Herramienta de sincronización

SEMAFOROS

sem_init (sem_t *sem, int pshared, unsigned int value)

- sem_t *sem → Puntero al semaforo para saber de cual hablamos
- int pshared → Si se comparte entre hilos del mismo proces → 0 sino se comparte entre procesos
- unsigned int value → recursos disponibles es decir cuantos pueden acceder concurrentemente

sem_wait (sem_t *sem)

- puntero al semaforo del que coger recursos

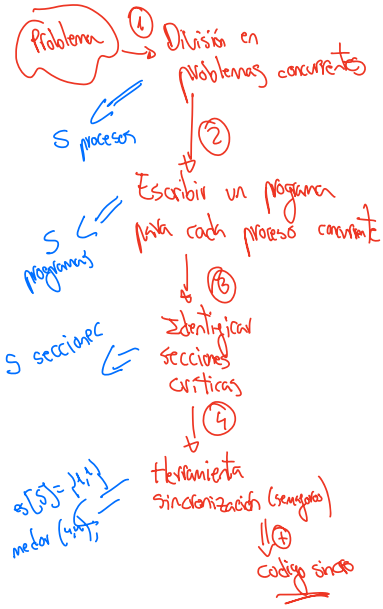
- Lo que hace es decrementar en 1 el recurso del semaforo que se le indica, PERO primero comprueba si ese valor es mayor que cero (hay recursos) y si los hay lo decrementa y continua el hilo, sino lo bloque hasta que haya recursos.

- sem_post(sem_t *sem) ⇒ Puntero al semaforo del que libero recursos

- Incrementa en 1 el recurso del semaforo que indica y despierta cualquier hilo bloqueado si lo hay

---

Resolución ejercicio 4



① División en problemas concurrentes

Tantos procesos como lectores haya y un proceso extra para gestionar el input del nº de lectores y lo que pueden hacer

② Programa para cada proceso

Programa main

input nº lectores;
input nº concurrentes lectura

crear hilos;

Programa leer_terminar ①

permitir leer;
leer;
terminar;

Programa leer_terminar ②

opciones {
① permitir_leer
② terminar

leer;

terminar;

③ Identificar S.C

Caso mas simple 2 lectores y solo puede leer 1

Lector 1

permitir leer;
leer;
terminar;

Lector 2

permitir leer;
leer;
terminar;

← S.C →

④ Codigo sincro

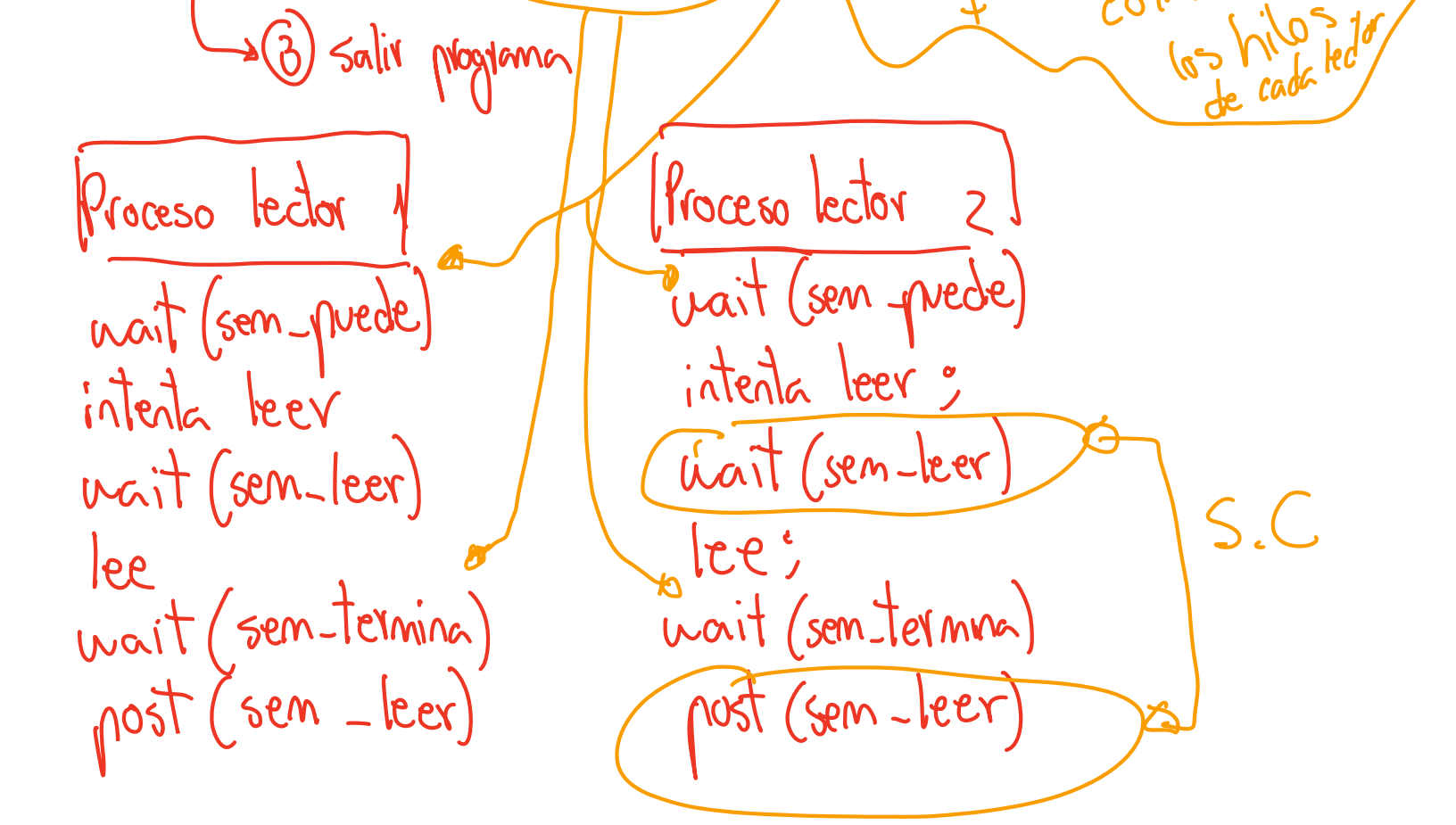
Caso mas simple 2 lectores

Proceso "main"

- input 1° lectores;
  - creo sem_leer(2,N)
- hilos para lector; → creo semaforo para lector sem_puede(0,1) sem_termina(0,1)
- switch opciones
  - lectores;
    - ① permiso → post(sem_puede)
    - ② termina → post(sem_termina)

???

No es S.C pero es la unica forma que encontre de comunicarme con



Ejercicio 5

Igual que el 4 pero solo 1 a la vez puede escribir no cambia nada mas.

Ejercicio 6

wait (sem_leer);
if (nº escr. > 0 y nº concurr == 0){
wait(sem_paso)
}

nº concurr ++;
lee;
wait(sem_termina);
nº concurr -- ;

if (nº escr. > 0 y nº concurr == 0) {
post (sem_ocupado)
}

no porque si ya no puedo otro lector no podrá da igual el orden

wait (sem_escribir);
if (nº concur > 0) {
wait(sem_ocupado)
}

escribir;

wait (sem_termina);
nº escr. --
if (nº escr. == 0 || nº concur > 0){
post(sem_paso)
}
}

Ejercicio 7

Necesitaria diferenciar entre escritores en papel y escribies que quieren el papel

un bucle while en lector

con sem_paso y

ya esta??, bueno

y un if despues

de ese while comprobando

si hay escritor en el papel