

# Concurrente Práctica 3

HOJA 1

FECHA

1) Si, el código funciona correctamente, pero no respeta el orden de llegada. (No debería pasar a como parámetro)

b) Podríamos simplificar el programa si el Monitor únicamente representa el cruce del puente y no la entrada y salida del mismo. Al hacer esto y gracias a la exclusión mutua que garantizan los monitores, podemos resolver el problema de la siguiente manera:

Monitor Puente {

procedure cruzarPuente () {

// El auto cruza el puente

}

}

\* PREG:

- xq uso un if y no un while?

Process Auto [a:1..M] {

Puente. cruzarPuente();

}

- Es lo mismo usar un bool B o un int s  
s == 0  
if s > 0  
if !B

c) Ninguna de las soluciones respeta el orden de llegada.

• Nuestra solución que sí lo respeta:

Process Auto [a:1..M] {

puentE. entraPuente();

// el auto cruza puente

puentE. salirPuente();

}

Monitor Puente {

bool libre = true; cond cv; int espera = 0;

procedure entraPuente()

{ if (!libre) { espera++; wait(cv); }

else libre = false;

procedure salirPuente()

{ if (espera > 0) { espera--; signal(cv); }

else libre = true; }

}

PASSING THE CONDITIONS

2) process Proceso [i=1..N]

```
{ BD.pedirAccess();
  // leer BD
  BD.libear();
}
```

Monitor BD {

```
{ int cant = 0;
  Cond cv;
```

procedure pedirAccess()

```
{ while (cant == 5) wait(cv);
  cant++;
}
```

procedure libear()

```
{ cant--;
  signal(cv);
}
```

# Concurrente Práctica 3

HOJA 2

FECHA

3) a) process Persona [id: 1 .. N]

```
{ Documento d;
  Fotocopia F;
```

Fotocopiadora. sacarCopia(d, F);

}

Monitor Fotocopiadora

{

```
procedure SacarCopia (d: in text, F: out text)
{ F = Fotocopiar(d);
```

}

}

b) process Persona [id: 1 .. N]

```
{ Documento d;
  Fotocopia F;
```

Fotocopiadora. solicitarCopia();

F = Fotocopiar(d);

Fotocopiadora. liberar();

}

Monitor Fotocopiadora

```
{ bool libre = true;
  cond cv;
  int espera = 0;
```

procedure solicitarCopia()

```
{ if (not libre)
{ espera++;
  wait(cv);
} else { libre=false; }
```

procedure liberar()

```
{ if (espera > 0)
{ espera--;
  signal(cv);
} else { libre=true; }
```

c) process Persona [id:1 .. N]

```
{ int edad = ...;  
Documento d;  
Fotocopia F;
```

Fotocopiadora. solicitarCopia(id, edad);

F = Fotocopiar(d);

fotocopiadora . liberar();

```
}
```

Monitor Fotocopiadora

```
{ bool libre = true;  
Cond cv[N];  
int idAux, espera = 0;  
colaOrdenada Q;
```

procedure solicitarCopia(idP, edad: in int)

```
{ if (not libre)  
{ insertar(Q, idP, edad);  
espera ++;  
Wait (cv[idP]);  
}
```

else libre = false;

```
}
```

procedure liberar()

```
{ if (espera > 0)  
{ espera --;  
sacar(Q, idAux);  
signal (cv[idAux]);  
}
```

else libre = true;

```
}  
NOTA
```

d) process Persona [id: 1..N]  
{ documento d;  
Fotocopia F;

Fotocopiadora. solicitarCopia (id);

F = Fotocopiar (d);

fotocopiadora. liberar();

Monitor Fotocopiadora

{ int proximo = 1;  
cond cv[N];

procedure .solicitarCopia (idP: in int)

{ if (idP <= proximo)  
} { wait (cv[idP]);

}

procedure liberar()

{ if (proximo < N)

{ proximo ++;

signal (cv[proximo]);

}

}

// para que el último  
no haga signal  
de cv[N+1]  
que no existe

e) process Persona [id : 1..N]

{  
Documento d;  
Fotocopia f;

Fotocopiadora. solicitarCopia();

F = Fotocopiar(d);

Fotocopiadora. liberar();

}

Process Empleado

{ for (int i=1; i < N; i++)  
    { Fotocopiadora. usarFotocopiadora();

}

}

Monitor Fotocopiadora

{ cond cv;  
cond empleado;  
cond impresoraLibre;  
int espera = 0;

procedure solicitarFotocopiadora()

{ espera ++;

signal (empleado);

wait (cv);

// aviso a empleado

// espero que empleado me de ok

procedure usarFotocopiadora()

{ if(espera == 0) { wait (empleado); }

// espero solicitud

espera --;

signal (cv);

wait (impresoraLibre); }

// aviso a persona

// espero impresora libre

procedure liberar()

