Examen Diciembre 2008

Lea detenidamente las siguientes instrucciones. No cumplir los requerimientos puede implicar la pérdida del examen.

Formato

- **Indique su nombre completo y número de cédula en cada hoja (No se corregirán las hojas sin nombre, sin excepciones) Numere todas las hojas e indique la cantidad total de hojas que entrega en la primera.**
- **Escriba las hojas de un solo lado y empiece cada problema en una hoja nueva. (No se corregirá la hoja que tenga el ejercicio compartido, sin excepciones)**
- **Si se entregan varias versiones de un problema solo se corregirá el primero de ellos.**

Dudas

- **Sólo se contestarán dudas de letra.**
- **No se aceptarán dudas en los últimos 30 minutos del examen.**

Material

- **El examen es SIN material (no puede utilizarse ningún apunte, libro ni calculadora). Sólo puede tenerse las hojas del examen, lápiz, goma y lapicera en su banco. Todas sus demás pertenencias debe colocarlas en el piso debajo de su asiento.**

Aprobación

- **Para aprobar el examen se debe tener un ejercicio entero bien hecho y medio más.**

Finalización

- **El examen dura 4 horas.**

Problema 1

1. Describa la taxonomía de Flynn en cuanto a los sistemas paralelos.

2. ¿ Qué permite el modo dual de ejecución que brindan ciertos MicroProcesadores ?

3. En un sistema de tiempo compartido que características se le pedirían a un planificador de CPU.

4. Dada las características de los procesadores actuales (multi-cores) que ventaja presenta un sistema operativo tenga soporte de hilos (threads) a nivel del sistema operativo frente a otro que no lo tenga.

5. Describa el modelo de hilos (threads) MxN (Many-To-Many).

6. Sea un sistema monoprocesador que utiliza un planificador de tipo round-roubin con quantum = 2.
Si se tiene la siguiente secuencia de ejecución de instrucciones para los procesos P0, P1 y P2:

P0: C C C C E E C
P1: C E C C C
P2: E C C C C

donde C representa un cómputo en una unidad de tiempo y E una operación de Entrada/Salida que tarda 3 unidades de tiempo (1 ejecución y 2 de espera).
A su vez, P0 está listo para ejecutar en el tiempo 0, P1 en el tiempo 1 y P2 en el tiempo 2.

Dibuje como será la distribución del recurso procesador en una línea del tiempo. A su vez, mencione en que estado está cada instante del tiempo.

Nota:
- Asuma que el cambio de contexto es despreciable.
- La cola de procesos listos es FIFO.

7. Describa los tipo asociación de direcciones (address binding) que conoce.

8. Sea un sistema con 4 marcos y la siguiente secuencia de accesos a páginas por parte de un proceso:

1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2

a. ¿ Cuántos fallos de página ocurrirán si el algoritmo de reemplazo a utilizar es el óptimo ?
b. ¿ Cuántos fallos de página ocurrirán si el algoritmo de reemplazo a utilizar es el LRU ?

9. ¿ Por qué razón se tienden a utilizar más de un nivel de indirección en la traducción de direcciones de paginación?

10. En una arquitectura de 16 bits con páginación y marcos (frames) de 4096 bytes. Proponga un sistema de traducción de memoria virtual a física.

11. Describa para que es útil un servicio de buffering a nivel del subsistema de entrada/salida.

```
Solución:
Ver libro teórico y/o notas del curso
```

Problema 2

Se desea modelar con semáforos las tareas VENDEDOR, CLIENTE y SUPERVISOR de una cafetería. Esta cafetería dispone de una caja, 8 vendedores y 2 supervisores.
Los vendedores toman el pedido, cobran y elaboran el pedido de cada cliente. Tienen orden de no dejar la caja sola, por lo que para poder ir a elaborar el pedido debe haber otro vendedor en el área de la caja. Por razones de espacio no puede haber más de 2 vendedores en el área de caja a la vez, de los cuales solo uno de ellos puede estar atendiendo. Los vendedores solo toman pedidos cuando están en la caja. Los vendedores deben atender a los clientes lo antes posible, no se debe hacer esperar a un cliente si la caja esta libre.
Los vendedores le avisarán al grupo de supervisores cada 10 pedidos cobrados por ese vendedor. Los supervisores estarán esperando ser avisados para llenar la planilla. El primer supervisor libre recibirá el número de vendedor y con ese dato llenar la planilla.

