

2 – Processi e Thread

Sommario

Processi

modello
operazioni: creazione, chiusura
gerarchie
stati, ciclo di vita
transizioni di stato
descrittore di processo Process Control Block (PCB)
sospensione, ripresa, cambio di contesto
Interrupt
comunicazione tra processi: segnali e messaggi

Thread

modello e uso

Scheduling

Obiettivi

Scheduling di processi: algoritmi
Vari tipi di sistemi
Scheduling di thread

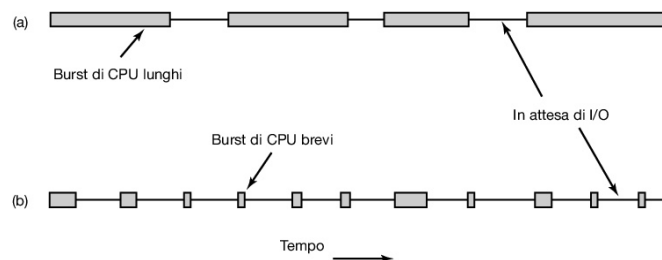
S. Balsamo – Università Ca' Foscari Venezia – SO2.80

Obbiettivi

- Tipi di scheduling
- Scopi dello scheduling del processore
- Scheduling con e senza **prelazione**
- Uso della **priorità** nello scheduling
- Criteri di scheduling
- Algoritmi
- Scheduling con scadenza e real-time

S. Balsamo – Università Ca' Foscari Venezia – SO.5.81

Introduzione allo Scheduling



- Bursts of CPU alternati a periodi di attesa per operazioni di I/O
 - (a) un processo CPU-bound
 - (b) un processo I/O bound

A. Tanenbaum – Modern Operating Systems

S. Balsamo – Università Ca' Foscari Venezia – SO.5.82

Introduzione allo Scheduling

- **Politica** di scheduling del processore
 - Decide **quale processo** viene eseguito ad un certo istante
 - Diversi schedulers possono avere diversi obiettivi
 - Massimizzare il **throughput**
 - Minimizzare la **latenza**
 - Prevenire la starvation (**attesa infinita**)
 - Completare i processi entro una scadenza temporale
 - Massimizzare l'uso del processore

S. Balsamo – Università Ca' Foscari Venezia – SO.5.83

84

Criteri di Scheduling

- Processi *processor-bound*
 - Usa tutto il tempo di CPU disponibile
- Processi *I/O-bound*
 - Genera richieste di I/O velocemente e lascia il processore
- Processi *batch*
 - Richiedono lavoro da eseguire senza l'interazione dell'utente
- Processi *interattivi*
 - Richiede frequenti input dell'utente

S. Balsamo – Università Ca' Foscari Venezia – SO.5.84

85

Tipi di Sistemi – obiettivi dello Scheduling

- Sistemi *Batch*
- Sistemi *Interattivi*
- Sistemi *Real-time*

S. Balsamo – Università Ca' Foscari Venezia – SO.5.85

86

Livelli di Scheduling

- Scheduling di *alto livello*
 - Determina *quale job* può competere per le risorse
 - Controlla il *numero* di processi nel sistema ad un dato tempo
 - Livello di *multiprogrammazione*
- Scheduling di *livello intermedio*
 - Determina *quali processi* possono competere per l'uso del processore
 - *Risponde* a *fluttuazioni* del carico del sistema
- Scheduling di *basso livello*
 - Assegna le *priorità*
 - *Assegna* i processori ai processi

S. Balsamo – Università Ca' Foscari Venezia – SO.5.86

87

Livelli di Scheduling



© Deitel & Ass. Inc.

