Departamento de Arquitectura

Examen de Sistemas Operativos

20 de febrero de 2025

Lea detenidamente las siguientes instrucciones. No cumplir los requerimientos puede implicar la pérdida del examen.

- **Indique su nombre completo, número de cédula y número de parcial en cada hoja** (no se corregirán las hojas sin datos). Numere todas las hojas e indique la cantidad total de hojas en la primera.
- **Escriba las hojas de un solo lado y empiece cada problema en una hoja nueva**.
- Sólo se contestarán dudas de letra y no se aceptarán dudas en los últimos 30 minutos del examen.
- El examen es **SIN material** y dura **3 horas**. Al momento de finalizar el examen **no se podrá escribir absolutamente nada en las hojas**. El estudiante debe ponerse de pie e ir a la fila de entrega.
- Para aprobar el examen es necesario un mínimo de **60 puntos**. **Debe justificar todas las respuestas, siempre.**

Problema 1 (33 pts)

(a) Sobre los sistemas de archivos:

I. (3 pts) Explique los problemas que enfrenta su diseño.

> **Solución:** (Notas de teórico del tema Sistemas de archivos, slide 21)
> El diseño de un sistema de archivo enfrenta dos problemas:
>
> - Visión del usuario: implica definir los atributos y las operaciones válidas de los archivos, y definir la estructura de directorios para la organización de los archivos.
> - La creación de algoritmos y estructuras de datos para implementar la correspondencia entre el sistema de archivos lógico con los dispositivos físicos para almacenamiento de datos.

II. (4 pts) Describa las estructuras que se definen en los dispositivos físicos para implementarlos.

> **Solución:** (Notas de teórico del tema Sistemas de archivos, slide 26)
> En los dispositivos físicos deben definirse las siguientes estructuras
>
> - Bloque de control para el arranque por volumen (boot control block), que es necesario para iniciar el sistema operativo en el volumen. Típicamente es el primer bloque del volumen y puede estar vacío.
> - Bloque de control de volumen (volume control block), que contiene información de cada volumen (cantidad de bloques, tamaño del bloque, bloques usados y libres, etc.)
> - Estructura de directorios para cada sistema de archivos, necesarias para la organización de los archivos. Incluye información de directorios, nombres de archivos, punteros al archivo, etc.
> - Bloque de control del archivo (file control block, FCB), que contiene los detalles de cada archivo. En particular, identifica qué bloques del disco contienen la información del archivo.

III. (4 pts) Considere una estructura de i-nodos con bloques de 1024 bytes. Las entradas en los i-nodos dedican 64 bits al tamaño del archivo y 16 bits a los punteros de los bloques. Cada i-nodo tiene ocho entradas de direccionamiento directo, una de direccionamiento indirecto simple y otra de direccionamiento indirecto doble. Para cada proceso, se tiene una tabla de

archivos abiertos con un campo de 64 bits que indica la ubicación (desplazamiento) dentro del archivo abierto. Calcule el tamaño máximo de un archivo en el sistema descrito.

Solución: Se estudian todos los parámetros del sistema que pueden limitar el tamaño de un archivo:

- campo tamaño del archivo en el i-nodo (64 bits). El tamaño máximo de un archivo es 2^{64} bytes ($2^{64}/2^{10} = 2^{54}$ bloques).
- campo desplazamiento en la tabla de archivos abiertos (64 bits). El offset máximo que puede tener un archivo es 2^{64} bytes ($2^{64}/2^{10} = 2^{54}$ bloques).
- número máximo de bloques asignados a un archivo en su i-nodo. Directo: 8 bloques, indirecto simple: (tamaño de bloque/tamaño de puntero) 1024/2 = 512 bloques, indirecto doble: (1024/2)×(1024/2) 262.144 bloques. Total: 262.664 bloques.
- tamaño de un puntero a bloque (16 bits), el máximo número de bloques que se puede referenciar con un puntero es 2^{16} = 65536 bloques.

El valor más restrictivo es el de los punteros a bloques: 65536 bloques = 65536 KiB = 64 MiB

(b) Considere una estructura RAID 5:

I. (3 pts) Explique sus características y la cantidad de fallas de disco que puede tolerar.

Solución: (Notas de teórico del tema Estructuras de dispositivos masivos de datos, slides 43 y 44)
En RAID 5, los datos se dividen en bloques y se distribuyen entre los discos junto con bloques de paridad. Si el arreglo tiene N discos, cada raya (conjunto de bloques alineados en los discos) tendrá N-1 bloques de datos y un bloque de paridad. Los bloques de paridad se distribuyen equitativamente entre los discos, por lo que RAID 5 tolera la pérdida de un disco, sin importar cuál sea.

