Universidad Nacional de Tierra del Fuego
Instituto de Desarrollo Económico e Innovación
Analista Universitario en Sistemas - Licenciatura en Sistemas

Introducción a la Concurrencia

Trabajo Práctico

Profesor: Lic. Horacio Pendenti
Ayudante: Lic. Luis Rojas
Ushuaia - Tierra del Fuego

Índice de Contenidos

Cronograma

Resumen de prácticas y duración estimada en módulos de 2 hs.

TP	Tema	Duración Práctica
TP1	ISSD – Introducción, problemática, soluciones soft y Deadlocks.	3
TP2	VCS - Variables Compartidas - Semáforos	4
TP3	VCM - Variables Compartidas - Monitores	4
TP4	CPM – Concurrencia con Pasaje de Mensajes	4

Bibliografía específica para la práctica Además de la Bibliografía sugerida inicialmente para toda la materia, se recomienda:

- **Operating Systems.** William Stallings.5th Edition.
- **Concurrent Programming, Principles and Practice.** Gregory R. Andrews.
- **Foundations of Multithreaded, Parallel, and Distributed Programming.** Gregory R. Andrews.
- **Principles of Concurrent and Distributed Programming.** M. Ben-Ari.
- Apunte: **The JR Programming Language.**
- **Concurrent programming in an Extended Java Ronald A. Olson, Aaron W. Keen.**
- Web: https://baptiste-wicht.com/categories/jr.html
- Web: https://web.cs.ucdavis.edu/~olsson/research/jr/

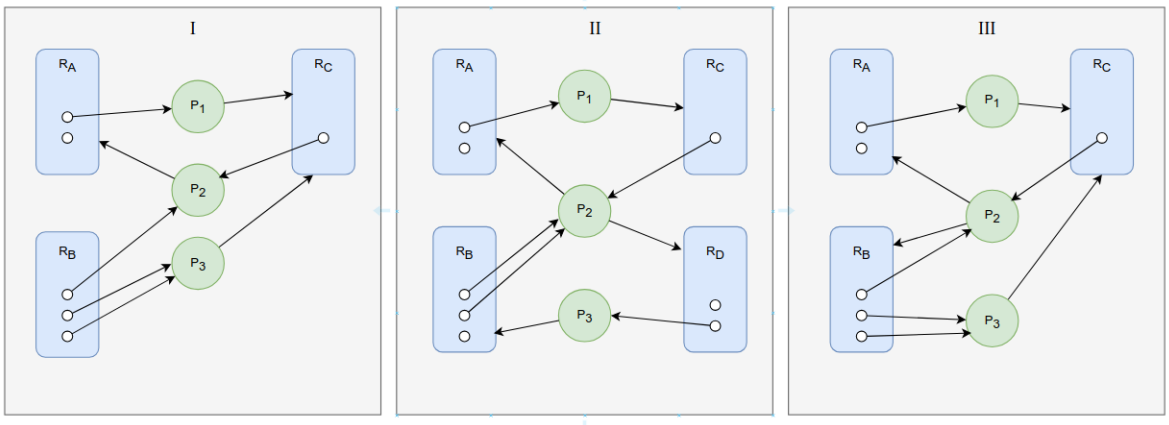
Trabajo Practico N° 1

ISS – Introducción, Soluciones Soft y Deadlock.

Ejercicios a entregar: 7, 10, 13, 18, 19.

1. Revise el ciclo básico de trabajo del CPU, haga el esquema y explique el funcionamiento.
2. Explique el mecanismo de trabajo bajo interrupciones. Analice su importancia respecto a sistemas estrictamente secuenciales y monoproceso.
3. ¿En qué consiste el concepto de enmascaramiento de interrupciones?.
4. Proporcione algunas definiciones de Proceso. Interprete y explique el diagrama de estado de procesos.
5. ¿Que es condición de concurso (race condition)?
6. En un sistema con 5 procesos y 4 recursos se tiene el siguiente estado:
   a) P_1 solicita R_1 y R_3 y tiene asignado R_2.
   b) P_2 solicita R_1 y R_2 y tiene asignado R_3.
   c) P_3 solicita R_4.
   d) P_4 solicita R_3 y tiene asignado R_1.
   e) P_5 tiene asignado R_4.

