Examen 19 de diciembre de 2015

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.**
- **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)

Para las siguientes partes, conteste justificando brevemente, cada una de las preguntas.

1. Describa dos métodos a través de los cuales el sistema operativo se entera de que un dispositivo de E/S ha finalizado un pedido generado por el sistema operativo, mencionando las principales características de cada uno y compare los mismos.

2. Una vez ocurrido un deadlock, describa un mecanismo que permita recuperarse del mismo. Mencione sus ventajas y desventajas.

3. En una arquitectura multiprocesador, describa las ventajas y desventajas de contar con cache a nivel de los procesadores.

4. Describa brevemente los siguientes algoritmos de reemplazo de paginas en el contexto de memoria virtual paginada: FIFO, LRU, Segunda-Oportunidad y el “algoritmo optimo”.

5. Describa brevemente las ventajas/desventajas de los sistemas operativos diseñados con el enfoque monolítico frente al enfoque en capas.

6. Describa dos mecanismos de protección que le brinda el hardware al sistema operativo.

7. Describa las alternativas vistas en el curso para administrar el espacio libre en el contexto de sistemas de archivos.

8. Para invocar un llamado al sistema, ¿es necesario estar en modo monitor? ¿Por que?

Problema 2 (35 puntos)

Se tiene un sistema con un procesador sobre el que ejecuta un sistema operativo con un planificador Round Robin con un cuanto de 20ms**.**

El sistema utiliza memoria virtual con direcciones virtuales de 12 bits, paginación bajo demanda con asignación de marcos local de 3 marcos por proceso, algoritmo de reemplazo LRU (*Least Recently Used*) y un esquema de traducción de dos niveles con 2 bits de direccionamiento en cada nivel.

El siguiente programa comienza su ejecución en el sistema planteado:

Programa

```
Ejecuta durante 10ms
pid1 = fork()
Ejecuta durante 15ms
Se bloquea durante 10ms
pid2 = fork()
Ejecuta durante 25ms
if pid1 <> 0 and pid2 <> 0 then
   wait()
   wait()
   Ejecuta durante 15ms
else if pid2 <> 0 then
   wait()
   Ejecuta durante 10ms
end if
Ejecuta durante 5ms
Termina.
```

Las operaciones fork-wait son estilo Unix donde wait() espera por la finalización de un único hijo. Sabiendo que este es el único programa en ejecución en el sistema, y que el tiempo de ejecución de la funciones fork y wait es de 5ms cada una:

a) Realice un diagrama de planificación (tiempo vs procesos) de la ejecución del programa, comenzando en tiempo t=0, indicando el estado de cada uno de los procesos (listo/ejecutando/bloqueado/terminado) en cada intervalo de tiempo

b) Calcule el tiempo de espera de cada proceso

La secuencia de accesos a memoria realizada por el programa durante los primeros 10ms de su ejecución es la siguiente:

(1,1,D)	(0,0,0)	(0,1,0)	(0,0,1)	(1,3,1)	(1,1,F)	(0,0,4)	(1,2,0)

Donde (X, Y, Z) representa: *X* referencia sobre la tabla de primer nivel, *Y* referencia sobre la tabla de segundo nivel, *Z* desplazamiento. Sabiendo que todos los accesos a memoria son válidos y que inicialmente el proceso tiene todos sus marcos vacíos:

c) Muestre un esquema que indique el estado de los marcos de memoria y del algoritmo LRU luego de cada acceso.

d) Indique la cantidad de fallos de página ocurridos.

