

Sistemas Operativos I

2 Control - 10 de Mayo de 2002

Modelo 111

TEORÍA (5 puntos)

Se proponen 10 preguntas tipo test. Cada pregunta tiene cuatro opciones, de las cuales sólo una es válida. Cada pregunta acertada valdrá 0.5 puntos. Cada fallo restará un tercio del valor asignado a la pregunta. Una pregunta no contestada valdrá 0 puntos.

- Referido al método de asignación de memoria llamado de los “asociados” o “colegas”, dí cual de las siguientes afirmaciones es cierta:
 - Elimina la fragmentación externa, al proporcionar un método eficiente para compactar todos los huecos existentes en la memoria en un único hueco grande.
 - Elimina la fragmentación interna, al proporcionar huecos de todos los tamaños posibles
 - Permite fusionar cualquier par de huecos contiguos de tamaño 2^{n-1} en un nuevo hueco de tamaño 2^n
 - Proporciona un método eficiente de fusionar pares de huecos de tamaño 2^{n-1} en huecos de tamaño 2^n cuando las direcciones de comienzo de los dos huecos (en binario) difieren únicamente en el bit de posición $(n-1)$.
- Sobre los mecanismos de compactación, di cual de las siguientes afirmaciones es falsa:
 - Son aplicables en sistemas que tengan vinculación de direcciones en tiempo de carga
 - Reducen el problema de la fragmentación externa en sistemas con asignación dinámica de memoria.
 - No se pueden aplicar en sistemas con asignación dinámica por particiones fijas
 - Son muy caros desde el punto de vista computacional, por lo que no siempre es práctico su uso.
- Sean tres procesos P_0 , P_1 , P_2 , con los siguientes tiempos de carga y solicitudes de ráfagas de tiempo de CPU y de operaciones de Entrada/Salida (en unidades de tiempo):

Proceso	Carga	CPU	E/S	CPU	E/S	CPU
P_0	$t=0$	4 ut	2 ut	4 ut	--	--
P_1	$t=2$	3 ut	3 ut	1 ut	--	--
P_2	$t=3$	1 ut	2 ut	2 ut	2 ut	2ut

Sea el siguiente diagrama de Gantt de la ejecución de los procesos,



En el diagrama, cada recuadro significa una unidad de tiempo, un recuadro negro (sólido) indica una unidad de tiempo de ejecución, un recuadro con rallo inclinado indica una unidad de tiempo en la que

el proceso está en espera en la cola de procesos listos y un recuadro con rallado horizontal indica una unidad de tiempo dedicada a una operación de Entrada Salida.

Di cual de las siguientes políticas de planificación se ha aplicado:

- a) Por tiempo de llegada (FCFS)
- b) Por el trabajo más corto (SJF)
- c) Por menor tiempo restante (SRT)
- d) Circular (Round Robin) con un cuanto de tiempo de CPU de 2 unidades de tiempo

Nota: Para los cálculos de tiempos restantes o solicitudes de trabajo, se debe considerar el tiempo correspondiente a la próxima ráfaga de CPU solicitada (o la que se está ejecutando).

4. Sean dos procesos A y B. Los dos procesos comparten una zona de memoria compartida compuesta por las siguientes variables:

/* Compartido por los procesos */

```
int X=0;
semaforo Y=0;
semaforo Z=10;
semaforo M=1;
```

El código de lo procesos es el siguiente:

Proceso A	Proceso B
<pre>main() { int i=30; while(i) { down(M); down(Y); X--; if(X==9) up(Z); --i; if(i==0) printf("%d\t",X); up(M); } }</pre>	<pre>main() { int j=15; while(j) { down(M); down(Z); X++; if(X==1) up(Y); --j; if(j==0) printf("%d\t",X); up(M); } }</pre>

Suponiendo que los dos procesos se ejecutan de manera concurrente, di cuál será el valor impreso en la salida estandar del sistema.

- a) -15
- b) Un número entre 1 y 10
- c) Dos números cualesquiera entre 1 y 10.
- d) 10 0

5. Referido a los sistemas operativos que soportan Hilos (Threads), ¿cuál de las siguientes afirmaciones es falsa?

- a) Los hilos a nivel de usuario permiten que distintos procesos implementen distintas políticas de planificación de los hilos que los componen.
- b) Los hilos a nivel del núcleo permiten que un bloqueo de E/S de uno de los hilos del proceso no bloquee a los demás hilos del proceso

- c) Los hilos a nivel de núcleo se implementan mediante llamadas a funciones de librerías de hilo, y por lo tanto son independientes del sistema operativo sobre el que se ejecutan
- d) Los beneficios de los hilos a nivel de usuario no pueden aprovecharse en sistemas multiprocesador.

6. ¿Cuál de las siguientes afirmaciones sobre los semáforos es falsa?:

- a) Causan pérdidas de tiempo debido a esperas ocupadas.
- b) Permiten realizar la sincronización de procesos.
- c) Pueden implementarse mediante paso de mensajes.
- d) Las operaciones up y down son operaciones atómicas.

7. ¿Cuál de las siguientes afirmaciones es falsa con respecto a la predicción de interbloqueos?

