Examen Julio 2008

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.

Conceptos generales

1. ¿Qué es un sistema operativo?
2. En un sistema de propósito general, ¿qué tipo (simétrico o asimétrico) de sistema operativo utilizaría? Justifique.
3. Describa lo que entiende por un sistema operativo de micronúcleo (*microkernel*). A su vez, describa sus ventajas.

Administración de Procesos

4. ¿Qué estado de proceso debe tener un proceso cuando realiza una invocación a un llamado a sistema? Justifique.
5. El despachador (*dispatcher*) realiza el cambio de contexto de dos procesos en un procesador. Enumere las tareas que debe realizar.

Administración de memoria

6. ¿Qué función cumple el cargador (*loader*) del sistema?

Memoria virtual

7. Describa, a través de un esquema, como se divide en componentes la memoria virtual de un proceso.
8. Sea un sistema de 16 bits, que utiliza memoria virtual con páginas de 256 bytes, y con una estructura de tabla de páginas de dos niveles. Construya una dirección virtual para acceder al byte 5119.
   Nota:
   - La componente para el número de página se debe dividir a la mitad (para el primer y segundo nivel).
   - Trabaje siempre con valores que empiezan en 0. Ej. El primer valor de la entrada de una tabla es 0.
9. En un sistema que implementa memoria virtual, de qué forma dos hilos (*threads*) de un mismo proceso pueden compartir información (estructuras de datos).

Dispositivos masivos

10. ¿Los sistemas RAID (*Redundant Array of Inexpensive Disk*), a qué mejoras apuntan?

Subsistema de Entrada/Salida

11. ¿Qué servicios brinda el núcleo del sistema operativo para el manejo de Entrada/Salida?

Solución:

Ver libro teórico y/o notas del curso

Problema 2

Se desea modelar en ADA una agencia de viajes la cual dispone de un portero, dos vendedores, un operador de reservas y una capacidad para albergar hasta cinco clientes dentro de ella.

Los clientes se contactarán con su vendedor preferido y construirán juntos el itinerario del viaje.

El vendedor, una vez que obtiene el itinerario de un cliente, solicitará las reservas correspondientes al operador de reservas. El vendedor esperará hasta dos minutos máximo por la confirmación o no de la reserva por parte del operador. Si esta no llegara dentro del lapso mencionado, el vendedor le dirá al cliente que contacte al operador de reserva directamente (de forma de dejar libre al vendedor para que ordene su escritorio).

El operador de reserva recibe los itinerarios de los vendedores, los procesa y con ellos obtiene la confirmación o no del mismo. Además el operador atenderá a los clientes a los que se les indicó que debían comunicarse directamente con él para saber si se confirmó o no su reserva.

Notas:

- No se podrá utilizar tareas auxiliares.
- El cliente solo podrá comunicarse con el vendedor y con el operador a través de una entrada para cada uno, dejándose en libertad quien hace la solicitud y quien el accept.
- El vendedor solamente ordenará su escritorio si tuvo que esperar los dos minutos y no recibió respuesta del operador

Se dispone de las siguientes funciones:

- armar_itinerario (): itinerario Ejecutada por el vendedor
- ordenar_escritorio() Ejecutada por el vendedor.
- vendedor_preferido():1..2 Ejecutada por el cliente, retorna el número de vendedor.
- procesar_itinerario(itinerario): boolean Ejecutada por el operador, devuelve la confirmación de la reserva.

Solución:

```
Task cliente is
End cliente

Task body cliente is
      portero.damenum(num);
      vendedores[vendedor_preferido()].atender(num, confirm, veroperador);
      if (veroperador) then
            operador.dameconfirm(num, confirm);
      endif
      portero.salir(num);
end cliente


Task type vendedor is
      Entry atender(num:in integer, confirm:out boolean, veroperador:out boolean);
      Entry respuesta(conf:in boolean);
      Entry num(id: in integer);
end vendedor

Task body vendedor is
      var numvend: integer
      accept num (id:in integer)
```

```
            numvend = id;
      end accept

      loop
         accept atender (num:in integer, confirm:out boolean, veroperador:out boolean)
            itin=armar_itinerario();
            operador.pedido(num, numvend, itin);
            select
                 accept respuesta (conf: in boolean)
                         confirm=conf;
                 end accept
                 veroperador=false;
                 ordenar=false;
             or delay 120
                  veroperador=true;
                  ordenar=true;
             end select
          end accept

          if (ordenar) then
             ordenar_escritorio();
             accept respuesta(conf:in boolean);
          endif
      end loop
end vendedor

vendedores = array[1..2] of vendedor;

Task operador is
      Entry pedido (num:in integer, numvend:in integer, itin: in itinerario)
      Entry dameconfim (num:in integer, conf:out boolean)
end operador

Task body operador is
   loop
      select
            accept pedido (num:in integer, numvend:in integer, itin: in itinerario)
                  numcli=num;
                  itincli=itin;
                  numvendvend=numvend;
            end accept
            confirmaciones[numcli]=procesar_itinerario(itincli);
            vendedor[numvendvend].respuesta(confirmaciones[numcli];
      or
            accept dameconfirm(num: in integer, conf:out boolean)
                  conf=confirmaciones[num];
            end accept
    end select
    end loop
end operador


Task portero is
      Entry damenum(num: out integer);
      Entry salir(num: in integer);
end portero

Task body portero is
      var
            libres = array [1..5] of boolean
            numero: integer

      vendendor[1].num(1);
      vendendedor[2].num(2);

      loop
            select
```

```
                    when hayLibres(libres) =>
                    accept damenum(num: in integer)
                          num = buscarLibre(libres);
                    end accept
                    libres[num] = false;
              or
                    accept salir(num:in integer)
                          numero = num;
                    end accept
                    libres[numero] = true;
              end select
       end loop
end portero
```