Solución:

a)

Proceso/tiempo	0	10	15	20	35	45	50	55	65	85	90	105
P1 (Principal)	E	E(fork)	E	L/1	E	B	B	L/2	L/1	E(fork)	E	L/3
P2 (Primer hijo de P1)	-	-	L/1	E	B	E(fork)	E	E	L/2	L/1	L/1	E
P3 (Segundo hijo de P1)	-	-	-	-	-	-	-	-	-	-	L/3	L/2
P4 (Hijo de P2)	-	-	-	-	-	-	L/1	L/1	E	L/2	L/2	L/1

115	120	130	150	160	165	180	190	195	210	215				
L/3	L/2	L/1	E	E(wait)	B	L/1	E(wait)	E	E	T				
E(wait)	B	L/2	L/1	L/1	E	T								
L/2	L/1	E	L/2	L/2	L/1	E	T							
L/1	E	T												

Donde L/X indica que el proceso se encuentra en la posición X en la cola de listos.

b)

P1 = (35-20) + (85-55) + (150-105) + (190-180) = 15 + 30 + 45 + 10 = 100ms.

P2 = (20-15) + (105-65) + (165-130) = 5 + 40 + 35 = 80ms.

P3 = (130-90) + (180-150) = 40 + 30 = 70ms.

P4 = (65-50) + (120-85) = 15 + 35 = 50ms.

c)

Acceso	(1,1)	(0,0)	(0,1)	(0,0)	(1,3)	(1,1)	(0,0)	(1,2)
Marco 1	(1,1)*	(1,1)	(1,1)	(1,1)	(1,3)*	(1,3)	(1,3)	(1,2)*
Marco 2		(0,0)*	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)
Marco 3			(0,1)*	(0,1)	(0,1)	(1,1)*	(1,1)	(1,1)
LRU	(1,1)	(0,0)	(0,1)	(0,0)	(1,3)	(1,1)	(0,0)	(1,2)
		(1,1)	(0,0)	(0,1)	(0,0)	(1,3)	(1,1)	(0,0)
			(1,1)	(1,1)	(0,1)	(0,0)	(1,3)	(1,1)

Con * se marcan los fallos de página.

d) Se produjeron 6 fallos de páginas en total.

Problema 3 (33 puntos)

Se desea modelar usando monitores el control de pasajeros y equipaje en un aeropuerto. El aeropuerto cuenta con un escaner para las valijas y cuatro controladores que chequean a los pasajeros.

Al llegar al aeropuerto los pasajeros dejan su valija (una por pasajero) en una cinta para ser escaneada y seleccionan la más corta de las cuatro colas para ser atendidos por uno de los cuatro controladores. Si la cinta no tiene lugar para la valija el pasajero deberá esperar a poder dejar la valija antes de pasar a la cola.
Las valijas van siendo procesadas por el escaner que determina si la misma es válida o no.
El pasajero puede pasar al avión unicamente si es aceptado por el controlador y su valija es válida.
Si el pasajero no es aceptado por el controlador debe ir inmediatamente a la sala de interrogatorios. Si es aceptado pero su valija no es válida también debe ir a esa sala.
Si el pasajero es aceptado pero su valija aún no fue escaneada, debe esperar en la sala de espera a que la misma se revise. En este caso, una vez que la valija es revisada el pasajero debe ir al avión o a la sala de interrogatorios de acuerdo a la validez de la valija.
La sala de espera tiene capacidad para 50 pasajeros y la cinta por donde van pasando las valijas tiene capacidad para 50 valijas.

Se pide: Implementar las tareas Pasajero, Controlador y Escáner.

Se tienen las siguientes funciones auxiliares:

- documento(): pasaporte
  - Retorna el pasaporte del pasajero que ejecuta la función.
- dejar_valija_en_cinta()
  - Ejecutada por el pasajero para poner la valija en la fila del escaner
- controlar_pasajero(): boolean
  - Ejecutada por el controlador para verificar al primer pasajero de la fila
- controlar_valija(): (pasaporte, boolean)
  - Ejecutada por el escaner para verificar la primer valija de la fila
- pasar_pasajero(destino:{avion, espera, interrogatorio})
  - Ejecutada por el pasajero para ir al avión, a la sala de espera o a la sala de interrogatorios

