

# esercizi

# processi cpu bound

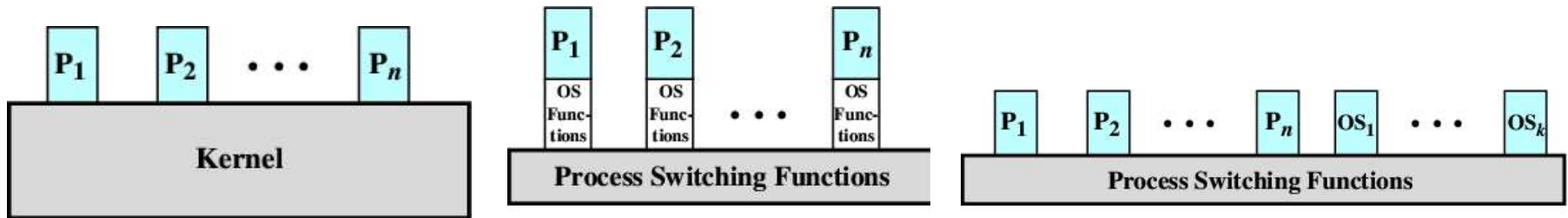
- Considera  $N$  processi che non eseguono system call (cpu bound)
- Ciascun processo ha bisogno, per concludere l'esecuzione, di 1 s di CPU.
- Nel sistema non ci sono altri processi.
- Gli  $N$  processi sono inizialmente in stato "ready"
- Il time slice è per tutti di 3 s.
- Quante volte verrà eseguito il dispatcher prima che tutti gli  $N$  processi terminino l'esecuzione? <sub>2</sub>

# processi cpu bound

- Supponi che il time slice sia per tutti di 100 ms.
- Quante volte verrà eseguito il dispatcher prima che tutti gli  $N$  processi terminino l'esecuzione?

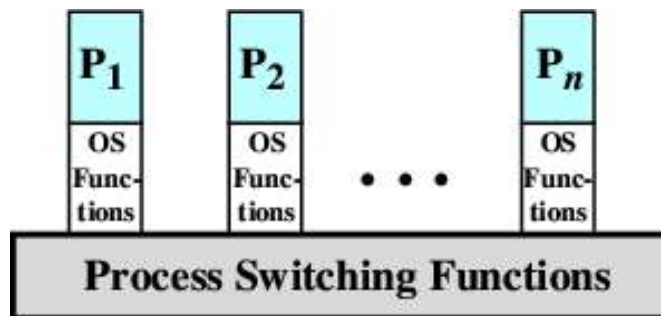
# processi cpu bound

- Quanti sono i process switch?
- Quanti sono i mode switch?
- Per fare questo calcolo è importante sapere che modello di sistema stiamo considerando?



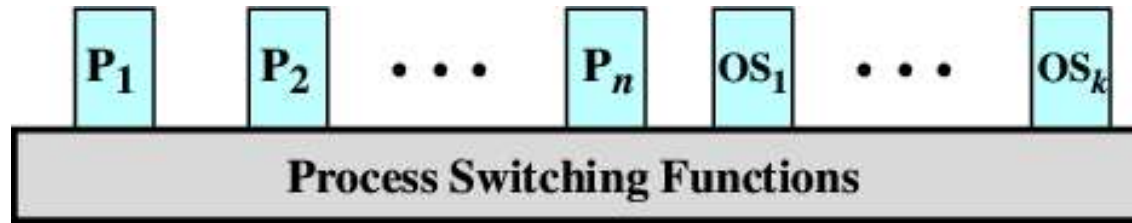
# processi i/o bound

- Supponi che nel sistema ci siano due processi  $P_1$  e  $P_2$ .  $P_1$  fa  $n$  chiamate di sistema, ciascuna bloccante.  $P_2$  non fa alcuna chiamata di sistema e terminerà dopo  $P_1$ . Supponi che la CPU venga restituita a  $P_1$  non appena  $P_1$  va in stato “ready” (preemption).
- Conta i mode switch e i process switch nel modello “kernel execution within process” fino al soddisfacimento dell' $n$ -esima chiamata di sistema.



# processi i/o bound

- considera lo stesso problema nel modello “process-based”



- pensi si possa rispondere alla domanda? quali informazioni mancano?

# first/next/best fit

- vedi Stallings esercizio 7.6

# buddy system

- Spazio di memoria da allocare 1 MB. Inizialmente nessun blocco allocato.
- Viene richiesto un blocco  $A$  di 64 KB. Quante volte viene richiamata la procedura `get_hole()` ?
- Com'è l'albero che rappresenta sistema dopo l'allocazione? Cosa contengono le liste  $L_i$ ?



# buddy system

- considera due indirizzi iniziali di due blocchi  $b_1$  e  $b_2$  della stessa grandezza che vengono identificati dall'indirizzo del primo byte
- dai un metodo basato sulla rappresentazione binaria di  $b_1$  e  $b_2$  per capire se sono due buddies.

# buddy system

- Mostra un algoritmo `free_block(b)` ricorsivo per aggiornare le liste  $L_i$  a seguito del rilascio di un blocco  $b$  (due campi:  $b.addr$ ,  $b.i$ ).
- indirizzo di  $b$ :  $b.addr$
- taglia di  $b$ :  $2^{b.i}$
- Adotta la seguente strategia: due buddies vengono uniti appena possibile

# buddy system

- varianti
  - i buddies possono nascere in più di due alla volta (3, 4, ecc.)
  - i buddies possono non essere tutti uguali in taglia (es. serie di fibonacci vedi esercizio 7.10 Stallings)
- implementazione
  - come faccio a implementare le liste  $L_i$ ?
    - non mi posso avvalere della allocazione dinamica della memoria!

# paging

- 32 bit per un indirizzo fisico
- 32 bit per un indirizzo logico
- frame di 4KB
- quanti bit uso per l'offset?
- con quanti bit identifico un frame? quanti frame abbiamo in memoria fisica?
- con quanti bit identifico una pagina? quante pagine abbiamo nello spazio di indirizzamento logico?
- se un processo usa tutte le pagine quanto è grande la page table?
- ... e se un processo usa la prima e l'ultima pagina?
- vedi anche esercizio 7.12 di Stallings

# paging

- p: numero di bit per identificare una pagina nello spazio di indirizzamento logico
- f: numero di bit per identificare un frame nello spazio di indirizzamento fisico
- discuti potenzialità e/o svantaggi dei tre casi
  - $p=f$
  - $p<f$
  - $p>f$