

Architettura dei sistemi di elaborazione

A. Simonetta¹



¹Dipartimento di Informatica
Università degli Studi di Roma «Tor Vergata»

13 maggio 2024



Introduzione
ai Sistemi
operativi

Processi e
thread

Comunicazione
tra processi
(IPC)

Scheduling

Gestione della
memoria

File System

Input e
Output

Deadlock



Introduzione
ai Sistemi
Operativi

A. Simonetta

Introduzione
ai Sistemi
operativi

Processi e
thread

Comunicazione
tra processi
(IPC)

Scheduling

Gestione della
memoria

File System

Input e
Output

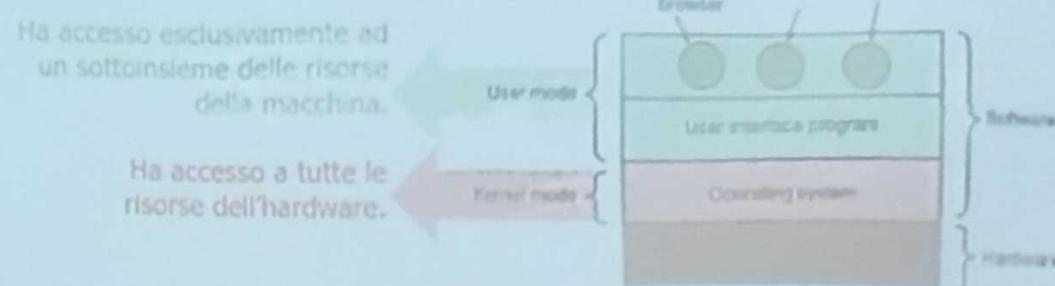
Deadlock

Argomenti

- 1 Introduzione ai Sistemi operativi
- 2 Processi e thread
- 3 Comunicazione tra processi (IPC)
- 4 Scheduling
- 5 Gestione della memoria
- 6 File System
- 7 Input e Output
- 8 Deadlock

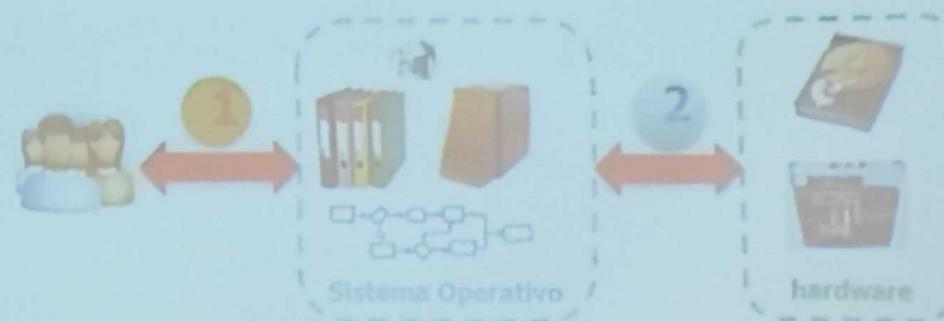
Cosa è un sistema operativo?

- I Sistema Operativo (SO) è quello strato software che avvolge la macchina e che permette l'utilizzo ottimizzato ed astratto delle risorse che la compongono
- Un errore comune è considerare la GUI (Graphical User Interface) come parte del SO, in realtà è solo un'applicazione che lo utilizza.
- La maggior parte dei computer ha due modalità operative: **kernel** (o **supervisor**) e **user**.
- È difficile dare una definizione formale per il SO, al di là della semplice affermazione «*quel sw che gira in modalità kernel*» che poi non è sempre vera.



Cosa è un sistema operativo?

- Il SO svolge due distinte funzioni:
 - fornisce ai programmati ed alle applicazioni sw un insieme di risorse astratte invece del complicato hardware;
 - gestisce le risorse hardware.
- Considereremo il sistema Operativo quindi come
 - una macchina estesa
 - un gestore di risorse



Introduzione
ai Sistemi
Operativi

A. Simonetta

Introduzione
ai Sistemi
operativi

Processi e
thread

Comunicazione
tra processi
(IPC)

Scheduling

Gestione della
memoria

File System

Input e
Output

Deadlock

Hardware di un computer

- La CPU è il «cervello» del computer; ogni CPU ha uno specifico set di istruzioni che è in grado di eseguire.
- Dato che accedere alla memoria richiede tempo, tutte le CPU contengono a proprio interno delle locazioni di memoria dette **registri**.
- La maggior parte delle CPU ha dei registri speciali:
 - il program counter, che contiene l'indirizzo di memoria della prossima istruzione che deve essere caricata
 - lo stack pointer, che punta alla cima dello stack in memoria
 - il Program Status Word (PSW) che contiene il risultato delle istruzioni di confronto, la priorità della CPU, la modalità di accesso (user o kernel) e altri bit di controllo.

