Resolución de practica 2 - Semáforos 🚦

> Un contador podes consultarlo , al semáforo no lo podes consultar

Ejercicio 1

a. El detector no es un proceso sino el recurso que es usado. Las personas deben ser procesos.

```
B.

sem libre =1;

process esperar{

    P(libre)
    // usa el detetor
    V(libre)

}
```

```
C.

sem libre = 3;

process esperar{

    P(libre)
    // usa el detetor
    V(libre)

}
```

```
D.

sem libre = 3;

process esperar{
    while (true) {
```

```
        P(libre)
        // usa el detetor
        V(libre)

    }

}
```

Ejercicio 2

Un sistema de control cuenta con 4 procesos que realizan chequeos en forma colaborativa. Para ello, reciben el historial de fallos del día anterior (por simplicidad, de tamaño N). De cada fallo, se conoce su número de identificación (ID) y su nivel de gravedad (0=bajo, 1=intermedio, 2=alto, 3=crítico). Resuelva considerando las siguientes situaciones:

a) Se debe imprimir en pantalla los ID de todos los errores críticos (no importa el orden).
b) Se debe calcular la cantidad de fallos por nivel de gravedad, debiendo quedar los resultados en un vector global.
c) Ídem b) pero cada proceso debe ocuparse de contar los fallos de un nivel de gravedad determinado.

Ejercicio 3

Un sistema operativo mantiene 5 instancias de un recurso almacenadas en una cola. Además, existen P procesos que necesitan usar una instancia del recurso. Para eso, deben sacar la instancia de la cola antes de usarla. Una vez usada, la instancia debe ser encolada nuevamente para su reúso.

```
cola c;
sem libre = 5
sem mutex = 1


process usoDeRecurso[id : 0..P-1]{
instancia x;

while (true) {
    P(libre)
    // hay recursos, ahora espero que la cola este libre
    p(mutex)
     x = c.pop()
    v(mutex)
    //Lo usa
```

```
    p(mutex)
     c.push(x)
    v(mutex)
    // Sale y libera el recurso
    V(libre)

}
}
```

Ejercicio 4

Suponga que existe una BD que puede ser accedida por 6 usuarios como máximo al mismo tiempo. Además, los usuarios se clasifican como usuarios de prioridad alta y usuarios de prioridad baja. Por último, la BD tiene la siguiente restricción:

- no puede haber más de 4 usuarios con prioridad alta al mismo tiempo usando la BD.
- no puede haber más de 5 usuarios con prioridad baja al mismo tiempo usando la BD. Indique si la solución presentada es la más adecuada. Justifique la respuesta.

<table>
<tr><td colspan="2">Var
total: sem := 6;
alta: sem := 4;
baja: sem := 5;</td></tr>
<tr><td>Process Usuario-Alta [I:1..L]::
{ P (total);
P (alta);
//usa la BD
V(total);
V(alta);
}</td><td>Process Usuario-Baja [I:1..K]::
{ P (total);
P (baja);
//usa la BD
V(total);
V(baja);
}</td></tr>
</table>

```
Se debe cambiar el P(alta) - P(baja) estas deben ir primero, ya que sino puede
quitar un recurso.
```

Ejercicio 5

En una empresa de logística de paquetes existe una sala de contenedores donde se preparan las entregas. Cada contenedor puede almacenar un paquete y la sala cuenta con capacidad para N contenedores. Resuelva considerando las siguientes situaciones:

a) La empresa cuenta con 2 empleados: un empleado Preparador que se ocupa de preparar los paquetes y dejarlos en los contenedores; un empelado Entregador que se ocupa de tomar los paquetes de los contenedores y realizar la entregas. Tanto el Preparador como el Entregador trabajan de a un paquete por vez.
b) Modifique la solución a) para el caso en que haya P empleados Preparadores.
c) Modifique la solución a) para el caso en que haya E empleados Entregadores.
d) Modifique la solución a) para el caso en que haya P empleados Preparadores y E empleadores Entregadores.

```
a.

sem vacias = N
sem hayPaquete = 0
vector vec[N]


process preparador{
int indiceP = 0

    while(true){
        //prepara paquete
        P(vacias)
        indiceP = ( (indiceP + 1 )  MOD N )
        vec[indiceP] = paquete
        V(hayPaquete)
    }
}


// Este esta raro

process entregador{
paquete x;
int indiceE = 0;

    while(true){
        P(hayPaquete)
        x = vec[indiceE]
        V(vacias) --> Ya que lo podes liberar
        indiceE = ( (indiceE + 1 )  MOD N )
        //realiza entrega
    }
```

```
}
```

