Ejercicio 1

Cuantas “a” se escriben en test.txt si se descomenta cada linea comentada (1 caso a la vez)

```
// close(dup(open("test.txt", ...)))   #0
// dup(close(open("test.txt", ...)))   #1
// close(open("test.txt", ...))        #2
// dup(open("test.txt", ...))          #3
write(3, "a", 1)
write(4, "a", 1)
```

linea	Cantidad de “a”	explicacion
0		
1		
2		
3		

línea	Cantidad de “a”	explicación
0	1	Se abre un fd 3, se duplica en un fd 4, se cierra el fd 4, luego solo el primer write es exitoso.
1	0	No se puede hacer dup de un fd cerrado.
2	0	Se cierra el fd que se abrió.
3	2	Se abre fd 3, se duplica en fd4, ambos writes suceden con exito.

Ejercicio 2

Tipico ejercicio de paginacion (10, 10, 12), con un page dir y dos page tables. (Las 2 page tables que hay son iguales)

a) Pasar direcciones de virtual a física
b) Modificar tabla para que baje el consumo de memoria sin cambiar a donde apuntan las direcciones virtuales.

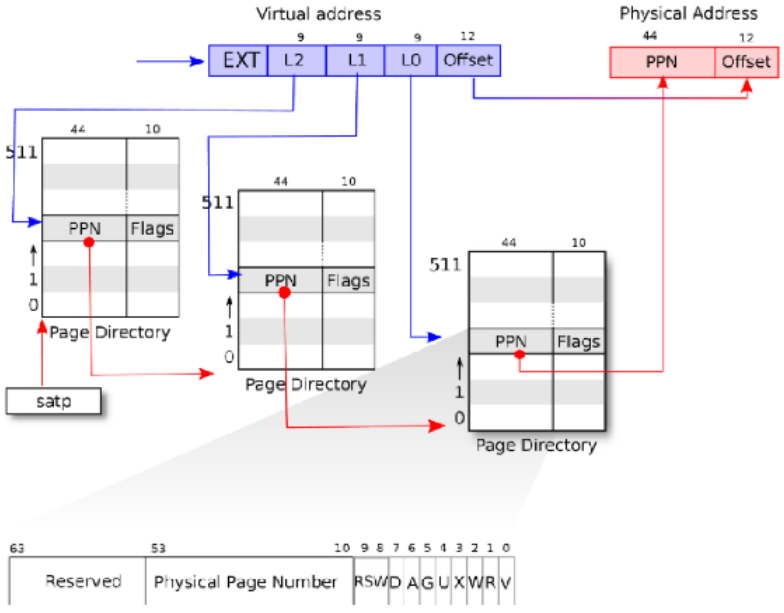
NO ESTÁ

Ejercicio del final 2023

2) Tenemos un esquema RISC-V, de 3 Niveles osea (9,9,9,12) -> (44,12) con paginas de 4KiB (no copie las tablas xd)
a) Traducir de virtual a fisica direcciones de memoria(No las copie xd)
b) Traducir de fisica a todas las virtuales (Tampoco las copie xd)

{correccion, encontre los ejercicios, son iguales a los del parcial 1 de 2022}

Tenemos un esquema de paginación RISC-V con páginas de 4 KiB de 3 niveles con formato 9,9,9,12 -> 44,12 como muestra la figura.



Bits de control

V: *válido*

R: se puede leer, *readable*

W: se puede escribir, *writable*

X: se puede ejecutar, *executable*

Supongamos que tenemos el registro de paginación apuntando al marco físico `satp=0x00000000FE0`.

0x00000000FE0	0x00000000FEA	0x000000AD0BE
`0x1FF: 0x00000000000, ----`	`0x1FF: 0x00000000000, ----`	`0x1FF: 0x00000000000, ----`
⋮	⋮	⋮
`0x004: 0x00000000000, ----`	`0x004: 0x00000000000, ----`	`0x004: 0x00000000000, ----`
`0x003: 0x00000000000, ----`	`0x003: 0x00000000000, ----`	`0x003: 0x00000D1AB10, XWR-`
`0x002: 0x00000000FEA, XWRV`	`0x002: 0x000000AD0BE, XWRV`	`0x002: 0x00000DECADA, -WRV`
`0x001: 0x00000000FEA, XWRV`	`0x001: 0x000000AD0BE, XWRV`	`0x001: 0x000CAFECAFE, ----`
`0x000: 0x00000000FEA, XWRV`	`0x000: 0x000000AD0BE, XWRV`	`0x000: 0x0000000ABAD, X--V`

a) Traducir de **virtual a física** las direcciones:

`0x0000`

`0x1000`

`0x2000`

`0x3000`

b) Traducir la direccion fisica 0xDECADA980 a TODAS LAS VIRTUALES que la apuntan.

(a)

0x0000 = 0b 000 0000 0000 0000 0000 0000 0000 0000 0000 0000

Los primeros 9 bits nos llevan a 0x00000000FEA

Los segundos 9 bits nos llevan a 0x000000ADOBE

Finalmente los últimos 9 bits nos dan 0x0000000ABAD
Concatenando obtenemos la dirección física:
0x0000000ABAD000

0x1000 = 0b 000 0000 0000 0000 0000 0000 0001 0000 0000 0000
Los primeros 9 bits nos llevan a 0x00000000FEA
Los segundos 9 bits nos llevan a 0x000000ADOBE
Finalmente los últimos 9 bits nos dan 0x000CAFECAFE
La cual no es una dirección válida.