S. Balsamo – Università Ca' Foscari Venezia – SO.5.87

88

Scheduling con e senza prelazione

- Processi soggetti a **prelazione**
 - Possono essere rimossi dall'attuale processore
 - Si può avere un miglioramento del **tempo di risposta**
 - Importante per ambienti **interattivi**
 - I processi soggetti a prelazione rimangono in memoria
- Processi **non** soggetti a **prelazione**
 - Eseguiti fino al completamente o fino a quanto utilizzano il processore
 - Processi non importanti possono bloccare indefinitamente altri più importanti

S. Balsamo – Università Ca' Foscari Venezia – SO.5.88

89

Priorità

- Priorità **statica**
 - La priorità assegnata ad un processo non cambia
 - **Facile** da implementare
 - **Basso** overhead
 - **Non reattiva** a variazioni dell'ambiente
- Priorità **dinamica**
 - Reattiva a cambiamenti
 - Favorisce una certa **interattività**
 - Richiede **maggior overhead** della statica
 - Giustificato dalla maggior capacità di reagire

S. Balsamo – Università Ca' Foscari Venezia – SO.5.89

90

Obbiettivi dello Scheduling

- Diversi **obbiettivi** dipendono dal tipo sistema
 - Massimizzare il **throughput** molto rilevante in sistemi **batch**
 - Massimizzare il **numero dei processi interattivi** che ricevono un tempo di risposta accettabile
 - Minimizzare il **tempo di risposta** (turnaround) anche in sistemi **batch**
 - Massimizzare l'**uso** delle risorse (utilizzazione)
 - Evitare l'**attesa infinita**
 - Forzare **priorità**
 - Minimizzare l'**overhead**
 - Garantire la **predicibilità**

S. Balsamo – Università Ca' Foscari Venezia – SO.5.90

91

Obbiettivi dello Scheduling

- Diversi obbiettivi comuni a molti scheduler per sistemi **generali**
 - **Equità (Fairness)**
ogni processo riceve la CPU in modo equo
 - **Predicibilità**
la politica dichiarata deve essere attuata
 - **Bilanciamento**
impegnare tutte le parti del sistema

S. Balsamo – Università Ca' Foscari Venezia – SO.5.91

Obbiettivi dello Scheduling

92

Diversi obiettivi per

- Sistemi **Batch**
 - **Throughput**
massimizzare il numero di processi serviti per unità di tempo
 - **Tempo di turnaround**
minimizzare il tempo totale di residenza nel sistema
 - **Utilizzo della CPU**
massimizzare l'uso del processore
- Sistemi **Interattivi**
 - **Tempo di risposta** minimizzare
 - **Adeguatezza** alle richieste e aspettative degli utenti
- Sistemi **Real-time**
 - **Scadenze** rispettarle
 - **Prevedibilità** mantenere la qualità del servizio

S. Balsamo – Università Ca' Foscari Venezia – SO.5.92

Algoritmi di Scheduling

93

Decidono **quando** e **quanto a lungo** porre in esecuzione ogni processo

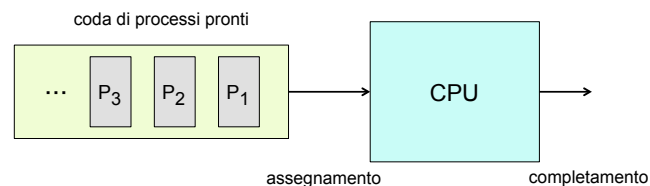
- alla creazione di un processo figlio chi eseguire
- alla terminazione di un processo quale altro processo eseguire
- se un processo si blocca quale altro processo eseguire (relazioni)
- alla gestione di interrupt
- Fa scelte su
 - **Prelazione**
 - **Priorità**
 - **Tempo** di esecuzione
 - Tempo fino al completamento
 - **Equità**

S. Balsamo – Università Ca' Foscari Venezia – SO.5.93

Scheduling **First-In-First-Out (FIFO)**

94

- Scheduling **FIFO**
 - Lo schema più **semplice**
 - I processi sono trattati in base al tempo di arrivo
 - Senza prelazione
 - Utilizzato raramente come algoritmo principale di scheduling



S. Balsamo – Università Ca' Foscari Venezia – SO.5.94

Shortest-Job-First (SJF) Scheduling

95

- Scheduler seleziona il processo con il **minimo tempo per terminare stimato**
 - Tempo media di attesa minore di FIFO
 - Riduce il numero di processi in attesa
 - Potenzialmente larga **varianza** del **tempo di attesa**
- **Senza prelazione**
 - Può portare a tempi di risposta lenti a richieste interattive
- Si **basa** sulla **stima del tempo** per completare l'esecuzione
 - Potrebbe essere inaccurata o falsata
 - Correzioni possibili
 - Tutti i tempi devono essere disponibili
- Non sempre adatto per moderni sistemi interattivi