{ signal (impresoraLibre);

// aviso que libero impresora

NOTA

# Concurrente Práctica 3

HOJA 4

FECHA

F) Process Persona [id: 1..N]

```
{ Documento d;           int fotocopiadora;
  Fotocopia F;
```

fotocopiadora. SolicitarCopia(id, fotocopiadora)

F = Fotocopiadora, Fotocopiar(d);

Fotocopiadora.liberar(fotocopiadora);

}

Process Empleado

```
{ for (int i=1; i < N; i++)
  { fotocopiadora.usarFotocopiadora();
```

}

no revisa si  
el arreglo  
↑

Monitor Fotocopiadora

Cond empleado; Cond cv[N]; // necesario que sea  
un arreglo?

Cond impresoraLibre;

cola impresoras[10] = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10);

cola espera;

int fotocopiadoraAsignada[N] = (N) 0;

Procedure SolicitarFotocopiadora(id: in int; fotocopiadora: out int)

```
{ espera.push(id); // me encolo
```

signal(impLibre); // aviso pedido

wait(cv[id]); // espero fotocopiadora

fotocopiadora = fotocopiadoraAsignada[id]; // asigno la fotocopiadora

}

procedre usar fotocopiadora()

{ if (espera. empty())  
    { wait (empleado);

// me duermo si no hay nadie esplando

int id = espera.pop();

if (impresoras. empty())

{ wait (impresora.libre);

// me duermo si no hay  
impresoras libres

fotocopiadora.asignada[id] = impresoras.pop();

signal (CV[id]);

// aviso que asigne  
impresora

procedure liberar\_fotocopiadora (fotocopiadora: in int)

{ impresoras.push (fotocopiadora);

// devuelvo impresora

signal (impresora.libre);

// aviso que hay  
impresora libre

# Concurrente Práctica 3

HOJA 5

FECHA

4) a) process Cliente [id:1..N]  
 { var lista, comprobante;  
 corralon.solicitarAtencion();  
 corralon.enviarLista(lista,  
 comprobante);  
 }

Monitor Corralon  
 { cond llegada; txt list, compra;  
 cond c; cond l, comp;  
 cond retiro;  
 procedure solicitarAtencion()  
 { signal (llegada);  
 wait (c);  
 }

process Empleado  
 {  
 for [i=1 .. N] ;  
 {  
 corralon.atender();  
 }  
 }

procedure enviarLista (lista: in txt,  
 comprobante: out txt)  
 { list = lista;  
 signal (l);  
 wait (comp);  
 comprobante = compra;  
 signal (retiro); }

procedure atender()  
 { wait (llegada); // espera cliente  
 signal (^c); // le da atención  
 wait (l); // espera lista  
 compra = generalCompra (list);  
 signal (comp);  
 wait (retiro); }

Necesito una var esperando que lleve control de clientes  
 operando?

y en procedure atender  
 if (esperando == 0)

{ wait (llegada); }

b) process Cliente [id:1..N]

```
{ txt lista, comprobante;  
int idE;
```

```
corralon.solicitarAtencion(idE);  
escritorio[idE].EnviarLista(lista,  
comprobante);
```

}

process Empleado [id:1..E]

```
{ txt lista, resul;
```

while (true)

```
{ corralon.motivo(id);  
escritorio[id].esperarLista(lista);  
resul = generarComprobante(lista);  
escritorio[id].enviarComprobante(resul);
```

}

}

Monitor corralon

/cola elibres; cond espera;

int cantLibres=0, esperando=0;

procedure solicitarAtencion(idE, out int)

```
{ if (cantLibres == 0)
```

```
{ esperando ++;
```

```
wait (espera);
```

}

```
else { cantLibres --;
```

```
pop (elibres, idE); // me trago id
```

}

procedure proximo(idE: in int)

```
{ push (elibres, idE);
```

```
if (esperando > 0)
```

```
{ esperando --;
```

```
signal (espera);
```

}

```
else cantLibres ++;
```

}

Monitor Escritorio[id:1..E]

```
{ cond c; cond e; boolean listo=false; txt datos, res ;
```

procedure enviarLista (lista: in txt,

comprobante: out txt){

datos=lista; listo=true;

signal (e);

wait (c);

comprobante= resul;

signal (e);

}

procedure esperarLista (lista: out txt)

```
{ if (not listo) wait (e);
```

datos=datos; }

procedure enviarComprobante (resul: in txt)

```
{ res= resul;
```

signal (c);

wait (e);

listo= False;