II. (3 pts) Indique los pasos para realizar una escritura de tamaño menor a un bloque.

Solución: (Notas de teórico del tema Estructuras de dispositivos masivos de datos, slide 45)
Para realizar una escritura de tamaño menor a un bloque se requiere:

1. Leer el bloque de datos donde se realizará la escritura.
2. Leer el bloque de paridad correspondiente a la raya del bloque de destino.
3. Comparar el bloque de datos original con la solicitud de escritura. Por cada bit modificado, invertir el bit correspondiente en el bloque de paridad.
4. Escribir el nuevo bloque de datos.
5. Escribir el bloque de paridad modificado.

(c) Sobre el mecanismo de creación de procesos en un sistema operativo genérico:

I. (4 pts) Explique qué decisiones deben tomarse en el diseño del sistema operativo.

Solución: (Notas de teórico del tema Procesos, slides 14 y 15)
Los procesos de un sistema son creados a partir de otro proceso. Al proceso creador se le denomina padre y al nuevo proceso hijo. En el diseño del sistema operativo se debe

decidir: i) cuáles recursos compartirán el proceso padre e hijo (nunca comparten memoria ni el estado de la CPU) y ii) qué sucede con los hijos cuando muere el padre (pueden morir también o cambiar de padre).

II. (3 pts) Explique cómo se gestiona la ejecución del nuevo proceso creado.

Solución: (Notas de teórico del tema Procesos, slide 15)
Un nuevo proceso tendrá un hilo de ejecución propio, con su propio program counter. El sistema generará un nuevo PCB para el proceso creado.

III. (3 pts) Presente un diagrama de ejecución e indique la salida del siguiente código C.

```
void main() {
   if (fork() == 0) {
        if (fork() == 0) {
             printf("5");
        } else if ((wait(NULL)) > 0)
                printf("2");
    } else {
        if (fork() == 0) {
            printf("1");
            exit(0);
        }
        if (fork() == 0)
            printf("4");
    }
    printf("0");
}
```

Solución: Diagrama de ejecución:

```
(padre) *----*----*--- 0 ---exit (return del main)
        |    |    |
        |    |    + (hijo3) --- 4 -- 0 --- exit (return del main)
        |    |
        |    +  (hijo2) --- 1 --- exit (explicito)
        |
        + (hijo1) --*--- wait -- 2--- 0 -- exit (return del main)
                    |
                    + (nieto 1) --- 5 -- 0 ---exit (return del main)
```

El orden de los dígitos que se imprimen como salida no está determinado. La única condición está dada por el wait: el 2 no puede imprimirse antes de que el proceso nieto1 imprima 5 y 0. Una posible salida es 50201040, pero otras combinaciones también son posibles (01405020, 10504020, 40150020, etc.).

(d) Sobre el tema planificación:

I. (3 pts) Defina sistema operativo con planificador no expropiativo y sistema operativo con planificador expropiativo.

Solución: (Notas de teórico del tema Planificación, slide 9)
Un sistema operativo con planificador no expropiativo asigna el recurso procesador a un proceso y hasta que éste no lo libere, ya sea porque finaliza su ejecución o se bloquea, no se vuelve a ejecutar el planificador.

Un sistema operativo con planificadores expropiativo puede expropiar el recurso procesador al proceso que lo tiene asignado. La expropiación puede darse cuando otro proceso entra en estado pronto (ya sea porque es nuevo o porque se desbloqueó) o porque el proceso en ejecución alcanzó un límite de tiempo.

II. (3 pts) Indique en qué momentos es invocado el planificador. Explique qué casos corresponden a un planificador expropiativo.

Solución: (Notas de teórico del tema Planificación, slide 8)
El planificador es invocado:

1. Cuando un proceso se bloquea (inicia una operación de E/S, espera a que termine un hijo, etc.)
2. Cuando un proceso cambia del estado ejecutando al estado pronto (por ejemplo, al ocurrir una interrupción).
3. Cuando ocurre una interrupción de E/S y un proceso pasa del estado bloqueado a pronto.
4. Cuando se crea un nuevo proceso.
5. Cuando un proceso finaliza su ejecución o libera voluntariamente la CPU.