S. Balsamo – Università Ca' Foscari Venezia – SO.5.95

96

Shortest-Remaining-Time- First (SRT) Scheduling

- SRT scheduling
 - Versione **con prelazione** di SJF
 - I processi più corti in arrivo effettuano prelazione sui processi in esecuzione
 - Varianza del tempo di risposta molto grande**: i processi lunghi aspettano ancora di più che non con SPF
 - Teoricamente **ottimo** per il **tempo media di attesa**
 - Non sempre ottimale in pratica
 - I processi in arrivo corti possono effettuare prelazione su processi quasi completati
 - Overhead** di cambio di contesto che può diventare significativo

S. Balsamo – Università Ca' Foscari Venezia – SO.5.96

97

Scheduling in sistemi **batch**

8	4	4	4
A	B	C	D

Esecuzione di quattro lavori
nell'ordine di arrivo

4	4	4	8
B	C	D	A

Esecuzione di quattro lavori
In ordine di tempo di esecuzione
crescente

Un esempio di scheduling shortest job first **SJF**

A. Tanenbaum – Modern Operating Systems
S. Balsamo – Università Ca' Foscari Venezia – SO.5.97

98

Scheduling **Round-Robin** (RR)

- Scheduling **Round-Robin**
 - Basato su FIFO
 - I processi son eseguiti solo per un **periodo di tempo limitato** detto intervallo o **quanto** di tempo
 - Con prelazione
 - Facile da implementare
 - Richiede al sistema di mantenere parecchi processi in memoria per minimizzare l'overhead
 - Spesso utilizzato come parte di algoritmi più complessi
 - Spesso usato per sistemi **interattivi**

completamento

assegnamento

S. Balsamo – Università Ca' Foscari Venezia – SO.5.98

99

Scheduling **Round-Robin**

Scheduling Round Robin in sistemi **interattivi**

lista di processi eseguibili

Processo attuale

Processo successivo

Processo attuale

lista di processi eseguibili dopo che
il processo B ha usato il suo quanto

A. Tanenbaum – Modern Operating Systems

Scheduling Round-Robin

100

Cambio di contesto – costo

Definire la **dimensione del quanto**

overhead di cambi contesto
 efficienza ridotta della CPU
 limitare l'attesa in coda
 rispetto alla lunghezza media del burst CPU

es. vantaggi /limiti quanto piccolo:
 quanto grande:

Scheduling Round-Robin

101

- Dimensione del **quanto**
 - Determina il tempo di risposta alle richieste interattive
 - Dimensione del quanto molto **grande**
 - Processi eseguiti per lungo tempo
 - Degenera nella FIFO
 - Dimensione del quanto molto **piccola**
 - Il sistema passa più tempo nel **cambio di contesto** che nell'esecuzione di processi
 - Dimensione del quanto **media**
 - Abbastanza a lungo per processi interattivi per fare richieste I/O
 - I processi batch ancora ottengono maggior la parte del tempo del processore

S. Balsamo – Università Ca' Foscari Venezia – SO.5.101

Scheduling a Priorità

102

Classi di priorità

I processi di classi a maggiore priorità sono eseguiti prima

Priorità

fissa
 dinamica (variabile)

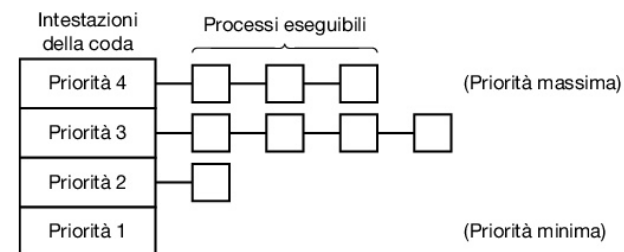
basata sui tempi
 astratta

Combinazione algoritmo di scheduling round-robin e a priorità
 classi di priorità
 round robin all'interno della classe

