Examen Julio 2007

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

Justifique brevemente cada respuesta.

1. ¿Cuál es el beneficio de la multiprogramación?
2. ¿Describa ventajas/desventajas del diseño de un sistema monolíticos frente al sistema en capas?
3. ¿Cómo representa el sistema operativo el concepto de proceso?
4. Clasifique el tipo de planificador (expropiativos o no expropiativos) a utilizar en cuanto a sistemas por lotes, interactivos y de tiempo real.
5.
    a. ¿Qué entiende por direccionamiento virtual de un proceso?
    b. ¿A través de qué técnicas se logra implementar un sistema de memoria virtual?
    c. ¿Qué es la memoria residente de un proceso?
6.
    a. Describa la fórmula del Tiempo Efectivo de Acceso (*Effective Access Time – EAT*) a memoria.
    b. ¿Qué nos brinda está fórmula?
7. Describa el algoritmo de reemplazo LRU.
8. Describa dos métodos de acceso a un archivo del sistema.
9.
    a. ¿Qué gran ventaja tiene el método de entrada/salida DMA frente al de entrada/salida programada (*Programmed IO*).
    b. Describa los 6 pasos de una entrada/salida a través de un DMA.
10.
    a. ¿Qué tipo de mejoras brindan los sistemas RAID (*redundant array of inexpensive disk*)?
    b. Describa el sistema RAID 1.

Solución:

Nota: Las respuestas se muestran como una guía de la solución. No necesariamente están completas.

1) La protección de CPU permite que los procesos no se apoderen de forma indefinida del recurso procesador. Si no se existiera en una ambiente de tiempo compartido, los procesos podrían hacer un uso excesivo del procesador, limitando a otros del recurso.

2)

- Mantener que partes de la memoria están siendo utilizadas y por quién.
- Decidir cuales procesos serán cargados a memoria cuando exista espacio de memoria disponible.
- Asignar y quitar espacio de memoria según sea necesario.

3)

- Ejecución de programas.
- Operaciones de Entrada/Salida.
- Manipulación del sistema de archivos.

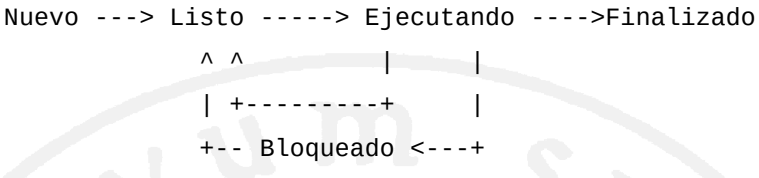
- Comunicación entre procesos.
- Manipulación de errores.

Faltan descripciones.

4) Para poder representar hilos (threads) es necesario tener estructuras para la pila (stack), el contador de programa (program counter) y los registros de cada hilo. Se debe modificar el PCB de forma tal de que cada hilo tenga estas estructuras de forma independiente.

5)

a.



Falta mencionar las transiciones.

b. Si el planificador es invocado solamente cuando un proceso finaliza su ejecución o cuando se bloquea, el núcleo no es expropiativo. Si, además, el planificador es invocado cuando un proceso pasa de estado bloqueado a listo o de ejecutando a listo, el núcleo se dice que es expropiativo.

6) La prioridad sobre un recurso es inversamente proporcional al uso del mismo. Los procesos intensivos en CPU (CPU-bound) deberán tener una prioridad baja sobre el recurso procesador, por lo que se les asignará un quantum de tiempo menor con respecto a los procesos intensivos en entrada-salida (I/O bound).

De estar forma, los procesos que no hacen un uso desmedido del procesador podrán acceder con mayores posibilidades.

7) Ejecutarán de la siguiente forma:

Procesador	Cola listos
(P1,P2)	(P3,P4)
(P3,P4)	(P1,P2)
(P1,P2)	(P3,P4)
(P3,P4)	(P1,P2)
(P1,P2)	(P4)
(P1,P4)	(P2)
(P1,P2)	()
(P2)	()
(P2)	()

(TE(P1) + TE(P2) + TE(P3) + TE(P4)) / 4 = (2 + 3 + 2 + 3) / 4 = 2.5.

