# Programación Concurrente

Cuestionario

Ramiro Martínez D'Elía

2021

**1. Defina programa concurrente, programa paralelo y programa distribuido.**

Apunte I: sección 1.1, Apunte II: sección 1.1 y Apunte IV: 1.1

**2. Responda**

***a*) ¿A qué se denomina propiedad del programa? ¿Qué son las propiedades de seguridad y vida? Ejemplificar.**

***b*) Defina fairness. Relacionar dicho concepto con las políticas de scheduling.**

***c*) Describa los distintos tipos de fairness**

***d*) ¿Cuáles son las propiedades que debe cumplir un protocolo de E/S a una sección crítica?**

***e*) Cuáles son los defectos que presenta la sincronización por busy waiting? Diferencie esta situación respecto de los semáforos.**

***f*) Explique la semántica de la instrucción de grano grueso AWAIT y su relación con las instrucciones Test & Set o Fetch & Add.**

a) Apunte I: sección 1.7

b) Apunte I: sección 1.8

c) Apunte I: sección 1.8

d) Apunte II: sección 1.2

e) Apunte II: sección 2.3

f) Apunte I: sección 1.6

**3. Definir el problema general de alocación de recursos y su resolución mediante una política SJN. ¿Minimiza el tiempo promedio de espera? ¿Es fair? Si no lo es, plantee una alternativa que lo sea.**

Apunte II: sección 3.5.2

**4. ¿En qué consiste la técnica de passing the baton? Aplicar este concepto a la resolución del problema de lectores y escritores.**

Apunte II: secciones 3.5 y 3.5.1

**5. Explique el concepto de broadcast y sus dificultades de implementación en un ambiente distribuido, con pasaje de mensajes sincrónico y asincrónico.**

*Sobre primitiva broadcast: Apunte III sección 2.1.4*

En un ambiente con memoria compartida, este tipo de primitivas es más sencilla de implementar.

En ambientes distribuidos, al no haber variables compartidas, se debe utilizar pasaje de mensajes. Esto, supone ciertas dificultades para cada caso.

Con pasaje de mensajes asincrónicos (PMA), podemos plantear los siguientes escenarios de implementación:

Utilizar un único canal

El emisor deberá enviar tantos mensajes, como receptores existan. Este escenario evita demoras innecesarias, en el envío de mensajes, ya que `send` no es bloqueante.

Un canal por proceso

El emisor deberá enviar el mensaje por tantos canales, como receptores existan. Con este escenario, tenemos demoras innecesarias: un proceso listo debe esperar a que el emisor itere sobre su canal.

Con pasaje de mensajes sincrínicos (PMS), los canales son punto a punto. Por este motivo, solo podemos llevar a cabo la implementación "un canal por proceso" planteada en PMA. Aunque, en este caso, resulte aún más ineficiente ya que `send` causa demora.

**6. Responder sobre exclusión mutua selectiva**

***a*) ¿Por qué el problema de los filósofos es de exclusión mutua selectiva? Si en lugar de 5 filósofos fueran 3, ¿el problema seguiría siendo de exclusión mutua selectiva? ¿Por qué?**

***b*) El problema de los filósofos resuelto de forma centralizada y sin posiciones fijas ¿es de exclusión mutua selectiva? ¿Por qué?**

***c*) El problema de los lectores-escritores es de exclusión mutua selectiva? ¿Porque?**

***d*) Si en el problema de los lectores-escritores se acepta sólo 1 escritor o 1 lector en la BD, ¿tenemos un problema de exclusión mutua selectiva? ¿Por qué?**

a) Apunte II: sección 3.4.1

b) Apunte II: sección 3.4.1

c) Apunte II: sección 3.4.2

d) Apunte II: sección 3.4.2

**7. Sea el problema en el cual N procesos poseen inicialmente cada uno un valor V, y el objetivo es que todos conozcan cual es el máximo y cuál es el mínimo de todos los valores.**

***a*) Plantee conceptualmente posibles soluciones con las siguientes arquitecturas de red: Centralizada, simétrica y anillo circular. No implementar.**

***b*) Analice las soluciones anteriores desde el punto de vista del número de mensajes y la performance global del sistema.**

a) Apunte III: sección 2.1

b) Apunte III: sección 2.1.4

**8. Defina el concepto de granularidad. ¿Qué relación existe entre la granularidad de programas y de procesadores?**

Apunte IV: sección 1.6

**9. ¿En qué consiste la comunicación guardada (introducida por CSP) y cuál es su utilidad?**

Apunte III: sección 3.1

**10. ¿Cuál es la utilidad de la técnica de passing the baton? ¿Qué relación encuentra con la técnica de passing the condition?**

