Processi e Thread

- Processi
- Thread
- Meccanismi di comunicazione fra processi (IPC)
- Problemi classici di IPC
- Scheduling
- Processi e thread in Unix
- Processi e thread in Windows

Scheduling
Introduzione al problema dello Scheduling (1)

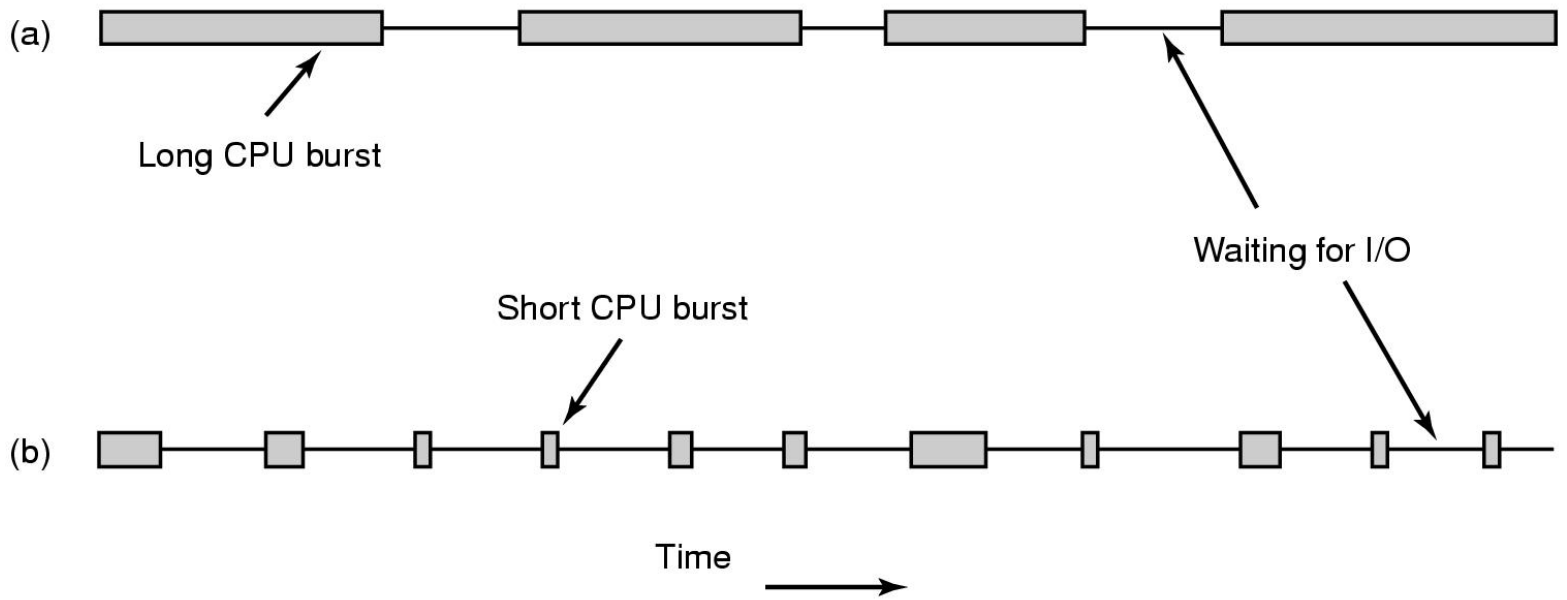
- Lo scheduler si occupa di decidere quale fra i processi *pronti* può essere mandato *in esecuzione*
- L'algoritmo di scheduling ha impatto su:
  - prestazioni percepite dagli utenti
  - efficienza nell'utilizzo delle risorse della macchina
- Lo scheduling ha obiettivi diversi in diversi sistemi (batch,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

Introduzione al problema dello Scheduling (2)

Obiettivi principali degli Algoritmi di Scheduling:

- *Fairness* - processi della stesso tipo devono avere trattamenti simili
- *Balance* - tutte le parti del sistema devono essere sfruttate (CPU, dispositivi …)
- Sistemi batch
  - *Throughput* **(produttivita')**- massimizzare il numero di job completati in un intervallo di tempo
  - *Turnaround Time* (**tempo medio di completamento**)- minimizzare il tempo di permanenza di un job nel sistema
- Sistemi interattivi
  - *Response time* - minimizzare il tempo di riposta agli eventi
  - *Proportionality* - assicurare che il tempo di risposta sia proporzionale alla complessità dell'azione richiesta

Introduzione al problema dello Scheduling (3)