8)

a. Su función principal es para poder realizar la traducción de página virtual a frame físico.

b. La estructura es guardada a nivel del sistema operativo. De otra forma, el proceso podría modificarla. También se puede mencionar que el proceso la puede acceder, pero solo en formato lectura.

c. Cumple la función de memoria cache de las entradas de la tabla de páginas. Esto permite mejor velocidad de acceso, ya que si se logra un hit en la cache no es necesario acceder a memoria física a buscar la entrada correspondiente de la tabla de páginas.

9) El algoritmo de segunda chance conserva una cola circular con los frames según el orden de carga. Cada vez que se carga un frame se lo pone al final de la cola y se marca el bit de referencia en 0. A su vez, cada vez que el frame es accedido se marca el bit de referencia en 1. Posteriormente, al ejecutarse el algoritmo de reemplazo, se hace una recorrida desde el comienzo de la cola buscando el frame a reemplazar. Para cada frame se tienen dos posibilidades:

   a. Si el frame tiene el bit de referencia en 1, entonces se le da una segunda oportunidad moviéndolo hacia el final de la cola y marcando su bit de referencia en 0.

   b. Si el frame tiene el bit de referencia en 0, es el elegido para ser reemplazado y el algoritmo finaliza.

10)

   a. Software.

Hacer la traducción de los pedidos que el sistema realiza sobre el dispositivo “dialogando” con la controladora del dispositivo. El manejador de dispositivo (device driver), por lo general, implementa un conjunto de primitivas de alto nivel (impuestas por el sistema operativo), que realizan la traducción de los pedidos y respuestas que genera el sistema. Para esto, es necesario que se pueda comunicar con el controlador del dispositivo en el lenguaje que éste propone.

Problema 2

Se dispone de una arquitectura paginada (tamaño de página 1Kbyte) con direcciones lógicas de 32 bits y 32Mbytes de memoria RAM. Para intentar reducir el espacio en memoria principal destinado al almacenamiento de tablas de traducción, se opta por un esquema de paginación de 2 niveles. Se pide:

a) Diseñar y dibujar el esquema de tablas de gestión de memoria. Describa brevemente cada componente perteneciente al diseño.

b) El esquema de gestión de memoria utiliza un algoritmo de reemplazo cuyo ámbito denominaremos global por prioridad. Es decir, a la hora de seleccionar un página víctima, el conjunto de páginas candidatas se limita a aquellas que pertenecen a procesos de igual o menor prioridad que aquél que produjo el fallo de página. En un instante dado se va a iniciar la ejecución de dos procesos, P_1 y P_2, ambos de la mínima prioridad. La memoria está ocupada por páginas de procesos de prioridad máxima y sólo existen 5 marcos libres. Suponiendo el siguiente esquema de ejecución de los dos procesos (con el rango de direcciones virtuales a las que acceden durante cada uno de los cuantos de tiempo q_i):

q_0	P_1	0-100
q_1	P_2	0-500
q_2	P_1	100-500
q_3	P_2	500-1500
q_4	P_1	3500-3700
q_5	P_2	500-600
q_6	P_1	2500-3000

Detallar los fallos de página que se producen con un algoritmo de reemplazo LRU y un algoritmo de reemplazo FIFO. Justifique su respuesta.

c) Indique en que ubicación de la memoria debe residir el código de la rutina de atención al fallo de página. Justifique su respuesta.