Apunte II: sección 3.5

**11. Explique sintéticamente los 7 paradigmas de interacción entre procesos en programación distribuida. Ejemplifique.**

Apunte III: capítulo 5

**12. Responder sobre programación paralela**

***a*) ¿Cuál es el objetivo de la programación paralela?**

***b*) Defina las métricas de speedup y eficiencia. ¿Cuál es el significado de cada una de ellas (que miden)? ¿Cuál es el rango de valores para cada una?**

***c*) ¿En qué consiste la ley de Amdahl?**

a) Apunte IV: sección 1.1

b) Apunte IV: sección 1.2 y sección 1.3

c) Apunte IV: sección 1.5

**13. Suponga que el tiempo de ejecución de un algoritmo secuencial es de 1000 unidades de tiempo, de las cuales el 80 % corresponden a código paralelizable. ¿Cuál es el límite en la mejora que puede obtenerse paralelizando el algoritmo?**

Apunte IV: sección 1.5 (Ley de Amdahl) está resuelto un ejercicio similar.

**14. ¿Qué significa el problema de interferencia en un programa concurrente? ¿Cómo puede evitarse?**

Apunte I: sección 1.4

**15. Suponga que la solución a un problema es paralelizada sobre p procesadores de dos maneras diferentes. En un caso, el speedup (S) está regido por la función S=p-1 y en el otro por la función S=p/2. ¿Cuál de las dos soluciones se comportará más eficientemente al crecer la cantidad de procesadores? Justifique claramente.**

Apunte I: sección 1.4 (Escalabilidad). Hay un ejercicio similar resuelto.

**16. Sea el problema de alocacion Shortest Job Next**

***a*) ¿Funciona correctamente con disciplina de señalización Signal and continue? Justifique.**

***b*) ¿Funciona correctamente con disciplina de señalización signal and wait? Justifique.**

a) Con esta disciplina, todo proceso señalizado es enviado a una cola de listos; donde competirá contra el resto por el acceso al recurso. En cuyo caso, podría perder contra otro proceso y la meta de SJN no se estaría concretando.

b) Con esta disciplina, todo proceso señalizado retoma su ejecución de inmediato. Por tal motivo, esta disciplina es la más adecudada para la implementación.

*Nota$_1$: El enunaciado incluye el mismo algoritmo que el del apunte de monitores.*

*Nota$_2$: Para el LJN (Lowest Job Next) aplican las mismas respuestas.*

**17. En los protocolos de acceso a sección crítica vistos en clase, cada proceso ejecuta el mismo algoritmo. Una manera alternativa de resolver el problema es usando un proceso coordinador. En este caso, cuando cada proceso SC[i] quiere entrar a su sección crítica le avisa al coordinador, y espera a que éste le de permiso. Al terminar de ejecutar su sección crítica, el proceso SC[i] le avisa al coordinador. Desarrolle protocolos para los procesos SC[i] y el coordinador usando sólo variables compartidas (no tenga en cuenta la propiedad de eventual entrada).**

```
1 int request[n] = ([n] 0);
2 int grant[n] = ([n] 0);
```

```
Process Worker[i = 1..n]

# Seccion no critica ...

request[i] = 1;
while(grant[i] == 0) skip;
# Seccion Critica
grant[i] = 0;

# Seccion no critica ...
end;
```

```
Process Coordinator
while (true)
  for (j = 1; j < n; j++)
    if (request[j] == 1)
      request[j] = 0;
      grant[j] = 1;
      while (grant[j] == 1) skip;
    end;
  end;
end;
end;
```

**18. Dado el siguiente bloque de código, indique para cada inciso qué valor queda en aux, o si el código queda bloqueado. Justifique su respuesta.**

```
aux = -1;
...
if (A == 0); P2?(aux) -> aux = aux + 2; # (1)
[] (A == 1); P3?(aux) -> aux = aux + 5; # (2)
[] (B == 0); P3?(aux) -> aux = aux + 7; # (3)
fi
```

