Practica 2

jueves, 1 de octubre de 2020 18:53

1. Existen N personas que deben ser chequeadas por un detector de metales antes de poder ingresar al avión.
   a. Implemente una solución que modele el acceso de las personas a un detector (es decir si el detector está libre la persona lo puede utilizar caso contrario debe esperar).
   b. Modifique su solución para el caso que haya tres detectores.

```
1)
a)
sem mutex = 1;

Process Persona[0..N-1]
{
    P(mutex);
    persona.detectarse();
    V(mutex);
}

b)
sem mutex = 3;

Process Persona[0..N-1]
{
    P(mutex);
    persona.detectarse();
    V(mutex);
}
```

Sin robar: (solo un poco porque hacemos un vector porque no podemos leer el valor de un semáforo)

```
b)
int detectores[3] = (0,0,0)
sem mutex = 1;
sem hayLibre = 3;

Process Persona[0..N-1]
{
    P(hayLibre);
    P(mutex);
    int elegido = elegirDetector(detectores);
    detectores[elegido] = 1;
    V(mutex);
    //usar detector
    detectores[elegido] = 0;
    V(hayLibre);
}
```

Como sería sin robar?

2. Un sistema operativo mantiene 5 instancias de un recurso almacenadas en una cola, cuando un proceso necesita usar una instancia del recurso la saca de la cola, la usa y cuando termina de usarla la vuelve a depositar.

```
Cola cola[5];
sem hayRecurso = 5;
sem usaCola = 1;

Process consumidor[0..N-1]{
    while(True){
        P(hayRecurso);
        P(usaCola);
        cola.pop();
        V(usaCola);
        //usa recurso
        P(usaCola);
        cola.push();
        V(usaCola);
        V(hayRecurso);
    }
}
```

3. 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.

Var

3. 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.

```
Var
    sem: semaphoro := 6;
    alta: semaphoro := 4;
    baja: semaphoro := 5;
```

```
Process Usuario-Alta [I:1..L]::
{   P (sem);
    P (alta);
     //usa la BD
    V(sem);
    V(alta);
}
```

```
Process Usuario-Baja [I:1..K]::
{   P (sem);
    P (baja);
    //usa la BD
    V(sem);
    V(baja);
}
```

La solución más eficiente es la de cambiar de lugar los dos primeros semáforos, es decir primero P(alta); y luego P(sem); y b mismo con P(baja).
Por qué? Por ejemplo, si entran 6 usuarios a alta, 4 estarían usando la base y 2 esperando. Es innecesario que esos 2 estén esperando, porque podrían haber 2 usuarios de baja utilizando la base, pero están restringidos. En palabras sabias:

```
El ejercicio original tendría demoras innecesarias en casos como este: llegan 5 de alta prioridad primero, entran 4 y unc
queda esperando, generando una demora para los de prioridad baja que sí pueden entrar.
```

4. Se tiene un curso con 40 alumnos, la maestra entrega una tarea distinta a cada alumno, luego cada alumno realiza su tarea y se la entrega a la maestra para que la corrija, esta revisa la tarea y si está bien le avisa al alumno que puede irse, si la tarea está mal le indica los errores, el alumno corregirá esos errores y volverá a entregarle la tarea a la maestra para que realice la corrección nuevamente, esto se repite hasta que la tarea no tenga errores.

```
sem esperarTarea[40] = 0;
sem profeLibre = 1;
sem corregir = 0;
boolean estabien[40] = false;
int aprobados = 0;

process Alumno[id:0..39]{
    P(esperarTarea[id]);
    while(!estabien[id]){
        //hace la tarea
        P(profelibre);
        //entregar
        V(corregir);
        P(correccion);
    }
}

process Maestra{
    for(int i = 0 to 39){
        //entrega la tarea al alumno i
        V(esperarTarea[i]);
    }
    while(aprobados<40){
        P(corregir); //espera tarea
        //corrige y setea estabien[id]
        //incrementa aprobados si procede
        V(correccion);
        V(profelibre);
    }
}
```