b.

```
sem vacias = N
sem hayPaquete = 0
vector vec[N]

int indiceP = 0
sem mutex = 1

process preparador[id: 0..P-1]{

     while(true){
        //prepara paquete
        P(vacias)

        P(mutex)
        indiceP = ( (indiceP + 1 )  MOD N )
        vec[indiceP] = paquete
        V(mutex)

        V(hayPaquete)
    }
}


// Este esta raro

process entregador{
paquete x;
int indiceE = 0;

    while(true){
        P(hayPaquete)
        x = vec[indiceE]
        V(vacias) --> Ya que lo podes liberar
        indiceE = ( (indiceE + 1 )  MOD N )
        //realiza entrega
    }

}
```

c.

```
sem vacias = N
sem hayPaquete = 0
vector vec[N]

int indiceE = 0;
sem mutex = 1

process preparador{
int indiceP = 0

    while(true){
        //prepara paquete
        P(vacias)
        indiceP = ( (indiceP + 1 )  MOD N )
        vec[indiceP] = paquete
        V(hayPaquete)
    }
}


// Este esta raro

process entregador[id: 0..P-1]{
paquete x;

    while(true){
        P(hayPaquete)

        P(mutex)
        x = vec[indiceE]
        indiceE = ( (indiceE + 1 )  MOD N )
        V(mutex)

        V(vacias)

        //realiza entrega
    }

}
```

d.

```
sem vacias = N
sem hayPaquete = 0
vector vec[N]

int indiceE = 0;
sem mutex = 1

int indiceP = 0
sem mutex2 = 1

process preparador[id: 0..P-1]{

     while(true){
        //prepara paquete
        P(vacias)

        P(mutex)
        indiceP = ( (indiceP + 1 )  MOD N )
        vec[indiceP] = paquete
        V(mutex)

        V(hayPaquete)
    }
}

process entregador[id: 0..E-1]{
paquete x;

    while(true){
        P(hayPaquete)

        P(mutex2)
        x = vec[indiceE]
        indiceE = ( (indiceE + 1 )  MOD N )
        V(mutex2)

        V(vacias)

        //realiza entrega
    }

}
```

Ejercicio 6

Existen N personas que deben imprimir un trabajo cada una. Resolver cada ítem usando semáforos:

a) Implemente una solución suponiendo que existe una única impresora compartida por todas las personas, y las mismas la deben usar de a una persona a la vez, sin importar el orden. Existe una función Imprimir(documento) llamada por la persona que simula el uso de la impresora. Sólo se deben usar los procesos que representan a las Personas.
b) Modifique la solución de (a) para el caso en que se deba respetar el orden de llegada.
c) Modifique la solución de (a) para el caso en que se deba respetar estrictamente el orden dado por el identificador del proceso (la persona X no puede usar la impresora hasta que no haya terminado de usarla la persona X-1).
d) Modifique la solución de (b) para el caso en que además hay un proceso Coordinador que le indica a cada persona que es su turno de usar la impresora.
e) Modificar la solución (d) para el caso en que sean 5 impresoras. El coordinador le indica a la persona cuando puede usar una impresora, y cual debe usar.

```
a.

sem libre = 1

process persona[id: 0.. N-1]{

    P(libre)
    imprimir(documento)
    V(libre)

}
```

```
b.

sem mutex = 1
sem libre = 1
cola c;

process persona[id: 0..N-1]{

    //Llega
    P(mutex)
     c.push(id)  //Y se encola en la fila
    V(mutex)

    P(libre)
```

```
    imprimir(documento)
    V(libre)

}

> La duda es que si no hay nadie en la cola, deberia encolarse igual ?
> Se desencola ?




-------------------------no nenita es con semaforos-----------------------------
-----

sem mutex = 1                   sem espera([N] 0)
boolean libre = true
cola c;
int aux;

process persona[id:0.. P-1]{

    P(mutex) // llega a la impresora
    if (libre){ //  si esta libre

        libre = false   // lo pongo en falso xq llegue yo
        V(mutex)        //
    } else {

        c.push(id)    // me encolo
        V(mutex)      // Antes de dormirme libero el recurso
        P(espera[id]) // Como no esta libre me encolo y me duermo

    }

    // Usa la impresora

    P(mutex)
    if (c.isEmpty()){
        libre = true
    } else {
        aux = c.pop(id)
        V(espera[aux])
    }
    V(mutex)

}
```

```
c.

sem turno([N] 0)

process persona[id: 0.. N-1]{

    if (id <> 1){
        P(turno[id])
        imprimir(documento)
        V(turno[id + 1])
    }else {
        imprimir(documento)
        V(turno[id + 1])
    }
    }
}
```

```
d.

sem termine = 0
sem turno([N] 0)
sem llegada = 0
sem mutex  = 1
cola c;

process persona[id: 0..P-1]{

    P(mutex)
    c.push(id)
    V(mutex)
    V(llegada)
    P(turno[id])
    imprimir(documento)
    V(termine) --> Esto nos olvidamos, xq tiene que avisar que termino
    //se retira
}

process coordinador{
    int id1;
```

```
    while (true){ // Podemos cortar el while

        P(llegada)
        P(mutex)
        id1 = c.pop(id)
        V(mutex)
        V(turno[id1])
        P(termine) --> Esto nos olvidamos, xq tiene que avisar que termino
    }

}
```

```
e.

vector meToca[P]
vector impresoras[5];
sem turno([N] 0)
cola c;
sem mutex = 1

sem termine = 5
cola c2;
sem llegada = 0

process persona[id: 0..P-1]{
    int idImpresora;

    P(mutex)
    c.push(id)
    V(mutex)
    V(llegada)
    P(turno[id])
    // en mi posicion del vector me fijo
    idImpresora = meToca[id]
    imprimir(documento,idImpresora)
    P(mutex2)
    c.push[idImpresora]
    V(mutex2)
    V(termine)
    //V(termine) --> Esto nos olvidamos, xq tiene que avisar que termino

    //se retira
}
```

```
-----------------------------------------------------------------

process coordinador{
    int id1;
    while (true){ // Podemos cortar el while

        P(llegada)
        P(mutex)
        id1 = c.pop(id)
        V(mutex)
        V(turno[id1])
        P(termine) --> Esto nos olvidamos, xq tiene que avisar que termino
    }



    while (true){
    // acá el coordinador debe encolar las impresoras para cuando inicia.

    P( mutex )
    for i:= 1 to 5
        c.push(id)
    V(mutex)

    recurso r;
        P(termine)
        P(mutex) --> mutex de cola
        r = c.pop()
        V(mutex)

        P(llegada)



    }

}
```

Ejercicio 7

Suponga que se tiene un curso con 50 alumnos. Cada alumno debe realizar una tarea y existen 10 enunciados posibles. Una vez que todos los alumnos eligieron su tarea, comienzan a realizarla. Cada vez que un alumno termina su tarea, le avisa al profesor y se queda esperando el puntaje del grupo (depende de todos aquellos que comparten el mismo

enunciado). Cuando un grupo terminó, el profesor les otorga un puntaje que representa el orden en que se terminó esa tarea de las 10 posibles.

Nota: Para elegir la tarea suponga que existe una función elegir que le asigna una tarea a un alumno (esta función asignará 10 tareas diferentes entre 50 alumnos, es decir, que 5 alumnos tendrán la tarea 1, otros 5 la tarea 2 y así sucesivamente para las 10 tareas).

```
<Esta solción es correcta si no dice que el alumno le avisa al profesor>

int cantAlumnos = 0
sem mutex = 1

sem comenzar = 0
vec tarea([10] 0)
sem modificar([10] 1)

cola grupoTermino;
sem notas([10] 0)
vec calificacion([10] 0)

sem arrancar = 0;
sem entrega = 0

process alumno[id:0..49]{
    tarea t;
    // Llega el alumno
    t = elegir()

    P(mutex)
    cantAlumnos ++
    if (cantAlumnos == 50){
        for i:= 0 to 49 do
            V(arrancar)
    }
    V(mutex)

    P(arrancar)
    //Hacen la tarea
    P(modificar[t])
    //Esto lo debe hacer el profesor ya que el enunciado dice que se le avisa
a el
    tarea[t] = tarea[t] ++
    if ( tarea[t] == 5 ) {
        //PROTEG
        grupoTermino.push(t)
        // despro
```

```
        // proteg
        V(entrega)
        // despro
    }
    V(modificar[t])

    P(notas[t]) // Esperar a que tenga mi nota
    imprimirNota(calificacion[t])

}


process profesor{
int grupo;

    for i= 1 to 10 do
        P(entrega)
        P(mutexG)  // proteger con mutex
        grupo = grupoTermino.pop()
        V(mutexG)  // desproteger con mutex

        calificacion[grupo] = i  //Le pone el orden en que entregaron
        for i= 1 to 5
            V(notas[grupo])          // Avisa que ya corrigio


}
```

