Examen Abril de 2003

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.**
- **Escriba el nombre en la hoja de letra, la cual deberá entregar conjuntamente con la solución.**

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 (el 2 o el 3) bien hecho y medio más (de cualquiera de los 3).**

Finalización

- **El examen dura 4 horas.**

Problema 1

Se debe responder a las cuestiones aquí planteadas de manera concisa (no más de 15 renglones por pregunta. En todos los casos debe justificar su respuesta.

Pregunta1

a) Describa brevemente las siguientes técnicas de asignación de memoria: First Fit, Best Fit, Worst Fit. Indique las ventajas y desventajas de cada una de ellas

==========

First Fit

Asigna el primer espacio que tenga tamaño suficiente. La búsqueda puede comenzar en el inicio de la lista de espacios, o donde terminó la búsqueda anterior. La búsqueda termina al encontrar el primero de tamaño suficiente. Es el más rápido de los 3, si bien en la utilización de almacenamiento es comparable a Best Fit.

Best Fit

Asigna el menor espacio de tamaño suficiente. Hay un mayor costo en la búsqueda o en la inserción según se mantenga o no una lista ordenada de espacios libres. Produce el fragmento sobrante más pequeño, que tendrá una escasa probabilidad de volver a ser reusado.

Worst Fit

Asigna el hueco más grande, a fin de evitar el problema anterior. Sin embargo sigue siendo ineficiente como Best Fit.

==========

b) Describa brevemente los siguientes algoritmos de reemplazo de paginas en el contexto de memoria virtual paginada: FIFO, LRU, Segunda-Oportunidad y el “algoritmo optimo”. Indique justificando adecuadamente en cada caso, si su uso es o no frecuente.

==========

FIFO

Asocia (en forma explícita o no) a cada página el instante en que se trajo a memoria, eligiendo la más antigua cuando hay que hacer un reemplazo. Es fácil de implementar, pero su rendimiento no siempre es bueno. Un ejemplo es cuando se reemplaza una página traída hace mucho tiempo, pero de uso intensivo. También presenta la anomalía de Belady: con ciertos patrones de referencias, la tasa de fallos de página puede aumentar al incrementarse el número de marcos asignados, en contra de lo esperado.

OPTIMO

Busca reemplazar la página que no se usará durante el mayor período de tiempo, garantizando la menor tasa de fallas de páginas posible, para un número fijo de marcos. Es difícil de implementar, pues se requiere un conocimiento futuro de las referencias a páginas. Se utiliza (más precisamente sus resultados esperados) para estudios comparativos con otros algoritmos no óptimos.

LRU

Reemplaza la página que no se ha usado durante el mayor periodo de tiempo, basándose en el pasado reciente para suponer cuál será la página que no se usuará en el futuro cercano. Es utilizado con frecuencia, y existen 2 implementaciones:

a. Con un contador, que es incrementado con cada referencia a alguna página de memoria. cada vez que se referencia una página en particular, el contador se copia a un registro asociado a la misma.
   - Debe tratarse el desbordamiento del contador.
   - Este esquema requiere una búsqueda en toda la tabla de páginas, a fin de encontrar la que tiene el menor valor.
   - Observar que requiere apoyo de hardware.

b. Mantener una lista doblemente encadenada, extrayendo la página más recientemente utilizada.

LRU no presenta la anomalía de Belady. (Pertenece a una clase llamada algoritmos de pila: las páginas residentes en memoria para un sistema con N marcos son siempre un subconjunto de las que habría con N + 1)

SEGUNDA OPORTUNIDAD

Es una variante del algoritmo FIFO, que evita que una página muy utilizada sea eliminada por llevar mucho tiempo en memoria. Se mantiene un bit por página, que es seteado en 1 con cada referencia, y actualizado por el sistema a intervalos regulares. Se recorre la lista de páginas examinando estos bits de referencia. Si el bit de una página está en 1 se le da una segunda oportunidad, poniéndolo en 0 y enviándolo al final de la lista (como si recién hubiera llegado a memoria). La búsqueda termina cuando se encuentra el primer bit en 0, en cuyo caso se reemplaza la página.

==========

c) ¿Cuando ocurre un fallo de pagina?. Explique las acciones del sistema operativo frente a un fallo.

