

0x00000BAD = 0000 0000 00|00 0000 0000 |1011 1010 1101

PD = 0x000

PT = 0x000

offset = 0xBAD

dir fisica = marco \* 4kib + offset = 0x60D000 + 0xBAD =

| 0X0C0CA  —-----------------------------------  dir : marco fisico, XWRV  0x3FF: 0x0CO1A, XWRV  .  .  .  0x003: 0x0CO1A, XWRV  0x002: 0x0CO1A, XWRV  0x001: 0x0CO1A, XWRV  0x000: 0x0CO1A, XWRV | 0x0C01A  —-----------------------------------  dir : marco fisico, XWRV  0x3FF: 0X0CAFE  0x300: 0x0CAFE, \_ \_ \_ -  .  0x003: 0x\_ \_ \_ \_ \_, \_ \_ \_ \_  0x002: 0x\_ \_ \_ \_ \_, \_ \_ \_ \_  0x001: 0x\_ \_ \_ \_ \_, \_ \_ \_ \_  0x000: 0x 0CAFE , \_ \_ \_ V | 0x0CAFE  —-----------------------------------  dir : marco fisico, XWRV  0x3FF:0xCOCA. \_ \_ R V,  .  .  .  0x003: 0x\_ \_ \_ \_ \_, \_ \_ \_ \_  0x002: 0x\_ \_ \_ \_ \_, \_ \_ \_ \_  0x001: 0x\_ \_ \_ \_ \_, \_ \_ \_ \_  0x000: 0x0060D, xrv |
| --- | --- | --- |

dir fisica = marco \* 4kib + offset = 0x60D000+0xBAD = 0x60DBAD

dir fisica = marco \* 4kib + offset = C0000FEE

0Xffc00000 -> 0x0cafe000 wn ctm

1111 1111 11 | 00 0000 0000 | 0000 0000 0000

1111 1111 11 | 11 1111 1111 . 0000 0000 0000

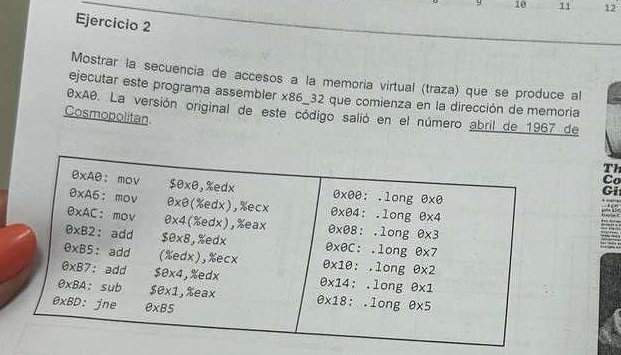
3FF | 3FF | 000 ya esta ocupado esto flaquito wn ctm

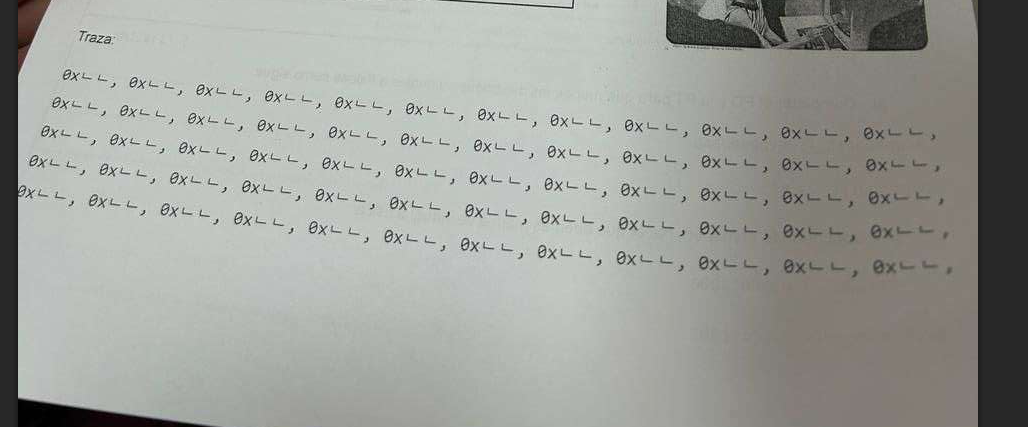
1100 0000 00 | 00 0000 0000 | 1111 1110 1110

