Examen 24 de julio de 2017

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 que es y que funcionalidad brinda el sistema de archivos virtual (VFS).
2. Describa los pasos que debe llevar a un proceso de usuario para invocar una system call y todos los pasos posteriores llevados a cabo por el hardware y el sistema operativo hasta que el resultado de la system call queda disponible. Indique quien ejecuta cada paso.
3. Realice un esquema donde muestre el PCB (Process Control Block) de un proceso con dos hilos (threads). Detalle los campos que considere importantes.
4. En el contexto del subsistema de E/S. Describa la diferencia entre una operación no bloqueante y una asincrónica. ¿En que caso utilizaría cada una de ellas?
5. Describa brevemente los mecanismos de hardware necesarios para garantizar la protección adecuada de los procesos en ejecución.
6. Describa las alternativas vistas en el curso para administrar el espacio libre en el contexto de sistemas de archivos.
7. ¿En qué circunstancias ocurren los fallos de página? Describa las acciones llevadas a cabo por el sistema operativo cuando ocurre un fallo de página.
8. Describa qué problemas enfrenta un hipervisor para poder soportar máquinas virtuales con sistemas operativos invitados que tienen soporte de memoria virtual y cómo puede resolverlos.

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 18 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 un nivel con 8 bits destinados al desplazamiento.

El siguiente programa comienza su ejecución en el sistema planteado:

Programa

```
%Ejecuta durante 5ms
pid1 = fork();
pid2 = fork();
if ( pid1 != 0 ) {
   if ( pid2 != 0 ) {
      %Se bloquea durante 10ms
      wait();
      wait();
   } else {
      %Ejecuta 5ms
      %Bloquea 10ms
      %Ejecuta 5ms
   }
} else if ( pid2 != 0 ) {
   %Ejecuta durante 10ms
   wait();
   %Ejecuta durante 5ms
} else {
   %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

c) En el milisegundo 35 se le envía una señal que finaliza inmediatamente el primer proceso hijo del proceso principal. Al matar un proceso también se matan todos sus hijos (finalización en cascada). **Se pide:** modificar el diagrama de la parte **a** para contemplar este nuevo escenario.

d) Durante los primeros 5ms de ejecución, el programa realiza los siguientes accesos a memoria:

(2,0D)	(5,10)	(3,10)	(5,A1)	(8,A1)	(2,0F)	(5,B4)	(9,C0)

Donde (X, Y) representa: *X* referencia sobre la tabla de primer nivel e Y referencia el desplazamiento (en hexadecimal). Sabiendo que: todos los accesos a memoria son válidos, que inicialmente el proceso tiene todos sus marcos vacíos, y que las tablas de páginas siempre se encuentran cargadas en memoria y no ocupan marcos del proceso.
Se pide: Muestre un esquema que indique el estado de los marcos de memoria, los eventuales fallos de página y del algoritmo LRU luego de cada acceso.

Solución:

a)

tiempo	0	5	10	15	20	25	30	35	40	45	50	55	60	65
P1 (Principal)	E	E(fork)	E(fork)	B	B	L/3	L/3	L/2	L/1	E(wait)	B(wait)	L/1	E(wait)	T
P2 (Primer hijo de P1)	-	-	L/1	E(fork)	E	E	E(wait)	B(wait P4)	B(wait P4)	L/1	E	T	-	-
P3 (Segundo hijo de P1)	-	-	-	L/1	L/1	L/1	L/1	E	B	B	L/1	E	T	-
P4 (Hijo de P2)	-	-	-	-	L/2	L/2	L/2	L/1	E	T	-	-	-	-

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

b)

P1 = (45-25) + (60-55) = 20 + 5 = 25ms.

P2 = (15-10) + (50-45) = 5 + 5 = 10ms.

P3 = (35-15) + (55-50) = 20 + 5 = 25ms.

P4 = (40-20) = 20ms.

c)

tiempo	0	5	10	15	20	25	30	35 (sigkill P2)	40	45	50	55	60
P1 (Principal)	E	E(fork)	E(fork)	B	B	L/3	L/3	L/1	E(wait)	E(wait)	B(wait)	T	-
P2 (Primer hijo de P1)	-	-	L/1	E(fork)	E	E	E(wait)	T	-	-	-	-	-
P3 (Segundo hijo de P1)	-	-	-	L/1	L/1	L/1	L/1	E	B	B	E	T	-
P4 (Hijo de P2)	-	-	-	-	L/2	L/2	L/2	T	-	-	-	-	-

d) Con * se marcan los fallos de página. Se produjeron 6 fallos de páginas en total.

Acceso	2	5	3	5	8	2	5	9
Marco 1	2*	2	2	2	8*	8	8	9*
Marco 2		5*	5	5	5	5	5	5
Marco 3			3*	3	3	2*	2	2
LRU	2	5	3	5	8	2	5	9
		2	5	3	5	8	2	5
			2	2	3	5	8	2

Problema 3 (33 puntos)

Los empleados de una empresa elaboran productos a partir de rollos de tela que se deben retirar en un depósito. El depósito solo admite 8 empleados simultáneamente. Cuando el empleado ve que en el depósito hay menos de 10 rollos avisa al reponedor y espera a que éste cumpla su tarea. Solamente el primer empleado que detecta el faltante debe avisar al reponedor (no debe haber notificaciones repetidas)

Cada cierto tiempo llega el inspector que supervisa el depósito. Para supervisar no debe haber otros empleados en el mismo y tiene prioridad de acceso sobre los otros empleados.

El reponedor solamente puede actuar cuando no hay empleados retirando ni el inspector supervisando en el depósito y tiene máxima prioridad.

Se disone de la siguientes funciones auxiliares:

- hay_stock(): boolean devuelve true si hay al menos 10 rollos
- retirar_rollo() retira un rollo del depósito
- manufacturar() ejecutada por los empleados para procesar el rollo retirado
- reponer() agrega rollos al depósito
- registrar_reposicion() registra la reposición
- supervisar() supervisa la mercadería del depósito
- otras_tareas() ejecutada por el inspector luego de supervisar

Se pide: Implementar en ADA las tareas Empleado, Inspector y Reponedor. Se pueden usar hasta 3 tareas auxiliares contando cada instancia de tarea.

Solución:

```
Task Type Empleado is
End Empleado;

Task body Empleado is
  var avisar: boolean;
begin
  Deposito.ingresoEmpleado();
  if not hayStock() then
    Deposito.avisoReponedor(avisar);
    if (avisar) then
      Reponedor.reponer();
    end
  else
    Deposito.finChequeo();
  end
  Deposito.retirar();
  retirar_rollo();
  Deposito.egresoEmpleado();
  manufacturar();
End Empleado;

Task Inspector is
End Inspector;
```

```
Task body Inspector is
begin
  loop
    Deposito.inicioSupervision();
    supervisar();
    Deposito.finSupervision();
    otras_tareas();
  endloop;
End Empleado;

Task Reponedor is
  entry reponer();
End Reponedor;

Task body Reponedor is
begin
  loop
    accept reponer;
    Deposito.inicioReponedor();
    reponer();
    Deposito.finReponedor();
    registar_reposicion();
  endloop;
End Reponedor;


Task Deposito is
  entry ingresoEmpleado();
  entry egresoEmpleado();
  entry inicioSupervision();
  entry finSupervision();
  entry inicioReponedor();
  entry finReponedor();
  entry avisoReponedor(out aviso: boolean);
  entry finChequeo();
  entry retirar();
End Deposito;
```

```
Task body Deposito is
  var   cantEmpleados, cantChequeo, retirando : int
    supervisando, reponedorAvisado: boolean;
begin
  cantEmpleados = 0;
  retirando = 0;
  cantChequeo = 0;
  supervisando = false;
  reponedorAvisado = false;

  loop
    select
      accept egresoEmpleado();
      cantEmpleados--;
      retirando--:
    or
      accept finSupervision();
      supervisando = false;
    or when cantChequeo == 0 =>
        -- La guarda es para que se consuman los eventuales
        -- avisos pendientes de empleados para los cuales
        -- tambien les habia dado que no hay stock pero no
        -- llegaron a encontrarse con avisoReponedor aún
      accept finReponedor();
      reponedorAvisado = false;
    or
      accept avisoReponedor(out aviso: boolean)
        aviso = reponedorAvisado;
      end;
      reponedorAvisado = true;
      cantChequeo--;
    or accept finChequeo();
      cantChequeo--;
    or
     when (cantEmpleados <= 8) AND (not supervisando)
           AND (inicioSupervision'count == 0) =>
      accept ingresoEmpleado();
      cantEmpleados++;
      cantChequeo++;
    or
      when (cantEmpleados == 0) AND (not reponedorAvisado) =>
      accept inicioSupervision();
      supervisando = true;
    or
      when (not supervisando) && (retirando == 0) =>
      accept ingresoReponedor();
    or
      when (not reponedorAvisado) =>
      accept retirar();
      retirando++;
    endselect;
  endloop;
End Deposito;
```