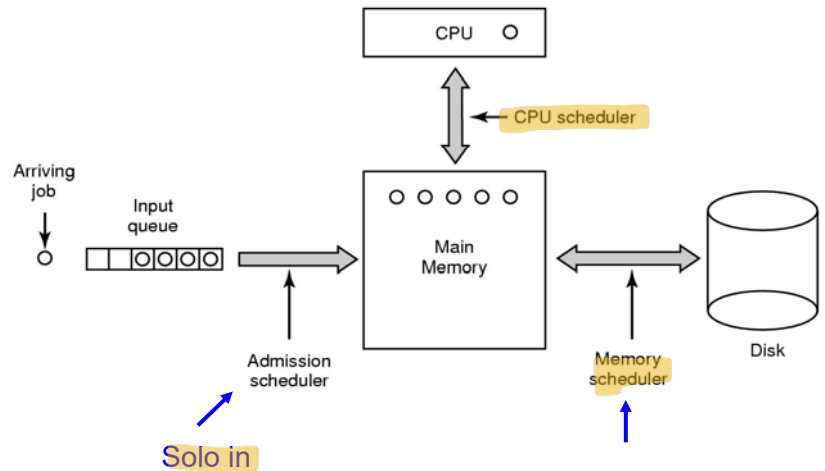
Scheduling

- Lo scheduler è la parte del SO che si occupa di decidere quale fra i processi *pronti* può essere mandato *in esecuzione*
- L'algoritmo di scheduling (la politica utilizzata dallo scheduler) ha impatto su:
  - prestazioni percepite dagli utenti
  - efficienza nell'utilizzo delle risorse della macchina
- Lo scheduling ha obiettivi diversi in diversi sistemi (batch, interattivi,real-time,…)

1

Diversi livelli di scheduling



Solo in sistemi batch (viene creato il processo corrispondente al job ammesso)

Parte della gestione memoria (non troppi processi in memoria)

2

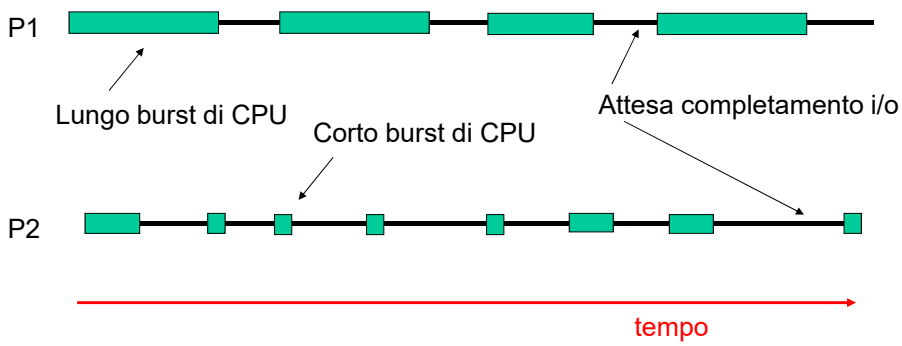
Obiettivi degli algoritmi di Scheduling

- *Fairness* (Equità) - **processi della stesso tipo devono avere trattamenti simili**
- *Balance* (Bilanciamento) - **tutte le parti del sistema devono essere sfruttate (CPU, dispositivi …)**
- Sistemi batch
  - *Throughput* - massimizzare il numero di job completati in un intervallo di tempo
  - *Tempo di Turnaround* - minimizzare il tempo di permanenza di un job nel sistema
- Sistemi interattivi
  - *Tempo di risposta* - minimizzare il tempo di riposta agli eventi
  - *Proporzionalità* - assicurare che il tempo di risposta sia proporzionale alla complessità dell'azione richiesta
- Sistemi real time
  - Rispettare le scadenze o non degradare la **QoS**

3

in genere conosce i suoi processi, sono previsti

Tipologie di processi



- processi *CPU-bound* -- lunghi periodi di elaborazione fra due richieste successive di I/O
- processi *I/O-bound* -- brevi periodi di elaborazione fra due richieste successive di I/O (forte tendenza a usare I/O)

