Examen Febrero de 2014

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 y cada parte del problema de teórico en una hoja nueva.**
- **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 mínimo de 60 puntos.**

Finalización

- **El examen dura 4 horas.**
- **Al momento de finalizar el examen no se podrá escribir absolutamente nada en las hojas, debiéndose parar e ir a la fila de entrega. Identificar cada una de las hojas con nombre, cédula y numeración forma parte de la duración del examen.**

Problema 1 (32 puntos)

1. Describa dos mecanismos implementados a nivel de hardware para que proteger al sistema operativo.
2. Explíque el fenómeno de hiperpaginación (*Trashing*) e indique un método para detectarlo.
3. Enumere y describa brevemente tres métodos de planificación de disco.
4. Describa tres servicios que debe brindar un Sistema Operativo.
5. Describa el nivel 5 de RAID.
6. Describa y compare los métodos de Entrada/Salida Programada y mediante acceso directo a memoria (*DMA*).
7.
    i. ¿Qué entiende por sistema de archivos virtual (VFS - Virtual File System)?
    ii. Indique dos métodos utilizados para administrar el espacio libre de un sistema de archivos
8.
    i. Describa la anomalía de *Belady.*
    ii. Describa el algoritmo de reemplazo de páginas Óptimo y justifique si es posible llevarlo a la práctica.

Problema 2 (33 puntos)

Sea un sistema operativo que utiliza un gestor de memoria, el cual implementa memoria virtual utilizando un modelo de paginación bajo demanda. Las direcciones virtuales son de 60 bits y la traducción se realiza a través de 5 niveles de tabla de página. El tamaño de los marcos (*frames*) de la memoria es de 8KB (2^{13} bytes) y se utilizan 120 bits (15 bytes) para identificar a cada uno de ellos.

Notas:

- La tabla correspondiente a el primer nivel utiliza cuatro páginas y las mismas siempre están en memoria.
- Las tablas de página del segundo, tercer, cuarto y quinto nivel tienen el mismo tamaño.
- El sistema utiliza un byte de control para cada entrada de la tabla de paginas.
- En las tablas de página se almacena los identificadores de marcos y la información de control.
- Las direcciones a partir de la dirección virtual 0 se utilizan para almacenar el área de código, el área de datos globales y el área de memoria dinámica (*heap*).
- El resto de las direcciones se destinan al espacio de pila (*stack*), que se almacena comenzando en la dirección virtual más alta y creciendo hacia las direcciones bajas.

Se pide (justifique cada respuesta):

1. Determine cuantos bytes son utilizados para el offset y para cada uno de los niveles de la tabla de pagina.
2. Asumiendo que tenemos un proceso que requiere de 200MB (8KB * 25600) para almacenar su código, datos globales en memoria y datos dinámicos, y que la memoria requerida por su pila es de 40MB (8KB * 5120). Determinar cuanta memoria es requerida por la tabla de paginas para almacenar toda la información de este proceso.

Solucions:

Parte A)

Por definición en tamaño de los marcos el igual al tamaño de las páginas. Por ende el tamaño de las páginas es de 8KB, como debemos direccionar a cada uno de los bytes debemos direccionar a 2^{13}bytes, entonces precisamos 13 bits para el offset.

Dado que cada marco se identifica utilizando 15bytes (120bits) y en cada entrada de la tabla de página se almacenan el identificador de marco junto a un byte de control, cada entrada de la tabla de páginas tendrá un tamaño de 16bytes. Por este motivo cada pagina podrá almacenar:

8KB/16B => $2^{13}/2^4 = 2^9$ entradas.

Dado que la tabla de primer nivel ocupa 4 páginas en memoria la cantidad de entradas que se pueden direccionar en la tabla de primer nivel son:

$4 * 2^9 = 2^2 * 2^9 = 2^{11}$ entradas.

Por tal motivo se requiere de 11bits para direccionar a cada una de las entradas de la tabla de primer nivel.

Entonces para direccionar en la tablas de segundo, tercer, cuarto y quinto nivel tenoms 60bits – 13bits – 11bits = 36bits. Dado que estas tablas tiene el mismo tamaño la cantidad de bits utilizados para direccionar en cada una de estas tablas es:

36bits/4 = 9bits.

Resumiendo, se requieren de:

- **11bits para direccionar la tabla de primer nivel.**
- **9bits para direccionar la tabla de segundo nivel.**
- **9bits para direccionar la tabla de tercer nivel.**
- **9bits para direccionar la tabla de cuarto nivel.**
- **9bits para direccionar la tabla de quinto nivel.**
- **13bits para el desplacamiento (offset).**

Parte b)

Dado que el proceso utiliza 25600 páginas para almacenar su código, datos globales y datos dinámicos en memoria, se requieren de 25600 entradas en la tabla de paginas de quinto nivel para poder direccionar cada uno de estas páginas. Cada página de quinto nivel puede almacenar 2^9 (512) entradas. Entonces se requieren de 25600/512 = 50 páginas de quinto nivel.

