Sistemi Operativi
Esercizi Ricapitolazione

Docente: Claudio E. Palazzi
cpalazzi@math.unipd.it

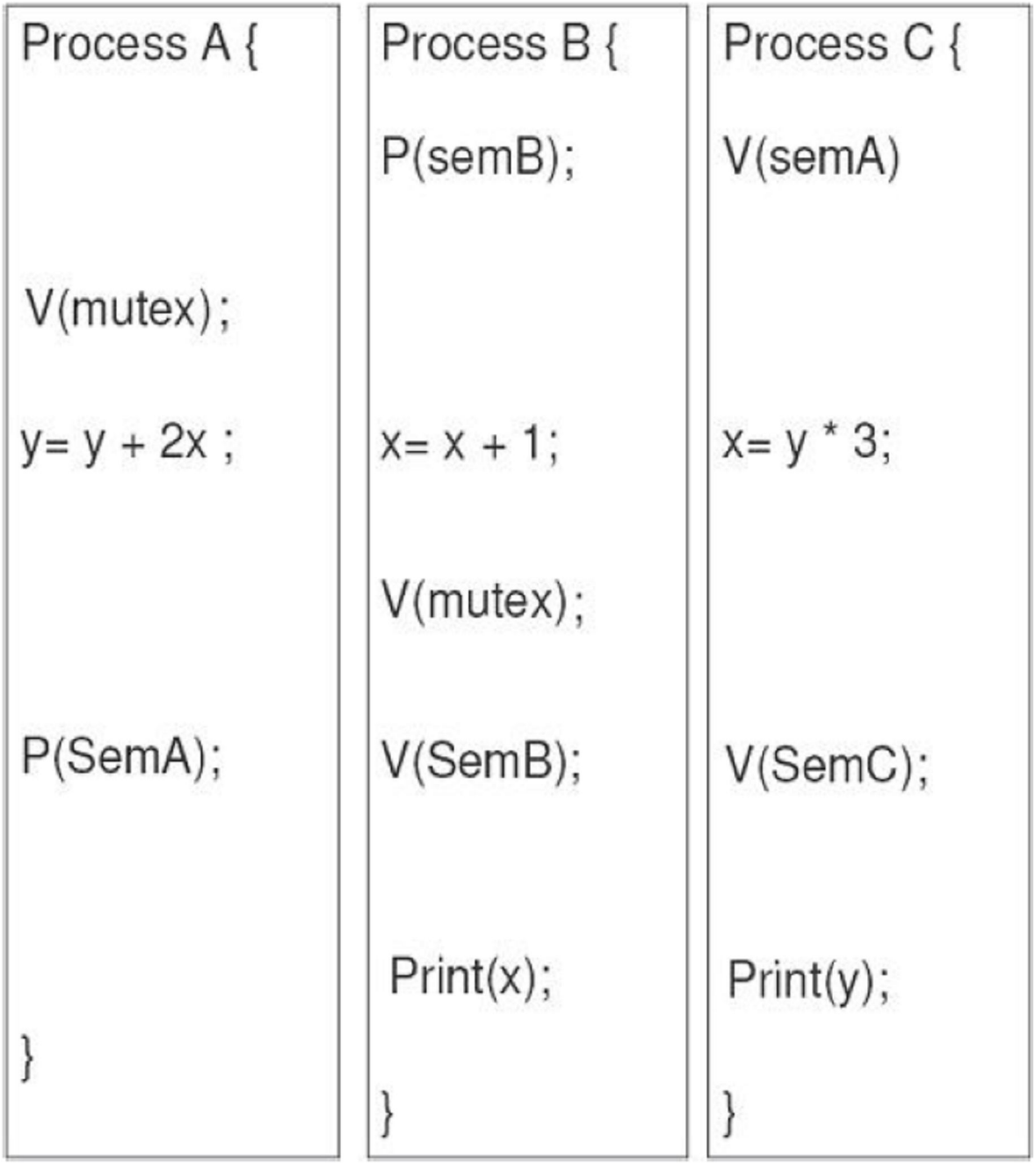
Problema Sincronizzazione Semafori

Si considerino tre processi (A, B e C) i quali devono eseguire alcune operazioni sulle variabili x e y e poi stamparne il risultato finale. A questo proposito, si consideri la seguente sequenza di avvenimenti:

- I processi A e C devono essere in attesa di essere risvegliati
- Il processo B sveglia il processo A, che a sua volta sveglia il processo C.
- I tre processi accedono concorrentemente (ma in mutua esclusione) alle variabili condivise x e y, poi B e C si bloccano in attesa che A termini la computazione su x e y (B e C avranno eseguito la loro computazione su x e y prima di bloccarsi)
- Terminata la computazione, è A stesso a svegliare B che a sua volta sveglia C.
- B stampa il valore di x e C stampa il valore di y (senza un ordine prestabilito).

[A] Assumendo che i semafori SemA, SemB e SemC siano inizializzati a 0, che il semaforo Mutex sia inizializzato a 1, e che inizialmente si abbia x = 2 e y = 1, si correggano e/o completino i frammenti di codice riportati nella slide seguente.

[B] Si discuta inoltre i possibili valori finali delle variabili x e y.


Process A {
V(mutex);
y= y + 2x ;
P(SemA);
}
Process B {
P(semB);
x= x + 1;
V(mutex);
V(SemB);
Print(x);
}
Process C {
V(semA)
x= y * 3;
V(SemC);
Print(y);
}

Soluzione [A]

Process A {	Process B {	Process C {
P(semB); V(semA);	V(semB);	P(semA);
P(mutex); y = y + 2*x V(mutex);	P(mutex); x = x + 1 V(mutex);	P(mutex); x = y*3; V(mutex);
V(semA);	P(semA); V(semC);	P(semC);
	Print(x);	Print(y);
}	}	}

Soluzione [B]

I tre processi potrebbero entrare e uscire dalla sezione critica limitata da P(mutex) e V(mutex) in un ordine qualsiasi dunque per determinare il valore finale di x e y occorre provare tutte le combinazioni.

- Ordine A, B, C: risultato x = 15 e y = 5
- Ordine A, C, B: risultato x = 16 e y = 5
- Ordine B, A, C: risultato x = 21 e y = 7
- Ordine B, C, A: risultato x = 3 e y = 7
- Ordine C, A, B: risultato x = 4 e y = 7
- Ordine C, B, A: risultato x = 4 e y = 9

Problema – 17/12/2008

Si consideri un elaboratore dotato di un sistema di memoria virtuale paginata e di un processore la cui frequenza di ciclo di clock sia 1 MHz. Si assuma che detto processore impieghi un ciclo di clock per eseguire istruzioni che non comportino riferimenti a pagine di memoria diverse da quella corrente. Si assuma inoltre che valgano le seguenti ipotesi:

- l'accesso a una pagina di memoria diversa da quella corrente comporti un costo temporale aggiuntivo di 1 µs;
- ciascuna pagina di memoria sia composta di 1.000 B;
- il disco fisso ruota alla velocit`a di 6.000 rpm (rotazioni/minuto);
- il trasferimento tra disco e memoria avvenga a 1.000.000 B/s;
- il tempo medio per spostare la testina dalla posizione corrente fin sopra la traccia dove si trova il punto di lettura/scrittura su disco sia di 10 ms;
- 1 blocco su disco corrisponda in dimensione a 1 pagina di memoria virtuale;
- 98% delle istruzioni eseguite facciano riferimento alla pagina corrente;
- 80% delle pagine accedute (diverse dalla corrente) si trovi gi`a in memoria;
- quando una nuova pagina debba essere caricata in memoria, nel 50% dei casi quella rimpiazzata sia stata precedentemente modificata.

Si calcoli il tempo medio effettivo di esecuzione di ciascuna istruzione su tale elaboratore assumendo che il sistema stia eseguendo un unico processo e che il processore rimanga inattivo (stato idle) durante i trasferimenti di dati.