Los casos de invocaciones 2, 3 y 4 se corresponden con un planificador expropiativo, ya que se quita el recurso procesador al proceso que estaba en ejecución.

Problema 2 (30 pts)

Sea un sistema operativo que implementa memoria virtual con paginación bajo demanda. Se requiere que los procesos puedan direccionar 16 TiB de memoria. La traducción de una dirección se realiza a través de tres niveles de tablas de páginas de igual tamaño. Las entradas en las tablas de páginas son de 32 bits y los marcos en memoria principal tienen un tamaño de 1MiB.

(a) (4 pts) Determine el largo y formato de las direcciones virtuales, justificando cada parte.

Solución: Dado que un proceso puede direccionar hasta 16 TiB, se requieren 44 bits para la dirección virtual. Como el tamaño de un marco es de 1 MiB, se necesitan 20 bits para el offset, ya que con esta cantidad de bits es posible direccionar cada byte del marco (2^{20}B = 1 MiB). Los 24 bits restantes de la dirección virtual se utilizan para referenciar las tablas de cada nivel. Al contar con tres niveles de tablas de páginas del mismo tamaño, un reparto equitativo de los bits asigna 8 bits a cada nivel.

N1 = 8bits	N2 = 8bits	N3 = 8bits	offset = 20 bits

(b) (4 pts) ¿Cuantos bytes ocupan las tablas de un proceso que tiene toda su memoria residente?

Solución: La tabla de primer nivel posee 2^8 entradas de 32 bits, lo que corresponde a 2^2 B por entrada. El tamaño de la tabla de primer nivel es $2^8 \times 2^2\text{B} = 2^{10}\text{B} = 1\text{KiB}$
Cada entrada en la tabla de primer nivel apunta a una tabla de segundo nivel, por lo que el número máximo de tablas de segundo nivel es 2^8. Dado que el tamaño de cada tabla de segundo nivel es igual que el de la tabla de primer nivel, el espacio total ocupado por las tablas de segundo nivel es $2^8 \times 1\text{KiB} = 2^8 \times 2^{10}\text{B} = 2^{18}\text{B} = 256\text{KiB}$.

Siguiendo el mismo razonamiento, en el tercer nivel contiene un máximo de 2^8 tablas por cada tabla de nivel 2, resultando un total de $2^8 \times 2^8 \times 2^{10}\mathrm{B} = 2^{26}\mathrm{B} = 64\mathrm{MiB}$.
Por lo tanto, el tamaño total ocupado por todas las tablas es $64\mathrm{MiB} + 256\mathrm{KiB} + 1\mathrm{KiB} \approx 64\mathrm{MiB}$.

(c) Suponga que el sistema permite a un marco alojar una única tabla:

I. (4 pts) Explique si el sistema sufre algún tipo de fragmentación.

Solución: El sistema sufre de fragmentación interna. Dado que una tabla es más pequeña que un marco, todos los marcos tendrán espacio asignado que nunca será ocupado por completo por la tabla que contienen. El sistema no sufre de fragmentación externa, porque este problema es resuelto por la técnica de paginación.

II. (4 pts) ¿Cuál es el costo en el peor caso (en MiB) de la restricción impuesta por el sistema?

Solución: Como el sistema operativo no permite más de una tabla por marco, y dado que el tamaño de una tabla (2^{10} B) es menor que el tamaño de un marco (2^{20} B), cada tabla debe ocupar un marco. En el peor caso se tienen $2^{16} + 2^8 + 1$ tablas y cada una utiliza un marco de 1 MiB ($2^{20}\mathrm{B}$). El total de memoria asignada es $(2^{16} + 2^8 + 1) \times 2^{20} = 2^{36} + 2^{28} + 2^{20}\mathrm{B}$. En el peor caso se asignan aproximadamente 65 GiB de memoria, cuando lo realmente necesario es aproximadamente 64 MiB. El porcentaje de utilización es de 0.1 % de cada marco.

(d) (4 pts) Realice un diagrama que muestre como se lleva a cabo la traducción de una dirección virtual en el sistema propuesto. Proponga y explique un diseño alternativo del sistema que permita, sin cambiar la estructura de las tablas de páginas, acelerar la traducción de direcciones.

