Lezione 15

Esercizi su Linux e Windows

Linux: Scheduling dei processi

- 40 livelli di priorita': da -19 (la massima) a 20 (la minima)
- I task vengono eseguiti secondo l'ordine di priorita'
  - Un array delle priorita' conserva i puntatori alle liste dei task con la stessa priorita'
- Ogni livello gestito in modalita' round robin
- Priorita' variabile a secondo del quanto di tempo q
  - Time slice piccolo per task a bassa priorita'
  - Time slice grande per task a alta priorita'
- Ogni task e' in esecuzione fino a:
  - Esaurimento del quanto di tempo
  - Operazioni di I/O
  - E' disponibile un task con priorita' maggiore

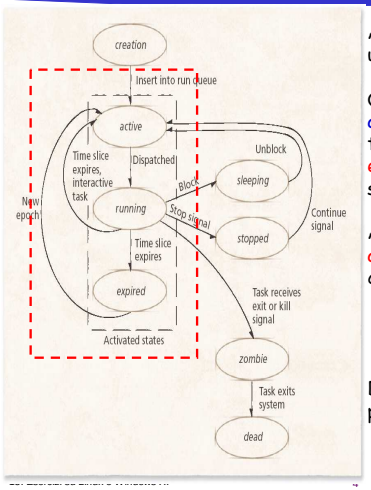
Epoche

- Per evitare la starvation dei processi il kernel divide il tempo di CPU in Epoche

EPOCA =
Periodo di tempo predefinito (starvation limit)
entro cui tutti i processi sono eseguiti almeno una volta

- All'inizio di ogni epoca viene assegnato un quanto di tempo q ad ogni processo, in funzione della sua priorita'.
- Durante un'epoca la CPU viene assegnata round robin piu' volte allo stesso processo, fino a quando q non e' esaurito
- Quando q e' esaurito per tutti i processi pronti scade l'epoca e nessun processo e' piu' rischedulato

Lo stato expired



Alla creazione di un processo si assegna un valore di default q= Q = 200ms

Quando i processi esauriscono il loro quanto oppure scade la starvation limit tutti i processi passano allo stato expired e non possono essere piu' schedulati

All'inizio di una nuova epoca il nuovo quanto di tempo q viene ricalcolato come

q = Q + T/2
(priorita' maggiore)

Dove T e' il tempo non usato nell'epoca precedente (es. erano bloccati per I/O)

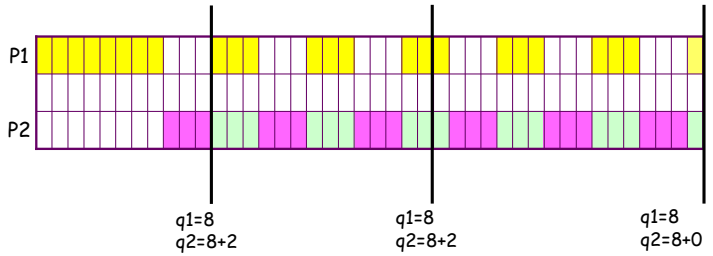
Esercizio

- P1 task CPU bound → 50 ms di uso continuo della cpu
- P2 task I/O bound → 3 ms di cpu e 3 ms di I/O (continuativo)
- A inizio epoca Q = q1 = q2 = 8
- Esecuzione di 3 epoche

- Determinare il diagramma di Gantt dell'esecuzione

Esempio : Q=8ms

- P1 task CPU bound → 50 ms di uso continuo della cpu
- P2 task I/O bound → 3 ms di cpu e 3 ms di I/O (continuativo)
- A inizio epoca q1=8 q2=8
- Esecuzione di 3 epoche



Notare come P2 riceve un bonus in termini di q

Linux: Algoritmo di paginazione

- Caratteristiche generali:
  - Quando le pagine sono lette in memoria sono inserite in una cache
  - Per migliorare le prestazioni, le pagine sporche sono copiate periodicamente (5-10sec) sul disco (write back caching)
- Due tipi di pagine:
  - Attive (referenziate di recente)
  - Inattive (candidate all'eleminazione)
- L'algoritmo di paginazione e' una variante del FIFO seconda chance
  - FIFO seconda chance a 2 livelli

FIFO seconda chance a 2 livelli

- 2 liste di dimensione variabile:
  - Pagine attive e pagine inattive
- FIFO seconda chance in ogni lista
  - Le pagine entrano nella lista delle pagine inattive con bit=1
- Nella lista delle pagine inattive un secondo riferimento "promuove" la pagina nella lista delle pagine inattive con bit=0
- Periodicamente si riorganizzano le liste:
  - Si spostano alcune pagine attive poco usate nella lista delle pagine inattive, con bit=0
  - Circa 2/3 nella lista delle pagine attive e 1/3 tra quelle inattive

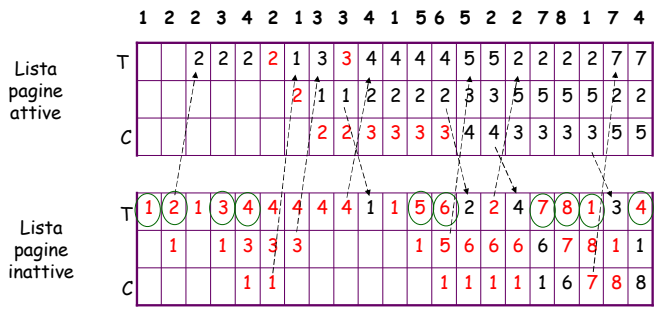
Esercizio

- Determinare il numero di page faults per la sequenza

  1, 2, 2, 3, 4, 2, 1, 3, 3, 4, 1, 5, 6, 5, 2, 2, 7, 8, 1, 7, 4

- Per semplicita'
  - si consideri fissata e costante la lunghezza di entrambe le liste L=3
  - Le liste si riorganizzano ad ogni riferimento

