

## Concurrente

## Práctica 2

HOJA 1

FECHA

- 1) sem mutex = 5; // Semáforo de recursos  
 colatEspecial C[5]; // tiene 5 recursos almacenados.  
 sem libre = 1; // semáforo para acceso a la cola

Process Proceso [id: 0..P-1]

{ while (true) // en el caso de que los procesos  
 quieran usar recursos más de una vez c/u

{ P(mutex);  
 P(libre);  
 recurso = pop(c, recurso);  
 V(libre);  
 // uso recurso  
 P(libre);  
 pop(c, recurso);  
 V(libre);  
 V(mutex);  
}

- 2) b) sem libre = 1;

Process Persona [id: 0..N-1]

{ P(libre);  
 // pasa por detector  
 V(libre);  
}

- c) Sem libre = 3;

Process Persona [id: 0..N-1]

{ P(libre);  
 // pasa por detector  
 V(libre);  
}

NOTA

3)<sup>a)</sup>

Fallos C[N];

Process Control [id:0..3]

```
{ inicio = id * N div 4;  
fin = ((id + 1) * N div 4) - 1;
```

for ( i = inicio to fin)

```
{ if ( c[i].nivel == 3 )
```

```
{ print("Error critico ID:", c[i].id);
```

}

}

}

b)

Fallos C[N];

```
int contador[4] = 0; // cuenta cant. de fallos de c/ nivel  
sem s[4] = 1;
```

process Control [id:0..3]

```
{ inicio = id * N div 4;
```

```
fin = ((id + 1) * N div 4) - 1;
```

for ( i = inicio to fin)

```
{ int g = c[i].nivel;
```

```
P(s[g]);
```

```
Contador[g]++;
```

```
V(s[g]);
```

}

}

3) c) fallas  $C[N]$ ,  
int contador[4] = (4)0;

Process Control [id: 0.. 3 ]

```
{ int local=0;  
  for [i: 0.. N-1]  
  {  
    if (C[i].nivel == id)  
    { local++;  
    }  
  }  
  contador[id] := local;  
}
```

No es muy eficiente porque todos los procesos  
deben recorrer todo el arreglo de historial  
de fallas.

4) a) sem vacío = N;  
sem lleno = 0;  
paquete c[N];

process Preparador  
{ while (true)  
{ // prepara paquete  
P(vacio);  
push (c, paquete);  
V(lleno);  
}  
}

Process Entregador  
{ while (true)  
{ P(lleno);  
paquete = pop (c, paquete);  
V(vacio);  
// entrega paquete  
}  
}

## Concurrente

## Práctica 2

4) b) Sem vacio = N; paquete C[N];  
 Sem lleno = O;  
 Sem mutexP = 1;  
 Sem mutexE = 1;

process Preparador [id: 0.. P-1]  
 { while (true)  
 { // prepara paquete  
 P(vacio);  
 P(mutexP);  
 push (c, paquete);  
 V(mutexP);  
 V(lleno);  
 }  
 }

Process Entregador [id: 0.. E-1]  
 { while (true)  
 { P(lleno);  
 P(mutexE);  
 paquete = pop(c, paquete);  
 V(mutexE);  
 V(vacio);  
 // entrega paquete  
 }  
 }

5) Sem mutex = 1;      cola finalizadas;      Sem avisarProfesor [10] = ([10] 0);  
Sem empezar = 0;      Sem avisarProfesor = 0;  
int cant = 0;      int puntuaje [10] = ([10] 0);

Process Alumnos [id:0..49]

{

tarea = elegir();

P(mutex);

cant = cant + 1;

if (cant == 50)

{ por (i: 0..49)

{ V(empezar);

}

V(mutex);

P(empezar);

// realizar tarea

P(mutex);

finalizadas.push(tarea); // guardo tarea al terminar

V(mutex);

V(avistarProfesor); // aviso que terminó

P(obtenerPuntaje[tarea]); // obtengo puntaje

nota = puntuaje[tarea]; // obtengo puntaje.