Ejercicio 8

Una fábrica de piezas metálicas debe producir T piezas por día. Para eso, cuenta con E empleados que se ocupan de producir las piezas de a una por vez. La fábrica empieza a producir una vez que todos los empleados llegaron. Mientras haya piezas por fabricar, los empleados tomarán una y la realizarán. Cada empleado puede tardar distinto tiempo en fabricar una pieza. Al finalizar el día, se debe conocer cual es el empleado que más piezas fabricó.

a) Implemente una solución asumiendo que T > E.
b) Implemente una solución que contemple cualquier valor de T y E.

```
// DUDAS
// Ver si son dos semaforos apartes (pieza y estoyTrabajando)
// Que cambia el a y b
```

```
int cantidad([E] 0)
int llegue = 0
sem mutex = 1

int fabricadas = 0
sem mutex2 = 1

int terminamos = 0
sem mutex3 = 1

sem listo = 0


process empleado[id:0..E-1]{
    P(mutex)
    llegue ++
    if (llegue == E){
        for 1 to E do
            V(arrancar)
    }
    V(mutex)
    ////////////////////////////////////
    P(arrancar)
    ////////////////////////////////////
    P(mutex2)
    while (fabricadas < T){
        fabricadas ++
        V(mutex2)
        //Fabrica
        cantidad[id]+= 1
        P(mutex2)
    }
    V(mutex2)

    P(mutex3)
    terminamos ++
        if (terminamos == E){
            V(listo) --> ACÁ PODES PONER LO DEL CALCULAR MAX
        }
    V(mutex3)
}


// Fabrica no existe en este caso.
process fabrica{
int max, idMax
```

```
    P(listo)
    calcularMaximo(cantidad,max,idMax)
    // Se informa el idMax junto a la cantidad de piezas que es max
}
```

Ejercicio 9

Resolver el funcionamiento en una fábrica de ventanas con 7 empleados (4 carpinteros, 1 vidriero y 2 armadores) que trabajan de la siguiente manera:

- Los carpinteros continuamente hacen marcos (cada marco es armando por un único carpintero) y los deja en un depósito con capacidad de almacenar 30 marcos.
- El vidriero continuamente hace vidrios y los deja en otro depósito con capacidad para 50 vidrios.
- Los armadores continuamente toman un marco y un vidrio (en ese orden) de los depósitos correspondientes y arman la ventana (cada ventana es armada por un único armador).

```
int cantMarcos = 0
sem mutex = 1

process carpinteros[id: 0..3]){

    P(mutex)
    while(CantMarcos< 30){
        CantMarcos ++
        V(mutex)
        // Armar el marco


    }

}

process vidriero{


}

process armadores[id:0..1]{
```

```
}
```

Ejercicio 10 - ??

A una cerealera van T camiones a descargarse trigo y M camiones a descargar maíz. Sólo hay lugar para que 7 camiones a la vez descarguen, pero no pueden ser más de 5 del mismo tipo de cereal.