Potenziale attesa infinita

Es. impostare la priorità dinamica di un processo a $1/f$ dove
 f = frazione dell'ultimo quanto usato dal processo

Scheduling a Priorità in sistemi interattivi



Un algoritmo di scheduling con quattro classi di priorità

A. Tanenbaum – Modern Operating Systems

103

104

Scheduling Selfish Round-Robin (RR)

- Selfish round-robin scheduling
 - Aumenta la priorità con l'età di processo
- Due code
 - Attivo
 - In attesa (Holding)
- Un processo entra nella coda dei processi 'nuovi' (in attesa) e invecchiando la sua priorità aumenta
- Quando la sua priorità è uguale a quella dei processi pronti (attivi) entra nella coda pronti e si applica il RR
- Favorisce i processi più anziani al fine di evitare ritardi irragionevoli
- Possibili diverse velocità di crescita della priorità

S. Balsamo – Università Ca' Foscari Venezia – SO.5.104

105

Highest-Response-Ratio-Next (HRRN) Scheduling

- HRRN scheduling
 - Migliora lo scheduling SJF
 - Ancora senza prelazione
 - Considera anche quanto a lungo un processo ha aspettato
 - Previene l'attesa infinita
- Priorità dinamica = $(\text{tempo di risposta}) / (\text{tempo d'esecuzione})$
- dove
- tempo di risposta = tempo di attesa + tempo d'esecuzione

S. Balsamo – Università Ca' Foscari Venezia – SO.5.105

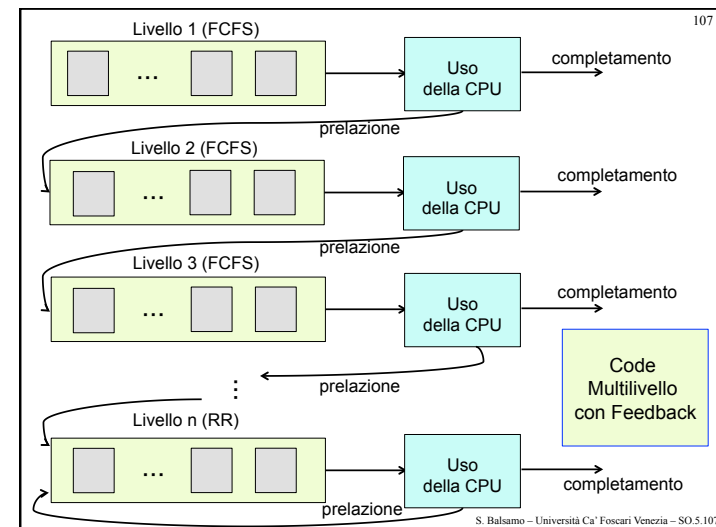
106

Code Multilivello con Feedback

- Diversi processi hanno diverse necessità
 - Processi corti interattivi e I/O-bound in generale dovrebbero essere eseguiti prima di processi processor-bound e batch
 - I modelli di comportamento non sono immediatamente evidenti allo scheduler
- Code multilivello con feedback
 - I processi che arrivano entrano nella coda di più alto livello e sono eseguiti con priorità maggiore rispetto ai processi nelle code inferiori
 - I processi lunghi scendono a livelli più bassi più volte
 - Fornisce maggiore priorità ai processi brevi e I/O-bound
 - I processi lunghi sono eseguiti quando quelli brevi e di I/O-bound sono terminati
 - I processi in ogni coda sono serviti utilizzando FIFO (livelli alti) o round-robin (sempre l'ultimo livello)
 - I processi che entrano in una coda ad alta priorità forzano la prelazione sui processi in esecuzione

S. Balsamo – Università Ca' Foscari Venezia – SO.5.106

107



S. Balsamo – Università Ca' Foscari Venezia – SO.5.107

108

Code Multilivello con Feedback

- L'algoritmo deve **rispondere ai cambiamenti** dell'ambiente
 - Sposta i processi in altre code quando alternano tra il comportamento interattivo e batch
- Esempio di un meccanismo di adattamento
 - Meccanismi **adattivo** richiedono un maggior **overhead** che spesso viene compensato da una maggiore sensibilità ai cambiamenti di comportamento dei processi

