Scheduling della CPU

Concetti fondamentali

> Lo scheduling è importante perché una sbagliata gestione dei core e dei processi porta a problemi per tutto il sistema

Un concetto fondamentale è quello dei punti di vista → una CPU deve scegliere il processo che richiede meno risorse per liberare quanto prima la memoria e poter accettare nuovi processi.

$T_a = T_f - T_i - D$ Tempo di attesa = Tempo finale - Tempo iniziale - Durata

Lo scopo finale è quello di massimizzare l'esecuzione, quindi avere sempre processi in esecuzione

Esecuzione del processo

Comincia con una sequenza di operazioni d'elaborazione svolte dalla CPU (CPU Burst), seguita da una sequenza di operazioni di I/O (I/O Burst). Queste operazioni si ripetono ciclicamente.

Scheduling della CPU

Ogni volta che la CPU passa per lo stato di inattività, il sistema operativo scegli per l'esecuzione uno dei processi presenti nella coda dei processi pronti

Le decisioni dello scheduling si prendono nelle seguenti circostanze:

1. Un processo passa da esecuzione ad attesa
2. Un processo passa da esecuzione a pronto
3. Un processo passa da attesa a pronto
4. Un processo termina

Nei casi 1. e 4. si dice che lo schema di scheduling è non-preemptive, negli altri due casi è preemptive

Dispatcher

Il dispatcher è un modulo che passa il controllo della CPU ai processi scelti dallo scheduler a breve termine.

Latenza di dispatch → Il tempo richiesto dal dispatcher per fermare un altro processo e avviare una nuova esecuzione

---

Criteri di scheduling

Si deve trovare un compromesso che garantisca:

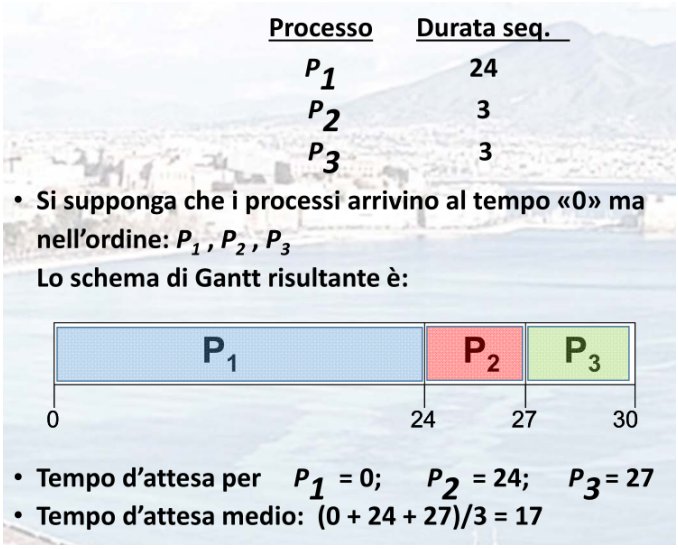
- Massimo utilizzo della CPU
- Massima produttività
- Minimo tempo di completamente
- Minimo tempo di attesa
- Minimo tempo di risposta

Algoritmi di scheduling

Scheduling First-Come, First-Served (FCFS)

Supponiamo che 3 processi P1, P2, P3 arrivino al tempo <<0>> nell'ordine P1→P2→P3

Creiamo uno schema di Gantt e calcoliamo il tempo di attesa medio:



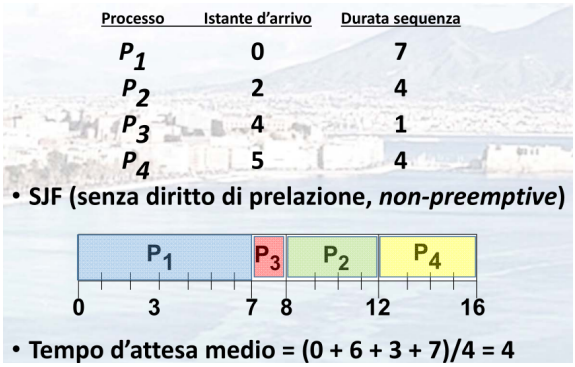
Nel caso in cui i processi arrivino nell'ordine P2→P3→P1, il tempo di attesa medio sarà:
$(6+0+3)/3=3$ considerando come tempi di attesa $P_1=6, P_2=0, P_3=3$
Seguendo lo schema di Gantt conseguente:

Scheduling Shortest Job First (SJF)

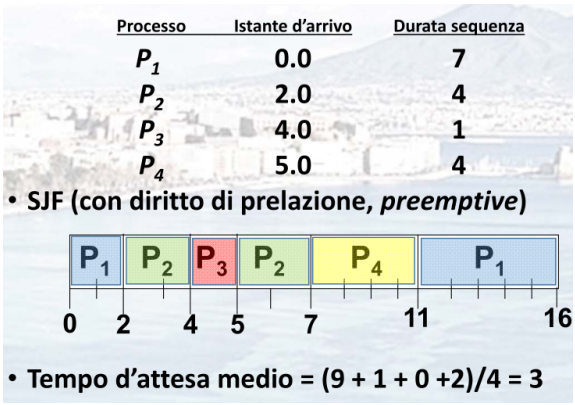
Associa a ogni processo la lunghezza della successiva sequenza di operazioni della CPU.