}

## Concurrente

## Práctica 2

HOJA: 4  
FECHA:

Process Profesor {

int puntojeADar = 1;

int tareaActual;

int alumnosPorTarea [20] = {[20]0};

for (i=0 to 49)

{ P(avisoProfesor); // aviso que alumno termine

P(mutex);

tareaActual = finalizados.pop(); // saco tarea

V(mutex);

alumnosPorTarea[tareaActual]++; // sumo alumno que realizó tarea

if (alumnosPorTarea[tareaActual] == 5) // si todos los alumnos terminaron la tarea

{ puntoje[tareaActual] = puntojeADar; // le doy puntoje al grupo

puntojeADar++;

for (j=0 .. 4)

{ V(obtenerPuntoje[tareaActual]); // aviso alumno

}

}

}

Ejercicio importante para recordar.

6) No es la más adecuada ya que podría generarse demora innecesaria al utilizar el semáforo general antes que el de prioridad

Solución correcta:

Sem: semaphore = 6; alta:Semaphore = 4; baja:Semaphore = 5;  
Process Usario-Alta [i: 1..L]:

{ P(alta);

P(sem);

//usa la BD

V(sem);

V(alta);

}

process Usario-Baja [i: 1..K]:

{ P(baja);

P(sem);

//usa la BD

V(sem);

V(baja);

}

## Concurrente

## Práctica 2

HOJA 5

FECHA

7) a) Sem  $S=1;$ 

process Persona [id: 0..N-1]

{ P(s),

Imprimir (documento);

V(s),

}

b) Sem  $S=1;$  Sem espera [N] = ([N] 0),  
boolean libre = lib = 1; colaEspera c;

process Persona [id: 0..N-1]

{ int aux;

P(sem);

if (libre)

{ libre = False;

V(sem);

} else

{ c.push(id);

V(sem);

P(espera [id]);

}

Imprimir (documento);

P(sem);

if (empty(c))

{ libre = True;

} else

{ aux = c.pop();

V(espera [aux]);

} V(mutex); }

NOTA

5

c) int siguiente = 0; Sem s = 1; Sem espera[N] = (EN 0);

Process Persona [id: 0..N-1]

{ P(sem);  
if (siguiente < id)  
{ V(sem);  
P(espera[id]);

} else {V(sem);}

// Imprimir Documento;

P(sem);

Siguiente ++;

V(espera[siguiente]);

V(sem);

}

La otra solución sería igual que el inciso b  
pero en vez de push utilizar un  
Añadir Ordenado (c, id);

d) sem sem=1;      sem turno[N] = ([N] 0);      sem pedidos=0;  
Cola Especial C;      boolean libre=true;      sem libre=0;

```
process Persona [id:0..N-1]
{
    P(sem);
    c.push(id);                                            // me encolo
    V(sem);
    V(pedidos);                                            // aviso que hay pedido
    P(turno[id]);                                            // espero a poder imprimir
    Imprimir(documento);
    V(libre);                                                // aviso que termine'
}
```

```
process Coordinador
{
    for (i=0..N-1)
    {
        P(pedidos);                                            // espero que haya pedido
        P(sem);
        aux = c.pop(id);                                        // saco de la cola
        V(sem);
        V(turno[aux]);                                            // aviso que puede imprimir
        P(libre);                                                // espero que termine de imprimir
    }
}
```

e) ColaEspera cola[N];

ColaImpresoras impresoras[s] = (0, 1, 2, 3, 4);

int impresoraPersona[N] = ([N] - 1);

Sem acceso(ColaPedidos = 1);

Sem acceso(ColaImpresoras = 1);

Sem pedidos = 0;

Sem cantImpresoras = 5;

Sem espera [N] = ([N] 0);

Process Persona[id: 0..N-1]

{ P(acceso(ColaPedidos));

cola.push(id); //encolo pedido

V(acceso(ColaPedidos));

V(pedidos); //aviso que hay pedido de impresión

P(espera[id]); //espero poder usar la impresora

Imprimir (documento, impresoraPersona[id]); //imprime en impresora asignada

P(acceso(ColaImpresoras));

impresoras.push(impressoraPersona[id]); //repongo impresora

V(acceso(ColaImpresoras));

V(cantImpresoras); } //aviso hay una impresora libre

Process Coordinador

{ For (i=0 to N-1)

{ P(pedidos); //espero pedido impresión

P(acceso(ColaPedidos));

sig = cola.pop();

V(acceso(ColaPedidos));

P(cantImpresoras); //espero que haya impresoras libres

P(acceso(ColaImpresoras));

impresora = impresoras.pop();