NOTA

# Concurrente Práctica 3

HOJA 6  
FECHA

c) process Cliente [id: 1..N]

```
{ txt lista, comprobante;
int idE;
Corralon. solicitarAtencion(idE);
escritorio[idE]. enviarlista(lista,
comprobante);
}
```

process Empleado [id: 1..E]

```
{ txt lista, resul;
bool seguir = true;
while (seguir)
{ corralon.proximo(id, seguir);
if (seguir)
    escritorio[id]. esperarlista(lista);
    resul = generarCompra(lista);
    escritorio[id]. enviarComprobante(resul);
}
}
```

Monitor Corralon

```
{ Cola libres, espera;
int cantlibres=0, esperando=0;
int cantAtendidos=0;
```

procedure solicitarAtencion (idE: out int)

```
{ if (cantlibres==0)
    { esperando++; wait(espera); }
else { cantlibres--;
    pop(libres,idE);
}
}
```

}

Monitor Escritorio [id: 1..E]

```
{ cond c, e; bool listo=false; txt datos, res;
```

procedure enviarlista (lista: in txt, comprobante: out txt)

```
{ datos = lista; listo=true;
signal(e);
wait(c);
comprobante = resul;
signal(e); }
```

procedure esperarlista (lista: out txt)

```
{ if (not listo) wait(e);
lista = datos; }
```

procedure enviarComprobante (resul: in txt)

```
{ res = resul;
signal(c);
wait(e);
listo=false; }
```

procedure proximo (idE: in int, seguir: out bool)

```
{ if (cantAtendidos == N) { seguir=false; }
```

```
{ else { cantAtendidos++;
```

seguir=true;

push(libres,idE);

if (esperando > 0)

{ esperando--;

signal(espera);

}

else cantlibres++;

5) process Alumno [id:1..50]

```
{ int grupo, nota;  
    tarea. llegada(id);  
    tarea. recibirGrupo(id, grupo);  
    // hacer tarea  
    tarea. obtenerNota(grupo, nota);  
}
```

process JTP

```
{ tarea. esperarAlumnos();  
    tarea. asignarGrupos();  
    tarea. asignarNotas();  
}
```

## Monitor tarea{

cola alumnos; int numGrupos[50] = ([50]-1); cola terminados;  
 Cond avisoProfes; Cond avisoNotas[25]; int notas[25] = ([25]-1);  
 Cond avisoGrupo[50]; int terminadosXGrupo[25] = ([25]-0);

procedure llegada(id: in int)  
 { alumnos.push(id);  
 if (alumnos.size() == 50)  
 } { signal(avisoProfes);  
 }}

procedure recibirGrupo(id: in int,  
 grupo: out int)  
 { wait(avisoGrupo[id]);  
 grupo = numGrupos[id]; }

procedure obtenerNota(grupo: in int,  
 nota: out int)  
 { terminados.push(grupo);  
 signal(avisoProfes);  
 wait(avisoNotas[grupo]);  
 nota = notas[grupo];  
 }

procedure esperarAlumnos()  
 { if (alumnos.size() < 50)  
 } { wait(avisoProfes);  
 }}

procedure asignarGrupos()  
 { for (int i=1 to 50)  
 { int id = alumnos.pop(id);  
 numGrupos[id] = asignarNroGrupos();  
 signal(avisoGrupo[id]);  
 }}

procedure asignarNotas()  
 int puestaje = 25;  
 { for (int i=1 to 50)  
 { if (terminados.empty())  
 { wait(avisoProfes);  
 int g = terminados.pop(grupo);  
 terminadosXGrupo[g]++;  
 if (terminadosXGrupo[g] == 2)  
 { notas[g] = puestaje;  
 puestaje--;  
 signal-all(avisoNotas[g]);  
 }}

6) process Jugador [id: 1..20]

{ int equipo, cancha;

equipo = DarEquipo();

Equipo[equipo]. pedirCancha(cancha);

Cancha[cancha]. Jugar();

process Partido [id: 1..2]

{ Cancha[id]. iniciar();

delay(50 min);

Cancha[id]. terminar();

Monitor Equipo [id: 1..4]

{ int llegados = 0; cond espera; int numCancha;

procedure pedirCancha(c: IN int)

{ llegados ++;

if (llegados < 5)

{ wait(espera);

} else

{ administrador. obtenerCancha(numCancha);

signal\_all(espera);

c = numCancha;

}

# Concurrente

## Práctica 3

HOJA 8  
FECHA

Monitor administrador

```
{ int grupos = 0;
```

proceder obtenerCancha (numCancha: ev1 int)

```
{ grupos++;
```

```
if (grupos <= 2)
```

```
{ numCancha = 1;
```

```
else numCancha = 2;
```

```
}
```

```
}
```