Solución:

```
Monitor lugares

var   slots: array[1..50] of pasaporte;
      espera: condition;

procedure obtenerLugar(p: pasaporte): integer
var   libre: integer
begin
      libre := 1;
      while slots[libre] != null and libre <= 50 do
            libre := libre + 1;
      end
      if libre = 51 then
            espera.wait();
            libre := 1;
            while slots[libre] != null and libre <= 50 do
                  libre := libre + 1;
            end
      end
      slots[libre] = p;
      return libre;
end

procedure conPasaporte(p: pasaporte): integer
var   lugar: integer
begin
      lugar := 1;
      while slots[lugar] != p do
            lugar := lugar + 1;
      end
      return lugar;
end

procedure liberar(lugar: integer)
begin
      slots[lugar] := null;
      espera.signal;
end

begin
      for i:= 1 to 50 do
            slots[i] := null;
      end
end

end Monitor
```

```
Monitor adm_colas

var   largo: array[1..4] of integer;
      cant: integer;
      valija_en_cinta: condition;

procedure obtener_cola(): integer
var   cola, min: integer
begin
      cola := 1;
      min := largo[1];
      for i := 2 to 4 do
            if largo[i] < min then
                  min := largo[i];
                  cola := i;
            end
      end
      largo[cola] := largo[cola] + 1;
      cant := cant + 1;
      valija_en_cinta.signal();
      return cola;
end

procedure retornar_cola(cola: integer)
begin
      largo[cola] := largo[cola] – 1;
end

procedure hay_valija()
      if cant == 0 then
            valija_en_cinta.wait();
      end
      cant := cant - 1;
end

begin
      for i:= 1 to 50 do
            largo[i] := 0;
      end
      cant := 0;
end

Monitor cola

var   espera: condition;
      controlador: condition;
      result, listo: boolean;
      largo: integer;

procedure esperar(): boolean
var   cola: integer;
begin
      largo := largo + 1;
      listo := false;
      controlador.signal();
      if(!listo)espera.wait();
      largo := largo – 1;
      return result
end
```

```
procedure procesar()
begin
      if largo == 0 then
            controlador.wait();
      end
      result = controlar_pasajero();
      listo := true;
      espera.signal();
end

begin
      largo := 0;
end


end Monitor

var colas: array[1..4] of Monitor cola


Monitor pasaje

var   valija, valijaOk: boolean;
      espera: condition;

procedure esperar(ok: boolean): boolean
var   result: boolean
begin
      if not ok then
            result := false;
      else if valija then
            result := valijaOk;
      else
            pasar_pasajero(espera);
            espera.wait;
            result := valijaOk;
      end

      valija := false;
      return result;
end

procedure valija(ok: boolean)
begin
      valija := true;
      valijaOk := ok;
      espera.signal;
end

begin
      valija := false;
end

end Monitor

var pasajeros: array[1..50] of Monitor pasaje
```

```
procedure pasajero()
var   lugar: integer;
      ok: boolean;
      p: pasaporte;
      cola: integer;
begin
      p := documento();
      lugar := lugares.obtenerLugar(p);
      dejar_valija_en_cinta();
      cola := adm_colas.obtener_cola();
      ok := colas[cola].esperar();
      adm_colas.retornar_cola(cola);
      ok := pasajeros[lugar].esperar(ok);
      lugares.liberar(lugar);
      if ok then
            pasar_pasajero(avion);
      else
            pasar_pasajero(interrogatorio);
      end
end


procedure controlador(num: integer)
var   ok: boolean;
begin
      while true do
            colas[num].procesar();
      end
end


procedure escaner()
begin
      while true do
            adm_colas.hay_valija();
            (p, ok) = controlar_valija()
            lugar := lugares.conPasaporte(p);
            pasajero[lugar].valija(ok);
      end
end


cobegin
   escaner();
   controlador(1); controlador(2); controlador(3); controlador(4);
   pasajero(); ... pasajero();
coend
```