Solución: La Figura 1 presenta el diagrama que muestra como se lleva a cabo la traducción de una dirección virtual

Para mejorar el tiempo de traducción de direcciones se puede recurrir a una caché de la tabla de páginas, que requiere soporte a nivel de hardware. Se utiliza una pequeña caché asociativa y de rápido acceso para la tabla de páginas: la Translation Look-aside Buffer (TLB). Cada entrada de la TLB consta de dos partes: una clave (tag) y un valor (el número de frame). La búsqueda de una clave en la TLB se realiza de manera simultánea entre todas las claves (tags). La Figura 2 muestra la incorporación de un TLB al diseño anterior para acelerar las búsquedas de números de frame en los casos de hit. Se realizan los mismos procedimientos que en un escenario sin TLB para los casos de miss.

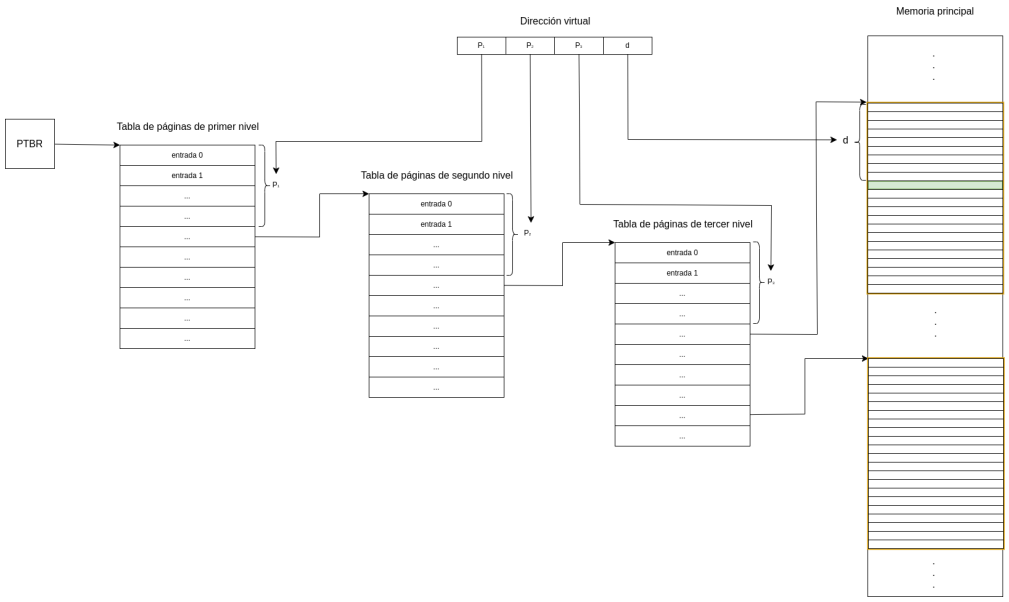

Figura 1: Diagrama que muestra como se lleva a cabo la traducción de una dirección virtual.

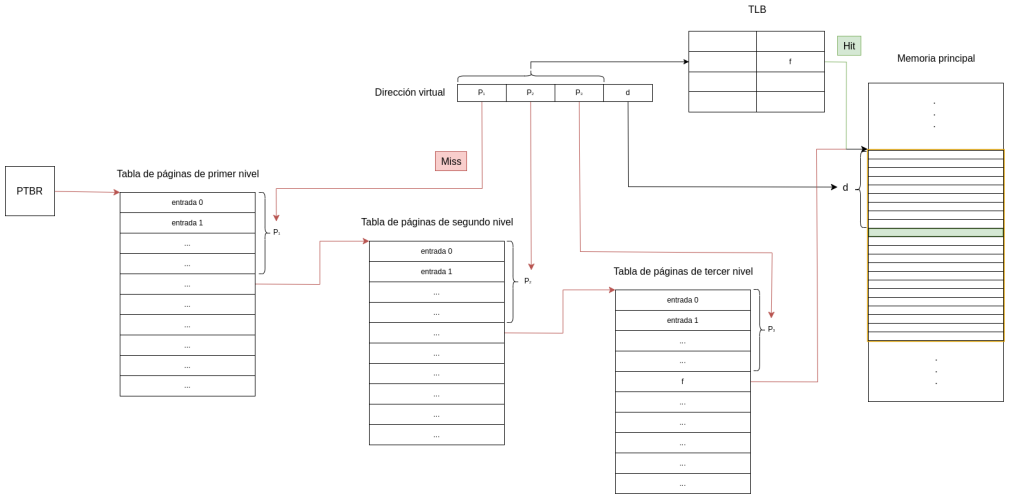

Figura 2: Diseño alternativo del sistema que utiliza una TLB para acelerar el proceso de traducción

(e) (5 pts) Uno de los procesos que ejecutan es el salvapantalla, que va pintando la pantalla píxel a píxel con un color diferente de una paleta de 8 bytes, formado por el primer byte de cada imagen en un conjunto de imágenes distintas de 1MiB cada una. El proceso no tiene páginas en memoria principal y se le asignan 4 marcos. Se aplica una política de reemplazo LRU. Presente un esquema en el tiempo para mostrar los fallos de página y el estado de la memoria en el peor caso.