==========

En el contexto de un sistema de paginación por demanda, sucede cuando un proceso intenta referenciar una página que no se incorporó a memoria. El hardware de paginación, al traducir la dirección, observa que la entrada correspondiente de la tabla tiene indicada esa página como inválida, y genera una trap para el sistema operativo (la instrucción que hizo la referencia no se completa).

Entonces el sistema

1. determina si la referencia memoria fue válida o inválida (por ejemplo por indizar mal un vector)
2. si fue una referencia válida, en vez de abortar el proceso, localizar en disco la página buscada
3. buscar un marco libre
   a. si lo hay, lo selecciona
   b. sino, aplica algoritmo de reemplazo para seleccionar un página a quitar, la guarda en disco y refleja el cambio en tabla de páginas.
4. mover la página deseada al nuevo marco libre, reflejar cambio en tabla de páginas

5. reejecutar la instrucción que provocó el fallo (el proceso no necesariamente cambio de estado).

Obs: Se planifica (CPU) en las transferencias entre disco y memoria de la página seleccionada para reemplazo y de la página entrante de acuerdo a las reglas habituales para E/S.

==========

Pregunta 2

a) Describa las acciones del kernel de un sistema operativo cuando ocurre un cambio de contexto:

1. entre hilos
2. entre procesos

==========

Para el caso de un cambio de contexto entre hilos de un mismo proceso, el sistema operativo debe salvar los registros, el contador de programa y el registro de pila del hilo en el PCB. Sin embargo no se requiere ninguna otra gestión relativa al espacio de memoria u otros recursos, ya que el contexto es idéntico para todos los hilos del proceso o tarea que contiene a los hilos. De acuerdo a lo expuesto, para un proceso activo, el planificador podría priorizar el despacho de hebras del mismo en el procesador o CPU corriente.
Si los hilos pertenecen a tareas o procesos diferentes, obviamente se hace necesario conmutar el contexto de acuerdo a la técnica habitual.

Para el caso de cambio de contexto entre procesos diferentes, el sistema operativo debe realizar las mismas tareas que con lo hilos, pero además debe movilizar información relativa a los recursos asignados al procesos, como ser bloques de memoria o disco asignado y procesos descendientes del proceso actual.

==========

b) En el contexto de planificación de procesos, describa los posibles estados en los cuales un proceso pueda encontrarse.

==========

En ejecución: el proceso tiene asignado el procesador, y el flujo de instrucciones del proceso se está ejecutando.

Bloqueado: el proceso está en espera de que se le asigne algún recurso físico (como el fin de una E/S) o lógico.

Listo: El proceso no está bloqueado, sino compitiendo para que se le asigne un procesador y pasar a estado de ejecución.

Suspendido: El proceso fue suspendido explícitamente, saliendo del estado listo o ejecutándose. Pasará al estado listo (y será elegible para asignársele la CPU) cuando sea reanudado, también en forma explícita.

Bloqueado suspendido: Análogo al anterior, se da cuando se suspende un proceso que estaba bloqueado. Volverá al estado bloqueado al ser reaunudado.

==========

c) Describa detalladamente que es el PCB

==========

Un PCB (Process Control Block) es una estructura de datos del sistema que representa un proceso, almacenando información del mismo. Incluye

a. El estado del proceso (listo, en ejecución, etc.)
b. El program counter, indicando la siguiente instrucción a ejecutar
c. Los registros, que son guardados junto con el PC por cada interrupción.
d. Información adicional para la planificación (prioridades, punteros a las colas donde se pudiera encontrar el proceso, según su estado, etc.)
e. Información de administración de memoria (registros límites, tablas de páginas)
f. Información contable: número de proceso, utilización de CPU, tiempo de creación, etc.
g. Información de E/S: archivos abiertos, E/S pendientes.
h. Referencias a la genealogía o contexto de procesos, hijos, padre, etc.