4

CLASSIFICAZIONE DINAMICA

Algoritmi di Scheduling

- Scheduling *senza prerilascio*
  - lo scheduler interviene solo quando un processo termina, si blocca in attesa di un evento o rilascia volontariamente la CPU
- Scheduling *con prerilascio*
  - lo scheduler può intervenire ogni volta che è necessario per ottenere gli obiettivi perseguiti
    - quando diventa pronto un processo a più alta priorità rispetto a quello in esecuzione
    - quando il processo in esecuzione ha sfruttato la CPU per un tempo abbastanza lungo

5

Momenti della schedulazione

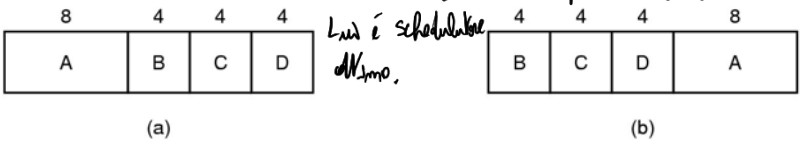
Quando va presa una decisione di scheduling?

- quando il processo corrente termina
- quando il processo corrente effettua un'operazione sospensiva
- quando viene creato un nuovo processo
- quando c'è un'interruzione da I/O: al termine della ISR può girare il processo che è stato interrotto, il processo in attesa dell'operazione di I/O, o un altro
- quando c'è un'interruzione da timer - oppure ogni k interruzioni: si toglie la risorsa CPU al processo che la stava usando

6

Scheduling nei sistemi Batch

Nota: so normalmente so solo il tempo di esecuzione. Lui è schedulabile dall'inizio.



- Un esempio di scheduling secondo la strategia che privilegia il job più corto (SJF "Shortest Job First")
  - l'insieme dei job da schedulare è noto all'inizio
  - si conosce il tempo di esecuzione *T* di ogni job
  - i job sono schedulati in ordine di *T* crescente
  - SJF minimizza il tempo di turnaround medio
  - non c'è prerilascio

7

Turnaround: entro ed esco quando ho finito, compreso il tempo che attesa.

Scheduling nei sistemi Batch (SJF)

Esempio:

4 job A,B,C,D con tempi di esecuzione *a*, *b*, *c*, *d*

- turnaround(A) -- *a*
- turnaround(B) -- *a* + *b*
- turnaround(C) -- *a* + *b* + *c*
- turnaround(D) -- *a* + *b* + *c* + *d*

turnaround totale $4a + 3b + 2c + 1d$

minimo quando *a,b,c,d* sono in ordine crescente

8

Scheduling per sistemi interattivi (Round Robin)

il processo corrente in esecuzione per un quanto di tempo (k timer interrupt, k≥1), va in fondo alla coda dei processi pronti e viene scelto il primo della lista

Non favorisce i processi I/O bound

Durata del quanto:

- almeno 10 volte il tempo di context switch (per avere meno del 10% di tempo sprecato nello switch), possibilmente di più
- non troppo (es. 100 msec) per evitare tempi di risposta alti per i processi molto interattivi
- tipicamente 20-50 msec

9

Y processi sono diversi però! Non ha tanto senso Round Robin così. Usiamo RR con code di priorità. Regola è partire dalla coda prioritaria. Quando è vuota passiamo alla prossima.

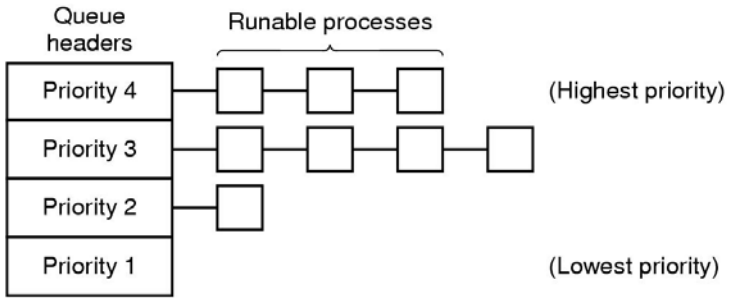
Scheduling per sistemi interattivi (Scheduling con priorità)

- Ogni processo ha una priorità
- Ogni volta va in esecuzione il processo a priorità più elevata
- Priorità diverse a utenti di diverso tipo
- Priorità diverse a processi di diverso tipo:
  es. I/O bound ⇒ priorità alta, per tenere i dispositivi di I/O alimentati

10

Scheduling con priorità a code multiple

Può essere organizzato per classi:



- Gira un processo della classe più alta, se esiste
- Round Robin all'interno di ogni classe

Appartenenza alle classi modificata dinamicamente per evitare che i processi nelle classi inferiori non ottengano mai la CPU (*starvation* = morte per fame)

11

Loro lasciano il controllo prima.