- Due tipologie di processi :
  - processi *CPU-bound* -- lunghi periodi di eleborazione fra due richieste successive di I/O
  - processi *I/O-bound* -- brevi periodi di elaborazione fra due richieste successive di I/O
- Conviene dare priorità ai processi *I/O-bound*

Introduzione al problema dello Scheduling (4)

Quando si attiva lo schedulatore:

- Creazione di un processo
- Uscita di un processo
- Quando c'e' richiesta di I/O
- Quando avviene un interrupt
- Interruzione di clock (prerilascio/no-prerilascio)

Introduzione al problema dello Scheduling (5)

- Scheduling *senza prerilascio*
  - lo scheduler interviene solo quando un processo viene creato, termina o si blocca
- Scheduling *con prerilascio*
  - lo scheduler può intervenire ogni volta che è necessario per ottenere gli obiettivi perseguiti
    - quando diventa ready 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

Introduzione al problema dello Scheduling (6)

- Scheduling in sistemi *batch*
  - *FCFS (First-Come First-Served)*
  - SJF (Shortest Job First)
  - SRTN (Shortest Remaining Time Next)

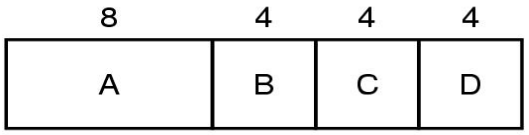
- Scheduling in sistemi *interattivi*
  - Round Robin
  - Code Multiple

_____ _____ _____ CPU BOUND

- - - - - - -

- - - - - - -

Scheduling nei sistemi Batch (1)



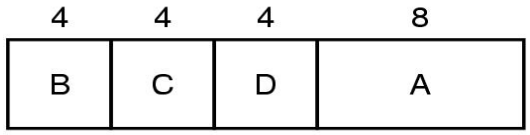
(a)

Tempo di completamento

A	8 min
B	12 min
C	16 min
D	20 min

==========

medio 14 min



(b)

Tempo di completamento

B	4 min
C	8 min
D	12min
A	20 min

==========

medio 11 min

- 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

Scheduling nei sistemi Batch (2)

Perché SJF funziona?

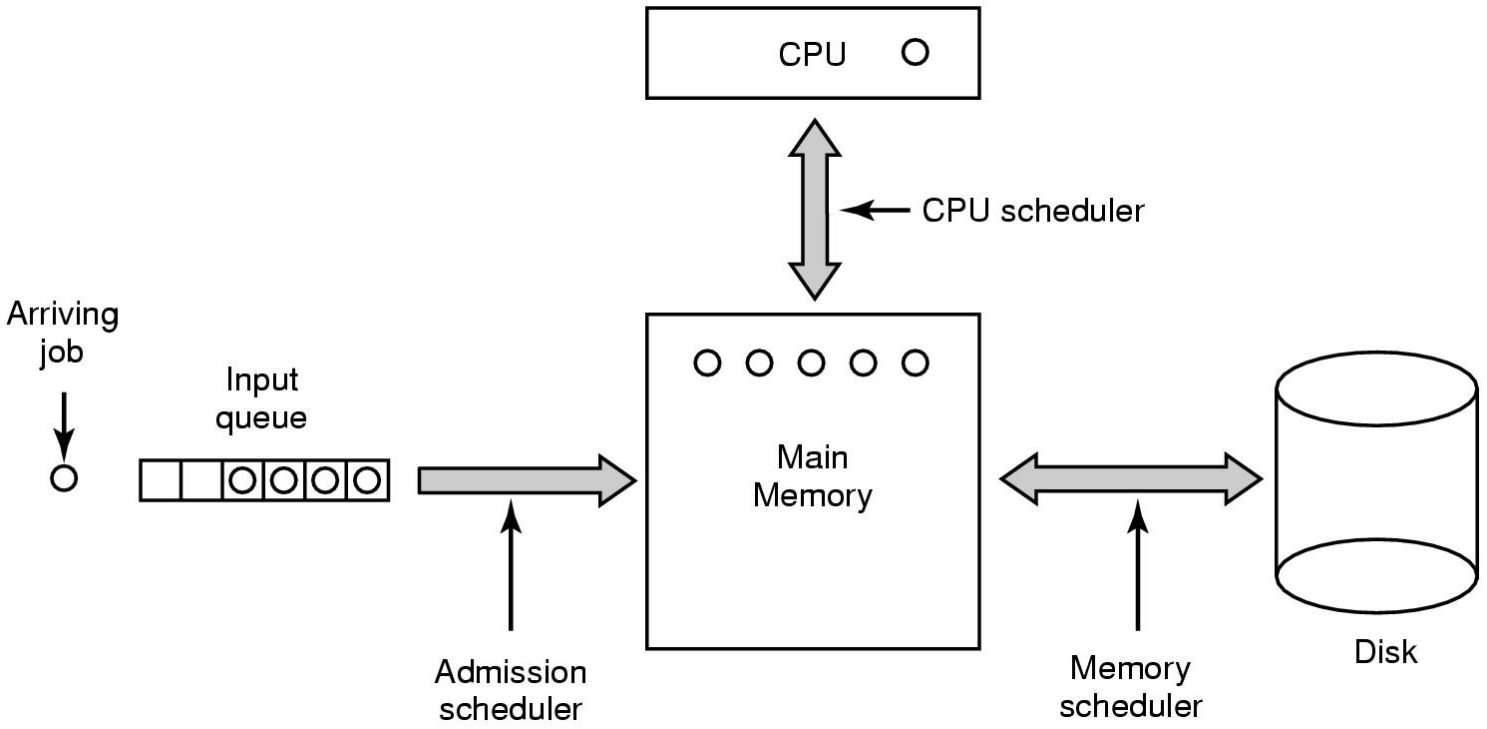
4 job A,B,C,D con tempi di esecuzione a, b, c, d