Introduzione
ai Sistemi
Operativi

A. Simonetta

Introduzione
ai Sistemi
operativi

Processi e
thread

Comunicazione
tra processi
(IPC)

Scheduling

Gestione della
memoria

File System

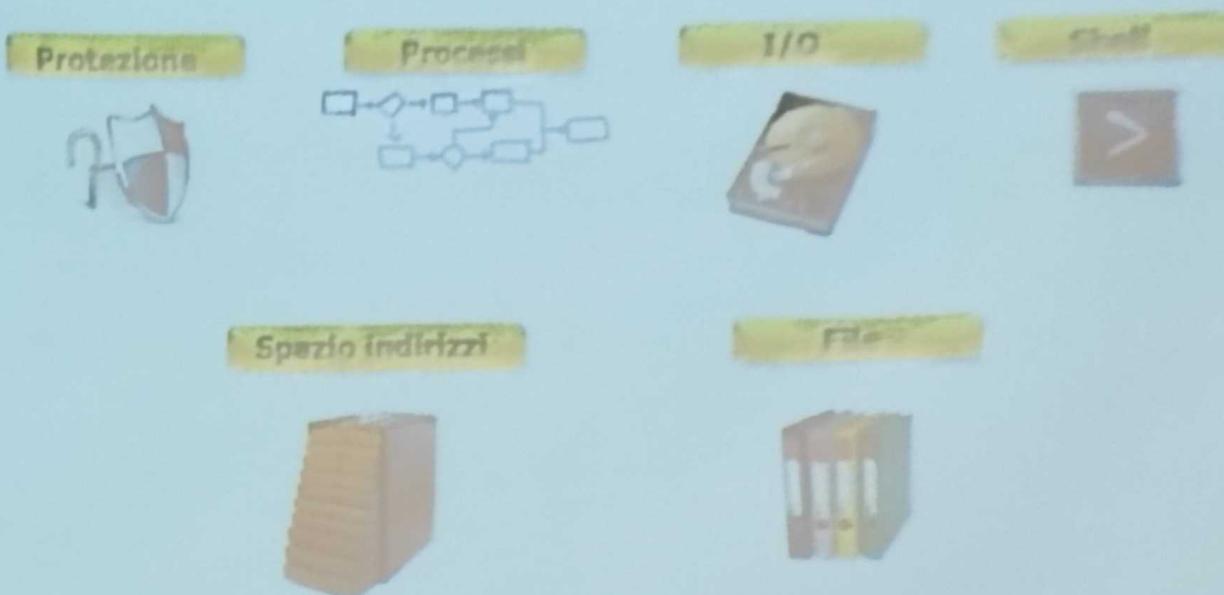
Input e
Output

Deadlock

Astrazione del Sistema Operativo

- I sistemi operativi forniscono concetti di base e astrazioni:

- processi
- file
- spazio degli indirizzi
- dispositivi di I/O
- protezione dei dati
- shell



Introduzione
ai Sistemi
Operativi

A. Simonetta

Introduzione
ai Sistemi
operativi

Processi e
thread

Comunicazione
tra processi
(IPC)

Scheduling

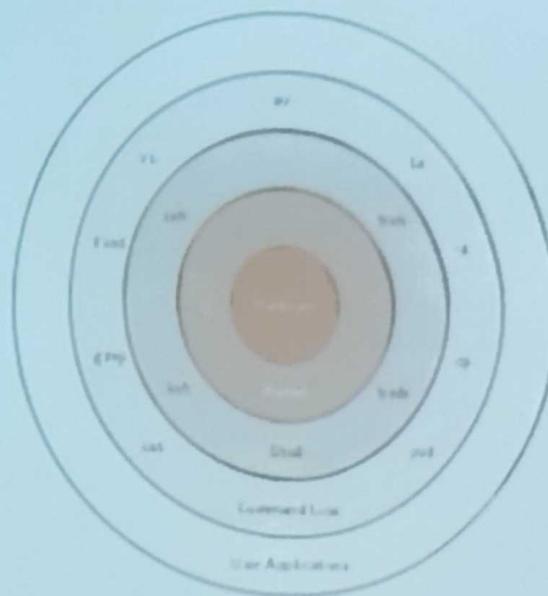
Gestione della
memoria

File System

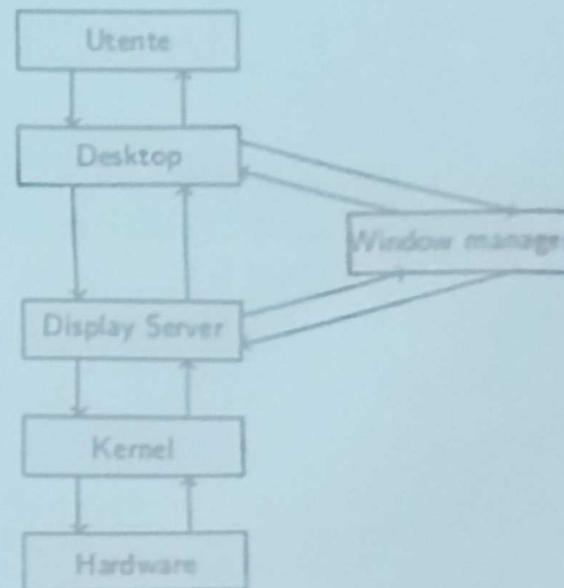
Input e
Output

Deadlock

- ## Shell
- Il SO è il codice che realizza le chiamate di Sistema.
 - Gli editor, i compilatori, gli assemblatori, i linker e l'interprete dei comandi (la shell in UNIX) non fanno parte del SO (vengono definiti software di base).
 - La GUI (Graphic User Interface) è un'applicazione che sta sopra al SO proprio come l'interprete dei comandi, ma non fa parte del SO.



