UNIVERSIDAD NACIONAL DE LA PLATA

Programación Concurrente - Practica 5

Tomatis, Vicens, Torchia, Olmos, Manzin, Petraccaro, Traberg, Villareal

(Ejercicio 1)

Consigna:
Se requiere modelar un puente de un único sentido que soporta hasta 5 unidades de peso. El peso de los vehículos depende del tipo: cada auto pesa 1 unidad, cada camioneta pesa 2 unidades y cada camión 3 unidades. Suponga que hay una cantidad innumerable de vehículos (A autos, B camionetas y C camiones). Analice el problema y defina qué tareas, recursos y sincronizaciones serán necesarios/convenientes para resolver el problema.

a) Realice la solución suponiendo que todos los vehículos tienen la misma prioridad.

b) Modifique la solución para que tengan mayor prioridad los camiones que el resto de los vehículos.

Respuesta:

Ejercicio 1

```
Procedure ej1 is
    task type Camion is
    end Camion;

    arrCamion: array(1..N/3) of Camion;
    task Body Camion is
    begin
        Puente.entrarCamion()
        -- pasa
        Puente.salir(3)
    end Camion;
```

Auto

```
    task type Auto is
    end Auto;

    arrAuto: array(1..N/3) of Auto;
    task Body Auto is
    begin
        Puente.entrarAuto()
        -- pasa
        Puente.salir(1)
    end Auto;
```

Camioneta

```
    task type Camioneta is
    end Camioneta;

    arrAuto: array(1..N/3) of Camioneta;
    task Body Camioneta is
    begin
        Puente.entrarCamioneta()
        -- pasa
        Puente.salir(2)
    end Camioneta;
```

Continuacion

```
    task Puente is
        Entry entrarCamioneta();
        Entry entrarAuto();
        Entry entrarCamion();
        Entry salir(peso: IN Integer);
    end Puente;
    task body Puente is
        totalPeso:Integer:=5;
    begin
        loop
            select
                accept salir(peso: IN Integer) do
                    totalPeso+=peso;
                end salir;
            or when (totalPeso>=3) =>
                accept entrarCamion();
                totalPeso-=3
            or when (totalPeso>=2) =>
                accept entrarCamioneta();
                totalPeso-=2
            or when (totalPeso>=1) =>
                accept entrarAuto();
                totalPeso-=1
        end loop;
    end Puente;
begin
end ej1;
```

b) Modifique la solución para que tengan mayor prioridad los camiones que el resto de los vehículos.

Ejercicio 1

```
Procedure ej1 is
    task type Camion is
    end Camion;

    arrCamion: array(1..N/3) of Camion;
    task Body Camion is
    begin
        Puente.entrarCamion()
        -- pasa
        Puente.salir(3)
    end Camion;
```

Auto

```
    task type Auto is
    end Auto;

    arrAuto: array(1..N/3) of Auto;
    task Body Auto is
    begin
        Puente.entrarAuto()
        -- pasa
        Puente.salir(1)
    end Auto;
```

Camioneta

```
    task type Camioneta is
    end Camioneta;

    arrAuto: array(1..N/3) of Camioneta;
    task Body Camioneta is
    begin
        Puente.entrarCamioneta()
        -- pasa
        Puente.salir(2)
    end Camioneta;
```

Continuacion

```
    task Puente is
        Entry entrarCamioneta();
        Entry entrarAuto();
        Entry entrarCamion();
        Entry salir(peso: IN Integer);
    end Puente;
    task body Puente is
        totalPeso:Integer:=5;
    begin
        loop
            select
                accept salir(peso: IN Integer) do
                    totalPeso+=peso;
                end salir;
            or when (totalPeso>=3) =>
                accept entrarCamion();
                totalPeso-=3
            or when (totalPeso>=2 and
            ↪  entrarCamion'count=0) =>
                accept entrarCamioneta();
                totalPeso-=2
            or when (totalPeso>=1 and
            ↪  entrarCamion'count=0) =>
                accept entrarAuto();
                totalPeso-=1
        end loop;
    end Puente;
begin
end ej1;
```

(Ejercicio 2)