- 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

Scheduling nei sistemi Batch (3)



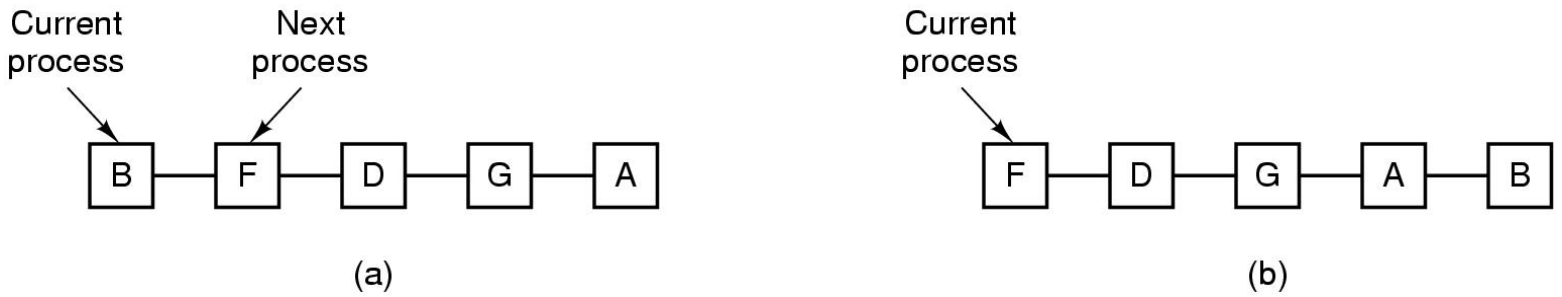
Tre livelli di scheduling

Scheduling nei sistemi Batch (4)

- Admission scheduler
  - decide quali job (sottomessi, memorizzati su disco) ammettere nel sistema (viene creato il processo corrispondente)
- Memory scheduler
  - i job ammessi devono essere caricati in memoria centrale prima di poter essere eseguiti
  - se non tutti i job entrano in MC, il memory scheduler sceglie quali job caricare in memoria e quali tenere su disco (*swapped out*)
- CPU scheduler
  - lo scheduler che abbiamo trattato finora

Scheduling nei sistemi Interattivi

Scheduling *Round Robin* (1)



- (a) lista dei processi pronti
- (b) lista dei pronti dopo che B ha usato il suo *quanto* (*quantum*) di tempo

loop: jmp loop ; (idle)

Scheduling *Round Robin* (2)

- Come fissare il quanto di tempo
  - deve essere abbastanza lungo da ammortizzare il costo di un *context switch* (ordine 1 ms)
  - deve essere abbastanza breve da permettere una risposta veloce agli utenti interattivi
  - in sistemi reali tipicamente 20-50 ms
- RR *non* favorisce i processi I/O bound