Stratificazione shell



Stratificazione GUI

Introduzione
ai Sistemi
Operativi

A. Simonetta

Introduzione
ai Sistemi
operativi

Processi e
thread

Comunicazione
tra processi
(IPC)

Scheduling

Gestione della
memoria

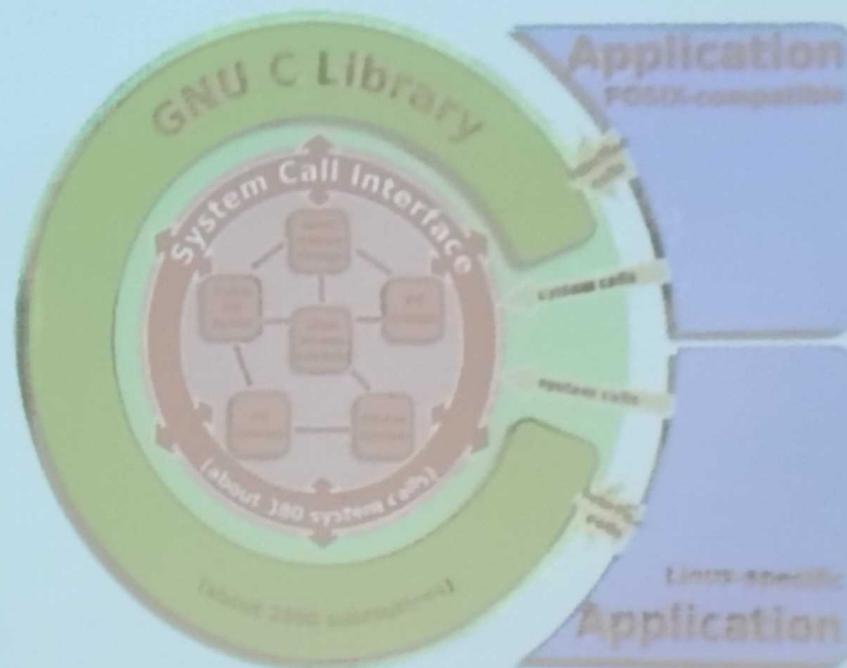
File System

Input e
Output

Deadlock

API e chiamate di sistema

- I SO hanno due funzioni principali:
 - forniscono astrazioni ai programmi utente,
 - gestiscono le risorse del computer.
- Per fornire l'accesso a queste astrazioni ai programmi utente, il SO mette a disposizione **API** (Application Program Interfaces) e **chiamate di sistema**, ossia procedure di interfaccia che invocano specifici servizi (API) o procedure speciali che vengono eseguite in modalità kernel (chiamate di sistema)

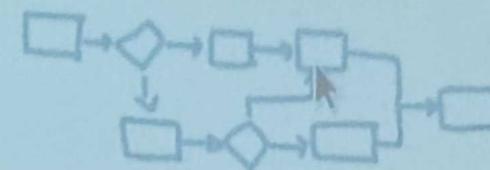


Il linguaggio C

- I sistemi operativi sono normalmente scritti in C (o, talvolta, in C++) in quanto:
 - fornisce la possibilità di usare puntatori, costrutti estremamente potenti ma allo stesso tempo molto delicati: non avendo controlli è facile indirizzare la memoria in modo improprio e commettere errori (motivo per il quale un programmatore java non può utilizzarli, se non implicitamente),
 - La gestione della memoria in C è statica o allocata in modo esplicito (con la funzione `malloc()`) e rilasciata dal programmatore al termine del lavoro svolto (con la funzione `free()`).
 - i sistemi operativi sono dei veri e propri *sistemi real-time* utilizzati per scopi generali, quindi l'esistenza di un *garbage collector* che entra in azione in modo arbitrario non è ammissibile,
 - è un linguaggio che ha un livello concettuale non troppo distante dalla macchina fisica.

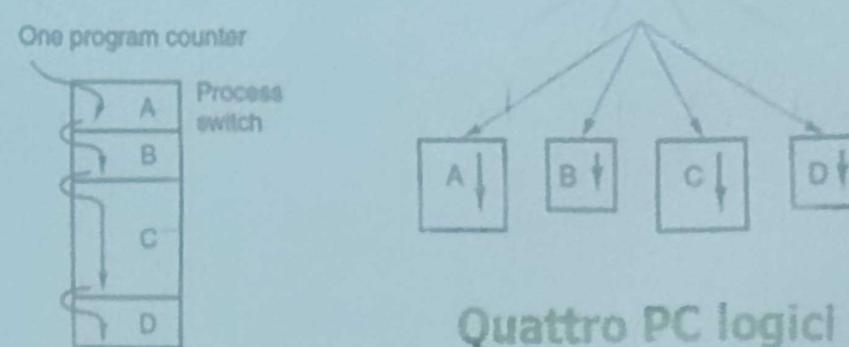
I processi

- Tutti i computer spesso fanno molte cose allo stesso tempo: per gestire questa attività è necessario un sistema multiprogrammato con diversi processi (sistema multitasking).
- Il concetto centrale in qualsiasi sistema operativo è il processo: l'astrazione di un programma in esecuzione.