   Dibuje el grafo correspondiente y explique si el sistema está en un estado seguro o no.
7. En relación a los deadlocks, para los tres gráficos de asignación de recursos que siguen (I, II y III), indique si:
   a) ¿Se encuentra en estado seguro o inseguro?.
   b) ¿Se encuentra en una situación de deadlock?.
   c) Proponga una modificación en alguno de los diagramas correspondiente a un sistema que se encuentre en estado seguro, de forma tal que pase a estar en deadlock.



8. Proporcione un ejemplo mediante un gráfico de asignación de recursos donde figuren al menos tres procesos, dos recursos de tipo A, dos recursos de tipo B y dos recursos de tipo C y se pueda observar un deadlock.

9. Dados los siguientes estados de un sistema, establezca si son o no estados seguros. En caso afirmativo, mostrar cómo podrían completarse los procesos.

caso a)			caso b)			caso c)		
Proceso	Asignado	Máximo	Proceso	Asignado	Máximo	Proceso	Asignado	Máximo
P_1	2	5	P_1	4	5	P_1	2	9
P_2	4	6	P_2	3	7	P_2	1	4
P_3	5	7	P_3	3	9	P_3	16	19
P_4	0	3	Rec. Disponibles: 1			P_4	1	2
Rec. Disponibles: 2						Rec. Disponibles: 1		

10. Un sistema que usa el algoritmo del banquero para evitar deadlocks tiene 5 procesos y recursos de cuatro tipo diferentes. El estado del sistema se encuentra definido en las siguientes matrices:

Proceso	Recursos Asignados				Matriz de Máxima				Recursos Totales			
	A	B	C	D	A	B	C	D	A	B	C	D
P_1	1	0	2	0	3	2	4	2	13	13	9	13
P_2	0	3	1	2	3	5	1	2				
P_3	2	4	5	1	2	7	7	5	Recursos Disponibles			
P_4	3	0	0	6	5	5	0	8	A	B	C	D
P_5	4	2	1	3	6	2	1	4	3	4	0	1

a) ¿Está el sistema en un estado seguro?. Justifique su respuesta.

b) Si está en un estado seguro, muestre cómo podrían terminar su ejecución todos los procesos exitosamente. Si no lo está, muestre mediante una secuencia de asignaciones de recursos cómo podría ocurrir un deadlock.

11. Considere un programa concurrente con dos procesos, **p** y **q**, que se muestran a continuación. Asuma que **A**, **B**, **C**, **D** y **E** son instrucciones arbitrarias pero atómicas (indivisibles).

```
Process P;
begin
  A;   B; C;
end.
```

```
Process Q;
begin
  E;
  D;
end.
```

```
Program intercalacion;
begin
  cobegin
    p; q;
  coend
end.
```

a) Muestre todas las posibles intercalaciones en la ejecución de los dos procesos anteriores.

12. Reescriba el siguiente cálculo en paralelo como una secuencia simple de cálculo.

```
process p1;
begin
  d := b * c - x;
end;
```

```
Process p2;
begin
  e := a / 6 + n ^ 2;
end.
```

```
Program consec;
var
a, b, c, d, e, n, x: real;
begin
  cobegin
    p1; p2;
  coend
end.
```

13. Divida el cálculo de la siguiente expresión entre procesos independientes con el fin de lograr el máximo grado de paralelismo. Utilice JR.

$$y = \frac{3 \cdot a \cdot b + 4}{(c - d)^{(e-f)}}$$

14. Considere un programa concurrente con los siguientes dos procesos:

```
process p1;
begin
  x := x + 1;
  y := y + x;
end;
```

```
process p2;
begin
  x := x + 2;
  y := y - x;
end;
```

```
program eje14;
var
x,y: real;
begin
  x := 0; y := 0;
  cobegin
    p1; p2;
  coend;
end;
```

a) Suponga que las sentencias de asignación utilizadas son implementadas por una única instrucción de máquina, por lo tanto atómicas.

   i. ¿Cuántas posibles historias hay?

   ii. ¿Cuales son los posibles valores de x e y?

b) Suponga que cada sentencia de asignación es implementada por 3 acciones atómicas que cargan un registro, incrementan o decrementan el valor del registro y almacenan el resultado.

   i. ¿Cuantas posibles historias hay ahora?