0x2000 = 0b 000 0000 0000 0000 0000 0000 0010 0000 0000 0000
Los primeros 9 bits nos llevan a 0x00000000FEA
Los segundos 9 bits nos llevan a 0x000000ADOBE
Finalmente los últimos 9 bits nos dan 0x00000DECADA
Concatenando obtenemos la dirección física:
0x00000DECADA000

0x3000 = 0b 000 0000 0000 0000 0000 0000 0011 0000 0000 0000
Los primeros 9 bits nos llevan a 0x00000000FEA
Los segundos 9 bits nos llevan a 0x000000ADOBE
Finalmente los últimos 9 bits nos dan 0x00000D1AB1O
La cual no es una dirección válida.

Virtual	Fisica
0x0000	0x0000000ABAD000
0x0001	No válida.
0x0002	0x00000DECADA000
0x0003	No válida.

(b)

0xDECADA980 = 0x00000DECADA980
offset = 980
0x00000DECADA
El índice de esta dirección en el último page directory es 0x002, es decir, estos siempre serán los últimos 9 bits.
¿Quién apunta a la tabla 0x00000ADOBE?
Las entradas 0x000, 0x001 y 0x002 en la tabla 0x00000000FEA. Es, decir, tenemos 3 posibles valores para los ante últimos 9 bits.

¿Quién apunta a la tabla 0x00000000FEA?
Las entradas 0x000, 0x001 y 0x002 en la tabla 0x00000000FE0. Es, decir, tenemos 3 posibles valores para los primeros 9 bits.

Finalmente las direcciones virtuales son:

0x2980
0x202980
0x402980
0x40002980
0x40202980
0x40402980
0x80002980
0x80202980
0x80402980

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: $|barco - raton| \leq N$

(a) Hacer varias veces raton = raton + 1
N = 3
hacemos 5 veces raton = raton + 1 → ratón = 1
Luego el inv = | 0 - 5 | = 5 > 3 NO se cumple.
Lo mismo para barco.

(b) | barco - raton | ≤ N sii -N ≤ barco - raton && barco - ratón ≤ N

ratón - N ≤ barco ≤ N + ratón la diferencia es de N
barco - N ≤ ratón ≤ N + barco la diferencia es de N

Entonces, mientras la diferencia de aumentos entre ambas no sea mayor a N, estamos bien.

(c)

```
init_sem(s1,0)
init_sem(s1,0)


while (true) {
        barco = barco + 1
        wait(s1);
        if( | barco - ratón | ≤ N )
                post(s2)
}

while (true) {
        while( | barco - ratón | > N )
                wait(s2)
        raton = raton + 1
        post(s1)
}
```

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.

while (1) {
a0 = i;
a0 = a0 + 1;
	while (1) {
	a1 = i;
	a1 = a1 - 1;
		while (1) {
		a[i] = 1;
		}

i = a0; }	i = a1; }	

Solución:

```
sem_init(s0,0)
sem_init(mutex,1)

while (1) {
        wait(mutex);
        a0 = i;
        a0 = a0 + 1;
        i = a0;
        post(mutex);
        if(0 ≤  i && i ≤  15)
                post(s0)
}

while (1) {
        wait(mutex);
        a1 = i;
        a1 = a1 - 1;
        i = a1;
        post(mutex)
        if(0 ≤  i && i ≤  15)
                post(s0)
}

while (1) {
        while(i < 0 || i > 15){
                wait(s0);
        }
        wait(mutex);
        a[i] = 1;
        post(mutex);
}
```

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?

(a)

T_seek = 8.5ms

7200 RPM / 60 = 120 RPS → 1 / 120 RPS = 0.0083s → 0.0083 * 1000 = 8.3ms
el tiempo de T_rotation = 8.3ms / 2 = 4.15ms

T_trasnfer = 1MiB / 220 MiB/s = 0.004545s → 4.54ms

T_i/o = 8.5ms + 4.15ms + 4.54ms = 17.19ms

R_i/o = 1MiB / (17.19ms / 1000) = 1MiB / 0.01729s = 58.1 MiB/s

(b)

440 MiB/s
T_rotation = 4.15ms / 2 = 2.075ms
T_transfer = 1MiB / 440 MiB/s = 0.0022s → 2.27ms
T_seek = 8.5ms

T_i/o = 12.846ms

R_i/o = 1MiB / (12.846ms / 1000) = 77.8 MiB/s

58.1 → %100
77.8 →%133.9

No vale la pena pagar un %50 más.

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
d-bmap = 2^32 / 8 = 2^29 = 512 MiB
—
data_region = 2^32 = 4 GiB
—
32 bits = 4 bytes y en un bloque de 4 KiB / 4 = 1024 bloques
max_file = 4 KiB (12 + 1024 + 1024^2 + 1024^3) ~ 2^42
maxfiles* = 2^20 * 2^42 = 4 HiB

El mismo proceso para 48b