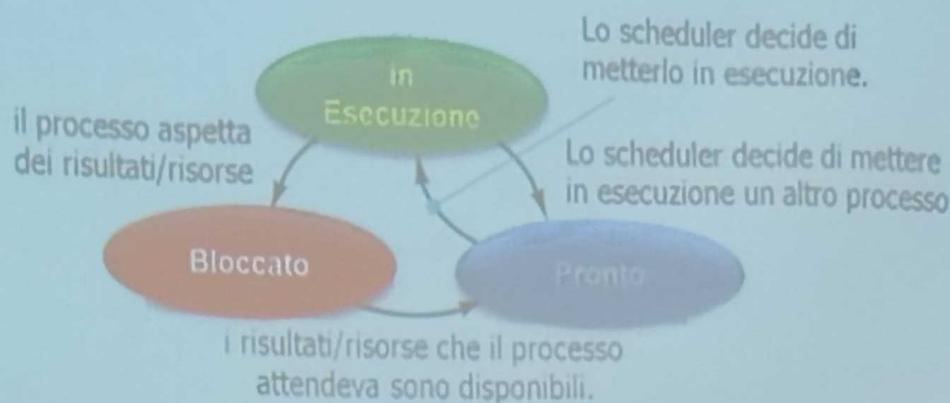
- In un sistema multiprogrammato, la CPU cambia da un processo a un altro velocemente ($10 \div 100$ ms)
- La CPU dà l'illusione di eseguire diversi processi in maniera concorrente (pseudo-parallelismo), in contrasto con il vero parallelismo hardware di un sistema multiprocessore (che ha più CPU che condividono la stessa memoria fisica).
- Grazie a questo concetto gli utenti possono immaginare che ci siano più operazioni concorrenti anche se dispone di una sola CPU (time sharing).

- Tutto il software eseguibile sul computer è organizzato in un numero di **processi sequenziali o processi**.
- Un processo è un'istanza di un programma in esecuzione, che include il valore corrente del **Program Counter**, stack, registri e variabili.
- Ogni processo ha la sua CPU virtuale, in realtà la CPU fisica commuta il controllo tra processi (sistema multiprogrammato)



Stati del processo

- Ogni processo è un'entità indipendente con il suo Program Counter e il suo stato interno.
- I processi spesso hanno bisogno di interagire tra loro.
- Un processo può generare dell'output che un altro processo può usare come input.
- Un processo può trovarsi in uno dei seguenti stati:
 - **Esecuzione:** il processo sta usando la CPU in quell'istante
 - **Pronto:** il processo è in attesa di poter usare la CPU
 - **Bloccato:** il processo non può essere eseguito per qualche motivazione esterna





Segnali

- Un processo in esecuzione può ricevere dei segnali di allarme che, analogamente agli interrupt nel caso hardware, ne bloccano l'esecuzione, causano il salvataggio dello stato e fanno sì che il processo esegua una procedura di gestione dei segnali speciali.
 - A termine della procedura il processo potrà riprendere l'esecuzione dal punto dove si era interrotto.
- Molti trap rilevati dall'hardware, come l'esecuzione di una istruzione illegale o di un indirizzo errato, sono convertiti in segnali per il processo responsabile.
- I segnali possono essere attivati anche in modo temporizzato, come nel caso dell'invio di un messaggio per eseguire un controllo sulla ricezione entro un certo timeout.



Introduzione
ai Sistemi
Operativi

A. Simonetta

Introduzione
ai Sistemi
operativi

Processi e
thread

Comunicazione
tra processi
(IPC)

Scheduling

Gestione della
memoria

File System

Input e
Output

Deadlock

Realizzazione dei processi

- Per implementare il modello di processo, il sistema operativo mantiene una **tabella dei processi** che ha una voce (o riga) per ogni processo.
- La voce è detta **process control block** e mantiene lo stato del processo:
 - il **Program Counter**
 - lo **Stack Pointer**
 - l'allocazione della memoria;
 - lo stato dei file aperti;
 - le informazioni relative alla gestione e allo scheduling.
- Nella voce è memorizzato tutto ciò che serve salvare nello stato bloccato e pronto affinchè sia possibile riavviare il processo come se non si fosse mai fermato.

PC	SP	mem	file	scheduling



Introduzione
ai Sistemi
Operativi

A. Simonetta

Introduzione
ai Sistemi
operativi

Processi e
thread

Comunicazione
tra processi
(IPC)

Scheduling

Gestione della
memoria

File System

Input e
Output

Deadlock

Un modello per la multiprogrammazione

- L'utilizzo della multiprogrammazione migliora l'utilizzo della CPU.
- Se un processo in media esegue calcoli solo per il 20% del tempo in cui risiede in memoria, con 5 processi contemporaneamente in memoria la CPU dovrebbe essere occupata tutto il tempo (100%).
- In realtà i processi attendono anche l'I/O.
- Da un punto di vista probabilistico: se un processo spende una frazione p del suo tempo in attesa dell'I/O, con n processi in memoria la probabilità che stiano tutti aspettando l'I/O è p^n .
- L'utilizzo della CPU è la probabilità complementare: $1 - p^n$.