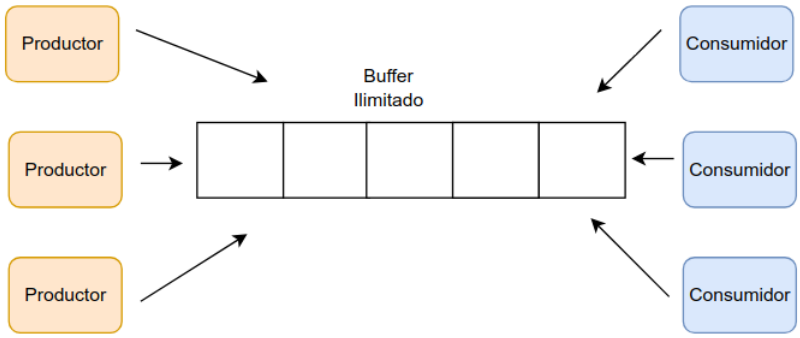
   ii. ¿Cuales son los posibles valores finales de **x** e **y**?

15. Escriba, ejecute y experimente en JR los primeros intentos de exclusión mutua vistos en la teoría. Arme un cuadro con sus conclusiones, en el que debe constar por cada uno de ellos si cumple o no con el objetivo y porqué, problemas, ventajas y desventajas y otras consideraciones.

16. Ídem anterior pero con los algoritmos de Dekker y Peterson.

17. Utilizando JR, escriba un programa concurrente que encuentre el máximo valor en un arreglo entero a[1..n], buscando en los elementos con subíndice par e impar en paralelo.

18. Productor consumidor. El problema describe dos procesos, el productor y el consumidor, que comparten un búffer en común utilizado como una cola. El trabajo del productor es el de generar una pieza de información, ponerla en el búffer y comenzar de nuevo. Al mismo tiempo, el consumidor está consumiendo los datos (es decir, sacándolo de la memoria intermedia) una pieza a la vez. El problema es asegurarse de que el productor no va a tratar de agregar los datos en el búffer si está lleno y que el consumidor no va a tratar de eliminar los datos de un búffer vacío.



Escriba en JR un programa concurrente con un proceso productor y otro consumidor. El productor introducirá elementos enteros en un arreglo de dimensión **x** relativamente pequeño (por ej. 20) en orden. El consumidor deberá leerlos en el orden correcto y mostrarlos por pantalla. Corrobore mediante la ejecución repetida del código que su solución no está libre de interferencia.

19. Explique el concepto de:

- Sección crítica
- Exclusión mutua
- Sincronización
- Sincronización por condición
- Protocolo de entrada
- Protocolo de salida.

Dé un ejemplo en pseudocódigo donde se efectúe una sincronización por exclusión mutua y otro donde se sincronice por condición; identificando sección crítica, protocolo de entrada y de salida.

Trabajo Practico N° 2

SCS – Sincronización con Semáforos

Ejercicios a entregar: 3, 4, 5c, 8b, 10, 11, 12.

1. Explique las diferencias entre espera ocupada o activa (busy waiting) y bloqueo. ¿Es la espera activa menos eficiente (en términos de uso del procesador) que el bloqueo?

2. ¿Porqué las operaciones sobre los semáforos deben ser atómicas?

3. Suponga que una máquina tiene instrucciones atómicas de incremento y decremento: **INC(var)** y **DEC(var)**. Estas operaciones suman y restan respectivamente 1 a var. Asumiendo que las lecturas y escrituras a memoria son operaciones atómicas:

   a) Es posible simular las operaciones wait y signal de un semáforo general s?. Si lo es, proporcione la simulación y explique el funcionamiento. Caso contrario, explique detalladamente porqué no es posible.

   b) Ahora asuma que INC y DEC también devuelven el bit de signo del valor final de var. En particular, si el valor final de var es negativo, las operaciones devuelven 1, en otro caso, 0. Es posible en este caso simular las operaciones wait y signal para un semáforo general s?. Si lo es, escriba la simulación y explique su funcionamiento, sino, explique detalladamente porqué no es posible.

4. Implemente la operación **TestAndSet** en JR, utilizando los semáforos provistos.

5. Utilice semáforos en JR para proporcionar una solución al clásico problema "productores / consumidores" con las siguientes variantes:

   a) búffer simple, **1** productor y **1** consumidor.

   b) búffer ilimitado, **1** productor y **1** consumidor.

   c) búffer ilimitado, **n** productores y **m** consumidores.

   d) búffer acotado, **1** productor y **1** consumidor.