==========

Pregunta 3

a) Describa las principales características de los siguientes tipos de sistemas operativos:

1. Por lotes
2. Interactivos
3. De tiempo compartido
4. De tiempo real
5. Distribuidos

==========

Por lotes

Recibe un flujo de trabajos separados, los ejecuta y vuelca los resultados de los mismos, devolviéndolos por una impresora, medio de almacenamiento o similar. Se caracterizan por la ausencia de interacción entre el usuario y el trabajo en tanto que este se ejecuta. El término <<por lotes>> se refiere a un modelo en que los trabajos similares se agrupaban en lotes, a fin de reducir el tiempo total dedicado a preparar el computador entre distintos trabajos. Los sistemas modernos no presentan esta limitación.

Interactivos

Provee una comunicación en línea entre el usuario y el sistema, donde el usuario da instrucciones al sistema o a las aplicaciones, y recibe una respuesta a la misma.

De tiempo compartido

Es un caso particular de la multiprogramación, donde la CPU es compartida entre los procesos con una frecuencia alta, presentándose al usuario como si este tuviera su propio computador.

De tiempo real

En general se usan para el control de aplicaciones dedicadas. Tiene restricciones temporales bien definidas, por lo que si el procesamiento no se lleva a cabo dentro de límites esperados, el sistema falla.

Distribuidos

Son un conjunto de computadoras independientes (desacopladas en cuanto a hardware, no comparten memoria ni reloj) comunicadas por una red, que se presentan al usuario en la mayor medida que sea posible, como un único sistema. Su motivación son compartir recursos (mejorar la utilización) y aumentar tanto la escalabilidad como la confiabilidad.

==========

b) En el contexto de un sistema operativo diseñado en capas, ¿ Por que el soporte para la memoria virtual no utiliza las primitivas de sistemas de archivos ?

==========

En un sistema operativo correctamente estructurado, habitualmente el filesystem está en una capa superior a la de gestión de memoria, hecho que inhibe la posibilidad de uso del mismo para la implementación de la memoria virtual.

En algún escenario, el uso de las primitivas del filesystem podría implicar un mayor overhead. Si bien esta razón como tal resulta invalida, ya que la corrección debería ser el hacer eficiente el filesystem.

==========

c) Cual es la diferencia entre un sistema monolítico y uno basado en microkernel. Indique ventajas y desventajas.

==========

Un sistema monolítico no tiene una estructura bien integrada. Todas las funcionalidades del sistema se ejecutan en modo monitor. Son difíciles de extender, mantener y depurar. Los sistemas con microkernel son sistemas estructurados, donde la mayoría de los servicios y funciones se implementan en módulos, dejándose sólo una pequeña parte del sistema (el microkernel) que ejecute en modo monitor. En general el microkernel se limita a gestión de interrupciones, gestión básica de procesos, gestión de memoria a bajo nivel y comunicación entre procesos.

Sistemas de este tipo, son mas portables y escalables que los convencionales. No obstante, debido a que los distintos componentes ejecutan en espacios de direccionamiento distintos, se presenta un overhead mayor al de los sistemas monolíticos.

==========

Pregunta 4

En el contexto del algoritmo del banquero:

a) ¿ Que ventajas y desventajas presenta ?

b) Considere la siguiente situación:

	Asignado				Máximo				Disponible			
	A	B	C	D	A	B	C	D	A	B	C	D
P0	0	0	1	2	0	0	1	2	1	5	2	0
P1	1	0	0	0	1	7	5	0				
P2	1	3	5	4	2	3	5	6				
P3	0	6	3	2	0	6	5	2				
P4	0	0	1	4	0	6	5	6				

1. ¿ Esta el sistema en estado seguro ? Justifique su respuesta.
2. Si para P1 llega el requerimiento (0,4,2,0), ¿ puede ser satisfecho inmediatamente ? Justifique su respuesta

==========

Solución:

a) Ventajas y desventajas del banquero

b.1) Verificamos si el sistema esta en un estado seguro aplicando el algoritmo

Trabajo = (1,5,2,0)
Fin = (false, false, false, false, false)

Necesidad = Máximo – Asignado =

	A	B	C	D
P0	0	0	0	0
P1	0	7	5	0
P2	1	0	0	2
P3	0	0	2	0
P4	0	6	4	2

Iniciamos el algoritmo, buscando una secuencia que muestre si el sistema esta en estado seguro o no:

1.- Elegimos P0, Necesidad0 <= Trabajo
    Trabajo = (1,5,2,0) + (0,0,1,2) = (1,5,3,2)
    Fin = (true, false, false, false, false)
2.- Elegimos P2, Necesidad2 <= Trabajo
    Trabajo = (1,5,3,2) + (1,3,5,4) = (2,8,8,6)
    Fin = (true, false, true, false, false)
3.- Elegimos P3, Necesidad3 <= Trabajo
    Trabajo = (2,8,8,6) + (0,6,3,2) = (2,14,11,8)
    Fin = (true, false, true, true, false)
4.- Elegimos P4, Necesidad4 <= Trabajo
    Trabajo = (2,14,11,8) + (0,0,1,4) = (2,14,12,12)
    Fin = (true, false, true, true, true)
5.- Elegimos P1, Necesidad1 <= Trabajo
    Trabajo = (2,14,12,12) + (1,0,0,0) = (3,14,12,12)
    Fin = (true, true, true, true, true)

Como Fin[i] = true para toda i, entonces el sistema esta en estado seguro. La secuencia obtenida es <P0, P2, P3, P4, P1>

b.2) Verificamos si el sistema esta en un estado seguro luego de hacer la asignación pedida para P1, (0,4,2,0). En este caso la situación es:

Trabajo = (1,5,2,0)
Fin = (false, false, false, false, false)

Necesidad = Máximo – Asignado =

	A	B	C	D
P0	0	0	0	0
P1	0	4	7	0
P2	1	0	0	2
P3	0	0	2	0
P4	0	6	4	2

Basta repetir el algoritmo anterior, dejando P1 para el final nuevamente. En este caso también se cumple que Necesidad1 <= Trabajo por lo que llegamos nuevamente a la misma secuencia valida. El sistema por tanto también esta en estado seguro.

==========

Problema 2

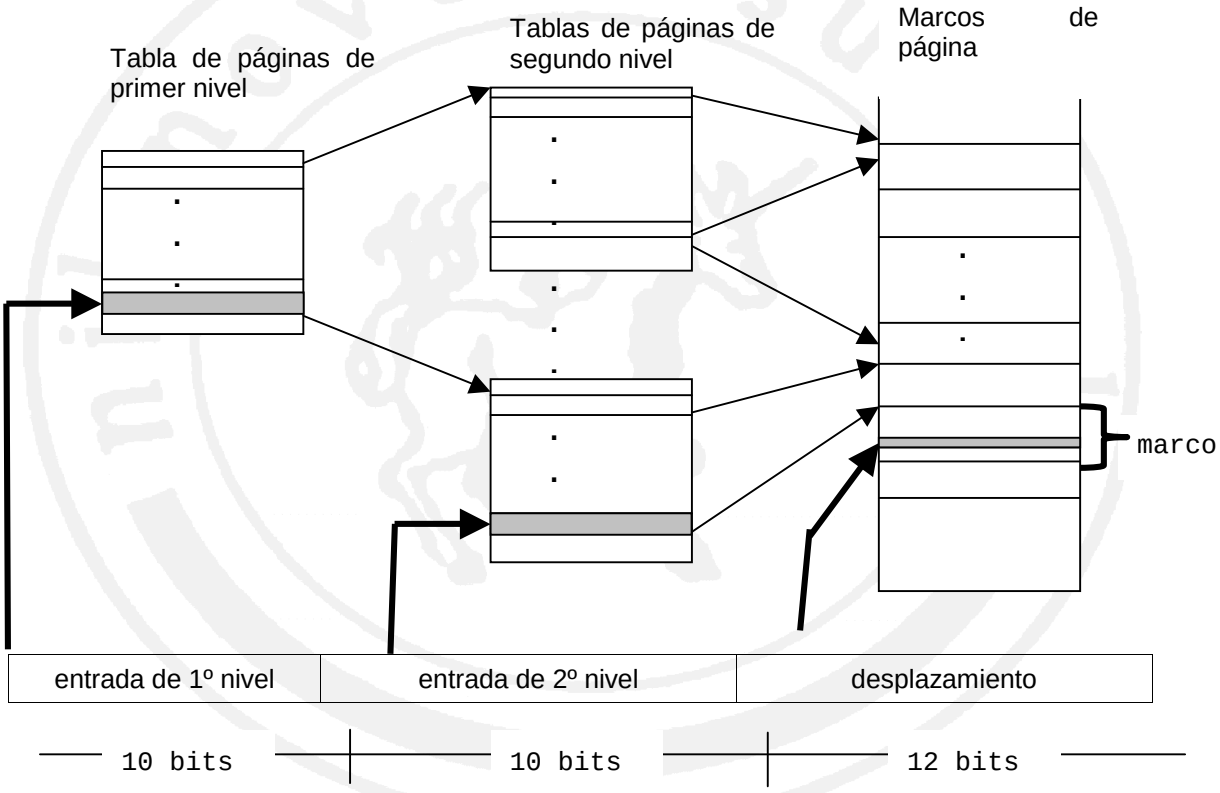
Un sistema operativo usa un sistema de memoria virtual paginada con direcciones virtuales de 32 bits y con páginas de 4KB. La tabla de páginas reside completamente en memoria principal, siendo el tamaño del descriptor de cada entrada de la tabla de páginas de 32 bits.

