Semafori

Strumenti di sincronizzazione utilizzabili da thread e processi

Non assomigliano ai semafori stradali!

semaforo =
  variabile intera che può assumere solo valori non negativi
  sono ammesse solo 2 operazioni:
    sem_post: incrementa di 1 (a teoria chiamata V)
    sem_wait: decrementa di 1 (a teoria chiamata P)
  queste operazioni sono atomiche, avvengono automaticamente
  in mutua esclusione.

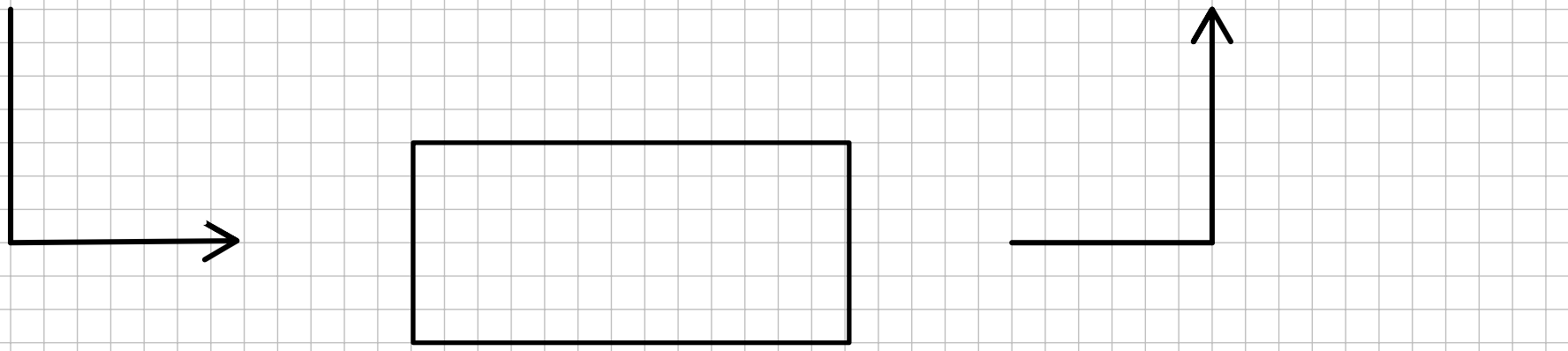
la sem_wait deve eventualmente attendere che il valore sia positivo

Tecnica produttori-consumatori

(vale per thread e processi)

Un produttore che fornisce dati da elaborare

Un consumatore che effettua l'elaborazione



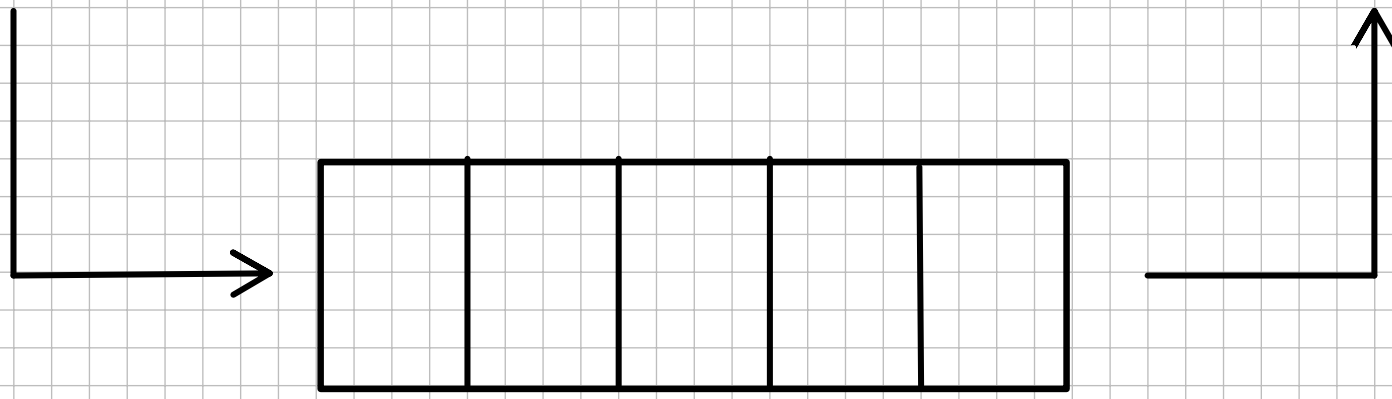
Memoria condivisa dove vengono passati i dati.
Necessari semafori per la sincronizzazione

Tecnica produttori-consumatori

(vale per thread e processi)

Un produttore che fornisce
dati da elaborare

Un consumatore che
effettua l'elaborazione



Memoria condivisa dove vengono passati i dati.
Necessari semafori per la sincronizzazione

Per motivi di efficienza conviene avere un buffer con più slot!

Per gestire un buffer con b posizioni utilizziamo due semafori

sem_free_slots: numero di slot dove il produttore può scrivere un oggetto
(inizializzato a b)

sem_data_items: numero di oggetti scritti dal produttore che il consumatore deve elaborare
(inizializzato a 0)

Se il produttore deve scrivere qualcosa effettua
sem_wait(sem_free_slots)
dopo la wait scrive l'oggetto ed effettua una
sem_post(sem_data_items)
Queste operazioni mantengono i semafori allineati con il numero effettivo di slot liberi e oggetti presenti.

Quando il consumatore vuole un nuovo dato effettua
sem_wait(sem_data_items)
che aspetta ci sia un dato disponibile e mantiene aggiornato il numero di oggetti sul buffer

Il consumatore legge l'oggetto e esegue
sem_post(sem_free_slots)
che mantiene aggiornato il numero di slot liberi

Dopo ogni operazione è mantenuto l'invariante:
sem_free_slots+ sem_data_items= b (dimensione buffer)

Come gestiamo le posizioni libere/occupate nel buffer?

Usiamo un indice p per la prossima posizione dove scriverà il produttore
e un indice c per la prossima posizione dove legge il consumatore

Grazie all'uso dei semafori abbiamo che

c <= p <= c + b

quando c=p sem_data_items è 0 e c non può avanzare oltre,
quando p=c+b sem_free_slots è 0 e p non può avanzare oltre.

Facciamo finta di avere un buffer infinito ma accediamo alle posizioni
c%b e p%b che sono tra 0 e b-1

Problema della terminazione: il produttore prima o poi termina i dati da elaborare.

Come lo segnala al consumatore?

La strategia più semplice consiste nel concordare un dato dummy che indica che sono finiti i dati da elaborare ad esempio un puntatore NULL, un valore <0, etc. ....