***a*) Si el valor de A = 1 y B = 2 antes del if, y solo P2 envia el valor 6.**

***b*) Si el valor de A = 0 y B = 2 antes del if, y solo P2 envia el valor 8.**

***c*) Si el valor de A = 2 y B = 0 antes del if, y solo P3 envia el valor 6.**

***d*) Si el valor de A = 2 y B = 1 antes del if, y solo P3 envia el valor 9.**

***e*) Si el valor de A = 1 y B = 0 antes del if, y solo P3 envia el valor 14.**

***f*) Si el valor de A = 0 y B = 0 antes del if, P3 envia el valor 9 y P2 el valor 5.**

a) La guarda 1 queda bloqueada; su condición booleana es verdadera pero, su sentencia de comunicación generá bloqueo.

b) Solo la guarda 1 tiene éxito. Su sentencia guardada es ejecutada. El programa finaliza con `aux = 10`.

c) Solo la guarda 3 tiene éxito. Su sentencia guardada es ejecutada. El programa finaliza con `aux = 13`.

d) Todas las guardas fallan porque sus condiciones booleanas se evalúan como falsas. El programa finaliza con `aux = -1`.

e) Las guardas 2 y 3, tienen éxito. Se elegirá una de las dos, de forma no deterministica y se ejecutará su sentencia guardada. El programa finalizará con `aux = 19 o 21`.

f) Las guardas 1 y 3, tienen éxito. Se elegirá una de las dos, de forma no deterministica y se ejecutará su sentencia guardada. El programa finalizará con `aux = 16 o 7`.

**19. Dado el siguiente programa, indique si es posible que finalice.**

```
bool continue = true;
bool try = false;
co  while (continue) { try = true; try = false; } #(P)
/   <await(try) continue = false> #(Q)
oc
```

Con una política débilmente fair, el programa podría no terminar. Ya que, `try` no se mantiene verdadera hasta ser vista por (Q).

Con una política fuertemente fair, el programa podría terminar. Ya que, `try` se convierte en verdadera con infinita frecuencia. Ante este escenario, eventualmente, (Q) verá a `try` como vedadera.

**20. Suponga que $n^2$ procesos organizados en forma de grilla cuadrada. Cada proceso puede comunicarse solo con los vecinos izquierdo, derecho, de arriba y de abajo (los procesos de las esquinas tienen solo 2 vecinos, y los otros en los bordes de la grilla tienen 3 vecinos). Cada proceso tiene inicialmente un valor local v.**

***a*) Escriba un algoritmo heartbeat que calcule el máximo y el mínimo de los $n^2$ valores. Al terminar el programa, cada proceso debe conocer ambos valores.**

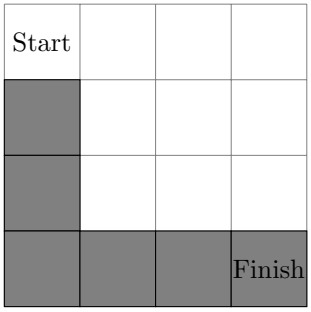
***b*) Analice la solución desde el punto de vista del número de mensajes.**

***c*) ¿Puede realizar alguna mejora para reducir el número de mensajes?**

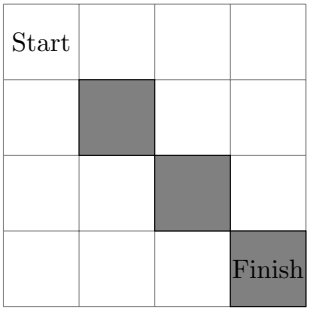
***d*) Indique que mecanismo de pasaje de mensajes utilizaría. Justifique.**

a) La idea del algoritmo es que, cada proceso pueda propagar su valor hasta el proceso más lejano. La mayor distancia entre procesos, sin diagonales, está dada por $2(n-1)$.