Scheduling con priorità a code multiple

Modifica dinamica della classe di appartenenza:

- La priorità cresce per i processi che passano da bloccato a pronto
- Legata alla percentuale f del quanto di tempo che è stato consumato l'ultima volta che il processo è andato in esecuzione (es. proporzionale a $1/f$, favorisce i processi I/O bound)
- I processi che usano tutto il quanto di tempo più volte vengono passati alla classe inferiore
- Alcuni sistemi danno quanti più lunghi ai processi nelle classi basse (*compute-bound*) per minimizzare l'overhead del cambio di contesto

12

Se un processo non fa altre cose se non usare la CPU non mi piace.

Scheduling per sistemi real-time

↱ completare entro deadline

Esigenza: reagire ad eventi in un tempo garantito

- sempre (*hard* real-time)
- con limitate eccezioni (*soft* real-time)

Gli eventi possono essere

- periodici
- aperiodici

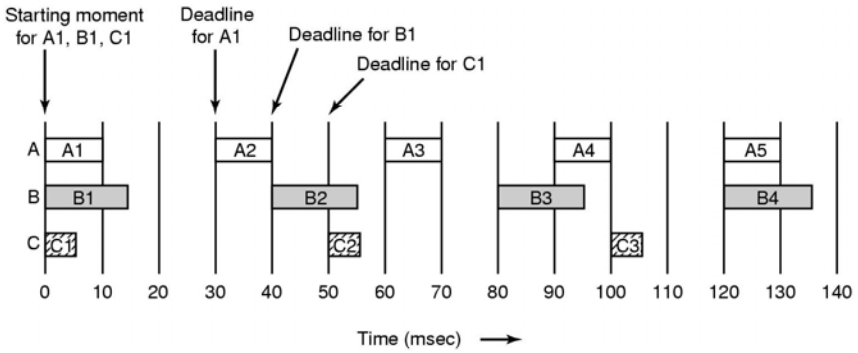
Eventi periodici di m tipi, se il tipo i occorre con periodo P_i e richiede C_i secondi di CPU per trattarlo, è necessario che:

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

13

hanno il vantaggio di conoscere meglio e bene le esigenze dei processi. In genere ho pochi processi e periodici

Schedulazione di un sistema multimediale



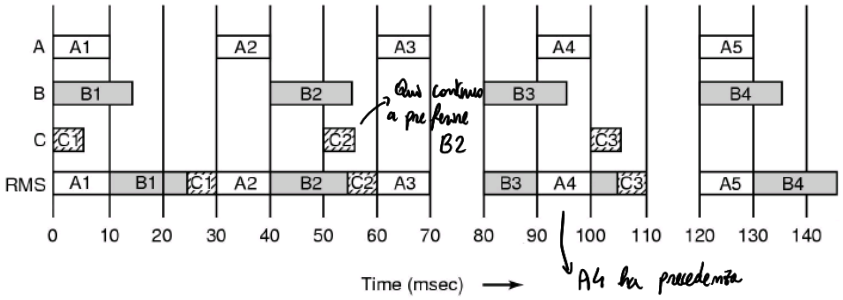
- 3 processi periodici per visualizzare un film:
  - **A (10 ms ogni 30 ms)** $C_1/P_1=1/3$
  - **B (15 ms ogni 40 ms)** $C_2/P_2= 0{,}375$
  - **C (5 ms ogni 50 ms)** $C_3/P_3= 0{,}1$

14

Obiettivo: fare sì che A1 abbia almeno 10ms di CPU come B e C.

Scegliere in caso di conflitto quello che ha periodo più piccolo (quello con deadline più vicina)

Schedulazione di un sistema multimediale (Shortest Period First)



- **A (10 ms ogni 30 ms)** $C_1/P_1=1/3$
- **B (15 ms ogni 40 ms)** $C_2/P_2= 0{,}375$
- **C (5 ms ogni 50 ms)** $C_3/P_3= 0{,}1$

15

Diamo di default la precedenza ai processi con periodo minore.