Solución

a) Ver 8.5.3 libro Silberschatz-Galvin 5ta edición (pag 267)

b) Como los procesos P1 y P2 tienen prioridad mínima, no reemplazan ninguna página ya cargada por otros procesos. Deben arreglarse (entre los dos procesos) con los cinco marcos existentes.

q0

PF. Se carga la página con direcciones 0-1023 de P1

Quedan cuatro marcos libres

q1

PF. Se carga la página con direcciones 0-1023 de P2

Quedan tres marcos libres

q2

No hay PF. La página con direcciones 0-1023 de P1 se cargó en q0

Quedan tres marcos libres

q3

En este punto se necesitan dos páginas. La que contiene las direcciones 0-1023 de P2 (para esta no hay PF ya que se cargó en q1) y la que contiene las direcciones 1024-2047 de P2 (para esta ocurre un PF).

Quedan 2 marcos libres.

q4

PF. Se carga la página con direcciones 3071-4095 de P1.

Quedan un marco libre.

q5

No hay PF. La página con direcciones 0-1023 de P2 se cargó en q1.

Quedan un marco libre.

q6

PF. Se carga la página con direcciones 2048-3071 de P1.

No queda ningún marco libre

Ya que alcanzan los cinco marcos libres para la secuencia, no se debe realizar ningún reemplazo, por lo que si se usa el algoritmo de reemplazo FIFO o LRU es lo mismo.

c) La rutina de atención al fallo de página debe residir en un área de memoria que no pagine, ya que si paginara entraría en un ciclo infinito cuando se va a ejecutar la rutina

---

Problema 3

Se desea modelar una juguetería con capacidad para 50 personas dentro del local. Los clientes esperan fuera del local cuando el comercio esta lleno, a la espera de que algún cliente salga.

Una vez que el cliente elige los juguetes a comprar elige de manera aleatoria una de las 5 posibles cajas donde hacer cola. Luego de haber pagado, el cajero le da el ticket y le indica a cual de las 3 empaquetadores dirigirse para obtener los juguetes. El cliente recibe los juguetes empaquetados luego de mostrarle al empaquetador el ticket de pago correspondiente.

La juguetería tiene un trato preferencial con las personas embarazadas al momento de hacer la cola para pagar. Una vez que el cliente preferencial se coloca en una cola para pagar, el cajero correspondiente debe atenderlo lo antes posible.

Se pide resolver el problema utilizando mailboxes (pero no más de 35). Se podrá utilizar a lo sumo una tarea auxiliar.

Se dispone de las siguientes funciones:

- cliente_embarazada():boolean ejecutada por el cliente devuelve TRUE si esta embarazada.
- elegir_caja():Integer ejecutada por el cliente, que indica en que cola colocarse para pagar.
- elegir_empaquetador(): Integer ejecutada por el cajero, que indica que cola debe hacer el cliente para recibir el paquete.
- pagar() ejecutada por el cliente.
- imprimir_ticket(): Ticket ejecutada por el cajero.
- empaquetar() ejecutada por el empaquetador luego de recibir el ticket.