5. 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 (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.

```
a)
sem libre = 1;

process persona[0..N-1]{
    P(libre);
    imprimir(documento);
    V(libre);
}

b)
sem esperar[N] = 0;
sem sacarNumero = 1;
int siguiente = 0;

process persona[0..N-1]{
    P(sacarNumero);
    id = siguiente + 1;
    V(sacarNumero);
    if(id != 1){
        P(esperar[id]);
    }
    imprimir(documento);
```

```
        P(esperar[id]);
    }
    imprimir(documento);
    V(esperar[id+1]);
}

c)
sem esperar[N] = 0;
sem usarCola = 1;
sem hayGente = 0;
sem usando = 0;
int siguiente = 0;
Cola cola[N];

process persona[id:0..N-1]{
    P(usarCola);
    cola.push(id)
    V(usarCola);
    V(hayGente);
    P(esperar[id]);
    imprimir(documento);
    V(usando);
}

process admin{
    while(True){
        P(hayGente)
        V(esperar[cola.pop()]);
        P(usando);
    }
}
```

6. Suponga que se tiene un curso con 50 alumnos. Cada alumno elije una de las 10 tareas para realizar entre todos. 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. Cuando todos los alumnos que tenían la misma tarea terminaron el profesor les otorga un puntaje que representa el orden en que se terminó esa tarea. ***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).

```
process alumno[id:0..49]{
    tarea = asignarTarea(); //devuelve el número de la tarea
    P(mutex);
    contador = contador + 1;
    if (contador == 50) {
        for i = 1..50 → V(barrera);
        V(espera_profesor);
    }
    V(mutex);
    P(barrera);
    //realizar tarea
    P(mutex);
    contador[tarea] = contador[tarea] + 1;
    //entrega la tarea
    if(contador[tarea] == 5){
        P(usarCola);
        entregas.push(tarea); //tarea es la tarea de los 5 flacos
        V(usarCola);
        V(hayEntregas);
    }
    V(mutex);
    P(esperarTarea[tarea]); //espera la nota
    //recibe la nota y saluda
}

process profesor{
    orden = 0
    p(espera_profesor);
    while(True){
        p(hayEntregas);
        orden++;
        P(usarCola);
        tarea = entregas.pop();
        V(usarCola);
        //corrige la tarea y devuelve la nota con el orden
        for i = 1..5 → V(esperarTarea[tarea]);
    }
}
```

7. A una empresa llegan E empleados y por día hay T tareas para hacer (T>E), una vez que todos los empleados llegaron empezaran a trabajar. Mientras haya tareas para hacer los empleados tomarán una y la realizarán. Cada empleado puede tardar distinto tiempo en realizar cada tarea. Al finalizar el día se le da un premio al empleado que más tareas realizó.

```
int contador[E] = 0;
int salieron = 0;

process empleado[id:0..E-1]{
    P(usarContador);
    contador++;
    if(contador == E){
        for i = 1..E -> V(barrera);
    }
    V(usarContador);
    P(barrera);
    while(colaVacia()){
        P(usarCola);
        tarea = cola.pop()
        V(usarCola);
        tarea.realizar(); //hace la tarea
        contador[id]++;
    }
    P(semSalieron);
    salieron++;
    if(salieron = E){
        contador.EntregarPremio(); //calcula el máximo y entrega el premio a ese id
    }
    V(semSalieron);
}

function colaVacia(){
    P(usarCola);
    aux = !cola.empty();
    V(usarCola);
    return aux;
}
```

Otra forma sin la función del while:

```
//la cola es una cola con las tareas
int contador[E] = 0;
int salieron = 0;

process empleado[id:0..E-1]{
    P(usarContador);
    contador++;
    if(contador == E){
        for i = 1..E -> V(barrera);
    }
    V(usarContador);
    P(barrera);
    P(usarCola);
    while(!cola.empty();){
        tarea = cola.pop()
        V(usarCola);
        tarea.realizar(); //hace la tarea
        contador[id]++;
        P(usarCola);
    }
    V(usarCola);
    P(semSalieron);
    salieron++;
    if(salieron = E){
        contador.EntregarPremio(); //calcula el máximo y entrega el premio a ese id
    }
    V(semSalieron);
}
```