Introduzione
ai Sistemi
Operativi

A. Simonetta

Introduzione
ai Sistemi
operativi

Processi e
thread

Comunicazione
tra processi
(IPC)

Scheduling

Gestione della
memoria

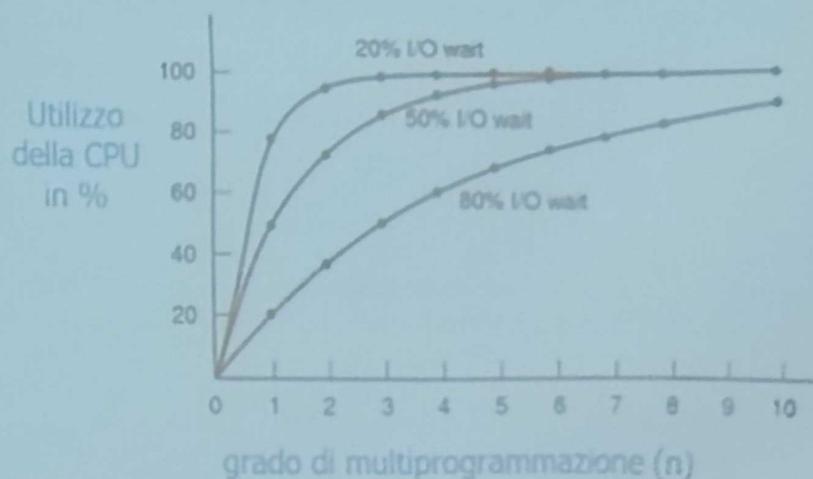
File System

Input e
Output

Deadlock

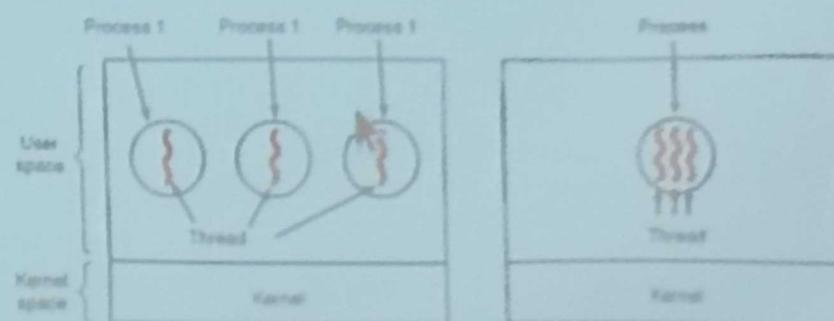
Un modello per la multiprogrammazione

- Se i processi attendono per l'80% del loro tempo in attesa dell'I/O, servono almeno 10 processi per portare la CPU ad un utilizzo del 90%. Questo modello semplificato permette di effettuare previsioni sull'utilizzo della CPU.
- Un computer ha 512 MB di memoria, se il Sistema Operativo occupa 128 MB e un processo 128 MB, può contenere in memoria fino a 3 processi.
- Se l'attesa media dell'I/O è dell'80%, l'utilizzo della CPU è pari a $1 - 0,8^3 = 49\%$. Aggiungendo 512 MB miglioriamo del $79\% = 1 - 0,8^7$



I Thread

- Nei sistemi operativi tradizionali, ogni processo ha uno spazio degli indirizzi e un singolo thread che controlla il flusso del programma.
- A volte è utile avere diversi thread che controllano lo stesso spazio degli indirizzi e che vengono eseguiti in modalità semi-parallela.
- I thread sono come miniprocessi
- I Thread hanno molte caratteristiche che ne determinano il successo:
 - il modello di programmazione diventa più semplice;
 - sono più leggeri dei processi e quindi più semplici e veloci da creare e distruggere;
 - hanno un miglior uso della CPU e ne possono migliorare le prestazioni;
 - sono utili in sistemi con molteplici CPU.



Thread vs Processi

- Quando un processo multithread viene eseguito su un sistema a singola CPU, i thread si alternano nell'esecuzione.
- La CPU passa rapidamente da un thread all'altro, dando l'illusione che i thread siano in esecuzione in parallelo.
- I diversi thread di un processo non sono indipendenti come i diversi processi.
- Poiché tutti i thread hanno lo stesso spazio di indirizzamento (le stesse variabili globali), possono persino leggere/scrivere o cancellare lo stack di un altro thread.
- Tra i thread non c'è protezione perché:
 - è impossibile;
 - non dovrebbe essere necessario.
- Mentre i processi possono essere di proprietà di utenti diversi e possono competere per raggiungere risorse comuni, ogni processo (di proprietà di un singolo utente) può creare più thread che dovrebbero cooperare.

Thread vs Processi

- Un'organizzazione per processi dovrebbe essere utilizzata quando i job non sono correlati.
- Mentre un'organizzazione per thread sarebbe appropriata quando i task fanno parte dello stesso job e collaborano attivamente e strettamente.
- Tutti i thread condividono lo stesso insieme di file aperti, processi figli, allarmi e segnali...

I thread condividono...

Spazio degli indirizzi

Variabili globali

File aperti

Processi figli

Allarmi in corso

Segnali e gestori dei segnali

Informazioni «contabili»

I thread tengono privati...