S. Balsamo – Università Ca' Foscari Venezia – SO.5.108

109

Scheduling **Fair Share** (FSS)

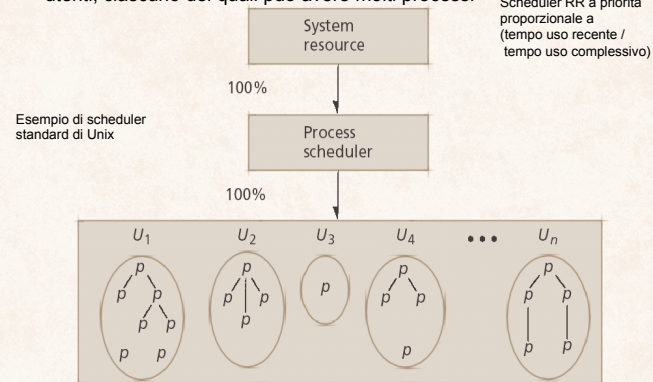
- FSS controlla l'accesso degli **utenti** alle risorse
 - Alcuni **gruppi di utenti** più importanti di altri
 - Processi appartenenti ad un utente (gruppo)
 - Assicura che i gruppi meno importanti non possano monopolizzare le risorse (**equità**)
 - Le **risorse** inutilizzate sono **distribuite** secondo la **proporzione** delle risorse già assegnata ad ogni **gruppo**
 - I gruppi che non soddisfano gli obiettivi della utilizzazione delle risorse ottengono una priorità maggiore

S. Balsamo – Università Ca' Foscari Venezia – SO.5.109

110

Scheduling **Fair Share**

Scheduler standard UNIX. Lo scheduler assegna il processore agli utenti, ciascuno dei quali può avere molti processi



(Property of AT&T Archives. Reprinted with permission of AT&T.)

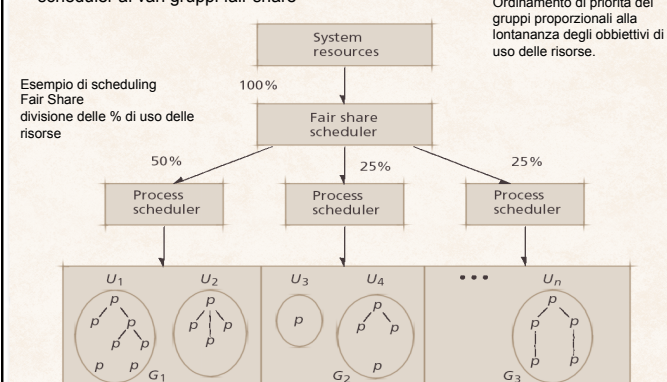
© Detel & Ass. Inc.

S. Balsamo – Università Ca' Foscari Venezia – SO.5.110

111

Scheduling **Fair Share**

Fair share scheduler: divide le **risorse** del sistema in **parti**, poi allocate dallo scheduler ai vari gruppi fair share



(Property of AT&T Archives. Reprinted with permission of AT&T.)

© Detel & Ass. Inc.

S. Balsamo – Università Ca' Foscari Venezia – SO.5.111

Scheduling per sistemi Real-Time

Sistemi con vincoli temporali di real-time, a scadenza

- Real-time scheduling
 - Riguarda lo scheduling a scadenza
 - I processi hanno **vincoli temporali**
 - Comprende anche le attività che vengono eseguiti **periodicamente**
- Due categorie
 - **Soft real-time** scheduling
 - Non garantisce che i vincoli temporali siano soddisfatti
 - Esempio: riproduzione multimediale
 - **Hard real-time** scheduling
 - I vincoli temporali devono essere sempre soddisfatti
 - Il mancato rispetto di scadenza potrebbe avere risultati catastrofici
 - Periodici o asincroni
 - Esempio: controllo del traffico aereo

112

Scheduling a scadenza (deadline)