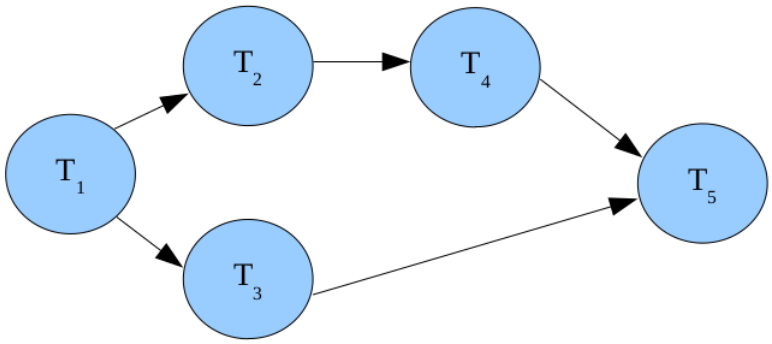
   e) búffer acotado, **n** productores y **m** consumidores.

6. Implemente semáforos binarios en JR valiéndose de los semáforos que éste provee.

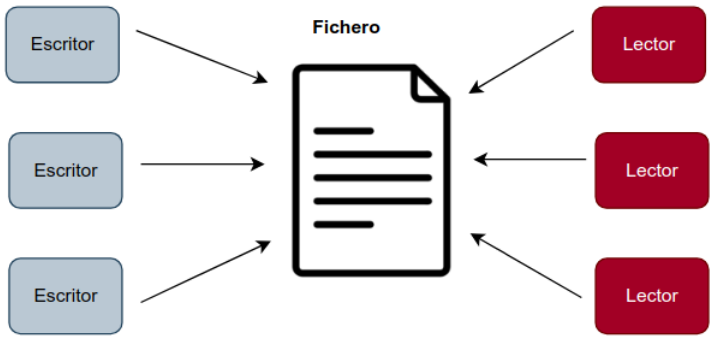
7. Un grafo de precedencia, es un grafo acíclico dirigido. Los nodos representan tareas y los arcos indican el orden en el que las tareas se ejecutan. En este caso, una tarea puede ejecutarse tan pronto como todos sus predecesores hayan terminado. Asumiendo que las tareas son procesos con el siguiente código:

```
process T[i];
  while(true)begin
    WAIT a los predecesores, si los hay; {toma tenedor izquierdo/derecho}
    cuerpo de la tarea;
    SIGNAL a los sucesores si loas hay;
  end;
end.

```

a) Utilizando semáforos, muestre cómo sincronizar cinco procesos en JR cuyo orden de ejecución está especificado por el siguiente grafo de precedencia:



8. Problema de lectores y escritores: Hay un objeto de datos (por ejemplo, un arreglo) que es utilizado por varios procesos, unos leen y otros que escriben. Mientras uno escribe, ningún otro proceso podrá acceder. Sí se permite a varios procesos leer simultáneamente.



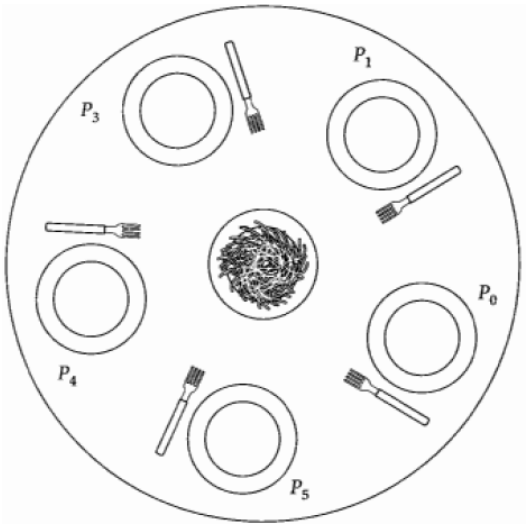
Implemente soluciones al problema de "lectores / escritores" utilizando semáforos en JR en las que:

a) Los lectores tienen prioridad.

b) Los escritores tienen prioridad.

c) Una solución “justa”.