El espacio virtual es único por proceso, aunque ningún proceso llega a necesitar más de 16 MB. De este espacio, 12 MB se utilizan para almacenar el área de código más el área de datos a partir de la dirección virtual 0. Los 4 MB restantes se destinan al espacio de pila, que se almacena comenzando en la dirección virtual más alta y creciendo hacia direcciones bajas.

a) ¿Cuánto ocupa la tabla de páginas?

b) ¿Cuál es la ocupación útil (entradas ocupadas) en % de la tabla de páginas?

Posteriormente, se decide usar un sistema de tablas de dos niveles, como se muestra a continuación.



Una dirección virtual se interpreta de la siguiente manera: Los 10 bits mas significativos sirven para seleccionar la entrada de la tabla de primer nivel. Esta entrada contiene el número del marco donde reside la tabla de páginas de segundo nivel. Los 10 bits siguientes se usan para seleccionar una entrada de la tabla de segundo nivel. Esta entrada contiene el número del marco donde está cargada la página. Una vez localizado el marco, solo hace falta sumarle los 12 bits del desplazamiento de la dirección virtual.

Supuesto un proceso de este sistema, que ocupa como máximo 16 MB, y sabiendo que la tabla de primer nivel siempre está cargada en memoria durante la ejecución del proceso, mientras que las tablas de segundo nivel se cargan bajo demanda, responda justificando adecuadamente a las siguientes preguntas:

c) ¿Qué tamaño tiene cada una de las tablas?

d) ¿Cuántas tablas de segundo nivel utiliza el proceso? ¿Qué índices se ocupan de la tabla de primer nivel?

e) ¿Cuánto espacio de memoria se necesita para almacenar simultáneamente todas las tablas necesarias?

f) ¿Qué ventaja tiene este sistema con respecto al de una sola tabla de páginas?

g) ¿Qué inconveniente presenta este modelo si la MMU no dispone de TLB?

==========

Solucion:

a) Con direcciones virtuales de 32 bits se puede direccionar hasta 4GB. Como las páginas son de 4KB, en 4GB existirán 4GB/4KB = 1 M páginas. Como cada descriptor ocupa 4 Bytes, entonces la tabla ocupará 1M páginas x 4B = 4MB.

b) Un proceso como máximo necesitará 16 MB. Como cada página soporta 4KB, entonces usará 16MB/4KB = 4K páginas, por lo que usará 4K descriptores de la tabla. Si cada descriptor ocupa 4 bytes y la tabla ocupa 4MB, entonces, la ocupación útil de la tabla será:

   (4K descriptores x 4 bytes)/4MB = 2-8 ➔ 0’39 %

   Es decir, se está desperdiciando casi 4MB por proceso.

c) Tanto la tabla principal como la secundarias tienen 210 entradas y ambas tienen descriptores de 4 bytes. Por tanto cualquier tabla ocupa 4 x210 = 4 KB.

d) Necesita 4 tablas de segundo nivel. Puesto que el área de código y datos empieza en la dirección virtual 0 y ocupa 12 MB, en la tabla de primer nivel se ocuparán los descriptores 0, 1 y 2. El espacio de pila va desde la dirección más alta a direcciones bajas. Por tanto, en la tabla de primer nivel aparece ocupada la última entrada, la 1023.