Soluzione – 17/12/2008 – Parte 1/2

- Il processore considerato esegue una istruzione ogni µs quando essa non comporti riferimenti a pagine di memoria diverse da quella corrente.

- Per ipotesi, il 2% delle istruzioni richiede 1 µs in più per accedere ad aree di memoria diverse da quella corrente; di queste, il 20% richiede tempo ulteriore per leggere dal disco un pagina di memoria virtuale e, nel 50% dei casi, la pagina rimpiazzata dovrà essere copiata su disco, aumentando quindi il tempo di esecuzione della istruzione corrispondente. Il tempo medio di esecuzione risulta quindi dalla somma delle seguenti componenti:
  - tempo di esecuzione sul processore: 1 µs per tutte le istruzioni;
  - tempo di accesso a un'area di memoria diversa da quella corrente: 1 µs per il 2% delle istruzioni;
  - tempo di lettura da disco della pagina referenziata: tempo medio di posizionamento, rotazione e trasmissione, per il 20% delle istruzioni che fanno riferimento a pagine diverse dalla corrente;
  - tempo di scrittura su disco della pagina rimpiazzata: come per punto precedente, ma solo nel 50% dei casi in cui si accede a disco;

Soluzione – 17/12/2008 – Parte 2/2

- Il disco ruota a 6.000 rpm, ovvero 100 rotazioni / secondo; una rotazione completa richiede quindi 10 ms.
  Statisticamente, quando la testina del disco si muove per leggere o scrivere su un'altra traccia, si troverà a mezzo giro di distanza dalla posizione cercata, aggiungendo quindi un ulteriore ritardo medio di 5 ms (equivalente a mezza rotazione) a ogni spostamento di testina.
  A ogni lettura o scrittura su disco occorre dunque aggiungere il tempo di spostamento della testina (per ipotesi, in media 10 ms), il tempo di rotazione del disco necessario per posizionare la testina sul punto iniziale di lettura o scrittura (in media 5 ms) e infine il tempo di trasferimento della pagina, pari a 1.000 B / 1.000.000 = 1 ms.

  In totale quindi, ogni accesso a disco comporta un ulteriore ritardo di: 5 ms + 10 ms + 1 ms = 16 ms. In conclusione, il tempo medio di esecuzione di un'istruzione risulta essere: 1 µs + 0,02 × [1 µs + 0,2 × (16 ms +0,5 × 16 ms)] = 97,02 µs

Domanda – 15/12/2010

- Gli hard disk sono componenti molto importanti di un computer che permettono di immagazzinare in modo permanente un insieme moderatamente grande di informazioni. Il sistema operativo si occupa di gestire anche queste componenti hardware permettendo, ad esempio, operazioni su file quali memorizzazione, recupero, cancellazione, ecc.
  I computer moderni sono dotati di hard disk di capacità sempre maggiore fornendo dunque un vantaggio in termini di spazio di memorizzazione agli utenti ma anche nuove complessità di gestione per il sistema operativo.

- Lo studente illustri, in massimo una pagina, le implicazioni (es. problematiche e possibili soluzioni, ma anche semplificazioni che diventerebbero possibili) per le varie componenti e strutture di un sistema operativo che si trovasse a dover gestire un hard disk di capacità infinita.

Risposta – 15/12/2010

- Molte funzioni del Sistema Operativo sarebbero coinvolte (e stravolte) nel dover gestire un hard-disk di dimensione infinita. Lo studente è invitato a rivisitare criticamente il programma del corso provando a riflettere sulle modifiche necessarie.

- Consideri ad esempio le implicazioni riguardo a:
  - dimensione degli indirizzi
  - dimensione strutture dati
  - utilizzabilità dei file system noti
  - reperire un file
  - contiguità file
  - lista blocchi liberi
  - importanza o meno della frammentazione
  - … (molto altro ancora)