Se dispone de los siguientes procedimientos:

- recibir_pedido(), cobrar_pedido(), elaborar_pedido(). Ejecutados por el vendedor
- enviar_pedido_pagar_y_recibir_pedido(). Ejecutado por el cliente, retorna cuando el vendedor termina de elaborar el pedido.
- llenar_planilla(int nro_vendedor). Ejecutado por el supervisor actualiza la planilla de ventas.

Aclaraciones:

- No se permite la implementación de tareas auxiliares
- Todos los procedimientos disponibles pueden demorar tiempos variables.

Solución:

```
semaphore sem_cli, sem_pedido, mutex_vend, mutex_caja, caja_cuidada,
espero_caja, area_caja, mutex_sup, sup_disp;

int cant_cajeros_en_area, num_vend=0, vend_sup;


void cliente () {
      P(sem_cli);
      V(sem_pedido);
      enviar_pagar_y_recibir_pedido();
}


void vendedor () {
      int cant_pedidos = 0, num;

      P(mutex_vend);
      num = num_vend++;
      V(mutex_vend);

      P(area_caja);
      P(mutex_caja);
      cant_cajeros_en_area++;
      if (cant_cajeros_en_area == 2) {
            V(mutex_caja);
            //Hay 2, uno puede seguir
            V(caja_cuidada);
            P(espero_caja);
      }
      else {
            V(mutex_caja);
      }
      //Estoy pronto para atender
      V(sem_cli);
      P(sem_pedido);
      recibir_pedido();
      P(caja_cuidada);
      V(espero_caja);
      P(mutex_caja);
      cant_cajeros_en_area--;
      V(mutex_caja);
```

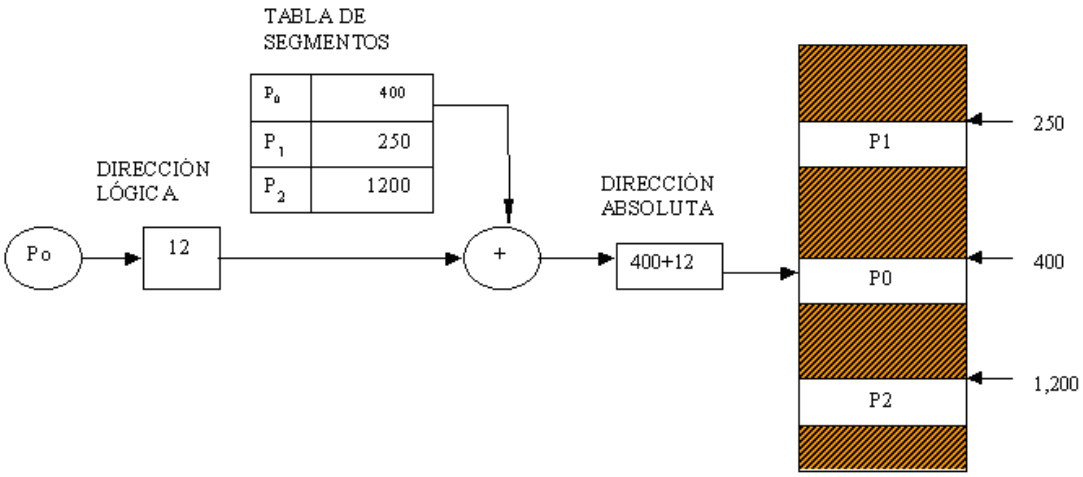
```
      V(area_caja);
      elaborar_pedido();
      cobrar_pedido();
      if(++cant_pedidos == 10) {
            cant_pedidos = 0;
            //AVISAR AL SUPERVISOR
            P(mutex_sup);
            vend_sup = num;
            V(sup_disp);
      }
}

void supervisor {
      int vendeder;
      while(true){
            P(sup_disp);
            vendedor = vend_sup;
            V(mutex_sup);
            llenar_planilla(vendedor);
      }
}

void main () {
      init(sem_cli,0);
      init(sem_pedido, 0);
      init(mutex_vend, 1);
      init(mutex_caja, 1);
      init(mutex_sup, 1);
      init(caja_cuidada, 0);
      init(espero_caja, 0);
      init(area_caja, 2);
      init(sup_disp, 0);
}
```