300 000 FEE

Very good

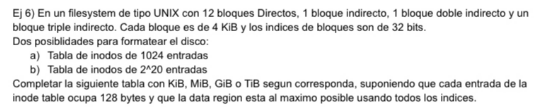
0X0 - 0X0 - 0X : 0x777





A0 A 6 0 0 A C 0 4 B 2 B 5 0 8 B 7 B A B D B5 0 C B 7 B A B D B5 1 0 B 7 B A B D 1 4

B 7 B A B D



| parte del formato de disco/opcion | 2^10 inodos | 2^20 inodos |
| --- | --- | --- |
| i-bmap | 2^10 = 1024/8= 128bytes | 2^20/2^3=2^17=128kib |
| d-bmap | 1024^3 = 128mb | 128mb |
| inode table | 2^10\*128=128kb | 2^20\*128=128mb y man |
| data region | 4tb | 4tb |

12 directos

1 indirecto

1 doble indirecto

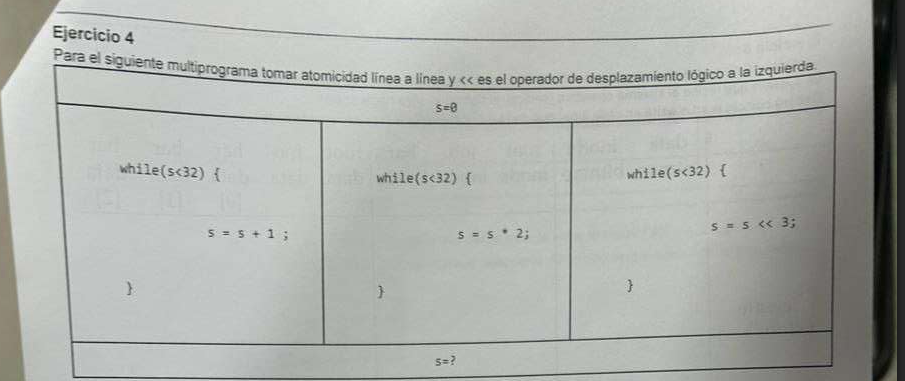
1 triple indirecto

cada bloque 4kib

punteros de bloque 32 bits = 4bytes

entran = 4kib/4bytes= 4096/4 = 1024 bytes por bloque

(12+1024+1024^2+1024^3)



