

RESUMEN PROGRAMACIÓN CONCURRENTE – 2DO PARCIAL

PASAJE DE MENSAJES ASINCRÓNICOS:

Más de un empleado para atender, sin que tengan que realizar otra tarea.

b. Ídem a) pero considerando que hay 2 empleados para atender, ¿qué debe modificarse en la solución anterior?

```
chan Atencion(int);
```

```
Process Cliente [id: 0..N-1]{  
    Respuesta R;  
    send Atencion (id);  
}
```

```
Process Empleado[id:0..1]{  
    int idC;  
    while (true){  
        receive Atencion(idC);  
        atiendo (idC);  
    }  
}
```

Más de un empleado, pero deben realizar una tarea mientras no hay pedidos.

```
chan Atencion(int);
chan Pedido (int);
chan Siguiente[0..1] (texto);

Process Cliente [id: 0..N-1]{
    Respuesta R;
    send Atencion (id);
}

Process Empleado[id:0..1]{
    int idC;
    while (true){
        send Pedido(id);
        if (!empty Siguiente[id]){
            receive Siguiente[id](idC);
            atiendo (idC);
        }
        else{
            delay (900);
        }
    }
}

Process Coordinador{
    Respuesta R; int idE; int idC;
    while (true){
        receive Pedido (idE)
        if (!empty(Atencion)){
            receive Atencion (idC);
            send Siguiente[idE] (idC);
        }
    }
}
```

Tres procesos + un administrador

```
chan Pedido (int);
chan SigVendedor [0..2] (int);
chan Cocinar (int);
chan Cocina (int);
chan Entrega[0..C-1](Pedido);
chan Venta (int);

Process Cocinero [id: 0..1]{
    int idCliente; Pedido p;
    while (true){
        receive Cocinar (idCliente);
        realizarPedido(p);
        send Entrega[idCliente] (p);
    }
}

Process Vendedor [id: 0..2]{
    int idCliente;
    while (true){
        send Venta (id);
        if (!empty (SigVendedor[id])){
            receive SigVendedor[id](idCliente)
            receive Cocina (idCocinero);
            send Cocinar(idCliente);
        }
        else{
            delay (120);
        }
    }
}

Process Admin{
    int idCliente, idEmpleado;
    while (true){
        receive Venta(idEmpleado);
        if (!empty(Pedido)){
            receive Pedido(idCliente);
            send SigVendedor[idEmpleado](idCliente);
        }
    }
}

Process Cliente [id: 0..C-1]{
    Pedido p;
    send Pedido (id);
    receive Entrega[id] (p);
}
```

Orden de Llegada CON PRIORIDADES:

b) Modifique la solución implementada para que considere la presencia de un director de oficina que también usa las impresas, el cual tiene prioridad sobre los administrativos.

```
chan Pedido (int);
chan ImprimirD (Documento);
chan ImprimirA (Documento);

Process Impresora [id:0..2]{
    Documento doc;
    while (true){
        if (empty(ImprimirD)){
            receive ImprimirA (doc);
            imprimir(doc)
        }
        else{
            receive ImprimirD (doc);
            imprimir (doc);
        }
    }
}

Process Admin [id: 0..N-1]{
    Documento doc;
    while (true){
        // trabajar
        send ImprimirA (doc);
    }
}

Process Director {
    Documento doc;
    while (true){
        // trabajar
        send ImprimirD (doc);
    }
}
```

Pedidos mientras los trabajadores deben hacer otra cosa:

Parcial 2022.

```
chan Pedido (Documento,int);
chan Respuesta [] (Producto);
chan Trabajo (int);
chan Siguiente [] (Documento);

Process Cliente [id: 0..C-1]{
    Doc p; Producto repuesto;
    send Pedido (p,id);
    receive Respuesta[id](repuesto);
}

Process Vendedor [id: 0..V-1]{
    pedido Doc; int idC; Producto repuesto;
    while(true){
        send Trabajo (id);
        if (!empty (Siguiente[id])){
            receive Siguiente[id](Doc,idC);
            realizarPedido(Doc,repuesto);
            send Respuesta[idC] (repuesto);
        }
        else DELAY 180
    }
}

Process Admin {
    int idV,idC; cola pedidos;
    while (true){
        receive Trabajo (idV);
        if (!empty(Pedido)) {
            receive Pedido (p,idC);
            send Siguiente[idV] (p,idC);
        }
    }
}
```

Barrera de Alumnos y Profesor:

```
chan Llegada ();
chan Empezar ();
chan termineExamen (texto,int);
chan Notas [] (int);

Process Estudiante [id: 0..E-1]{
    texto examen, mino;
    send Llegada ();
    receive Empezar();
    hacerExamen (examen)
    send termineExamen (examen,id)
    receive Notas [id] (miNota);
}

Process Profesor {
    int i, idA, nota; texto examen;;
    for (i: 1..E){
        receive Llegada (); // espera a que todos avisen que llegaron
    }

    for (i: 1..E){
        send Empezar();
    }

    for (i: 1..E){
        receive termineExamen(examen, idA);
        corregirExamen (examen,nota);
        send Notas [idA] (nota);
    }
}
```

Prioridades con IF no determinístico:

```

chan EntrarMenosBoletas (int, boletas, dinero);
chan EntrarEmbarazadas (int, boletas, dinero);
chan EntrarNormal (int, boletas, dinero);
chan Respuesta [] (recibo);
chan AvisoGeneral ();

Process Persona [id: 0..P-1]{
    int boletas;
    if (soyEmbarazada()){
        send EntrarEmbarazadas (id, boletas, dinero);
    }
    else if (boletas < 5){
        send EntrarMenosBoletas(id, boletas, dinero);
    }
    else EntrarNormal (id, boletas, dinero);

    send AvisoGeneral ();
    receive Respuesta [id](recibo);
}

Process Cajero[] {
    for i=1..P{
        receive aviso();
        if (not empty(atencionEmbarazada)) ->
            receive EntrarEmbarazadas(idP,b,d);
        [] (not empty(atencionMenos5Boletas) && empty(atencionEmbarazada)) ->
            receive EntrarMenosBoletas(idP,b,d);
        [] (not empty (atencionGeneral) && empty (atencionEmbarazada) && empty (atencionMenos5Boletas)) ->
            receive AvisoGeneral(idP,b,d);
        fi
        realizarPago(b,d,v,r);
        send Respuesta[id](v,r);
    }
}

```