Solución: El peor caso ocurre cuando se da un fallo de página por cada acceso a memoria. Un escenario posible de este caso se presenta cuando se solicitan imágenes distintas a las ya cargadas en un marco.

Las primeras 4 imágenes a cargar siempre provocarán un fallo de página. Al cargar la quinta imagen, ocurre un fallo de página y es necesario reemplazar la primera imagen que se cargó.

Lo mismo sucede al cargar la sexta imagen, que provoca un fallo y reemplaza la segunda imagen. Este patrón se repite con la séptima y octava imagen. En este punto, ya se tienen las 8 imágenes necesarias y se produjeron 8 fallos de página. Si se repite el proceso, ocurrirá exactamente lo mismo. La Tabla 1 presenta el esquema en el tiempo que muestra los fallos de página y el estado de la memoria en el peor caso.

imagen	fallo	memoria
1	FALLO	[1, -, -, -]
2	FALLO	[2, 1, -, -]
3	FALLO	[3, 2, 1, -]
4	FALLO	[4, 3, 2, 1]
5	FALLO	[5, 4, 3, 2]
1	FALLO	[1, 5, 4, 3]
2	FALLO	[2, 1, 5, 4]
3	FALLO	[3, 2, 1, 5]

Tabla 1: Esquema en el tiempo que muestra los fallos de página y el estado de la memoria en el peor caso

(f) (5 pts) Suponga que el programa usa todas las imágenes de la tabla a continuación, en la misma secuencia, una y otra vez. Manteniendo la asignación de 4 marcos y política LRU. Indique el porcentaje de fallos de página y justifique su respuesta.

imagen	dirección virtual
1	(6, 24, 7 , 0)
2	(126, 78, 20, 0)
3	(15, 55, 255, 0)
4	(5, 44, 7, 0)
1	(6, 24, 7, 0)
5	(88, 88, 88, 0)
2	(126, 78, 20, 0)
3	(15, 55, 255, 0)

Solución: Los primeros cuatro accesos a memoria para obtener el primer byte generan un fallo de página, ya que el proceso recién inicia y aún no ha utilizado memoria.

En el primer ciclo del patrón, el estado de la memoria queda descrito según la Tabla 2.

Al ejecutar el mismo orden de solicitudes de imágenes, se el patrón presentado en la Tabla 3.

El estado de la memoria al final del segundo ciclo coincide con el inicio de la secuencia, lo que significa que los fallos se repetirán de la misma forma.

Considerando la última tabla, la tasa de fallos de página es del 50 %.

imagen	fallo	memoria
1	FALLO	[1, -, -, -]
2	FALLO	[2, 1, -, -]
3	FALLO	[3, 2, 1, -]
4	FALLO	[4, 3, 2, 1]
1	NO FALLO	[1, 4, 3, 2]
5	FALLO	[5, 1, 4, 3]
2	FALLO	[2, 5, 1, 4]
3	FALLO	[3, 2, 5, 1]

Tabla 2: Esquema en el tiempo que muestra los fallos de página y el estado de la memoria en el primer ciclo

imagen	fallo	memoria
1	NO FALLO	[1, 3, 2, 5]
2	NO FALLO	[2, 1, 3, 5]
3	NO FALLO	[3, 2, 1, 5]
4	FALLO	[4, 3, 2, 1]
1	NO FALLO	[1, 4, 3, 2]
5	FALLO	[5, 1, 4, 3]
2	FALLO	[2, 5, 1, 4]
3	FALLO	[3, 2, 5, 1]

Tabla 3: Esquema en el tiempo que muestra los fallos de página y el estado de la memoria en el segundo ciclo

Problema 3 (37 pts)

Se desea modelar un sistema automático para la administración de las cinco pistas de aterrizaje de un aeropuerto. Los aviones que desean despegar o aterrizar solicitan al sistema que se les asigne una pista de aterrizaje y esperan hasta que el sistema les indique que pista pueden usar.

