## Ejercicio 3

a) Dar planificacion que muestre que no se cumple el invariante.
b) Este caso se da siempre?
c) Reparar la sincronizacion con semaforos maximizando concurrencia.

pre: $barco = raton = 0; s1 = 1; 0 < N$

```
while (true) {
    barco=barco+1;
    wait(s1);
}
```

```
while (true) {
    raton=raton+1;
    post(s1);
}
```

post: $|barco - raton| \leq N$

| P0 | P1 |
|---|---|
| a : while (true) {<br>b : barco = barco + 1;<br>c : wait(s1);<br>d : } | A : while (true) {<br>B : raton = raton + 1;<br>C : post(s1);<br>D : } |

a)

| Planificación | Invariante |
|---|---|
| ABCD ABCD ABCD ABCD abcd | \| barco - raton \| = \| 1 - 4 \| = 3 > N = 2 |
| 5 ABCD abcd | 1 - 5 = 4 N = {1,2,3} |
| 6 ABCD abcd | |

b)
No siempre se da este caso.

c)

pre: barco = ratón = 1; 0 < N
postcondición: |barco - ratón | ≤ N

s1 = 0, s0 = 0

| P0 | P1 |
|---|---|
| a : while (true) {<br>b : barco = barco + 1;<br>c : post(s0)<br>d : wait(s1)<br>e : } | while (true) {<br>if( ! inv )<br>wait(s0)<br>raton = raton + 1; |

```
while (true) {
        if( inv )
                post(s0)
        barco = barco + 1;
        wait(s1)
```

```
while (true) {
        while ( ! inv )
                wait(s0)
        raton = raton + 1;
        post(s1)
```

Seguis con la misma lógica del programa base, barco puede aumentar tantas veces como aumenta ratón, pero si en ratón el inv ya no vale, se duerme en s0. Hasta que el barco aumenta tantas veces como sea necesario para despertar al ratón.

## Ejercicio 4

Multiprograma que nunca termina con atomicidad linea a linea, variables i y a[] compartidas y a[0, 16) (**Comentario:** Osea, indice máximo 15)

```
P0: while (1) {      P1: while (1) {      P2: while (1) {
    a0=i;                a1=i;
    a0=a0+1;             a1=a1-1;             a[i] = 1;
    i=a0;                i=a1;
}                    }                    }
```

a) Sincronizar con semaforos. El objetivo principal es no salirse de los limites del arreglo. Sin embargo, mientras mas posiciones del arreglo se toquen, más puntaje se tendra en el ejercicio. Pero lo más importante es que no se vaya de los limites.

¿Cómo podría salirme de los límites?
Se me ocurre que podría suceder es evitar un decremento o un aumento, eso podría llevarnos a obtener cualquier valor posible de i. Por ejemplo:
pre: barco = ratón = 1; 0 < N

```
a0 = i;              // a0 = 0
a0 = a0 + 1;         // a0 = 1

a1 = i;              // a1 = 0
a1 = a1 - 1          // a1 = -1

i = a0;              // i = 1
i = a1;              // i = -1
```

Ya nos salimos del arreglo.

De forma contraria podemos evitar un decremento:

```
a0 = i;              // a0 = 0
a0 = a0 + 1;         // a0 = 1

a1 = i;              // a1 = 0
a1 = a1 - 1          // a1 = -1
i = a1;              // i = -1

i = a0;              // i = 1
```

Con esta planificación podemos obtener cualquier valor positivo de i e salirnos del límite.

i = 0, s1 = 15, s2 = 0

| | | |
|---|---|---|
| po : while(1) {<br>: a0 = i;<br>: a0 = a0 + 1;<br>: i = a0<br>wait(s1)<br>post(s2) | p1 : while(1) {<br>wait(s2)<br>: a1 = i;<br>: a1 = a1 - 1;<br>: i = a1<br>post(s1) | p2 : while(1) {<br>: a[i] = 1;<br>: } |

## Ejercicio 5

HDD 7200 RPM, 8.5 latencia de busqueda, 220 MiB/s tasa de transferencia máxima.

a) Calcular tasa de transferencia al azar para bloques de 1MiB
b) Si se duplica la velocidad de rotacion, es decir, se pasa a 14400 RPM, la transferencia máxima se duplica y el tiempo de rotacion baja a la mitad. Si para duplicar la velocidad de rotacion aumenta un 50% el precio, vale la pena la ganancia en velocidad para bloques al azar de 1MiB?