Monitor Cancha [id:1..2]

```
{ int cant = 0; cond espera; cond inicio;
```

procedure jugar()

```
{ cant++;
```

```
if (cant == 10)
```

```
{ signal (inicio);
```

```
wait (espera); }
```

procedure iniciar()

```
{ if (cant < 10)
```

```
{ wait (inicio); }
```

procedure terminar()

```
{ signal-all (espera); }
```

7) process Corredor [id:1..C]

```
carrera.llegada();  
// correr  
carrera.accesoMaguina();  
maguina.retirarBotella();  
carrera.liberaMaguina();
```

```
process Repositor  
{ while(true)  
{ maguina.reponerBotellas();  
}}
```

Monitor Maquina

```
{ int botellas=20; (con) repositor; cond esperaBotellas;
```

```
procedure reponerBotellas  
{ if (botellas > 0)  
{ wait (repositor);  
}  
botellas = 20;  
signal (esperaBotellas);  
}
```

```
procedure retirarBotella  
{ if (botellas == 0)  
{ Signal (repositor);  
wait (esperaBotellas);  
}  
botellas --;
```

# Concurrente Práctica 3

HOJA 9  
FECHA

Monitor Carrera

```
{ int cant=0; cond espera; bool libre=true;  
    cond cola; int esperando=0;
```

procedure Negado()

```
{ cant++;  
if (cant < c)  
{ wait(espera);  
}
```

```
else signal(cola);  
}
```

procedure acosoMajina()

```
{ if (not libre)  
{ esperando++;  
wait(cola);  
}
```

```
else libre=false;  
}
```

procedure liberarMajina()

```
{ if (esperando > 0)  
{ esperando--;  
signal(cola);  
}
```

```
} else  
libre=true;  
}
```

estos 2  
procedimientos  
deberían  
estar en  
el monitor  
Majina?

8) process Alumno(id:0..45)

```
{ txt enunciado; resolucion;  
double nota;  
examen. llegada(id, enunciado);  
resolucion = HacerExamen(enunciado);  
examen. esperarCorreccion();  
escritorio. correccion(nota, resolucion);  
examen. liberarPrograma();  
}
```

process Preceptor

```
{ txt enunciado;  
slamen. esperarAlumnos();  
examen. entregarEnunciado(enunciado);  
}
```

process Profesora

```
{ txt resolucion;  
double nota;  
for (i:0 .. 44)  
{ escritorio. esperarEnunciado(resolucion);  
nota = corregirExamen();  
escritorio. entregarCorreccion(nota);  
}
```

33

podría hacer todo con un mismo monitor Examen ??

# Concurrente

## Práctica 3

HOJA

10

FECHA

### Monitor Examen

```
{ cond espesa; cond profe; esperarEnunciado[45]; bool profelibre=true;
  txt entregarEnunciado[45] = ("[45] ''"); int cant=0; int esperando=0;
```

procedure llegada (id: IN int, enunciado: OUT txt)

```
{ cant++;
```

```
if (cant == 45)
```

```
{ signal (profe);
```

// aviso profe que están todos

```
} wait (esperarEnunciado[id]);
```

// espero enunciado

```
enunciado = entregarEnunciado[id];
```

// agarré enunciado

```
}
```

### procedure esperarAlumnos

```
{ if (cant < 45)
```

```
{ wait (profe);
```

// espero a todos los alumnos

```
}
```

### procedure entregarEnunciado (enunciado: IN txt)

```
{ for (i:0..44)
```

```
{ entregarEnunciado[i] = enunciado;
```

// entrego enunciado

```
signal (esperarEnunciado[i]);
```

// aviso que ya entregué

```
}
```

### procedure esperarCorrección()

```
{ if (!profelibre)
```

// si profe está ocupada

```
{ esperando++;
```

// sumo 1 a esperando

```
wait (espresa);
```

// espreso profesora

```
}
```

```
else profelibre = false;
```

// owo profesra

NOTA

```

procedure liberarProfesora()
{
    if (esperando > 0)           // si hay alumnos esperando
        {
            esperando --;       // resto 1 a esperando
            signal(espero);     // desperto alumno qd espero
        }
    else profesolibre = true;   // aviso que prof esté libre
}

```

### Monitor Corrección

```

txt res; bool hayRes; double mark;
Cond profesor; Cond alumno;

```

procedure Corrección(nota: OUT double, resolución: IN txt)

```

res = resolución;           // copio resolución en var. del monitor
hayRes = true;              // var. booleana en true
signal(profesor);          // aviso al profesor
wait(alumno);               // espero nota
nota = mark;                // me guarda la nota
}

```

procedure esperoEnunciado(resolución: OUT txt)

```

if (!hayRes)
    {
        wait(alumno);      // espero alumno me avise.
        resolución = res;  // me guarda la resolución
    }
}

```

procedure entregarCorrección(nota: IN double)

```

mark = nota;                 // copio nota en var. del monitor
signal(alumno);             // aviso alumno
hayRes = false;              // var. booleana en false
}

```