Program counter

Registri

Stack

Stato

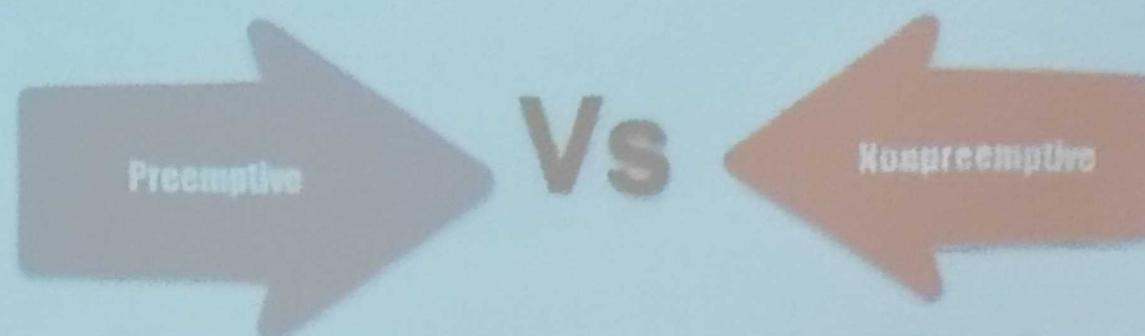


Scheduling

- Quando un computer è multiprogrammato, ha più processi o thread che competono per la CPU allo stesso tempo.
- Questa situazione si verifica ogni volta che due o più di essi si trovano contemporaneamente nello stato di pronto.
- Con una CPU disponibile, solo un processo può essere selezionato per l'esecuzione successiva.
- La parte del sistema operativo che effettua la scelta si chiama **scheduler** e l'algoritmo si chiama **algoritmo di scheduling**.
- Molti dei problemi di schedulazione dei processi possono essere applicati anche alla schedulazione dei thread, sebbene siano molto diversi.
- Quando il kernel gestisce i thread, lo scheduling viene solitamente effettuato per thread, senza tenere conto del processo a cui il thread appartiene.

Tipi di scheduling

- Gli algoritmi di scheduling possono essere suddivisi in due categorie rispetto alla capacità dello scheduling di interrompere l'esecuzione dei processi:
 - ① **non preemptive**: lo scheduler una volta mandato in esecuzione un processo lo lascia andare finché non si blocca (attesa di I/O o di un altro processo) o rilascia spontaneamente la CPU.
 - ② **preemptive**: lo scheduler sceglie un processo da eseguire per un tempo massimo stabilito. Se il processo è ancora in esecuzione al termine dell'intervallo di tempo, lo scheduler lo sospende e sceglie un altro processo da eseguire (se disponibile).





Introduzione
ai Sistemi
Operativi

A. Simonetta

Introduzione
ai Sistemi
operativi

Processi e
thread

Comunicazione
tra processi
(IPC)

Schedulng

Gestione della
memoria

File System

Input e
Output

Deadlock

Obiettivi degli algoritmi di scheduling

- Per progettare un algoritmo di scheduling ci sono obiettivi comuni e altri che dipendono dal tipo di ambiente (batch, interattivo o real-time).
- In ogni caso, l'**equità** (dando ad ogni elaborare una congrua parte della CPU) è molto importante.
- Forzare le **politiche** del sistema è scorretto. In un centro di calcolo reattore nucleare, se la politica locale di sicurezza è che il ciclo di controllo deve prendere 10 msec, lo scheduler deve assicurarsi questa politica venga **sempre** rispettata.
- Tutte le parti del sistema devono essere **impegnate** in modo **bilanciato**: se la CPU e i dispositivi di I/O sono tenuti sempre impegnati vengono eseguiti più lavori al secondo rispetto ad avere componenti inattivi.



Introduzione
ai Sistemi
Operativi

A. Simonetta

Introduzione
ai Sistemi
operativi

Processi e
thread

Comunicazione
tra processi
(IPC)

Scheduling

Gestione della
memoria

File System

Input e
Output

Deadlock

Algoritmi di scheduling

- Aree applicative e sistemi operativi diversi hanno obiettivi diversi e richiedono algoritmi di schedulazione specifici.
- Considereremo tre tipi di ambiente:
 - sistemi **batch**,
 - sistemi **interattivi**,
 - sistemi in **real-time**.

Introduzione
ai Sistemi
Operativi

A. Simonetta

Introduzione
ai Sistemi
operativi

Processi e
thread

Comunicazione
tra processi
(IPC)

Schedulng

Gestione della
memoria

File System

Input e
Output

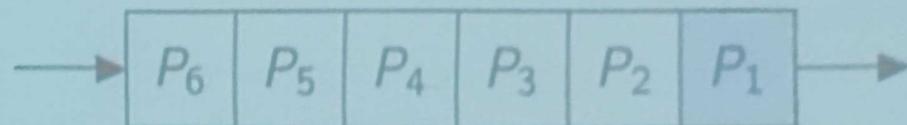
Deadlock

Scheduling nei sistemi batch

- Algoritmi usati nei sistemi batch
 - First-come first-served
 - Shortest job first
 - Shortest remaining time next

First-come first-served

- È il più semplice algoritmo di scheduling non preemptive: il primo che arriva è il primo ad essere servito.



- La CPU è assegnata ai processi in ordine di arrivo; c'è un'unica coda dei processi pronti.
- Questo algoritmo è facile da capire e da programmare.
- Problema:
 - un sistema con processi eterogenei per tempo di esecuzione potrebbe rallentare i processi veloci in assenza di preemption.