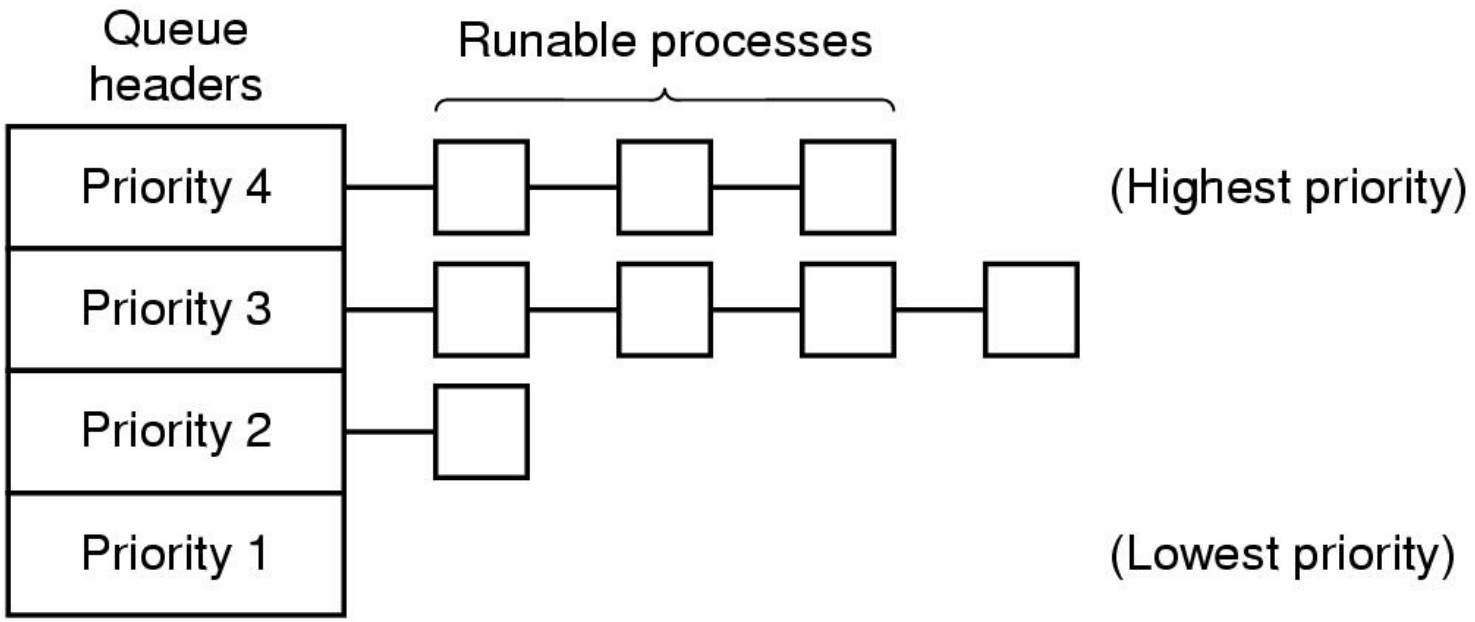
Scheduling con priorità (1)

- Ogni processo ha una priorità
- Ogni volta va in esecuzione il processo a priorità più elevata
- Punti chiave :
  - come assegnare le priorità (statiche, dinamiche…)
  - come evitare attesa indefinita della CPU nei processi a priorità più bassa
  - come individuare i processi I/O bound

Scheduling con priorità (2)

- Molte strategie per il calcolo della priorità
- Tipicamente :
  - priorità dinamica (es. più elevata per i processi che passano da blocked a ready)
  - 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)
  - decrescente nel tempo per i processi che rimangono ready (es. per impedire l'attesa indefinita)

Scheduling con Code multiple (1)



Esempio di algoritmo di scheduling a code multiple con 4 classi di priorità

Scheduling con Code multiple (2)

- Scheduling Round Robin all'interno della classe con priorità più elevata
- I processi che usano tutto il quanto di tempo più di un certo numero di 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

Shortest Process Next

Tempo stimato per comando usando euristica di *aging*

***Tnew*= a Told+(1-a)T1**

scelgo a=1/2

Tnew=T0

Tnew=1/2(T0)+1/2(T1)=1/2(T0+t1)

Tnew=T0/4+T1/4+T2/2

Tnew=T0/8+T1/8+T2/4+T3/2

.....