- Deadline scheduling
 - I processi devono essere completati **entro un tempo stabilito**
 - Usato quando i risultati sarebbero inutili se non consegnati in tempo
 - **Difficile** da implementare
 - Deve **conoscere i requisiti** delle risorse di anticipo
 - Comporta notevole **overhead**
 - Il servizio offerto ad altri processi può degradare
 - Teoricamente: allocare prima i processi a scadenza più vicina

S. Balsamo – Università Ca' Foscari Venezia – SO.5.113

Real-Time Scheduling

- Eventi per sistemi real-time
 - **periodici** regolari
 - **non periodici** imprevedibili
- dati
 - m eventi periodici
 - L'evento i avviene nel periodo P_i e richiede C_i sec di CPU
- allora si può gestire il carico solo se

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

- e il sistema è **schedulabile**
- **Algoritmi statici / dinamici** prima o durante l'esecuzione

S. Balsamo – Università Ca' Foscari Venezia – SO.5.114

Real-Time Scheduling

- real-time scheduling **statico**
 - Non adeguare le priorità nel corso del tempo
 - Basso overhead, semplice
 - Adatto per sistemi dove le condizioni cambiano raramente
 - Hard real-time schedulers
 - **Rate-monotonic (RM)** scheduling
 - Aumenta la priorità del processo monotonicamente con la frequenza con cui deve essere eseguito
 - Round-robin, con prelazione e priorità
 - Favorisce i processi periodici eseguiti spesso
 - Deadline RM scheduling
 - Utile per un processo periodico che ha una scadenza diversa dal suo periodo

S. Balsamo – Università Ca' Foscari Venezia – SO.5.115

Real-Time Scheduling

- real-time scheduling **dinamico**
 - Regola priorità in risposta alle condizioni variate in esecuzione
 - Può portare ad un overhead significativo, e deve garantire che l'overhead non porti ad un aumento di scadenze mancate
 - Le priorità sono di solito basate sulle scadenze dei processi
 - **Earliest-deadline-first (EDF)**
 - Con prelazione
 - Sceglie sempre il processo con la **scadenza più vicina**
 - Massimizzazione del throughput e minimizzare il tempo di attesa
 - **Minimum-laxity-first**
 - Simile a EDF, ma basa la priorità sulla lassità
 - Lassità: misura il tempo alla scadenza del processo e il suo tempo rimanente di esecuzione (C)
 - $L = D - (T + C)$ D *deadline*, T tempo corrente
 - disponibilità delle informazioni

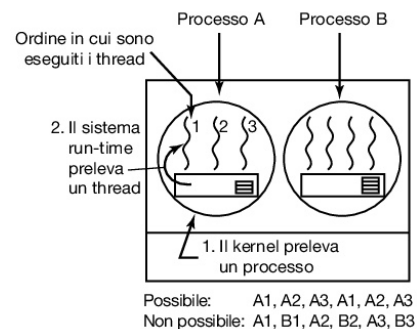
S. Balsamo – Università Ca' Foscari Venezia – SO.5.116

Politica e meccanismo di scheduling

- Separazione fra
 - **meccanismo** di scheduling come
 - **politica** di scheduling cosa è permesso
- un processo conosce la rilevanza dei thread figli e chi necessita di maggior priorità
- Occorrerebbe usare algoritmi di Scheduling parameterizzati
 - il meccanismo risiede solitamente nel nucleo
- I parametri dovrebbero essere scelti dai processi utente
 - La politica definita dai processi utente

117

Scheduling di Thread

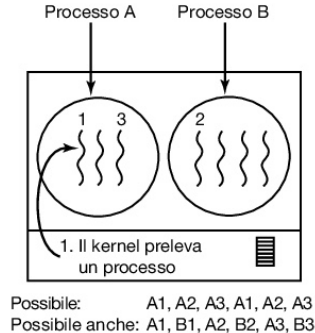
Possibile **scheduling di thread a livello utente**

- quanto di processo 50-msec
- threads eseguito per 5 msec per ogni CPU burst

A. Tanenbaum – Modern Operating Systems

118

Scheduling di Thread

Possibile **scheduling di thread a livello nucleo**

- quanto di processo 50-msec
- threads eseguito per 5 msec per ogni CPU burst

A. Tanenbaum – Modern Operating Systems

119