Shortest job first

- Si tratta di un algoritmo batch senza preemption in cui i tempi di esecuzione dei processi sono noti in anticipo (la predizione del tempo di esecuzione per job ripetitivi non è difficile): i job entrano nella coda in ordine, ma lo scheduler sceglie quello che termina prima.
- Si supponga che arrivino 4 job P_1 , P_2 , P_3 , e P_4 con la rispettiva stima dei tempi di esecuzione (in minuti):

8	4	4	4
P_1	P_2	P_3	P_4

- Se si facessero entrare in esecuzione nell'ordine di arrivo P_1 avrebbe un tempo di turnaround di 8', P_2 di 12', P_3 di 20' e P_4 di 20' minuti (tempo medio $T_m = 14'$). Usando l'algoritmo shortest job first, i tempi di turnaround diventano 4', 8', 12', and 20' minuti (tempo medio $T_m = 11'$):

4	4	4	8
P_2	P_3	P_4	P_1

- Attenzione: l'algoritmo shortest job first è ottimale solo se i job sono tutti disponibili al momento della scelta.

Shortest Remaining time next

- È una versione preemptive del precedente algoritmo.
- Il tempo di esecuzione è noto a priori.
- Lo scheduler sceglie il processo a cui manca meno tempo al termine dell'esecuzione.
- Se arriva un nuovo job che ha bisogno di meno tempo per terminare rispetto al job corrente, quest'ultimo è sospeso e viene eseguito il nuovo arrivato.



Introduzione
ai Sistemi
Operativi

A. Simonetta

Introduzione
ai Sistemi
operativi

Processi e
thread

Comunicazione
tra processi
(IPC)

Sched^{uling}

Gestione della
memoria

File System

Input e
Output

Deadlock

Scheduling nei sistemi interattivi

- I seguenti algoritmi possono essere utilizzati nei sistemi interattivi, nei PC, nei server e anche in altri tipi di sistemi:
 - Round-Robin Scheduling.
 - Priority Scheduling.
 - Shortest process next.
 - Guaranteed Scheduling.
 - Lottery Scheduling.
 - Fair-Share Scheduling.

Introduzione
ai Sistemi
Operativi

A. Simonetta

Introduzione
ai Sistemi
operativi

Processi e
thread

Comunicazione
tra processi
(IPC)

Scheduling

Gestione della
memoria

File System

Input e
Output

Deadlock

Scheduling run-robin

- È uno degli algoritmi più vecchi, più semplice, più equilibrato e diffusamente utilizzato.
- Ad ogni processo è assegnato un «quantum» di tempo di CPU per l'esecuzione, allo scadere del quale viene interrotto e si passa, con modalità circolare e parietaria, al successivo processo.
- La scelta del «quantum» è un fattore critico: un valore troppo breve provoca troppe interruzioni dei processi causando **overhead**, mentre un valore troppo alto provoca una coda di attesa eccessiva.



Introduzione
ai Sistemi
Operativi

A. Simonetta

Introduzione
ai Sistemi
operativi

Processi e
thread

Comunicazione
tra processi
(IPC)

Sched

Gestione della
memoria

File System

Input e
Output

Deadlock

Scheduling con priorità

- Nello scheduling round-robin c'è l'assunzione implicita che tutti i processi sono ugualmente importanti, in molti contesti reali questa è una ipotesi troppo restrittiva.
- Un approccio differente assegna una priorità a ciascun processo e lo scheduling tiene conto della priorità nel momento in cui deve scegliere il processo da mandare in esecuzione.
- Per evitare che i processi ad alta priorità girino per un tempo indeterminato, lo scheduler può:
 - ① diminuire la priorità del processo in esecuzione ad ogni ciclo di clock;
 - ② utilizzare un quantum di tempo massimo di CPU.
- Le priorità possono essere assegnate in modo statico o dinamico.
- È conveniente dividere i processi per classi di priorità utilizzando lo scheduling con priorità per le classi e quello round-robin all'interno di ciascuna classe.

Shortest process next

- Poiché l'algoritmo shortest job first ha sempre il minor tempo medio di risposta nei sistemi batch, sarebbe interessante poterlo utilizzare anche per i processi interattivi.
- La difficoltà nei sistemi interattivi è nel calcolare il tempo di esecuzione che non è sempre lo stesso.
- Un possibile approccio è di misurare i tempi di esecuzione e di utilizzarli per le stime successive attraverso una media pesata tenendo conto che le misure più recenti sono più attendibili rispetto a quelle passate (**aging**):

Iterazione	0	1	2	3
Tempo di esecuzione	T_0	T_1	T_2	T_3
Calcolo	T_0	$\frac{T_0}{2} + \frac{T_1}{2}$	$\frac{T_0}{4} + \frac{T_1}{4} + \frac{T_2}{2}$	$\frac{T_0}{8} + \frac{T_1}{8} + \frac{T_2}{4} + \frac{T_3}{2}$
Stima di	T_1	T_2	T_3	T_4



Introduzione
al Sistemi
Operativi

A. Simonetta

Introduzione
ai Sistemi
operativi

Processi e
thread

Comunicazione
tra processi
(IPC)

Sched^{uling}

Gestione della
memoria

File System

Input e
Output

Deadlock

Scheduling garantito

- Un approccio completamente diverso per lo scheduling è quello di fare promesse sulle performance reali dei processi utente e poi mantenerle.
- Se ci sono n utenti collegati dovrebbero ricevere $\frac{1}{n}$ del tempo della CPU.
- Per mantenere questa promessa, il sistema memorizza quanta CPU ogni processo ha avuto fin dalla sua creazione e calcola il rapporto tra avuto/promessa:
 - se un qualsiasi processo ha un valore più basso di tale rapporto è il successivo candidato ad essere messo in esecuzione fino a quando il suo rapporto è il più basso tra tutti quelli pronti.