V(acceso(ColaImpresoras));

impresoraPersona[sig] = impresora;

//asigno impresora al siguiente

NOTA V(espera[sig]); }

//habilito uso impresora

## Concurrente

## Práctica 2

HOJA 7

FECHA

```

8) int resueltos[E] = ([E] 0);
    Sem SemFinalizo=1;
    Sem sem=1;
    Sem empesar=0;
    int cantE=0;
    int piezas=0;
    int Finalizo=0;
  
```

```

process Empleado [id.0 .. E-1]
{
  P(sem);
  cantE++;
  // cuenta cant. empleados
  if (cantE == E)
  {
    for (i=0..E-1)
    {
      V(empezar); }
    // les aviso que pueden empezar
  }
  V(sem);
  P(empezar);
  P(sem);
  while (piezas < T)
  { piezas++;
    // mientras haga piezas
    V(sem);
    // actualiza piezas
    // fabrica pieza
    resueltos[id]++;
    // sumo cant piezas que fabrique
    P(sem);
  }
  V(sem);
  P(sem Finalizo);
  Finalizo++;
  if (Finalizo == E)
  {
    V(despierto Empresa); }
  // desperto empresa
  V(semFinalizo);
  P(detPremio);
  // espero anuncio de premio
}
  
```

NOTA: if (premio == id) { print ("Fabrique más piezas que el resto"); }

## Process Empresa

```
{ P(despiertoEmpresa);           // espero que terminen todos
    premio = resultados[indexDF(resultados.max())]; // guarda empleado ganador
    for (j=0..E-1)
    { V(detPremio);               // desperto empleado.
    }
```

9) Sem capacidadM = 30; Sem capacidadV = 50; colallenturas colaventana;  
Sem colaM = 1; Sem colav = 1; Sem colaventana = 1;  
cola Marcos marcos; colavidrios vidrios;  
Sem hayM = 0; Sem hayV = 0;

## Process Carpintero [id:0...3]

```
{ while (true)
{ marco = hacerMarco();          // hago marco
P(capacidadM);                  // controlo que haya lugar
P(colaM);
marcos.push(marco);             // encola marco
V(colaM);                       // sumo 1 marcos disponibles
} V(hayM);
```

## Process Vidriero

```
{ while (true)
{ vidrio = hacerVidrio;          // hago vidrio
P(capacidadV);                  // controlo que haya lugar
P(colaV);
vidrios.push(vidrio);            // encola vidrio
V(colaV);                       // sumo 1 marcos disponibles
} V(hayV);
```

NOTA

# Concurrente

## Práctica 2

HOJA 8

FECHA

Process Armador [id: 0..1]

{ while (true)

{ P(hayM);

// controlo que haya m Marcos

P(colaM);

marcos = marcos.pop();

// saco marco

V(colaM);

V(capacidadM);

// capacidad marcos sumo 1

P(hayV);

// controlo que haya vidrio

P(colaV);

vidrio = vidrios.pop();

// saco vidrio

V(colaV);

V(capacidadV);

// capacidad marcos sumo 1

ventana = armarVentana(marco, vidrio);

P(colaVentana);

colaVentana.push(ventana);

// guardo ventana en cola

V(colaVentana);

}

}

10) Sem SemTrigo = 5;  
Sem SemMaiz = 5;  
Sem SemTotal = 7,

process CamionTrigo [id: 0... T-1]

```
{ P(SemTrigo);  
P(SemTotal);  
// descargar  
V(SemTotal);  
V(SemTrigo);  
}
```

process CamionMaiz [id: 0... T-1]

```
{ P(SemMaiz);  
P(SemTotal);  
// descargar  
V(SemTotal);  
V(SemMaiz);  
}
```

11) cola espera;  $\text{Sem } \text{esperaVacuna}[P] = ([P] 0);$

$\text{Sem } \text{SemEspera} = 1;$

int cant = 0;

$\text{Sem } \text{despiertoEmpleado} = 0;$

process Persona [id: 0...P-1]

{ P(SemEspera);

espera.push(id);

cant++;

if (cant == S)

V(despiertoEmpleado);

cant = 0;

}

//agrego a cola de personas

//incremento cantidad

//despierto empleado

V(SemEspera);