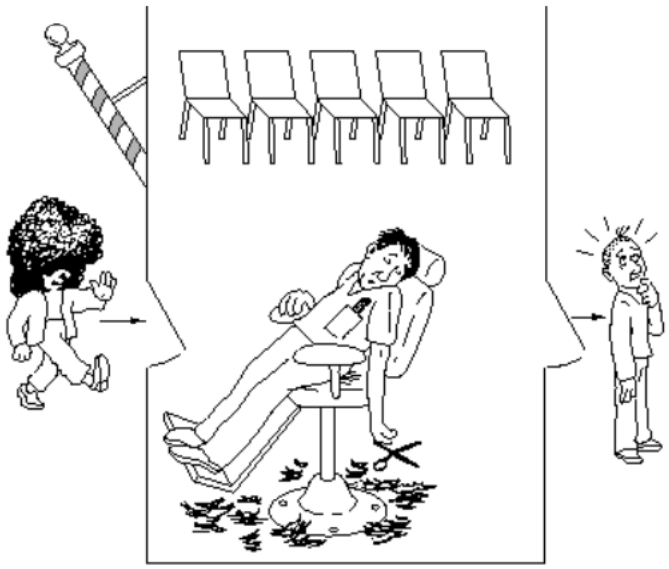
9. En 1965 Dijkstra planteó y resolvió un problema de sincronización llamado el problema de la cena de los filósofos, que se puede enunciar como sigue. Cinco filósofos se sientan alrededor de una mesa y pasan su vida cenando y pensando. Cada filósofo tiene un plato de arroz y un palillo a la izquierda de su plato. Para comer el arroz son necesarios dos palillos y cada filósofo sólo puede tomar los que están a su izquierda y derecha. Si cualquier filósofo coge un palillo y el otro está ocupado, se quedará esperando, con el palillo en la mano, hasta que pueda coger el otro palillo, para luego empezar a comer. Si dos filósofos adyacentes intentan tomar el mismo palillo a la vez, se produce una condición de carrera: ambos compiten por tomar el mismo palillo, y uno de ellos se queda sin comer. Si todos los filósofos cogen el palillo que está a su derecha al mismo tiempo, entonces todos se quedarán esperando eternamente, porque alguien debe liberar el palillo que les falta. Nadie lo hará porque todos se encuentran en la misma situación (esperando que alguno deje su palillo). Entonces los filósofos se morirán de hambre. Este bloqueo mutuo se denomina interbloqueo. El problema consiste en encontrar un algoritmo que permita que los filósofos nunca se mueran de hambre.



Se propone la siguiente solución al problema de la cena de filósofos. Usted deberá indicar si la misma es correcta o no, en caso negativo proporcione alguna solución, en caso positivo, justifique.

```
program filosofos;
procedure COMER; begin sleep(random(10)); end;
procedure PENSAR; begin sleep(random(10)); end;


process type tfilosofo(i:integer);
begin
  repeat
    wait(fork[i]):       {toma tenedor izquierdo}
    wait(fork[(i+1) mod 5]): {toma tenedor derecho}
    COMER;
    wait(fork[i]):
    wait(fork[(i+1) mod 5]): {libera tenedor izquierdo}
    PENSAR;              {libera tenedor derecho}
  forever;
end;

var
  i:integer;
  fork: array[1..5] of semaphore;
  filosofo: array[1..5] of semaphore;

begin                    {programa principal}
for i:=1 to 5 do        {5 semaforos inicializados a 1}
  initial(fork[i],1);

  cobegin                {comienza tareas}
    for i:=1 to 5 do
    filosofo[i](i);
  coend;
end.

```

10. El problema del barbero durmiente consiste en una barbería en la que trabaja un barbero que tiene un único sillón de barbero y varias sillas para esperar. Cuando no hay clientes, el barbero se sienta en una silla y se duerme. Cuando llega un nuevo cliente, éste o bien despierta al barbero o —si el barbero está afeitando a otro cliente— se sienta en una silla (o se va si todas las sillas están ocupadas por clientes esperando). El problema consiste en realizar la actividad del barbero sin que ocurran condiciones de carrera.



Implemente una solución al mismo utilizando semáforos en JR con las siguientes restricciones:

a) Puede haber N clientes esperando en la barbería, además de quien está sentado en el sillón de corte.

b) Hay un único sillón para corte y un único Barbero que además cobra en la única caja disponible.