Soluzione



Rosso → bit =1
Nero → bit =0

Windows : Scheduling dei thread

- Scheduling basato su priorita' con prelazione
- 32 livelli di priorita'
  - 0 = priorita' minore
  - 31 = priorita' maggiore
- 32 ready queue (una per ogni priorita')
- Ogni coda gestita round robin.
- Due classi di thread
  - Thread real time (priorita' statiche 16 → 31)
  - Thread normali (priorita' dinamiche 0 → 15)

Scheduling dei thread

Clsssi di priorita'

Livelli di priorita'	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1

- I thread normali sono divisi in classi di priorita'
  - es. proc di sistema o processi utente
- All'interno di ogni classe la esistono livelli di priorita'
  - Es. background/foreground

Scheduling dei thread

- I thread normali hanno una priorita' dinamica
  - finestra che riceve input da tastiera
    ➔ Aumento della priorita' +2
  - Completamento I/O, ottenimento di una risorsa,
    ➔ aumento della priorita' +1
  - Scadenza del quanto
    ➔ riduzione della priorita' -1
- comunque mai sopra highest o sotto lowest

Esercizio

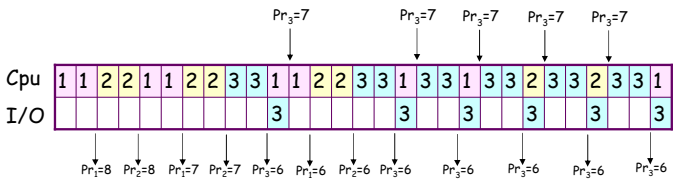
- 3 processi in classe normal
  - Highest = 10 Lowest = 6

	cpu	I/O	liv.pr.	
P1	9	0	9 (above norm.)	
P2	8	0	9	
P3	2	1	7 (below norm.)	(in sequenza)

- Incremento di 1 dopo completamento di I/O
- Decremento di 1 dopo esaurimento del quanto di tempo
- Quanto di tempo q=2

Determinare il diagramma di Gantt

soluzione



A regime la cpu e' schedulata in base alle necessita' di P3

Sostituzione delle pagine

- Windows XP utilizza un modello ibrido di sostituzioine delle pagine:
  Working set + Localized Least recently used (LLRU)
- Vengono assegnati ad ogni processo
  - Working set minimo
  - Working set massimo
  - Tali valori dipendono dal carico del sistema
- Quando un processo richiede piu' pagine del working set massimo il VMM sposta una pagina sul disco secondo la politica LLRU
- Localized ➔ politica focalizzata sul singolo processo

Windows: sostituzione delle pagine

- Elimina tutte le pagine che eccedono la dimensione del working set massimo
  - Le pagine eliminate sono dette anche ridotte
- Le pagine ridotte vengono spostate nella
  - Lista delle pagine in attesa
  - Lista delle pagine modificate
  - Lista delle pagine modificate ma da non scrivere
- La pagina viene spostata nuovamente nella lista delle pagine valide in caso di nuovi riferimenti (soft fault)
- La pagina viene spostata nella lista delle pagine libere se non viene piu' referenziata per un certo periodo di tempo
- Un thread di sistema a bassa priorita' la azzera e la sposta nella lista delle pagine azzerate

- Le pagine per i nuovi processi vengono prese dalla lista delle pagine azzerate

Esercizio

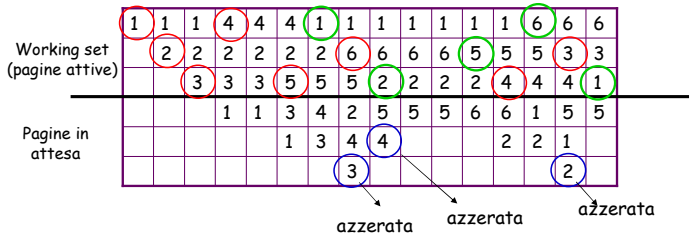
- Working set massimo = 3 (costante)
- Tempo di permanenza nella lista delle pagine in attesa = max 3 u.t.
- Si considerino solo gli stati attivo, attesa e azzerato
- Sequenza di riferimento:

1, 2, 3, 4, 2, 5, 1, 6, 2, 2, 1, 5, 4, 6, 3, 1

- Determinare il numero di page faults
- Determinare il numero di soft faults

Soluzione

1, 2, 3, 4, 2, 5, 1, 6, 2, 2, 1, 5, 4, 6, 3, 1



Page fault
Soft fault

8 page faults
5 soft faults

In conclusione

- Linux e Windows adottano strategie ibride per lo scheduling dei processi e per la sostituzione delle pagine
  - Linux
    - Scheduling CPU: priorita' dinamiche + round robin + starvation limit
    - Sost. Pagine: FIFO seconda chance a 2 livelli
  - Windows
    - Scheduling CPU: priorita' dinamiche + round robin
    - Sost. Pagine: working set + LLRU

- Entrambi cercano di privilegiare (mediante le priorita' dinamiche) i processi interattivi
- Entrambi non rimuovono subito dalla memoria le pagine "poco" refernziate