Problema 3

Se desea realizar una implementación en capas de un sistema de archivos. Para esto ya se cuenta con la capa que implementa un sistema de archivos anónimo, con las siguientes funciones:

- **`FileHandle := SAA_Abrir(inout fileNum: Integer [0..N], disposición : [Nuevo, Existente])`**
- **`SAA_Cerrar (handle : FileHandle)`**
- **`SAA_Leer (handle : FileHandle, buffer : char[])`**
- **`SAA_Escribir (handle : FileHandle, buffer: char[])`**

Se desea implementar un sistema de archivos con nombres, en donde cada nombre será de la forma "A/B/C/…", donde este denota el camino absoluto para hallar el archivo en una estructura de directorio arborescente. Cada nivel será una cadena de hasta **MAX_CARACTERES** de largo. Se utilizaran los servicios del sistema de archivos anónimo.

Se desea implementar un sistema de archivos con nombres, en donde cada nombre será de la forma **"A/B/C/…X.ext"**. **"A/B/C/…"** denota el camino absoluto para hallar el archivo en una estructura de directorio en forma de árbol y "**X.ext"** será el nombre del archivo.

Se pide implementar las siguientes operaciones

- **`FileHandle := SA_Abrir (nombreArchivo : string, disposición : [Nuevo, Existente])`**
- **`SA_Cerrar (handle : FileHandle)`**
- **`SA_Leer (handle : FileHandle, buffer : char[])`**
- **`SA_Escribir (handle : FileHandle, buffer: char[])`**

Considerar que:

- El estado inicial del SAA es vacío
- Todas las rutas son absolutas
- En el SA no está acotada la cantidad de niveles, excepto por la cantidad de archivos del SAA
- En la implementación de SAA_Abrir si no existe el camino falla.
- En la implementación de SAA_Abrir si si Disposicion=Nuevo, devuelve el numero de archivo asignado, además del FileHandle

Se dispone de las siguientes funciones auxiliares:

- **`Head(Camino: string) : string`**

  Recibe un camino de la forma "A/B/C/….X.ext", y retorna "A", esto es, el primer elemento de la lista. En caso de recibir un archivo "X.ext" retorna una cadena vacia. En cualquier otro caso, retorna una cadena vacia.

- **`Tail(Camino: string) : string`**

  Recibe un camino de la forma "A/B/C/….X.ext", y retorna "B/C/….X.ext", esto es, sacando el primer elemento de la lista, retorna el resto de la misma. En caso de recibir un archivo "X.ext" retorna una cadena vacia. En cualquier otro caso, retorna una cadena vacia.

Solución:

Leer , escribir y cerrar son directas, invocando la capa anterior.

Para SA_Abrir:

```
         Procedure SA_Abrir ((NombreArchivo : string,
                 disposición : TipoDisposición)
         Begin
            SA_Translate (Head(NombreArchivo,"/")+".Dir",Tail(NombreArchivo,"/"),
                 0, Disposicion)   % Directorio raiz
         End;

         SA_Translate (Nombre, RestoNombre: String;
                 Directorio :  TipoFileNum;
                 Disposicion : TipoDisposición)
                 :  TipoFileNum;   % Resultado de la traduccion

         Begin
         % Abro el directorio
         FileHandleDir := SAA_Abrir (Directorio,[Existente]);
         While Not  SAA_EOF(FileHandleDir) and
                 Nombre <> Dir.Nombre Do
            SA_Leer (FileHandleDir,Dir);
            If SAA_EOF(FileHandleDir) then % no existe
            If Length(RestoNombre) > 0 then % No existe parte del camino -> Error
              Return (-1)
            Else  % No existe al archivo
                If Disposicion = [Nuevo] then % Lo creo
                  FileHandleArch := SAA_Abrir (FileNumArch,[Nuevo]);
                  Return (FileNumArch);  % devuelvo file Num de nuevo arch.
                Else
                  Return (-2); % no existe archivo
                End If
            End If
         Else     % Existe Dir o Archivo
            If Length(RestoNombre) > 0 then % Resuelvo resto del camino
                Return (SA_Translate (
                        (Head(RestoNombre,"/")+".Dir",Tail(RestoNombre,"/"),
                        Dir.FileNum,
                        Disposicion )
                        )
            Else  % Existe al archivo
                If Disposicion = [Nuevo] then
                  Return (-3) % Ya existe, Duplicado
                Else
                        Return  (Dir.FileNum);  %  devuelvo  file  Num  de  arch.
         existente
            End If

         End If
End;
```