Altri approcci

- Scheduling garantito
- Scheduling a lotteria
- Scheduling equo

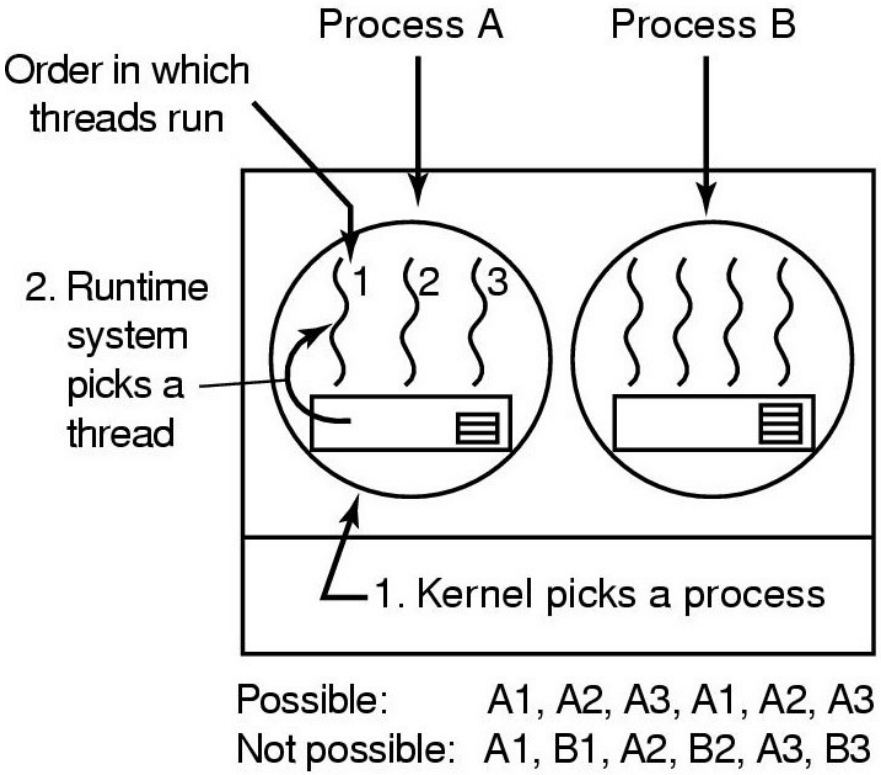
user #1 {A,B,C,D} user #2 {E}

AEBECEDEAEBECEDE...

Politiche e meccanismi

- Separare quello che si può fare da come viene fatto
  - un processo conosce le caratteristiche dei suoi figli e può schedularli in modo più efficace dello scheduler generico presente nel kernel
- Algoritmi di scheduling parametrici
  - il kernel mette a disposizione i meccanismi per la schedulazione in modo parametrico
- I parametri vengono forniti dal processo utente
  - i valori dei parametri fissano la politica da seguire (es: influenzano il calcolo della priorità)

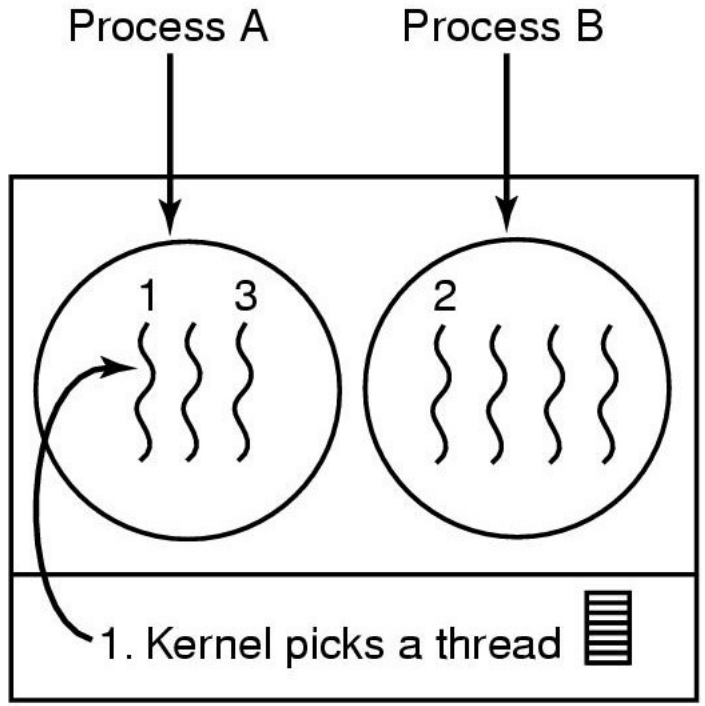
Scheduling dei Thread (1)



Un possibile scheduling per thread user-level

- il quanto di ogni processo è 50-msec
- ogni thread occupa la CPU per 5 msec
- algoritmi più usati: round robin, priorità

Scheduling dei Thread (2)



- Un possibile scheduling per thread kernel-level
- Tipicamente esiste una thread table unica per tutti i processi