- a) Para poder predecir bloqueos se debe conocer la demanda de recursos por anticipado
- b) El número total de procesos y total de recursos debe de ser fijo
- c) Supone que los procesos no pueden finalizar mientras mantengan recursos apropiados.
- d) El orden de ejecución de los procesos debe de estar forzado por condiciones de sincronización.

8. ¿Cuál de las siguientes afirmaciones es falsa relativa a la asignación de memoria mediante particiones dinámicas?

- a) Las particiones asignadas a los procesos son variables en longitud pero no en número.
- b) Asigna a los procesos exactamente la memoria que necesiten
- c) Produce fragmentación externa.
- d) Si hay fragmentación, se debe usar la compactación para desplazar los procesos que estén contiguos, de forma que toda la memoria libre quede junta en un bloque.

9. ¿Cuál de las siguientes afirmaciones es falsa en un sistema de memoria virtual paginada?

- a) No hay fragmentación interna
- b) Supone vinculación de direcciones en tiempo de ejecución
- c) Gran espacio virtual para el proceso con soporte de protección y compartición.
- d) No hay fragmentación externa

10. De los siguientes algoritmos de ubicación de páginas ¿cuál es el que produce mayor fragmentación externa?

- a) Primer ajuste
- b) Mejor ajuste.
- c) Siguiente en ajustarse
- d) Peor ajuste

Sistemas Operativos I

2 Control - 10 de Mayo de 2002

1

Nombre:

Apellidos:

Problema (5 ptos)

Sea un bar de dudosa reputación, que sirve bebidas alcoholicas sin permiso del ayuntamiento. La barra del bar está atendida por tres camareros, que realizan, a petición de los clientes, mezclas de seis bebidas (Whisky, Ginebra, Cognac, Anís del Mono, Cola-Cola Light, y Zumo de Piña).

Los clientes, piden su consumición, que será una combinación cualquiera de **dos de las seis bebidas**, a un camarero libre, si lo hay. Si todos los camareros están ocupados, esperan a que se libere uno de los tres. El camarero, una vez recibida la petición, coge las botellas correspondientes y hace la mezcla. Mientras tanto, si otro camarero recibe una nueva petición que requiere el uso de una de las botellas que se está utilizando, espera pacientemente a que el otro camarero termine de hacer su mezcla.

Se supone que el máximo número de clientes distintos que se pueden atender en un día está limitado a 100, aunque cada cliente puede pedir tantas combinaciones –gratis- como quiera.

La policía, que sospecha que en el bar se distribuye alcohol sin licencia, puede entrar en cualquier momento en el bar. Cuando así ocurre, los tres camareros sirven a todo el mundo Coca-Cola Light con Zumo de Piña, independientemente de lo que hayan pedido.

Escribe el código C **comentado** de tres procesos (Camarero, Cliente, Policía) que ejecuten las funciones descritas. Para la sincronización de los procesos, utiliza variables compartidas y semáforos.

No hace falta dar los detalles de la definición y creación de la memoria compartida ni de los semáforos. Utiliza un **tipo** genérico **semáforo** para las variables tipo semáforo, con dos operaciones elementales **up** y **down**.

```

/* Memoria compartida */
#define WHISKY      1
#define GINEBRA    2
#define COGNAC     3
#define ANIS       4
#define COCA       5
#define ZUMO       6
#define NUMERO_CLIENTES 100
semaforo camarero_libre = 0;
semaforo pedido = 0;
semaforo servir=0;
semaforo cliente[NUMERO_CLIENTES]={0,0,0,...,0};
semaforo mutex = 1;
semaforo mutex_policia = 1;
semaforo bebidas[6]={1,1,1,1,1,1};
int cuenta_clientes =0;
int policia=0;

```

/* Proceso Cliente */	/* Proceso Camarero */
<pre> main() { int numero_cliente; down(&mutex); cuenta_clientes++; if(cuenta_clientes>100) { up(&mutex); exit(); } else { numero_cliente = cuenta_clientes-1; up(&mutex) } while(1) { down(&camarero_libre); pide_consumicion(numero_cliente, bebida1, bebida2); up(&pedido); down(&cliente[numero_cliente]); down(&servir); bebe(); } } </pre>	<pre> main() { int numero_cliente, bebida1, bebida2, real1, real2; while(1) { up(&camarero_libre); down(&pedido); recibe_pedido(&numero_cliente, &bebida1, &bebida2); down(&mutex_policia); if(policia) { real1= COCA; real2 = ZUMO; } else { real1= bebida1; real2 = bebida2; } up(&mutex_policia); if(real1<real2) { down(&bebida[real1]); down(&bebida[real2]); } else { down(&bebida[real2]); down(&bebida[real1]); } up(&cliente[numero_cliente]); sirve(mezcla(real1,real2)); up(&bebida[real1]); up(&bebida[real2]); up(&servir); } } </pre>

/* Proceso Policia */	
<pre>main() { while(1) { patrulla(); down(&mutex_policia); policia=1; up(&mutex_policia); comprueba_bebidas(); down(&mutex_policia); policia=0; up(&mutex_policia); } }</pre>	