Como se requiere de 50 páginas de quinto nivel, el la tabla de cuarto nivel se requiere de 50 entradas, las cuales son almacenadas en unas sola pagina. Dado que la tabla de página de cuarto nivel se de una página se requiere tan solo de una entrada en la tabla de tercer nivel y por ende de una página para almacenarla. Siguiendo el mismo razonamiento de requiere de una sola página de segundo nivel.

De forma análoga realizaremos el calculo de la cantidad de paginas de la tabla de paginas son requeridas para poder direccionar a completamente la pila. La pila ocupa 5120 páginas en memoria por ende se requieren de 5120 entradas en la tabla de páginas de quinto nivel. Entonces la cantidad de páginas requeridas son 5120/512 = 10 páginas. Como ya vimos para direccionar esta cantidad de páginas se requiere de una pagina de cuarto nivel, una de tercer nivel y una de segundo nivel.

Dado que la página de primer nivel siempre esta completamente en memoria y que el tamaño de la misma es de 4 páginas, tenemos que, se requiere de 70 (4 + 1 + 1 + 1 + 50 + 1 + 1 + 1 + 10) páginas para para direccionar toda la información del proceso.

Estas 70 paginas equivalen a 560KB (70 * 8KB) de memoria.

Problema 3 (35 puntos)

Sea una estación de servicio con un tanque de combustible que abastece 5 surtidores y un tanque de combustible de reserva.

Los clientes se auto-abastecen combustible previo pago del mismo en el surtidor correspondiente mediante tarjeta de crédito. Si no hay suficiente combustible en el tanque de abastecimiento el cliente debe avisar al jefe de pista que lo rellene desde el tanque de reserva, salvo que este ya haya sido avisado. El cliente esperará hasta tanto el jefe de pista termine de rellenar el tanque de combustible para poder cargar.

El tanque solo debe ser rellenado si no hay clientes abasteciéndose de combustible. Los clientes que ya pagaron tendrán prioridad sobre los nuevos que llegan para cargar combustible luego del rellenado.

El jefe de pista pasará a descansar luego de rellenar. Se supone que el tanque luego de rellenado es capaz de abastecer al menos a 5 autos

Se dispone de las siguientes funciones:

- Pagar(): boolean

  Ejecutada por los clientes y retorna si hay suficiente combustible en el tanque de abastecimiento.

- Cargar()

  Es ejecutado por los clientes para cargar combustible.

- Rellenar()

  Esta función es ejecutada por el jefe de pista para rellenar los tanques

Se pide:

Implementar los procedimientos cliente y jefe de pista utilizando semáforos.

Solución:

```
semaforo mutex;
semaforo estacion;
semaforo cliente_espera;
semaforo cliente_nuevo;
semaforo mutex_clientes_esperando_rellenar;
semaforo mutex_nuevos;
semaforo surtidores_ocupados;

integer clientes_esperando_rellenar;
integer clientes_estacion;
integer nuevos;
boolean entran_nuevos;

main() {
      clientes_esperando_rellenar = 0;
      cliente_estacion = 0;
      nuevos = 0;
      entran_nuevos = true;

      init(mutex, 1);
      init(estacion, 1);
      init(cliente_espera, 0);
      init(cliente_nuevo, 0);
      init(mutex_clientes_esperando_rellenar, 1);
      init(mutex_nuevos, 1);
      init(surtidores_ocupados, 5);

      cobegin
            jefe();
            cliente();
            cliente();
            cliente();
            …
            cliente();
      coend
}

procedure cliente() {
      P(surtidores_ocupados);
      boolean hay_combustible = Pagar();
```

```
P(mutex_nuevos);
if(!entran_nuevos) {
     nuevos++;
     V(mutex_nuevos);

     P(cliente_nuevo);

     P(mutex_nuevos);
     nuevos--;
     if(nuevos == 0) {
          entran_nuevos = true;
     }
     else {
          V(cliente_nuevo);
     }
     V(mutex_nuevos);
}
else {
     V(mutex_nuevos);
}

if(!hay_combustible) {
     P(mutex_clientes_esperando_rellenar);
     if(clientes_esperando_rellenar == 0) {
          V(aviso_rellenar);
     }
     clientes_esperando_rellenar++;
     V(mutex_clientes_esperando_rellenar);

     P(cliente_espera);

     P(mutex_clientes_esperando_rellenar);
     clientes_esperando_rellenar--;
     if(clientes_esperando_rellenar > 0) {
          V(cliente_espera);
     }
     V(mutex_clientes_esperando_rellenar);
}

P(mutex);
if(clientes_estacion == 0) {
     P(estacion);
```

```
      }
      clientes_estacion++;
      V(mutex);

      Cargar();

      P(mutex_nuevos);
      if(nuevos > 0) {
            V(cliente_nuevo);
      }
      V(mutex_nuevos);

      P(mutex);
      clientes_estacion--;
      if(clientes_estacion == 0) {
            V(estacion);
      }
      V(mutex);

      V(surtidores_ocupados);
}

procedure jefe() {
    while (true) {
          P(aviso_rellenar);
          P(estación);

          P(mutex_nuevos);
          entran_nuevos = false;
          V(mutex_nuevos);

          Rellenar();
          V(estacion);
          V(cliente_espera);
    }
}
```