Ci sono due schemi:

- Senza prelazione → permette al processo correntemente in esecuzione di portare a termine le propria sequenza di operazioni della CPU



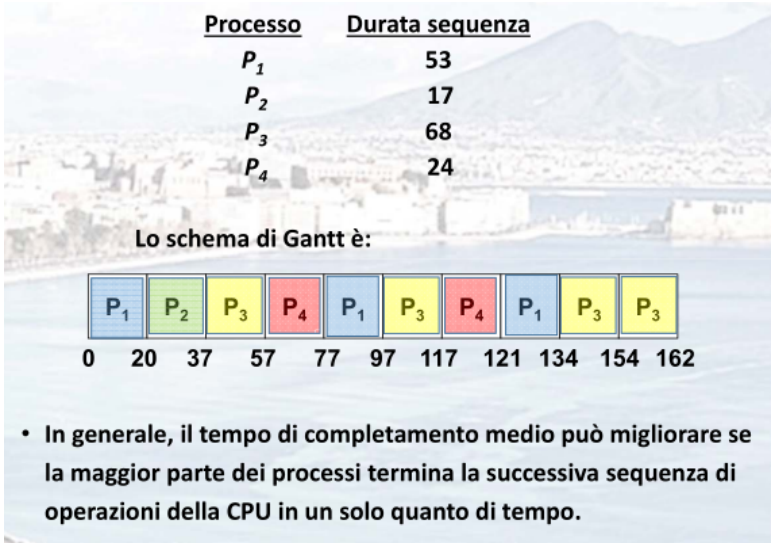
- Con prelazione → se arriva un nuovo processo con sequenza di CPU inferiore a quella del tempo restante per eseguire il processo in costo, lo sostituisce al processo attualmente in esecuzione



Questo algoritmo è ottimale nel senso che rende minimo il tempo d'attesa medio per un insieme di processi.

Scheduling circolare Round Robin (RR)

Ciascun processo riceve una quantità di tempo fissata (time quantum), che varia da 10 a 100ms e la cosa dei processi pronti è trattata come una coda circolare



Se q (quanto) è molto lungo l'algoritmo diventa uguale al FCFS

Se q è breve (es: 1microsec) la CPU viene condivisa (processor sharing)

Scheduling a code multiple

Si fa molta distinzione tra processi eseguiti in primo piano (o interattivi) e quelli in sottofondo (o a lotti). Questi due processi possono avere diverse necessità di scheduling:

- In primo piano → Round Robin
- In sottofondo → First Come First Served

Risulta sempre necessario avere uno scheduling tra le code:

- Se si ha uno scheduling con priorità fissa e prelazione si può incorrere in una starvation.
- Si possono quindi impostare i quanti di tempo, quindi per ogni coda si stabilisce una parte del tempo di elaborazione della CPU.

Scheduling a code multiple con retroazione

La retroazione (feedback) permette ai processi di spostarsi fra le code. In questo modo si attua una forma di invecchiamento che impedisce il verificarsi di un'attesa indefinita.

Caratterizzato dai seguenti parametri:

- numero di code
- algoritmo di scheduling per ogni coda
- metodo usato per determinare quando spostare un processo in una coda con priorità maggiore
- metodo usato per determinare quando spostare un processo in una coda con priorità minore
- Metodo usato per determinare in quale coda si deve mettere un processo quando richiede servizio

---

Scheduling per sistemi con più unità di elaborazione

Lo scheduling della CPU diventa più complesso nei sistemi con più univtà di elaborazione.

- Sistemi omogenei → Sistemi nei quali le unità di elaborazione sono identiche (in relazione alle loro funzioni)
- Multielaborazione asimmetrica → Tutte le attività del sistema sono gestite da un'unica unità di elaborazione, che quindi assume il ruolo di unità centrale

Scheduling per sistemi di elaborazione in tempo reale

- Sistemi in tempo reale stretto → Sistemi real-time hard → Sistemi capaci di garantire il completamento delle funzioni critiche entro un tempo definito, oltre il quale sarebbe innutile
- Sistemi in tempo reale debole → Sistemi real-time soft → Sistemi dei quali i processi critici hanno una priorità maggiore dei processi ordinari

Valutazione degli algoritmi

- Modello deterministico → Valutazione analitica che considera un carico di lavoro predeterminato e definisce le presstazioni di ciascun algoritmo per quel carico di lavoro
- Reti di code → Se sono noti le distribuzione degli arrivi e dei servizi si possono calcolare l'utilizzo, la lunghezza media delle code, il tempo medio di attesa, etc...
- Simlazioni e realizzazione

← Capitolo 5: Thread

Capitolo 7: Sincronizzazione dei

Processi →