8. 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).

```
sem hayLugarVidrios = 50;
sem usarBufferVidrios = 1;
sem hayLugarMarcos = 30;
sem usarBufferMarcos = 1;
vec marcos[30];
int proxMarcos = 0;
vec vidrios[50];
int proxConsMarcos = 0;
int proxConsVidrios = 0;


process carpinteros[id:0..3]{
    while(True){
        marco = carpintero.generarMarco(); //produce marco
        P(hayLugarMarcos);
        P(usarBufferMarcos);
        proxMarcos = (proxMarcos + 1) mod 30
        marcos[proxMarcos] = marco;
        V(usarBufferMarcos);
    }
}

process vidriero{
    while(True){
        vidrio = vidriero.generarVidrio(); //produce vidrio
        P(hayLugarVidrios);
        P(usarBufferVidrios);
        proxVidrios = (proxVidrios + 1) mod 50
        vidrios[proxVidrios] = vidrio;
        V(usarBufferVidrios);
    }
}

process armador[id:0..1]{
    while(True){

        P(usarBufferMarcos);
        marco = marcos[proxConsMarcos];
        proxConsMarcos = (proxConsMarcos + 1) mod 30;
        V(usarBufferMarcos);
        V(hayLugarMarcos);

        P(usarBufferVidrios);
        vidrio = Vidrios[proxConsVidrios];
        proxConsVidrios = (proxConsVidrios + 1) mod 50;
        V(usarBufferVidrios);
        V(hayLugarVidrios);

        armador.ensamblar(marco,vidrio);

    }
}
```

9. 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. ***Nota:*** no usar un proceso extra que actué como coordinador, resolverlo entre los camiones.

```
sem lugarMaiz = 5;
```

```
sem lugarMaiz = 5;
sem lugarTrigo = 5;
sem lugarCamion = 7;

process maiz[id:0..M]{
    P(lugarMaiz);
    P(lugarCamion);
    //camion.descargar();
    V(lugarCamion);
    V(lugarMaiz);
}

process trigo[id:0..T]{
    P(lugarTrigo);
    P(lugarCamion);
    //camion.descargar();
    V(lugarCamion);
    V(lugarTrigo);
}
```

10)

El algoritmo no cumple con las condiciones ya que el uso del vector estado no está en una sección crítica, lo que puede ocasionar que dos procesos lean la condición del if como verdadera y provocar un error.

Para solucionarlo deberían los procesos tener un semáforo para usar el vector como en este ejemplo:

```
Profesor A::
{ int idAlumno
  while (true)
    { P(llegoA)
      P(mutexA)
      idAlumno = pop(colaA)
      V(mutexA) P(verVector )
      If (estado[idAlumno] = ="Esperando")
        { estado[idAlumno] = "A"; V(verVector )
          V(esperando[idAlumno])
          //Se toma el examen//
          V(esperando[idAlumno])
        }
      else{
          V(verVector )
      }
    }
}
```

11. Existe una casa de comida rápida que es atendida por 1 empleado. Cuando una persona llega se pone en la cola y espera a lo sumo 10 minutos a que el empleado lo atienda. Pasado ese tiempo se retira sin realizar la compra.

```
process persona[i=1 to N]{
    P(sCola)
    cola.encolar(i)
    V(sCola)
    V(sPersona)
    V(sEsperando[i])
    P(sSalida[i])
}

process coordinador[i=1 to N]{
    P(sEsperando[i])
    delay(10min)
    P(sEstado[i])
    if (estado == "esperando"){
        estado[i] = "saliendo"
        V(sEstado[i])
        V(sSalida[i])
    }
}

process empleado{
    while(true)
        P(sPersona)
        P(sCola)
        persona = cola.desencolar()
        V(sCola)
        P(sEstado[persona])
        if (estado[persona] == "esperando"){
            estado[persona] = "atendido"
            V(sEstado[persona])
            atender(persona)
            V(pSalida[persona])
        }
    }
}
```