Solución

```
Type
  AdmEntrada = Record
              tipo : [no_emb, embarazada, fin_embarazada, fin_no_emb, caja];
              nro_caja : integer;
  End;
       RetTicket = Record
              ticket : integer;
              empaqutadora : integer;
       End;


Var
       mbx_entrada : mailbox of null;
       mbx_admin : mailbox of AdmEntrada;
       mbx_caja_embarazada : Array[1..5] of mailbox of null;
       mbx_caja_no_emb : Array[1..5] of mailbox of null;
       mbx_cajas : Array[1..5] of mailbox of [embarazada, no_emb];
```

```
	mbx_cajas_ticket : Array[1..5] of RetTicket;
	mbx_emps_in : Array[1..3] of mailbox of null;
	mbx_emps : Array[1..3] of Ticket;
	mbx_emps_juguetes : Array[1..3] of Juguetes;

Procedure Cliente;
Begin
	receive(mbx_entrada,null);
	elegir_juguetes();
	nro_caja = elegir_caja();
	if (cliente_embarazada()) then
		send(mbx_admin,[embarazada,nro_caja]);
		receive(mbx_caja_embarazada[nro_caja],null);  // Espero atencion
	else
		send(mbx_admin,[no_emb,nro_caja]);
		receive(mbx_caja_no_emb[nro_caja],null); // Espero ser atendido
	endif

	pagar();

	send(mbx_cajas[nro_caja],null);  // Aviso que pague
	receive(mbx_cajas_ticket[nro_caja],[ticket,nro_emp]); // Recibo ticket
	if (cliente_embarazada()) then
		send(mbx_admin,[fin_embarazada,nro_caja]);
	else
		send(mbx_admin,[fin_no_emb,nro_caja]);
	endif
	send(mbx_cajas[nro_caja],null);  // Aviso que 'recibi ticket'

	receive(mbx_emps_in[nro_emp], null); // Comienzo pedido de empaquetado
	send(mbx_emps[nro_emp], ticket); // Le doy el ticket
	receive(mbx_emps_juguetes[nro_emp],juguetes); // Recibo juguetes
	send(mbx_emps[nro_emp], null); // Aviso que 'recibi juguetes'

	send(mbx_entrada,null);  // Me voy...
End Cliente;

Procedure Caja(nro_caja : 1..5);
Begin
	while true do begin
		send(mbx_admin,[caja, nro_caja]); // Aviso disponibilidad de caja
		receive(mbx_cajas[nro_caja], tipo)
		if(tipo = embarazada) then   // Le aviso al cliente que me pague
```

```
                send(mbx_caja_embarazada[nro_caja],null);
            else
                send(mbx_caja_no_emb[nro_caja],null);
            endif
            receive(mbx_cajas[nro_caja],null); // Espero 'pago'

            ticket = imprimir_ticket(); // Imprimo ticket
            emp = elegir_empaquetador();

            send(mbx_cajas_ticket[nro_caja],[ticket,emp]);
            receive(mbx_cajas[nro_caja],null); // Espero confirmacion de que el cliente
                                               //  recibio el ticket
      end
End;

Procedure Empaquetadora(nro_emp : 1..3);
Begin
      while true do begin
            send(mbx_emps_in[nro_emp], null); // Aviso que estoy libre
            receive(mbx_emps[nro_emp], ticket); // Recibo ticket
            empaquetar();
            send(mbx_emps_juguetes[nro_emp],juguetes); // envio juguetes
            receive(mbx_emps[nro_emp], null); // Espero confirmacion
      end while;
End;

Procedure Administrador;
Var embarazadas : Array [1..5] of integer;
            no_emb: Array [1..5] of integer;
            caja_espara: Array [1..5] of boolean;
Begin
      For i = 1 to 5 do begin
            embarazadas[i] = 0;
            no_emb[i] = 0;
            caja_espera[i] = false;
      End;

      while (true) do begin
            receive(mbx_admin,[tipo,nro_caja]);
            if (tipo = embarazada) then
                  embarazadas[nro_caja]++;
                  if(caja_espera[nro_caja]) // Si no hay nadie en caja
                        send(mbx_cajas[nro_caja], embarazada);
```

```
                endif
            else if (tipo = fin_embarazada)
                embarazadas[nro_caja]--;
            else if (tipo = no_emb) then
                no_emb[nro_caja]++;
                if(caja_espera[nro_caja]) // Si no hay nadie en caja
                    send(mbx_cajas[nro_caja], no_emb);
                endif
            else if (tipo = fin_no_emb)
                no_emb[nro_caja]--;
            else // cuando 'tipo = caja'
                caja_espera[nro_caja] = false; // marco caja como no-libre
                if(embarazadas[nro_caja] <> 0)
                    send(mbx_cajas[nro_caja], embarazada);
                else if(no_emb[nro_caja] <> 0)
                    send(mbx_cajas[nro_caja], no_emb);
                else
                    caja_espera[nro_caja] = true; // marco caja como libre
                endif
            endif
        end while
End;

Begin
        for I = 1 to 50 do
            send(mbx_entrada,null);
        cobegin
            Administrador();
            Empaquetadora(1);
                ...
            Empaquetadora(3);
            Caja(1);
                ...
            Caja(5);

            Cliente();
                ...
            Cliente();
        coend
End;
```