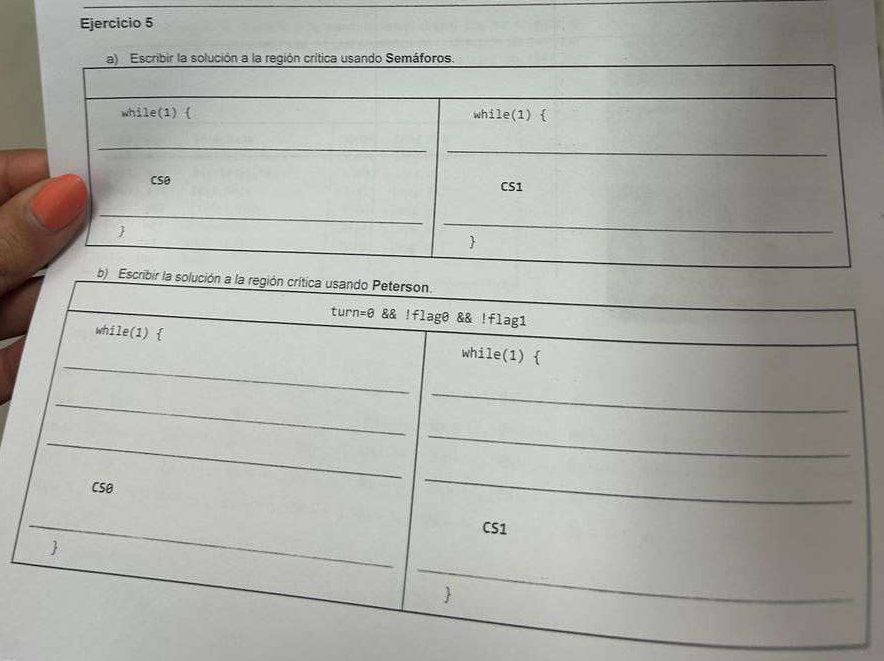
sem1 = 0

sem\_wait(sem1) sem\_wait(sem1)

sem\_post(sem1)

sem\_post(sem1)

s=32



sem1 = 0, sem2 = 1

sem\_wait(sem2) sem\_wait(sem1)

sem\_post(sem1) sem\_post(sem2)

