

write(file descriptor, thing to write, int count)

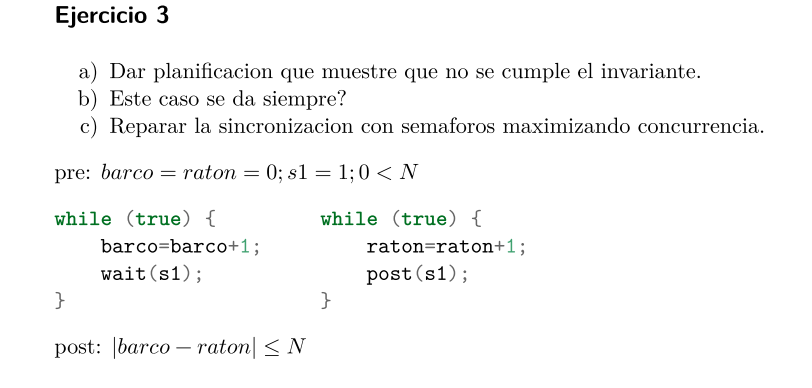
close() cierra el fd duplicado, no el original

#0 se escribe 1 a

#1 se escriben 0 a xq se cierra el fd antes del dup

#2 0 a xq solo se abre y se cierra

#3 2 a



c) deberia haber un mejor sistema de semaforos:

t\_sem s1 = N

t\_sem s2 = 0

while(true){

wait(s1);

barco++;

post(s2);

}

while (true){

wait(s2);

raton++;

post(s1);

}

uso N en s1 xq maximiza concurrencia

El problema de dejar avanzar a barco N veces sin raton

Si barco puede avanzar N veces antes que raton avance siquiera 1 vez, la diferencia barco - raton podría llegar a ser exactamente N.

Eso está permitido por la invariante |barco - raton| ≤ N.

Pero, ¿qué pasa si barco sigue avanzando más veces sin que raton avance?

El semáforo sem\_barco empieza con valor N, que se decrementa en cada wait(sem\_barco).

Cuando sem\_barco llega a 0, barco ya no puede avanzar más hasta que raton haga post(sem\_barco) liberando espacio.

Por eso, la diferencia no puede superar N porque barco se bloquea cuando llega a ese límite.

Y qué pasa con raton?

Ratón sólo puede avanzar cuando hay pasos adelantados de barco (wait(sem\_raton)).

En cada paso de ratón, libera espacio para barco con post(sem\_barco).

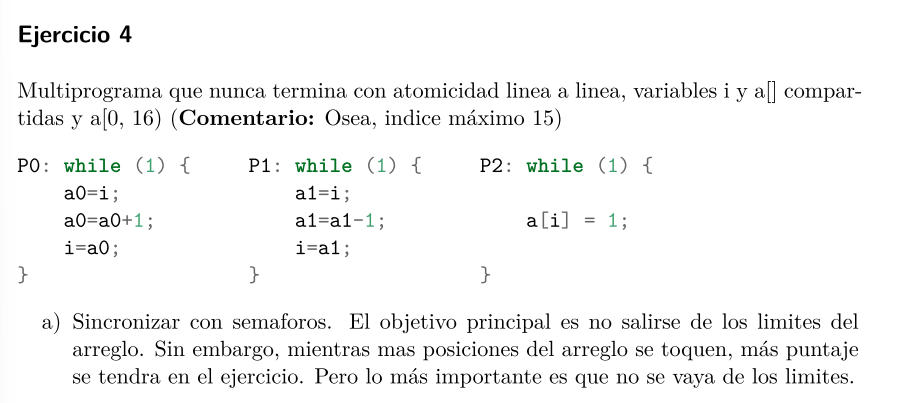
Por eso, el sistema se autorregula.

En conclusión

No, no terminaría siendo mayor que N nunca si ratón no avanza mucho, porque el semáforo bloquea a barco justo en ese punto.

Si raton no avanza, barco se queda bloqueado en wait(sem\_barco) cuando alcanza la diferencia N.

Esto impide que la diferencia sea mayor que N.



| pre: i = 0, pos i<16  i=0;  t\_sem s0 = 1;  t\_sem s1 = 0;  t\_sem s2 = 0 ; | | |
| --- | --- | --- |
| p0: (i++)  while(1){  wait(s0)  assert(i<16)  a0 = i;  a0 ++;  i=a0;  if(i<=15){  post(s2)  }else{  post(s1)  }  } | p1:(i - -)  while(1){  wait(s1);  if (i>0){  i–  post(s2)  }else {  post(s0)  }  } | p2:  while(1){  wait(s2)  a[i]=1;  post(s1)  } |