Lettori e scrittori MULTIPLI

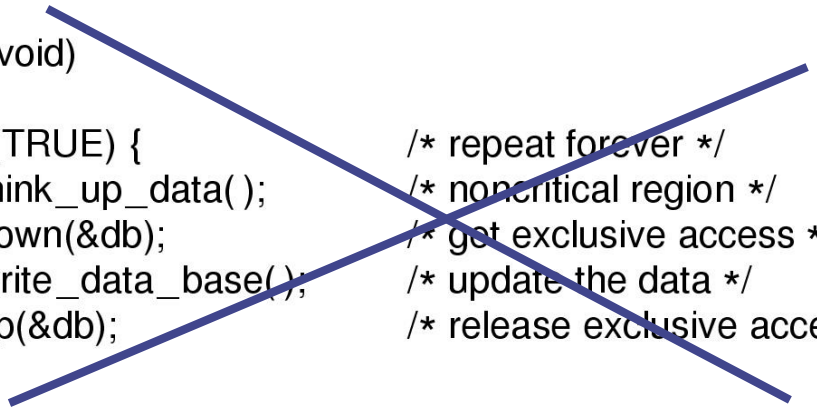
- Si consideri una versione modificata del problema dei lettori-scrittori in cui
  - Più lettori possono accedere contemporaneamente a db
  - Più scrittori possono accedere contemporaneamente a db
  - se uno o più scrittori chiedono di accedere mentre uno o più lettori stanno accedendo a db, lo scrittore deve attendere che i lettori abbiano finito
  - se uno o più lettori chiedono di accedere mentre uno o più scrittori stanno accedendo a db, il lettore deve attendere che gli scrittori abbiano finito

```
typedef int semaphore;              /* use your imagination */
semaphore mutex = 1;                /* controls access to 'rc' */
semaphore db = 1;                   /* controls access to the database */
int rc = 0;                         /* # of processes reading or wanting to */

void reader(void)
{
    while (TRUE) {                  /* repeat forever */
        down(&mutex);               /* get exclusive access to 'rc' */
        rc = rc + 1;                /* one reader more now */
        if (rc == 1) down(&db);     /* if this is the first reader ... */
        up(&mutex);                 /* release exclusive access to 'rc' */
        read_data_base( );          /* access the data */
        down(&mutex);               /* get exclusive access to 'rc' */
        rc = rc – 1;                /* one reader fewer now */
        if (rc == 0) up(&db);       /* if this is the last reader ... */
        up(&mutex);                 /* release exclusive access to 'rc' */
        use_data_read( );           /* noncritical region */
    }
}


void writer(void)
{
    while (TRUE) {                  /* repeat forever */
        think_up_data( );           /* noncritical region */
        down(&db);                  /* get exclusive access */
        write_data_base( );         /* update the data */
        up(&db);                    /* release exclusive access */
    }
}
```

SOLUZIONE DELLA VERSIONE CLASSICA

```
typedef int semaphore;              /* use your imagination */
semaphore mutex = 1;                /* controls access to 'rc' */
semaphore db = 1;                   /* controls access to the database */
int rc = 0;                         /* # of processes reading or wanting to */

void reader(void)
{
    while (TRUE) {                  /* repeat forever */
        down(&mutex);               /* get exclusive access to 'rc' */
        rc = rc + 1;                /* one reader more now */
        if (rc == 1) down(&db);     /* if this is the first reader ... */
        up(&mutex);                 /* release exclusive access to 'rc' */
        read_data_base( );          /* access the data */
        down(&mutex);               /* get exclusive access to 'rc' */
        rc = rc – 1;                /* one reader fewer now */
        if (rc == 0) up(&db);       /* if this is the last reader ... */
        up(&mutex);                 /* release exclusive access to 'rc' */
        use_data_read( );           /* noncritical region */
    }
}
```

```
void writer(void)
{
    while (TRUE) {                  /* repeat forever */
        think_up_data( );           /* noncritical region */
        down(&db);                  /* get exclusive access */
        write_data_base( );         /* update the data */
        up(&db);                    /* release exclusive access */
    }
}
```



SOLUZIONE NUOVA VERSIONE

La nuova procedura ***writer*** sarà molto simile a ***reader***:

- *wc* al posto di *rc* (ma funziona anche lasciando *rc* in entrambi… perché?)

- *write_data_base()* al posto di *read_data_base()*