flag0 = true

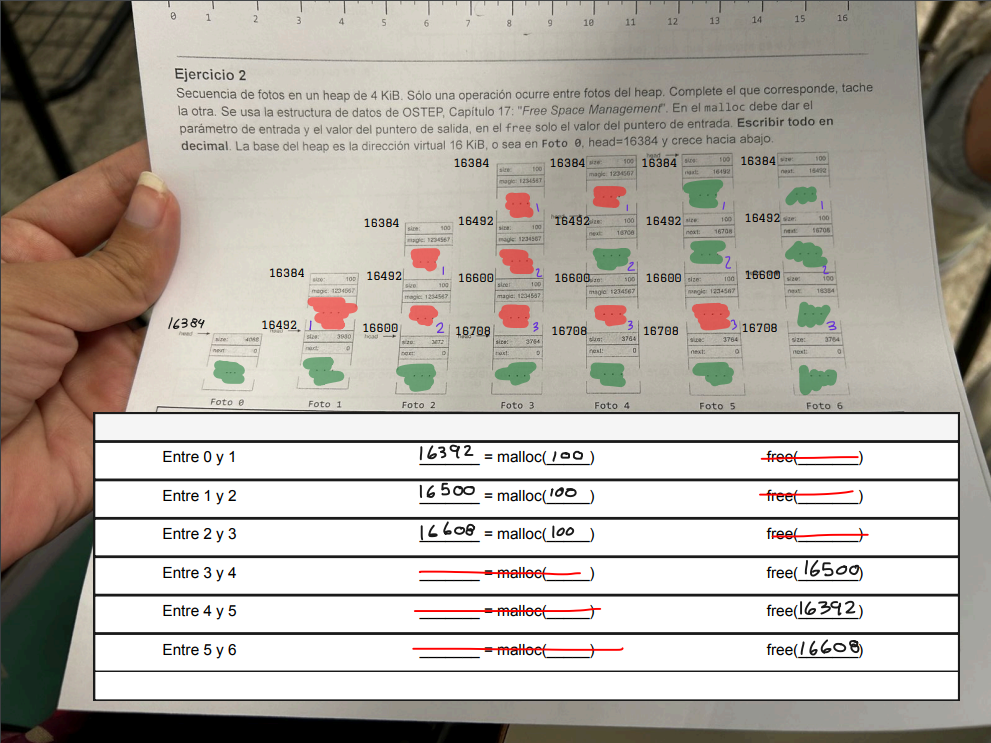
turn = 1 flag1 = true

if(flag1 ^ turn = 1){waits} turn = 0

if(flag0 ^ turn = 0){waits}

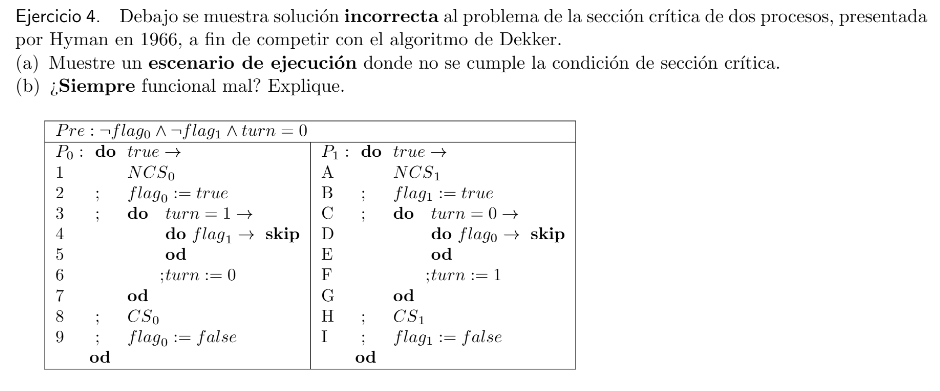
flag0 = false

flag1 = false



Por favor Fran AYudame con este ejercicio :((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((((

Fran no ayuda :’(

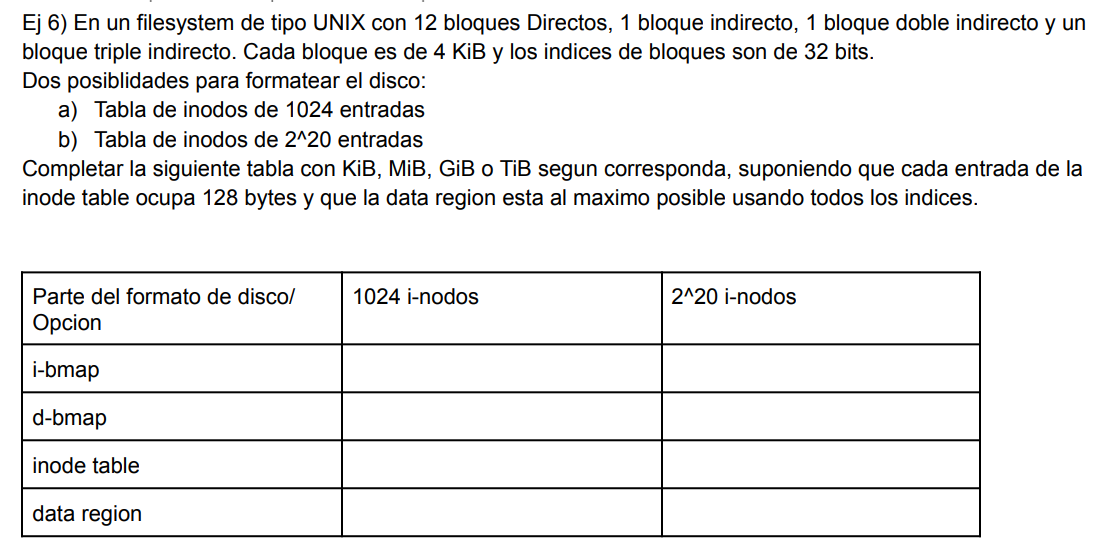


=> Estado inicial: flag0 = false, flag1 = false y turn = 0

1. P0 ejecuta lineal 1 (NCS0) y linea 2: flag0 = true
2. P1 ejecuta linea A (NCS1) y linea B: flag1 = true
3. 3 => es falso (turn == 1), salta a 8 (CS0)
4. C => es true (turn == 0), entra al bucle
5. D => es true, hace skip
6. se repite el bucle C, D mientras P0 esta en CS0
7. 9 => flag0 = false
8. D => es falso, no entra al bucle
9. F => turn = 1
10. G => salimos del bucle
11. H => estamos en CS1
12. P0 se vuelve a ejecutar del principio
13. 2 => flag0 = true
14. 3 => entramos al bucle
15. 4 => es true, hace skip
16. P0 Sigue en bucle mientras P1 esta en CS1

Ambos procesos P0 y P1 entraron a sus secciones criticas (CS0 y CS1) al mismo tiempo, violando la exclusion mutua.





inode table =1024 \* 128 | 2 a la 20 inodos.