a) Implemente una solución que use un proceso extra que actúe como coordinador entre los camiones. El coordinador debe atender a los camiones según el orden de llegada. Además, debe retirarse cuando todos los camiones han descargado.
b) Implemente una solución que no use procesos adicionales (sólo camiones). No importa el orden de llegada para descargar. Nota: maximice la concurrencia.

```
a.

cola c;
sem mutex = 1
sem llegue = 0

sem entrar = 7

sem maiz = 5
sem trigo = 5

sem esperaTrigo([T] 0)
sem esperaMaiz([M] 0)

sem termineMaiz = 0
sem termineTrigo = 0
int totalCamiones = 0

process coordinador{
int actualDescargando = 0

    while (tot < (M+T))
        // que haya un lugar libre

        P(llegue) // Llega una persona
        P(entrar) // Puede entrar ya que hay lugar

        //Si entra debe verificar que tipo es
```

```
        if ()


        }

}

process camionTrigo[id:0..T-1]{
    // ORDEN DE LLEGADA NO IMPORTA SI SE DESAPRO ESPACIOS.

    // llega el camion
    p(mutex)
    c.push(id,"Trigo")
    V(mutex)
    V(llegue)

    P(esperaTrigo[id])
    //Descarga
    V(termineTrigo)
}

process camionMaiz[id:0..M-1]{

    // llega el camion
    p(mutex)
    c.push(id,"Maiz")
    V(mutex)
    V(llegue)

    P(esperaMaiz[id])
    //Descarga
    V(termineMaiz)
}
```

```
sem maiz = 5
sem trigo = 5
sem lugares = 7


process camionTrigo[id:0..T-1]{

    // llega el camion
    P(Trigo)
    P(lugares)
```

```
    //Descarga
    V(Lugares)    // Acá que cambia si hago primero V(trigo) ? nop
    V(Trigo)
}

process camionMaiz[id:0..M-1]{
    // llega el camion
    P(Maiz)
    P(lugares)
    //Descarga
    V(Lugares)
    V(Maiz)

}
```

Ejercicio 11 - preguntar chechu

En un vacunatorio hay un empleado de salud para vacunar a 50 personas. El empleado de salud atiende a las personas de acuerdo con el orden de llegada y de a 5 personas a la vez. Es decir, que cuando está libre debe esperar a que haya al menos 5 personas esperando, luego vacuna a las 5 primeras personas, y al terminar las deja ir para esperar por otras 5. Cuando ha atendido a las 50 personas el empleado de salud se retira. Nota: todos los procesos deben terminar su ejecución; suponga que el empleado tienen una función VacunarPersona() que simula que el empleado está vacunando a UNA persona.

```
sem vacunado([50] 0)
sem llegue = 0
cola c;
sem mutex = 1

sem antender = 5

process empleadoSalud{
int id;
c cola;
    for x = 1 to 10 do
        for i = 1 to 5   // verifico que llegaron 5 personas
            P(llegue)
        for i = 1 to 5
            P(mutex)
            c.pop(id)
            V(mutex)
            // Avisarle que lo vacunan ??????????????????????????'
```

```
            VacunarPersona(id)
            cola.push(id)
        for i = 1 to 5
            V(vacunado[c.pop()])   // Le aviso que la vacunaron
}

process persona[id:0..49]{
    P(mutex)
    c.push(id)        //Se pone en la fila
    V(mutex)
    V(llegue)        //Avisa que llego

    // Espera a ser atiendido - completarrrrrrrrrrrrrrrrrrrrr
    // Se esta vacunando
    P(vacunado[id]) // Le avisan que ya lo vacunaron y se puede retirar
}
```

Ejercicio 12

Simular la atención en una Terminal de Micros que posee 3 puestos para hisopar a 150 pasajeros. En cada puesto hay una Enfermera que atiende a los pasajeros de acuerdo con el orden de llegada al mismo. Cuando llega un pasajero se dirige al Recepcionista, quien le indica qué puesto es el que tiene menos gente esperando. Luego se dirige al puesto y espera a que la enfermera correspondiente lo llame para hisoparlo. Finalmente, se retira.

a) Implemente una solución considerando los procesos Pasajeros, Enfermera y Recepcionista.
b) Modifique la solución anterior para que sólo haya procesos Pasajeros y Enfermera, siendo los pasajeros quienes determinan por su cuenta qué puesto tiene menos personas esperando.

Nota: suponga que existe una función Hisopar() que simula la atención del pasajero por parte de la enfermera correspondiente.

```
a.

// La persona denbe avisar que se fue para decrementar el semaforo de 3 . La
enfermera si decrementa pierde el tiempo .

sem personas([150] 0)

int cantidad = 0
int mutex = 1

process Pasajero[id : 0..149]{
```

```
    P(personas[id]) //Espera a ser despertado
    // Se esta  hisopando
    P(personas[id]) //Espera a poder retirarse

}


process Enfermera[id: 0..2]{
int pasajero;

    P(mutex)
    while (cantidad < 150){
        //saca el id de la persona
        cantidad ++
        V(mutex)

        V(personas[pasajero]) // despertamos a la persona
        Hisopar(pasajero)

        P(mutex)
    }
    V(mutex)

}


process Recepcionista{


}
```

b.