Sin diagonal y n=4




Con diagonal y n=4




```
1  chan valores[1:n,1:n](int);
2
3  Process P[i=1..n;j=1..n]
4
5  int v, nuevo;
6  int min = v;
7  int max = v;
8  int rondas = 2(n-1);
9
10 for(k=1; k <= rondas; k++)
11
12   foreach(vecinos => (x,y))
13     send valores[x,y](v);
14
15   foreach(vecinos => (x,y))
16     receive valores[i,j](nuevo);
17     if (nuevo < min) min = nuevo;
18     if (nuevo > max) max = nuevo;
19   end;
20 end;
21 end;
```

b) Desde el punto de vista de los mensajes, podemos obtener las siguientes conclusiones.

| Tipo | Cantidad | Vecinos | Mensajes |
|---|---|---|---|
| **Esquina** | 4 | 2 | $cantidad * vecinos * rondas$ |
| **Borde** | $4(n-2)$ | 3 | $cantidad * vecinos * rondas$ |
| **Internos** | $(n-2)^2$ | 4 | $cantidad * vecinos * rondas$ |

c) Podemos hacer que cada proceso, también, se comunique con sus diagonales. De esta forma; los procesos esquina tendrán 3 vecinos, los bordes 5 vecinos y el resto 8 vecinos.

Esto incrementará el número de mensajes enviados por cada proceso pero, también reducirá en la mitad la cantidad de rondas $(n-1)$. Generando una reducción en la cantidad final de mensajes.

| Tipo | Cantidad | Vecinos |
|---|---|---|
| **Esquina** | 4 | 3 |
| **Borde** | $4(n-2)$ | 5 |
| **Internos** | $(n-2)^2$ | 8 |

| n | Sin Diagonales | Con Diagonales |
|---|---|---|
| 4 | 288 | 252 |
| 6 | 1200 | 1100 |
| 8 | 3136 | 2940 |

d) En mi opinión, utilizaría PMA. Particularmente, porque su primitiva `send` no es bloqueante. Esto nos evita demoras innecesarias al momento de propagar el valor de un proceso, en particular con $n$ muy grandes.

**21. Suponga los siguientes programas concurrentes. Asuma que EOS es un valor especial que indica el fin de la secuencia de mensajes y que los procesos son iniciados desde el programa principal.**

P1

```
chan canal (double)
process Genera {
  int fila, col;  double sum;
  for [fila= 1 to 10000]
    for [col = 1 to 10000]
      send canal (a(fila,col));
  send canal (EOS)  }
```

```
process Acumula {
  double valor, sumT;
  sumT=0;
  receive canal (valor);
  while valor<>EOS {
    sumT = sumT + valor
    receive canal (valor); }
  printf (sumT);
}
```

P2

```
chan canal (double)
process Genera {
  int fila, col;  double sum;
  for [fila= 1 to 10000] {
    sum=0;
    for [col = 1 to 10000]
      sum=sum+a(fila,col);
    send canal (sum);  }
  send canal (EOS)  }
```

```
process Acumula {
  double valor, sumT;
  sumT=0;
  receive canal (valor);
  while valor<>EOS {
    sumT = sumT + valor
    receive canal (valor); }
  printf (sumT);
}
```

***a*) ¿Qué hacen los programas?**

***b*) Analice desde el punto de vista del número de mensajes.**

***c*) Analice desde el punto de vista de la granularidad de procesos.**

***d*) ¿Cuál de los programas le parece el más adecuado para ejecutar sobre una arquitectura tipo cluster PCs? Justifique.**

a) Ambos programas realizan la suma de los elementos de una matriz. Aunque, difieren en el modo en que lo hacen.

En el `Programa 1`: el proceso `Genera` envía todos los valores al proceso `Acumula`. El proceso `Acumula` suma los valores que recibe desde `Genera`.

En el `Programa 2`: el proceso `Genera` realiza la suma de los elementos de una fila y envia el resultado a `Acumula`. El proceso `Acumula`, recibe los resultados parciales y los adiciona en su acumulador.

b) En el `Programa 1`, el proceso `Genera` envía todos los elementos de la matriz. Por lo cual, en esta solución, se están enviando $n * n = n^2$ mensajes.

En el `Programa 2`, el proceso `Genera` envía los valores acumulados de cada fila. Por lo cual, en esta solución, se están enviando $n$ mensajes.

c) El programa `P2` tiene mayor granularidad, con respecto a `P1`. Esto se debe a que, requiere menor comunicación (menos mensajes) para completar la tarea.

d) Las arquitecturas cluster PC pueden ser catalogadas como grano grueso. Esto es, porque se componen de mucho procesadores; otorgando mayor potencia para cómputo. No obstante, tienen una menor performance en cuanto a comunicación.

Es deseable que la granularidad de la aplicación, se ajuste a la de la arquitectura. Para hacer un uso apropiado de los recursos y evitar rendimientos menores.

En este escenario, el programa, `P1` parece el más adecuado para ejecutar en la arquitectura planteada.