Consigna:
Se quiere modelar el funcionamiento de un banco, al cual llegan clientes que deben realizar un pago y retirar un comprobante. Existe un único empleado en el banco, el cual atiende de acuerdo con el orden de llegada. Los clientes llegan y si esperan más de 10 minutos se retiran sin realizar el pago. **Respuesta:**

Ejercicio 1

```
Procedure ej2 is

    Task type Cliente
        Entry pagar (pago:OUT int; comp: IN text);
    end Cliente;

    arrClientes: array(1..N) of Cliente;

    Task Body Cliente is
        pago: int = generarPago()
        comp: text;
    Begin
        SELECT
            Empleado.pagar(pago, comp);
        OR DELAY 10mins
            NULL;
        End SELECT;
    End Cliente;
```

Continuacion

```
    Task Empleado is
        Entry pagar (pago:IN int; comp: OUT text);
    end Empleado;

    Task Body Empleado is
    Begin
        loop
            Accept pagar(pago: IN int; comp: OUT text)
            ↪  do
                comp := generarComprobante(pago);
            End pagar;
        End loop;
    End Empleado;

Begin
    NULL;
End ej2;
```

(Ejercicio 3)

Consigna:
Se dispone de un sistema compuesto por 1 central y 2 procesos periféricos, que se comunican continuamente. Se requiere modelar su funcionamiento considerando las siguientes condiciones:

- La central siempre comienza su ejecución tomando una señal del proceso 1; luego toma aleatoriamente señales de cualquiera de los dos indefinidamente. Al recibir una señal de proceso 2, recibe señales del mismo proceso durante 3 minutos.
- Los procesos periféricos envían señales continuamente a la central. La señal del proceso 1 será considerada vieja (se deshecha) si en 2 minutos no fue recibida. Si la señal del proceso 2 no puede ser recibida inmediatamente, entonces espera 1 minuto y vuelve a mandarla (no se deshecha).

Respuesta:

Central

```
Procedure ej3 is

Task Central is
    entry enviar1(mensaje:IN String);
    entry enviar2(mensaje:IN String);
    entry terminoTimer;
End Central;

Task body Central is
    Accept enviar1();
    loop
        Select
            Accept enviar1(mensaje:IN String);
        or
            Accept enviar2(mensaje:IN String);
            Timer.empezarTimer();
            seguir=true;
            while seguir loop
                Select
                    Accept terminoTimer();
                    seguir=false;
                or when(terminoTimer'count=0) =>
                    Accept enviar2(mensaje:IN String);
                end select;
            end loop;
        End Select;
    end loop;
End body Central;
```

Timer

```
Task Timer is
    entry empezarTimer;
End Timer;

Task body Timer is
    loop
        Accept empezarTimer;
        delay 3min;
        Central.terminoTimer();
    end loop;
End body Timer;
```

Periferico 1

```

Task type Periferico1 is
End Periferico;

Task Body Periferico1 is
    loop
        select
            mensaje = GenerarMensaje()
            Central.enviar1(mensaje);
        or delay 2min
            Null;
        end select;
    end loop;
End Periferico;

```

Periferico 2

```
Task type Periferico2 is
End Periferico;

Task Body Periferico2 is
    loop
        mensaje = GenerarMensaje()
        seguir=true;
        loop seguir
            select
                Central.enviar2(mensaje);
                seguir=false;
            else
                delay 1min;
            end select;
        end loop;
    end loop;
End Periferico;

Begin
    NULL;
End ej3;
```

(Ejercicio 4)

Consigna:
En una clínica existe un médico de guardia que recibe continuamente peticiones de atención de las E enfermeras que trabajan en su piso y de las P personas que llegan a la clínica ser atendidos.
Cuando una persona necesita que la atiendan espera a lo sumo 5 minutos a que el médico lo haga, si pasado ese tiempo no lo hace, espera 10 minutos y vuelve a requerir la atención del médico. Si no es atendida en tres veces, se enoja y se retira de la clínica.
Cuando una enfermera requiere la atención del médico, si este no lo atiende inmediatamente le hace una nota y se la deja en el consultorio para que esta resuelva su pedido en el momento que pueda (el pedido puede ser que el médico le firme algún papel). Cuando la petición ha sido recibida por el médico o la nota ha sido dejada en el escritorio, continúa trabajando y haciendo más peticiones.
El médico atiende los pedidos dándole prioridad a los enfermos que llegan para ser atendidos. Cuando atiende un pedido, recibe la solicitud y la procesa durante un cierto tiempo. Cuando está libre aprovecha a procesar las notas dejadas por las enfermeras.
Respuesta:

Medico

```
Procedure ej1 is

Task Medico is
  PedirAtencion();
  pedirNota();
  agarrarNota();
End Medico;

Task body Medico is
  loop
    select
      accept PedirAtencion()
          -- Atencion
      end PedirAtencion;
    or
      when (PedirAtencion'count=0) =>
        select
          accept pedirNota(nota: IN text) do
              firmarNota(nota);
          end pedirNota;
        else
          Buffer.agarrarNota(nota)
          if (nota!="") then
            firmarNota(nota);
          end if;
        end select;
    end select;
  end loop;
End body Medico;
```

Enfermera

```
Task type Enfermera is
End Enfermera;

enfermeras: array (1..E) of Enfermera;

Task body Enfermera is
    id:Integer;
begin
    loop
        nota = GenerarNota(trabajo, papel, lapicera);
        select
            Medico.pedirNota(nota);
        else
            Buffer.Nota(nota);
        end select;
    end loop;
End body Enfermera;
```

Persona

```
Task type Persona is
End Persona;

personas: array (1..P) of Persona;

Task Body Persona is
    esperando:Boolean:=true;
    iteraciones:Integer:=0;
begin
    while iteraciones<3 and esperando loop
        select
            Medico.PedirAtencion();
            esperando:=false;
        or delay (5min)
            iteraciones++
            if iteraciones<3 then
                delay (10min);
            end if;
        end select;
    end loop;
    if esperando
        Enojar(self.animo);
        Irse(self.location);
    end if;
End body Persona;
```

Escritorio

```
Task Buffer is
    Entry Nota(id:IN Integer;nota:IN String);
    Entry AgarrarNota(nota:OUT String);
end Buffer;

Task Body Buffer is
    cola:ColaNotas;
begin
    loop
        select
            accept Nota(nota: IN String)
                push(ColaNotas,nota)
            end Nota;
        or
            accept AgarrarNota(nota: OUT String)
                if empty(ColaNotas) then
                    nota:=""
                else
                    nota:=pop(ColaNotas)
                end if;
            end AgarrarNota;
        end select;
    end loop
End body Buffer;

Begin
    null;
End ej4;
```

(Ejercicio 5)

Consigna:
En un sistema para acreditar carreras universitarias, hay UN Servidor que atiende pedidos de U Usuarios de a uno a la vez y de acuerdo con el orden en que se hacen los pedidos. Cada usuario trabaja en el documento a presentar, y luego lo envía al servidor; espera la respuesta de este que le indica si está todo bien o hay algún error. Mientras haya algún error,vuelve a trabajar con el documento y a enviarlo al servidor. Cuando el servidor le responde que está todo bien, el usuario se retira. Cuando un usuario envía un pedido espera a lo sumo 2 minutos a que sea recibido por el servidor, pasado ese tiempo espera un minuto y vuelve a intentarlo (usando el mismo documento).
Respuesta:

Servidor

```
Procedure ej5 is

Task Servidor is
    Entry recibir (pedido: IN text; ok: OUT bool);
End Servidor;

Task body Servidor is
    loop
        Accept recibir (pedido: IN text; ok: OUT bool)
        ↪ do
            ok = chequear(pedido);
        end recibir;
    end loop;
End body Servidor;
```

Usuario

```
Task type Usuario is
End Usuario;

usuarios: array (1..U) of Usuario;

Task body Usuario is
    text documento;
    bool esCorrecto = false;
    bool seguir = true;
    documento = trabajar();
    loop seguir
        select
            Servidor.recibir(documento, esCorrecto);
            if !esCorrecto then
                documento = corregir(documento);
            else
                seguir = false;
            end if;
        or delay(2)
            delay(1);
        end select;

    end loop;
End body Usuario;

Begin
    null;
End ej5;
```

(Ejercicio 6)