7200 RPM
8.5 seek
220 MiB/s tasa de transferencia máxima

a)

**T_seek = 8.5 ms**

---

60s — 7200 RPM
1s — 120 RPS

120 — 1s
1 — 0.00833s

Es decir, tardamos 0,008s en hacer una vuelta

1s → 1000ms
0,00833s → 8.33ms

Es decir, tardamos 8.33 ms en dar una vuelta.
Luego, latencia promedio de rotación:
**T_rotation = 8.33 / 2 = 4.165ms (promedio)**

---

tasa de transferencia máxima = 220 MiB/s

220 MiB → 1s
1 Mib → 0.004545s

1s → 1000ms

0,004545s → 4.545ms

Es decir, tarda 4.545 ms en transferir 1 MiB
**T_transfer = 4.545 ms**

Finalmente:

T(I/O) = T_seek + T_rotation + T_transfer
= 8.5 + 4.165ms + 4.545ms
= 17.21ms

Luego la tasa de transferencia es

R(I/O) = 1MiB / 17.21ms = 0.058ms = 58.1MiB/s

1048576 bytes / 17.21ms = 60928.2975 b/ms

60928297.5 b/s → 1000ms
60928.2975 b/ms → 1ms

1048576 b → 1 MiB
60928297.5 b → 50.1 MiB/s

**R(I/O) = 50.1 MiB/s**

b) Si se duplica la velocidad de rotacion, es decir, se pasa a 14400 R
encia máxima se duplica y el tiempo de rotacion baja a la mitad.
la velocidad de rotacion aumenta un 50% el precio, vale la pena
velocidad para bloques al azar de 1MiB?

b)

14400 RPM
440 MiB/s tasa de transferencia máxima
8.5 seek

**T_rotation = 4.165 / 2 = 2.0825 (promedio)**

**T_transfer = 1MiB / 440MiB/s = 2.27 ms**

**T_seek = 8.5 ms**

T(I/O) = T_rotation + T_transfer + T_seek
= 2.0825ms + 2.27ms + 8.5ms = **12.8525ms**

R(I/O) = (1MiB / 12.8525ms) * 1000 = 77.8

**58.1 MiB/s → %100**
**77.8 MiB/s → %133**

**Una mejora del %33 aproximadamente**

## Ejercicio 6

File system unix con 12 bloques directos, 1 indirecto, 1 doble indirecto, 1 triple indirecto. Con bloques de 4KiB y tabla de inodos de $2^{20}$.

a) Calcular tamaño de d-bmap, data-region y maxfiles para indices de bloque de 32b y 48b:

| | 32b | 48b |
|---|---|---|
| d-bmap | | |
| data-region | | |
| maxfiles* | | |

(*) Maxfiles es la cantidad máxima de disco que se puede usar con archivos de tamaño máximo.

| | 32b | 48b |
|---|---|---|
| d-bmap | 512 MiB | 32 TiB |
| data-región | 16 TiB | 1HiB |
| maxfiles* | 4 HiB | 1.18 HiB |

32b:

- Si son 2^32 bloques, necesitamos 2^32 bits para el d-bmap esto 2^32 / 8 = 536870912 bytes

  1048576 bytes → 1MiB

536870912 bytes → 512 MiB

- La data región son 2^32 * 4KiB = 2^32 * 2^12 = 2^44 = 16TiB

  4 KiB / 32 bits = 4 KiB / 4 bytes = 2^12 / 2^2 = 1024 bloques

- maxfiles = 4 KiB (12 + 1024 + 1024^2 + 1024^3)
- maxfiles* = 2^20 * maxfiles = 2^20 * 2*12 * (2*10)^3
  = 2^62 = 4 HiB

48b:

- Si son 2^48 bloques, necesitamos 2^48 bits para el d-bmap esto 2^48 / 8 bytes = 2^45 bytes

  2^20 bytes → 1MiB
  2^45 bytes → 2^25 MiB ≡ 2^25 * 2^20 = 2^45 = 32 TiB

- La data región son 2^48 * 4KiB = 2^48 * 2^12 = 2^60 = 1HiB

  4 KiB / 48 bits = 4 KiB / 6 bytes = 2^12 bytes / 6 bytes = 682 bloques

- maxfiles = 4 KiB (12 + 682+ 682^2 + 682^3)
- maxfiles* = 2^20 * maxfiles = 2^20 * 2^12 * 682^3 = 2^32 * 317214568
  = 1268858271 GiB
  = 1.18 HiB

  2^30 GB → 1HiB
  1268858271 GiB → 1.18 HiB

R(I/O) = 1MiB / 17.21ms = 1048576 bytes / 17.21ms = 60928.2975 b/ms

0.058 MiB/ms

60928.2975 b

1000ms →

60928297.5 b/s

1048576 b → 1 MiB
60928297.5 b → 50.1 MiB/s

R(I/O) = 50.1 MiB/s
**