Introduzione
ai Sistemi
Operativi

A. Simonetta

Introduzione
ai Sistemi
operativi

Processi e
thread

Comunicazione
tra processi
(IPC)

Sched^{ing}

Gestione della
memoria

File System

Input e
Output

Deadlock

Scheduling a lotteria

- Fare promesse agli utenti è una bella idea è ma difficile da realizzare.
- Si può utilizzare un algoritmo più semplice: si assegna un biglietto della lotteria alle varie risorse del sistema, come il tempo di CPU.
- Ogni volta che lo scheduling deve fare una scelta pesca caso un biglietto e il processo che ha quel biglietto si aggiudica la risorsa.
- Differentemente dallo scheduling con le priorità in cui è difficile dire cosa significa avere una certa priorità, in questo algoritmo se un processo possiede 20 biglietti su 100 ha il 20% delle probabilità di ottenere la risorsa (che al crescere del tempo approssima la frequenza).



Vantaggi dello scheduler a lotteria

- L'algoritmo di scheduling a lotteria ha due proprietà interessanti:
 - è reattivo: anche i processi neonati possono vincere la lotteria, fin dalle prime scelte dello scheduler;
 - i processi che cooperano, se lo desiderano, possono scambiarsi biglietti che detengono per alterare le priorità di esecuzione.
- Può essere utilizzato per problemi che sono di difficile risoluzione con altri metodi come ad esempio lo streaming video dove sono necessarie differenti velocità di trasferimento (espresse in frame al secondo) a seconda del processo. Assegnando tanti biglietti quanti sono i frame automaticamente la CPU approssima le proporzioni desiderate.

Scheduling fair-share

- Fino ad ora ogni processo è schedulato per proprio conto indipendentemente dal suo proprietario.
 - Se un utente X avvia 4 processi (A, B, C e D) ed un altro Y uno soltanto (E), il primo utente si prenderà l'80% del tempo di CPU.

A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
X	X	X	X	Y	X	X	X	X	Y	X	X	X	X	Y

- Per evitare questa situazione prima di qualsiasi schedulazione di controllo il proprietario del processo e si assegna una porzione di CPU in base agli utenti collegati (nel caso di prima 50% ai due utenti) e non ai processi:



Introduzione
ai Sistemi
Operativi

A. Simonetta

Introduzione
ai Sistemi
operativi

Processi e
thread

Comunicazione
tra processi
(IPC)

Sched^{uling}

Gestione della
memoria

File System

Input e
Output

Deadlock

Scheduling nei sistemi real-time

- In un sistema in tempo reale, il tempo gioca un ruolo essenziale.
- I sistemi real-time sono generalmente suddivisi in categorie:
 - **Hard real-time:** le scadenze devono essere sempre rispettate.
 - **Soft real-time:** il mancato rispetto di una scadenza non è auspicabile, ma comunque tollerabile.
- Gli eventi in un sistema real-time possono essere classificati come:
 - **Periodici**, quando si verificano a intervalli di tempo regolari.
 - **Non periodici**, quando si verificano in modo imprevedibile.

Scheduling nei sistemi real-time

- Un sistema **soft real-time con eventi periodici** è sostenibile se riesce a far fronte agli eventi stessi, ovvero se riesce a trattare un evento prima che ne arrivi un altro.
- Un sistema è in grado di rispondere a ogni evento dipendente da
 - la frequenza di arrivo degli eventi
 - il tempo necessario per processarli
- Se ci sono m eventi periodici e ogni evento avviene con frequenza pari a $1/P_i$, supponendo che ciascun evento richieda C_i secondi di tempo CPU per gestirlo, allora il sistema è in grado di reggere il carico se e solo se:

$$\sum_{i=0}^m \frac{C_i}{P_i} < 1$$

- Algoritmi di scheduling real-time possono essere statici o dinamici

Introduzione
ai Sistemi
Operativi

A. Simonetta

Introduzione
ai Sistemi
operativi

Processi e
thread

Comunicazione
tra processi
(IPC)

Schedul

Gestione della
memoria

File System

Input e
Output

Deadlock

Problema

- Si consideri un sistema soft real-time con tre eventi periodici aventi:
 - tempi di arrivo $P_0 = 100 \text{ ms}$, $P_1 = 200 \text{ ms}$, $P_2 = 500 \text{ ms}$
 - e tempi di elaborazione $C_0 = 50 \text{ ms}$, $C_1 = 30 \text{ ms}$, $C_2 = 100 \text{ ms}$
- Il sistema è sostenibile?
 - Sì perché il carico è dell'85%: $0,5 + 0,15 + 0,2 = 0,85 < 1$
- Se viene aggiunto un quarto evento con un periodo di 1 s quanto è il tempo limite di processamento affinché il sistema rimanga sostenibile?
 - $P_4 < 150 \text{ ms}$ del tempo di CPU: $1 - (0,5 + 0,15 + 0,2) = 0,15$