Consigna:
En una playa hay 5 equipos de 4 personas cada uno (en total son 20 personas donde cada una conoce previamente a que equipo pertenece). Cuando las personas van llegando esperan con los de su equipo hasta que el mismo esté completo (hayan llegado los 4 integrantes), a partir de ese momento el equipo comienza a jugar. El juego consiste en que cada integrante del grupo junta 15 monedas de a una en una playa (las monedas pueden ser de 1, 2 o 5 pesos) y se suman los montos de las 60 monedas conseguidas en el grupo. Al finalizar cada persona debe conocer el grupo que más dinero junto. Nota: maximizar la concurrencia. Suponga que para simular la búsqueda de una moneda por parte de una persona existe una función Moneda() que retorna el valor de la moneda encontrada.
Respuesta:

Persona

```
Procedure ej6 is

Task type Persona is
End Persona;

personas: array (1..20) of Persona;

Task body Persona is
  equipo:Integer:=saberEquipo();
  total:Integer:=0;
  id: Integer;
begin
  accept iden(id: IN int);
  equipos[equipo].llegue();
  accept equipos[equipo].empezar();
  for i in 1..15 loop
    total:=total+Moneda();
  end loop;
  Juego.monedas(total,grupo);
  accept maximo(grupoMax:IN Integer)
    if grupoMax == equipo then
      puts "Bokeeeee";
    end if;
    -- yay ya se el grupoMax
  end maximo;
End body Persona;
```

Equipo

```
Task type Equipo is
End Equipo;

equipos: array (1..5) of Equipo;

Task body Equipo is
  jugadores: array (1..4) of int;
  for i in 1..4 do
    accept llegue(in id int);
    jugadores[i]= id;
  end loop;
  for i in 1..4 loop
    jugadadores[i].empezar();
  end loop;
End body Equipo;
```

Juego

```
Task type Juego is
    entry monedas(total,grupo:IN Integer);
End Juego;

Task body Playa is
  maxGrupo,maxCantidad,grupoActual:Integer:=-1;
  arrContador(1..5) of Integer; -- inicializado en 0
  arrLlegaron(1..5) of Integer; -- inicializado en 0
begin
  for i in 1..20 loop
    accept monedas(total,grupo:IN Integer) is
    begin
      grupoActual:=grupo;
    end monedas;
    arrContador[grupoActual] := arrContador[grupoActual]+total;
    arrLegaron[grupoActual] := arrLegaron[grupoActual]+1;
    if (arrLegaron[grupoActual] == 4 and
    ↪  maxCantidad<arrContador[grupoActual]) then
      maxCantidad := arrContador[grupoActual];
      maxGrupo := grupoActual;
    end if
  end loop;
  for i in 1..20 loop
      personas(i).maximo(maxGrupo);
  end loop;
End body Equipo;

Begin
  for i in 1..20 do
    personas[i].iden(i);
  od
End ej6;
```

(Ejercicio 7)

Consigna:
Hay un sistema de reconocimiento de huellas dactilares de la policía que tiene 8 Servidores para realizar el reconocimiento, cada uno de ellos trabajando con una Base de Datos propia; a su vez hay un Especialista que utiliza indefinidamente. El sistema funciona de la siguiente manera: el Especialista toma una imagen de una huella (TEST) y se la envía a los servidores para que cada uno de ellos le devuelva el código y el valor de similitud de la huella que más se asemeja a TEST en su BD; al final del procesamiento, el especialista debe conocer el código de la huella con mayor valor de similitud entre las devueltas por los 8 servidores.
Cuando ha terminado de procesar una huella comienza nuevamente todo el ciclo. Nota: suponga que existe una función Buscar(test, código, valor) que utiliza cada Servidor donde recibe como parámetro de entrada la huella test, y devuelve como parámetros de salida el código y el valor de similitud de la huella más parecida a test en la BD correspondiente. Maximizar la concurrencia y no generar demora innecesaria.
Respuesta:

Especialista

```
Procedure ej7 is

Task Especialista is
  entry codigo(codNum, similitud: IN int);
End Especialista;

Task body Especialista is
  huella: text;
  maxSim, cod: int = -1;
begin
  loop
    huella = obtenerHuella();
    for i in 1..8 loop
      servidores[i].recibirHuella(huella);
    end loop;
    for i in 1..8 loop
      accept codigo(codNum, similitud: IN int) do
        if similitud > maxSim then
          cod = codNum;
          maxSim = similitud;
        end if;
      end codigo;
    end loop;
  end loop;
End body Especialista;
```