Los aviones que quieren aterrizar tendrán prioridad sobre los que quieren despegar y dentro de cada grupo se atenderán por orden de llegada. Si algún avión que quiere despegar ha esperado más de 15 minutos se pasará a atender con la misma prioridad que los que quieren aterrizar.

Puede pasar que un avión quiera aterrizar en una situación de emergencia. En ese caso tendrá prioridad sobre todos los demás y previo al aterrizaje de emergencia se deberán liberar todas las pistas de aterrizaje de modo de facilitar el trabajo de los bomberos y el personal de seguridad. En caso de haber varios vuelos en situación de emergencia aterrizarán también por orden de llegada, de a uno a la vez.

Modelar en ADA los procesos avión y aeropuerto. Se pueden usar tareas auxiliares.

Se dispone de las siguientes funciones auxiliares:

- que_hacer (aterrizar: out boolean, emergencia: out boolean), ejecutada por el avión para saber si tiene que aterrizar o despegar y en caso de aterrizar si se encuentra en situación de emergencia.
- aterrizar (int pista) y despegar (int pista), ejecutadas por el avión para aterrizar y despegar.

Solución:

```
task Aeropuerto is
   entry Aterrizaje_Emergencia (Pista : out Integer);
```

```
   entry Aterrizaje (Pista : out Integer);
   entry Despegue_Prioritario (Pista : out Integer);
   entry Despegue (Pista : out Integer);
   entry Liberar_Pista (Pista : in Integer);
   entry Liberar_Pista_Emergencia();
end Aeropuerto;

task body Aeropuerto is
   Pistas : array [1..5] of Boolean;
   Pistas_Libres : Integer;
   PistaLibre : Integer;

   function Pista_Libre() : Integer is
   begin
      for i in 1..5 loop
         if Pistas[i] then
            return i;
         end if;
      end loop;
      return 0;
   end Pista_Libre;

begin
   for i in 1..5 loop
      Pistas[i] := True;
   end loop:
   Pistas_Libres := 5;

   loop
      PistaLibre := Pista_Libre();
      select
         accept Liberar_Pista (Pista : in Integer) do
            PistaLibre := Pista;
         end Liberar_Pista;
    Pistas[PistaLibre] := True;
        Pistas_Libres++;
      or
         when Pistas_Libres = 5 =>
            accept Aterrizaje_Emergencia (Pista : out Integer) do
                Pista := PistaLibre;
            end Aterrizaje_Emergencia;
            -- bloquear aeropuerto hasta que termine el aterrizaje de emergencia
            accept Liberar_Pista_Emergencia();
      or
         when Aterrizaje_Emergencia'Count = 0 and Pistas_Libres > 0 =>
            accept Aterrizaje (Pista : out Integer) do
                Pista := PistaLibre;
            end Aterrizaje;
            Pistas[PistaLibre] := False;
            Pistas_Libres--;
      or
         when Aterrizaje_Emergencia'Count = 0 and Pistas_Libres > 0 =>
            accept Despegue_Prioritario (Pista : out Integer) do
                Pista := Pista_;
            end Despegue_Prioritario;
            Pistas[PistaLibre] := False;
```

```
            Pistas_Libres--;
      or
         when Aterrizaje_Emergencia'Count = 0 and Aterrizaje'Count = 0 and
               Despegue_Prioritario'Count = 0 and Pistas_Libres > 0 =>
            accept Despegue (Pista : out Integer) do
               Pista := PistaLibre;
            end Despegue;
            Pistas[PistaLibre] := False;
            Pistas_Libres--;
      end select;
   end loop;
end Aeropuerto;

task type Avion is
end Avion;

task body Avion is
   Aterrizar, Emergencia : Boolean;
   Pista : Integer;
begin
   Que_Hacer(Aterrizar, Emergencia);
   if Aterrizar then
      if Emergencia then
         Aeropuerto.Aterrizaje_Emergencia(Pista);
         Aterrizar(Pista);
         Aeropuerto.Liberar_Pista_Emergencia();
      else
         Aeropuerto.Aterrizaje(Pista);
         Aterrizar(Pista);
         Aeropuerto.Liberar_Pista(Pista);
      end if;
   else
      select
        Aeropuerto.Despegue(Pista);
      or
        delay 900.0; -- 15 minutos en segundos
        Aeropuerto.Despegue_Prioritario(Pista);
      end select;
      Despegar(Pista);
      Aeropuerto.Liberar_Pista(Pista);
   end if;
end Avion;

begin
end;
```