Problema 3

Considere un sistema operativo que utiliza segmentación como mecanismo de administración de memoria. Este mecanismo se basa en la implementación típica, la cual utiliza descriptores de memoria para organizar los diferentes segmentos.



Las direcciones virtuales se expresan como una **pareja (S,D)** siendo **S el número de descriptor** en el diccionario de segmentos y **D el desplazamiento del operando dentro del mismo**. Los descriptores describen la colección de segmentos que integran el espacio virtual mediante su dirección de comienzo y largo.

Se pide:

a) Implementar la función **Liberar (S)**, describiendo claramente el proceso a realizar en ocasión de liberar un segmento de memoria.
b) Describa en detalle el algoritmo **Reorganizar** () a ejecutar cuando la fragmentación de la memoria es elevada.
c) Indique como hacer para que dos procesos compartan un mismo segmento. Su solución, tiene algún impacto en el punto anterior?

Notas y aclaraciones:

- La implementación de las funciones o la descripción de los algoritmos puede hacerse utilizando pseudocódigo
- En caso de necesitar funciones auxiliares (por ejemplo, una función para ordenar una lista), puede aclarar el nombre de la función, los parámetros y el tipo de retorno sin necesidad de implementar la misma. Esto solo es válido para funciones que no resuelvan en su totalidad el problema.
- No se aceptaran respuestas o resultados que carezcan de la justificación apropiada

Solución:

a) Utilizando la propia memoria libre, ésta se describe mediante una lista doblemente encadenada, donde para cada sección contigua de memoria física libre se dispone la siguiente estructura:

- Libre-Ocupado (*)
- Tamaño (*)
- &Anterior
- &Siguiente
- … espacio libre
- Tamaño (*)

Los elementos indicados con (*) están presentes también en los segmentos ocupados.

Al Liberar (S) se realizan los siguientes pasos:

- Se extrae del descriptor S la dirección del segmento.
- Se examina el segmento físicamente anterior. Si esta libre entonces se colapsa con el anterior, simplemente actualizando su tamaño.
- Finalmente se examina el segmento físicamente siguiente. Si esta libre, también se debe colapsar con el anterior. Para ello, se lo da de baja de la lista encadenada de libres y nuevamente se actualiza el tamaño del segmento anterior.
- Se actualiza la indicación libre y tamaño dispuesto al final del segmento.

b) Ocasionalmente, ante una elevada presencia de muchas secciones de largo menor que algún K, podrá disponerse la realización de una reorganización de la memoria, corriendo segmentos en uso cualesquiera de lugar, permitiendo colapsar áreas libres que no eran contiguas. Para ello, todo segmento ocupado tendrá además de lo ya indicado la dirección del descriptor que lo apunta mediante una pareja (P,S). Al cambiar un segmento ocupado de lugar se deberá:

- Inhibir el uso del descriptor original (Lock)
- Copiar su contenido de un lugar a otro
- Actualizar la dirección del descriptor original
- Liberar el descriptor original
- Considerar que el descriptor original, debe ser inhibido ya que en un multiprocesador, otro CPU en forma concurrente podría indexarlo para acceder a su contenido, cuando éste se encuentra en proceso de ser corrido de lugar con la consiguiente falla.

c) Si dos procesos comparten un área hay 2 opciones:

- Que ambos posean un descriptor. En éste caso, al reorganizar, como sólo se puede ubicar uno de ellos, habría que recorrer todos los diccionarios buscando si hay una copia del descriptor padre original, para corregirlo. Esto no es una muy buena solución.

- Que el segundo proceso incorpore una referencia al descriptor original, de forma de evitar el descriptor copia y el mencionado problema en ocasión de la reorganización