Servidor

```
Task type Servidor is
  entry recibirHuella(nuevaHuella: IN text);
End Servidor;

servidores: array (1..8) of Servidor;

Task body Servidor is
  huella: text;
  codNum, similitud: int;
begin
  accept iden(unID: IN int) do
    id = unID;
  end iden;
  loop
    accept recibirHuella(nuevaHuella: IN text) do
      huella = nuevaHuella;
    end recibirHuella;
    dbs[id].checkHuella(huella, codNum, similitud);
    Especialista.codigo(codNum, similitud);
  end loop;
End body Servidor;
```

DB

```
Task type DB is
  entry iden(id: IN int);
End DB;

dbs: array (1..8) of DB;

Task body DB is
begin
  accept iden(unID: IN int) do
    id = unID;
  end iden;
  loop
    accept servidores[id].checkHuella(huella: IN text;
    ↪  codNum, similitud: OUT int) do
      codNum, similitud = analizarHuella(huella);
    end checkHuella;
  end loop;
End body DB;

Begin
  for i in 1..8 loop
    servidores.iden(i);
    dbs.iden(i);
  loop end;
End ej7;
```

(Ejercicio 8)

Consigna:
Una empresa de limpieza se encarga de recolectar residuos en una ciudad por medio de 3 camiones. Hay P personas que hacen continuos reclamos hasta que uno de los camiones pase por su casa. Cada persona hace un reclamo, espera a lo sumo 15 minutos a que llegue un camión y si no vuelve a hacer el reclamo y a esperar a lo sumo 15 minutos a que llegue un camión y así sucesivamente hasta que el camión llegue y recolecte los residuos; en ese momento deja de hacer reclamos y se va. Cuando un camión está libre la empresa lo envía a la casa de la persona que más reclamos ha hecho sin ser atendido. Nota: maximizar la concurrencia
Respuesta:

Empresa

```
Procedure ej8 is

Task Empresa is
  entry reclamar(idPers:IN Integer);
  entry estoyLibre(idCamion:IN Integer);
End Empresa;

Task body Empresa is
  reclamos: array (1..N) of Integer;
  cantPersonas: int = 0;
  maxReclamos,idReclamos: int;
begin
  loop
    select
      accept reclamar(idPers:IN Integer) do
        if reclamos[idPers] > -1 then
            reclamos[idPers] := reclamos[idPers] + 1;
            if reclamos[idPers]==1 then
                cantPersonas++
            end if;
        end if;
      end reclamar;
    or
      when (cantPersonas > 0) =>
        accept estoyLibre(idPersona: OUT Integer) do
          idPersona:=max(reclamos);
          cantPersonas--;
          reclamos[idPersona]:=-1;
        end estoyLibre;
    end select;
  end loop;
End body Empresa;
```

Funcion maximo

```
function max(arr:array (1..N) of Integer) return Integer;
is
  maxCant,maxId:Integer:=-1;
begin
  for i in 1..N loop
    if arr[i]>maxCant then
      maxCant:=arr[i];
      maxId:=i;
    end if;
  end loop;
  return maxId;
end max;
```

Camion

```
Task type Camion is
  entry limpiar(idPersona: IN Integer)
End Camion;

camiones: array (1..3) of Camion;

Task body Camion is
    idPersona: int;
begin
  loop
    Empresa.estoyLibre(idPersona);
    personas[idPersona].vinoCamion();
    --limpia
    personas[idPersona].terminoCamion();
  end loop;
End body Camion;
```

Persona

```
Task type Persona is
  entry iden(idPers:IN Integer);
  entry vinoCamion();
End Persona;

personas: array (1..N) of Persona;

Task body Persona is
  id:Integer:=-1;
  seguir: bool:=true;
begin
  accept iden(idPers:IN Integer) do
    id:=idPers;
  end iden;
  while seguir loop
    Empresa.reclamar(id);
    select
      accept vinoCamion();
      seguir:=false;
    or delay 15min
      null;
    end select;
    if not seguir then
        accept terminoCamion();
    end if
  end loop;
End body Persona;

Begin
    for i in 1..N  loop
        personas(i).iden(i);
    end loop;
End ej8;
```