11. Un Chef y tres ayudantes trabajan en una Pizzería de un modo muy particular. Para hacer una pizza se requiere de tres ingredientes (además de la masa): salsa, queso y morrones. Uno de los ayudantes tiene una cantidad infinita de salsa, otro de queso y el restante de morrones, mientras que el Chef cuenta con una cantidad infinita de los tres ingredientes. Las pizzas se hacen del siguiente modo: el Chef coloca sobre la masa dos de los ingredientes de la pizza, avisa que hay una pizza lista para completar y continúa con la preparación de la siguiente. El ayudante que tiene el ingrediente faltante la completa y avisa que terminó. Recién en este momento, el Chef puede poner nuevamente una pizza disponible a los ayudantes. Ud. debe escribir en JR utilizando semáforos un programa que simule esta situación y resuelva los problemas de sincronización que haya, asegurando que tampoco se produzcan interbloqueos. Esta solución debe arrojar además como resultado la cantidad de pizzas que se han hecho al final de la jornada.

12. Para poder determinar el futuro de un ecosistema, a Ud. se le encarga que construya un simulador sobre parte del mismo, en el que conviven diversos organismos y también gran diversidad de bacterias que los atacan. Particularmente el interés es el estudio de los organismos X que son atacados por las bacterias **Blancas**y **Rojas**. Estas bacterias compiten por atacar a los X y se sabe que aleatoriamente éstos se vuelven vulnerables; si esto ocurre, es a partir de ese momento que las bacterias comienzan su ataque que

siempre culmina exitosamente y en una forma muy precisa; cuando 4 bacterias Blancas y 2 Rojas lograron introducirse en el organismo X, éste muere. Se sabe además que las bacterias blancas se pueden "introducir" en X de a 2 a la vez, mientras que las rojas sólo de a 1.

a) Utilice semáforos en JR para construir el simulador que debe ser capaz de correr para 1 organismo X que es atacado por n bacterias Blancas y m bacterias Rojas (con n y m grandes).

b) Generalice para más de 1 X.

Trabajo Practico N° 3

SCM - Sincronización con Monitores

Ejercicios a entregar: 3, 6,12 y 13.

1. Explique en un cuadro las diferencias y similitudes en el uso de las primitivas de semáforos y monitores.

2. Demuestre que semáforos y monitores son funcionalmente equivalentes implementando un semáforo general utilizando monitores. Muestre su uso mediante un programa ejemplo.

3. Demuestre que semáforos y monitores son funcionalmente equivalentes implementando una librería que permita simular un monitor valiéndose de semáforos. Muestre su uso mediante un programa ejemplo.

4. Use monitores "de Hoare" para implementar una solución al problema de los lectores y escritores.

5. Use monitores de Hoare para implementar una solución al problema de los lectores y escritores de modo tal que se vean favorecidos los lectores.

6. Use monitores de Hoare para implementar una solución al problema de los lectores y escritores de modo tal que se vean favorecidos escritores.

7. Utilice monitores con notify / broadcast para resolver el problema de lectores y escritores.

8. Utilice monitores de Hoare para proporcionar una solución al clásico problema "productores / consumidores" con las siguientes variantes:

a) Búffer simple, **1** productor y **1** consumidor.

b) Búffer ilimitado, **1** productor y **1** consumidor.

c) Búffer ilimitado, **n** productores y **m** consumidores.

d) Búffer acotado, **1** productor y **1** consumidor.

e) Búffer acotado, **n** productores y **m** consumidores.

9. Utilizando monitores de Lampson & Redell, proporcione una solución al problema de productor / consumidor con un buffer limitado.

10. En relación al conocido problema de la Barbería, implemente una solución al mismo utilizando un monitor según la definición de Lampson & Redell en JR con las siguientes restricciones:

a) Puede haber N clientes esperando en la barbería, además de quien está sentado en el sillón de corte.

b) Hay un único sillón para corte y un único Barbero que además cobra en la única caja disponible.

11. Se necesita simular el funcionamiento de un pequeño módulo de un sistema que permite asignar N recursos de tipo X a m procesos que los requieren. Cada uno de los procesos necesita 2 recursos del tipo X para poder finalizar. Las experiencias muestran que la cantidad de procesos m es muy superior a la cantidad de recursos totales (N). Este simulador debe ser implementado

Utilizando monitores de Hoare, teniendo en cuenta:

a) Siempre N es > 1

b) Especifique los supuestos que asuma.

c) la solución debe mostrar también la desasignación de recursos.