e) Se necesitan 5 tablas: la de primer nivel más cuatro secundarias. Cada tabla ocupa 4KB, por tanto en memoria principal se usarán 20 KB.

f) El ahorro de espacio, puesto que se ha pasado de 4MB de una sola tabla a 20KB, con tablas multinivel.

g) Es necesario hacer 3 accesos por cada acceso a memoria: uno a la tabla principal, otro a la tabla secundaria y por último el acceso al marco que contiene la información.

==========

Problema 3

En la tienda de mascotas “La cuevita del cobayo” están teniendo problemas serios para mantener contentos a los cobayos. Para evitar esto, se decidió construir un nuevo bebedero y una rueda para hacer ejercicio. Sin embargo, debido al bajo presupuesto, la rueda es muy pesada para un solo cobayo y además el bebedero no soporta más de cinco cobayos a la vez.

Dado que los cobayos son naturalmente peleadores y territoriales al momento de la comida, si hay cobayos de distinto sexo a la vez en el bebedero, estos siempre se pelean hasta morir. Además, para evitar que los cobayos de un sexo se apoderen del bebedero, cuando llega alguien del sexo opuesto a los que están bebiendo, deberá evitar que sigan entrando cobayos del sexo de los que están bebiendo. También se desea evitar que quede olor a los animales del sexo opuesto en el bebedero por lo que antes de cambiar de sexo se debe dejar ventilar el bebedero.

Por otro lado, como la rueda es muy pesada para un cobayo solo, deberá permitirse el acceso a la rueda cuando haya al menos tres cobayos para jugar. Los cobayos entrarán en grupos de a tres a la rueda y estos podrán dejar el juego solo cuando los tres quieran abandonarlo.

A fin de evitar estos problemas, la tienda de mascotas quiere (con lo que le queda de presupuesto) implementar un sistema de control de cobayos que evite que estos se peleen y permita que se diviertan.

En resumen:

Este sistema deberá **evitar:**

- que haya cobayos de distinto sexo a la vez en el bebedero
- que haya más de cinco cobayos en el bebedero
- que entre al bebedero un nuevo cobayo del mismo sexo de los que están bebiendo cuando hay uno del sexo opuesto esperando

y deberá **permitir:**

- que únicamente deje entrar o salir a la rueda de a tres cobayos por vez
- que se ventile el bebedero antes de cambiar el sexo de los animales que lo usan
- mientras se está ventilando el bebedero que también se pueda entrar o salir de la rueda

Notas:

- Existen múltiples instancias de los cobayos aburridos en el sistema
- Los cobayos SOLO ejecutarán el código presentado (son reacios al nuevo código)

Se dispone de la siguientes rutinas auxiliares:

- `que_soy(): {MACHO, HEMBRA}`
- `ventilar()`

Para esto los cobayos “ejecutarán” siempre este código para entrar a beber o para jugar en la rueda:

```
Procedure Cobayo_Aburrido()
Var
      que_hago: Integer;

Begin
      While (true) do
      Begin
            que_hago = pensar_un_rato();
            If que_hago = Beber then
            Begin
                  Quiero_entrar_al_bebedero();
                  Beber_agua();
                  Quiero_salir_del_bebedero()
            End
            Else
                  If que_hago = Jugar then
                  Begin
                        Quiero_entrar_a_la_rueda();
                        Jugar_un_rato();
                        Quiero_salir_de_la_rueda()
                  End
      End
End
```

Se pide implementar en Ada los procedimientos:

- `Quiero_entrar_al_bebedero();`
- `Quiero_salir_del_bebedero();`
- `Quiero_entrar_a_la_rueda();`
- `Quiero_salir_de_la_rueda();`

y las tareas que crea conveniente para modelar este problema

==========

Solución:

```
---------------- bebedero ----------------
procedure Quiero_entrar_al_bebedero is
begin
      if que_soy() = 'MACHO' then
            bebedero.entrar_macho();
      else
            bebedero.entrar_hembra();
      end if;
end Quiero_entrar_al_bebedero;

procedure Quiero_salir_del_bebedero is
begin
      bebedero.salir();
end Quiero_salir_del_bebedero;
```

```
---------------- bebedero ----------------

task bebedero is
      entry entrar_macho;
      entry entrar_hembra;
      entry salir;
end bebedero_machos;

task body bebedero is
      sexo: {MACHO, HEMBRA, NADIE};
      cantb: integer;         // Cantidad en el bebedero
begin
      sexo = NADIE;
      cant = 0;
      loop
            select
                  ; Dejo entrar Cobayos
                  ; Si no hay nadie en el bebedero doy privilegio al sexo opuesto
                  ; al que habia anteriormente
                  when(cantb=0 and (sexo <> MACHO or entrar_hembra'count=0)) or
                     (cantb < 5 AND sexo=MACHO and entrar_hembra'count=0) =>
                       accept entrar_macho do
                         ; Si hubo cambio de sexo, debo ventilar antes
                         ; de terminar el encuentro, esto implica una pequeña
                         ; demora en caso en que no hay que ventilar pero es
                         ; aceptable
                         if cantb = 0 and sexo = HEMBRA then
                             ventilar();
                         end if;
                       end accept
                       sexo = MACHO;
                       cantb = cantb + 1;
            or
                  when(cantb=0 and (sexo <> HEMBRA or entrar_macho'count=0)) or
                     (cantb < 5 AND sexo=HEMBRA and entrar_macho'count=0) =>
                       accept entrar_hembra do
                         ; Si hubo cambio de sexo, debo ventilar antes
                         ; de terminar el encuentro
                         if cantb = 0 and sexo = MACHO then
                             ventilar();
                         end if;
                       end accept
                       sexo = HEMBRA;
                       cantb = cantb + 1;
            or
                  accept salir do
                       cantb = cantb – 1;
                  end accept
            end select
      end loop
end bebedero;


Otra opción para la tarea 'bebedero' que no tiene la demora en el accept:

task body bebedero is
      sexo: {MACHO, HEMBRA};
      cantb: integer;         // Cantidad en el bebedero
begin
      sexo = NADIE;
      cant = 0;
```

```
      loop
            select
                  ; Dejo entrar Cobayos
                  ; Solo dejo entrar cobayos de la misma especie
                  ; El caso inicial (sexo = NADIE) tambien lo considero aquí
                  ; pues suponemos que inicialmente no se necesita ventilar
                  when ((cantb=0 and (sexo = NADIE or entrar_hembra'count=0)) or
                        (cantb < 5 AND sexo=MACHO and entrar_hembra'count=0)) =>
                        accept entrar_macho do
                        end accept
                        cantb = cantb + 1;
            or
                  when ((cantb=0 and (sexo = NADIE or entrar_macho'count=0)) or
                        (cantb < 5 AND sexo=HEMBRA and entrar_macho'count=0)) =>
                        accept entrar_hembra do
                        end accept
                        cantb = cantb + 1;
            or
                  ; Ahora considero el cambio de sexo
                  when (cantb = 0 AND sexo = HEMBRA) =>
                        accept entrar_macho do
                        end accept
                        ventilar();
                        sexo = MACHO;
                        cantb = 1;
            or
                  when (cantb = 0 AND sexo = MACHO) =>
                        accept entrar_hembra do
                        end accept
                        ventilar();
                        sexo = HEMBRA;
                        cantb = 1;
            or
                  accept salir do
                        cantb = cantb – 1;
                  end accept
            end select
      end loop
end bebedero;


---------------- rueda ----------------

procedure Quiero_entrar_a_la_rueda() is
begin
      rueda.entrar
end Quiero_entrar_a_la_rueda;


procedure Quiero_salir_de_la_rueda() is
begin
      rueda.salir
end Quiero_salir_de_la_rueda;
```

```
task rueda is
      entry entrar;
      entry salir;
end rueda;

task body rueda is
begin
      loop
            select
                  accept entrar do
                        accept entrar do
                              accept entrar do
                              end accept
                        end accept
                  end accept
            or
                  accept salir do
                        accept salir do
                              accept salir do
                              end accept
                        end accept
                  end accept
            end select
      end loop
end rueda;
```

==========