad máxima es de 3 personas.

   Programe los procesos `Hombre()` y `Mujer()` para que se garanticen estas dos condiciones en el acceso al lavabo. Evite el interbloqueo y, si es posible, la inanición.

   AYUDA: El problema se asemeja en parte a uno de los problemas clásicos de programación concurrente estudiados.

3. Supóngase un sistema UNIX en el que ejecutan 3 procesos con la siguiente secuencia de sucesos a lo largo de su ejecución.

Proceso	Estado	Duración (ms)	Modo	Prioridad
1	Finaliza llamada a `fork()`	10	Núcleo	20
	Ejecuta en modo usuario	40	Usuario	-
	Interrupción por fallo de página	10	Núcleo	2
	Espera por fallo de página	10	-	–
	Completa interrupción	10	Núcleo	1
	Llamada a `exit()`	25	Núcleo	24
2	Finaliza llamada a `fork()`	10	Núcleo	20
	Ejecuta en modo usuario	100	Usuario	-
	Llamada al sistema	10	Núcleo	18
	Espera final de E/S	40	-	-
	Completa llamada al sistema	20	Núcleo	8
	Llamada a `exit()`	25	Núcleo	24
3	Finaliza llamada a `fork()`	10	Núcleo	20
	Ejecuta en modo usuario	50	Usuario	-
	Llamada al sistema	10	Núcleo	10
	Espera final E/S	40	-	-
	Completa llamada al sistema	20	Núcleo	8
	Llamada a `exit()`	25	Núcleo	24

Considere el algoritmo de planificación estándar del sistema operativo UNIX con un cuanto $q = 50$ ms y una prioridad base $P_b = 60$ ms. Considere que un proceso del sistema ejecuta siempre su ráfaga completa sin ser apropiado y que un proceso en modo usuario ejecuta siempre el cuanto completo, excepto cuando la ráfaga de CPU que ha de ejecutar termina antes de agotar el cuanto.

Realice la planificación de los tres procesos anteriores, teniendo en cuenta que el proceso 1 llega al sistema en el instante 0, el proceso 2 en el instante 1 y el proceso 3 en el instante 2. Indique el diagrama de Gant y el tiempo medio de espera. Considere que el tiempo de planificación y cambio de contexto son despreciables.

4. Sea la siguiente secuencia de referencias a memoria, de lectura y escritura, y una memoria real de 4 marcos de página y una memoria virtual de 8 páginas:

```
2 1 3 2 4 5 2 3 4 7 6 1 1 6 5 4 0 1 2 7
E E L L L E L E E L L E L L E E L E L E
```

Supóngase que modificamos el algoritmo de reemplazo de reloj añadiéndole un bit de modificación, que se pone a 1 cuando la página es accedida para escritura, además del bit de referencia usual. A continuación el algoritmo funciona de la siguiente forma:

a) Comenzando en la cabeza actual de la cola circular se recorre toda la cola sin modificar ningún bit, ni el de modificado ni el de reemplazo. El primer marco que tenga ambos bits a 0 es reemplazado.

b) Si en el paso anterior ninguna página cumplió el criterio, se realiza una segunda pasada hasta encontrar una página con el bit de referencia a 0. En esta segunda pasada los bits de referencia se van poniendo a 0 a medida que se pasan las páginas.

c) Si el paso anterior también falló, todas las páginas tendrán su bit de referencia a 0. Se repite entonces el paso 1, y el 2 si fuera necesario, que en esta segunda ocasión obtendrán una página a reemplazar con seguridad.

Indique el número de fallos de página si se utilizara este algoritmo en la secuencia de referencias de memoria anterior. En cada referencia indique el estado de la memoria con un gráfico como el de la figura:

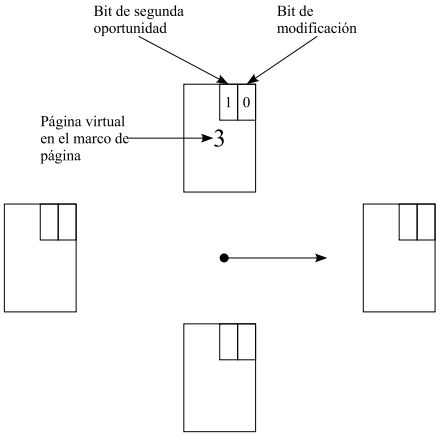

Figura 1: Gráfico para representar el estado de la memoria.

NOTA: Justifique siempre todas las respuestas.

Tiempo de realización: **4 horas**.