12. Implemente una solución para el problema de la pizzería ya enunciado, utilizando monitores de Hoare.

13. El sistema en cuestión es un doble procesador (cpu0 y cpu1), cuya carga de procesos es administrada por un monitor llamado despachador que Ud. debe implementar. Este monitor (despachador de procesos en este caso) se encarga entonces de asignar CPUs libre a los N procesos que la requieren por medio de una llamada al monitor **"despachador.adquirirCPU(Pid)"**. Cuando un proceso hace el requerimiento, el monitor verifica si se encuentra disponible alguna de las 2 CPUs y en caso afirmativo, sin mediar ninguna política de planificación en particular (salvo la propia disciplina de colas implementadas en el monitor) se le asigna. En caso de no haber ninguna CPU disponible el proceso que intenta la adquisición es bloqueado hasta que otro proceso en ejecución termine y ese procesador pueda ser nuevamente asignado. La solución debe ser tal que en todo momento el monitor debe conocer como mínimo el estado de cada uno de los procesadores (libre / ocupado) y al final de la corrida, debe proporcionar el total de asignaciones realizadas a cada uno de los procesadores.

Trabajo Practico N° 4

CPM – Concurrencia con Pasaje de Mensajes

Ejercicios a entregar: 1, 4e, 9 y 10.

1. En relación al pasaje de mensajes:

   a) Clasifique y defina los distintos métodos de direccionamiento conocidos.

   b) Clasifique y explique los distintos esquemas de sincronización posibles.

2. Explique en qué consiste la sincronización rendezvous utilizada entre otros por JR.

3. Implemente la operación TestAndSet utilizando mensajes.

4. Utilice mensajes para proporcionar una solución al clásico problema "productor/consumidorçon las siguientes variantes:

   a) Búffer simple, **1** productor y **1** consumidor en JR.

   b) Búffer ilimitado, **1** productor y **1** consumidor.

   c) Búffer ilimitado, **n** productores y **m** consumidores.

   d) Búffer acotado, **1** productor y **1** consumidor.

   e) Búffer acotado, **n** productores y **m** consumidores.

5. Implemente un semáforo general utilizando mensajes.

6. Implemente soluciones al problema de “lectores / escritores” utilizando pasaje de mensajes, con las siguientes variantes:

   a) Los lectores tienen prioridad.

   b) Los escritores tienen prioridad.

   c) Una solución “justa”.

7. En relación al conocido problema de la Barbería, implemente una solución al mismo utilizando mensajes y teniendo en cuenta lo siguiente:

   a) Puede haber N clientes esperando en la barbería, además de quien está sentado en el sillón de corte.

   b) Hay tres sillones de corte y tres barberos que atienden cada uno de ellos.

   c) Hay una única caja que es atendida por un único cajero.

8. Para el problema de la pizzería enunciado anteriormente, proporcione una solución que utilice pasaje de mensajes.

9. El sistema en estudio es una playa de estacionamiento que tiene una única puerta de entrada y salida, con capacidad para albergar N vehículos en cada una de sus 2 plantas (PB y 1P), lo que da una capacidad total de 2xN. Todos los vehículos intentan ingresar, con la única condición que haya espacio, una vez que consiguen el ingreso (por PB) deben ubicarse en una de las N cocheras si es que éstas no están todas ocupadas, en caso contrario, debe subir al 1er. Piso y ubicarse allí. Se sabe que luego de unos instantes, todos los vehículos, se retiran de la playa, obviamente por la misma puerta de entrada. Ud. Debe simular este sistema utilizando mensajes para solucionar los inconvenientes que se planteen de situaciones de concurso y sincronización. La solución debe contemplar como mínimo: a) que no ingresen mas vehículos que los permitidos, b) que no haya más de N vehículos en cada piso en todo instante, c) que obligatoriamente los vehículos llenen la PB y luego el 1P, d) debido a que la puerta de acceso es reducida, sólo puede pasar un vehículo a la vez, tanto entrando como saliendo, e) cada vehículo tiene sus propios tiempos, es decir la estancia en la playa puede ser distinta en todos casos, el instante en que desee entrar a la playa puede ser también distinto, f) el sistema debe proporcionar en todo momento que se requiera, el total de vehículos estacionados en cada piso.

10. Utilizando semáforos, implemente en JR las definiciones de estructuras de datos y primitivas necesarias para proveer un sistema de pasaje de mensajes con las siguientes características:

a) Con direccionamiento indirecto, utilizando mailboxes.

b) Provea forma de sincronización con send no bloqueante y receive bloqueante.

c) Indique la disciplina de cola que utiliza (no se requiere una en particular).

d) Funciona la solución obtenida en un sistema distribuido?.