P(esperaVacuna[id]);

//espero para irme

process Empleado

{ cola vacunados;

for [i = 1.. P/S];

{ P(despiertoEmpleado);

//espero que haya S

for [j = 1.. S]

{ P(SemEspera);

aux = espera.pop();

//desencolar persona

V(SemEspera);

VacunarPersona(aux);

//vacuno

Vacunados.push(aux);

//agrego persona a vacunados

}

for [j = 1.. S]

{ aux = vacunados.pop();

//desencolar vacunado

V(esperaVacuna[aux]);

//le aviso que puede irse

Sem esperapuestos[3] = ([3] 0);  
 Sem espero[150] = ([150] 0);  
 Sem mutex = 1;  
 Sem accesoPuestos[3] = ([3] 1);  
 Sem general = 0;  
 Cola Puestos[3];

process Pasajero [id: 0...149]

```

  { P(mutex);
    llegados.push(id);           // me encolo
    V(mutex);
    V(general);                // aviso a coordinador que llegue
    P(espero[id]);             // espero a poder irme
  }
  }
```

process Coordinador

```

  { For [i: 0...149]
    { P(general);           // espero pasajeros
      P(mutex);
      actual = llegados.pop(); // desencolo actual
      V(mutex);
      puesto = PuestoConMenosPasajeros(); // obtengo puesto con
      P(accesoPuestos[puesto]);            // menos pasajeros
      puestos[puesto].push(actual);        // encolo actual en
      V(sacarPuestos[puesto]);            // cola de puesto elegido
      V(esperaPuestos[puesto]);          // aviso a enfermera
    }                                // que hay pasajero esperando
  }
  }
```

For [i: 0...2)

```

  { V(esperaPuestos[i]); // aviso a enfermeras que no
    hay mas pasajeros esperando
  }
  }
```

process Enfermera [id: 0..2]

```
{ P(esperaPuestos[id]); // espero que haya pasajero
  P(accesoPuestos[id]); en mi puesto
  while (puestos[id].notEmpty)
  {
    actual = puestos[id].pop(); // desencolar pasajero
    V(accesoPuestos[id]); // liberar puesto
    atispar(actual); // avisar pasajero
    V(espera[actual]); // avisar pasajero que puede "se
    P(esperaPuestos[id]); // espero que haya pasajeros
    P(accesoPuestos[id]);
  }
  V(accesoPuestos[id]);
```

b) Sem mutex = 1;  
sem accesoPuestos [3] = ([3] 1);  
Cola Puestos [3];  
Sem esperaPuestos [3] = ([3] 0);

int hisopados = 0;  
Sem mutexHisop = 1;

process Pasajero [id: 0.. 149]  
{  
P(mutex);  
puesto = PuestoConMenosPasajeros(); // obtengo puesto con  
V(mutex); // menos pasajeros  
P(accesoPuestos [puesto]);  
puestos [puesto].push(id); // me agrego a la cola de  
pasajeros del puesto asignado  
V(accesoPuestos [puesto]);  
V(esperaPuestos [puesto]); // aviso que hay pasajero esperando  
P(espero [id]); // espero a ser hisopado  
// estoy siendo hisopado  
P(espero [id]); // espero a poder irme  
}

# Concurrente Práctica 2

HOJA 11

FECHA

```

process Enfermera [ i:0..2 ]
{
    while ( hisopados < 150 )
        {
            P ( esperaPuestos [ i : id ] );
            // espero que haya pasajero
            // en mi puesto
            if ( puestos [ i : id ].notEmpty )
                {
                    P ( accesoPuestos [ id ] );
                    actual = puestos [ id ].pop();
                    // desencolo pasajero
                    // a hisopar
                    V ( accesoPuestos [ id ] );
                    V ( espero [ actual ] );
                    // aviso a pasajero que
                    // lo voy a hisopar
                    hisopar ( actual );
                    V ( espero [ actual ] );
                    // aviso que se pide ir
                    P ( mutexHisop );
                    hisopados++;
                    if ( hisopados == 150 )
                        {
                            for ( j : 0 .. 2 )
                                {
                                    V ( esperoPuestos [ j ] );
                                    // aviso que
                                    // no hay mas
                                    // a hisopar
                                }
                            V ( mutexHisop );
                        }
                }
        }
}

```