Sistemi Operativi

Prof. **Marco Camurri**

Introduzione

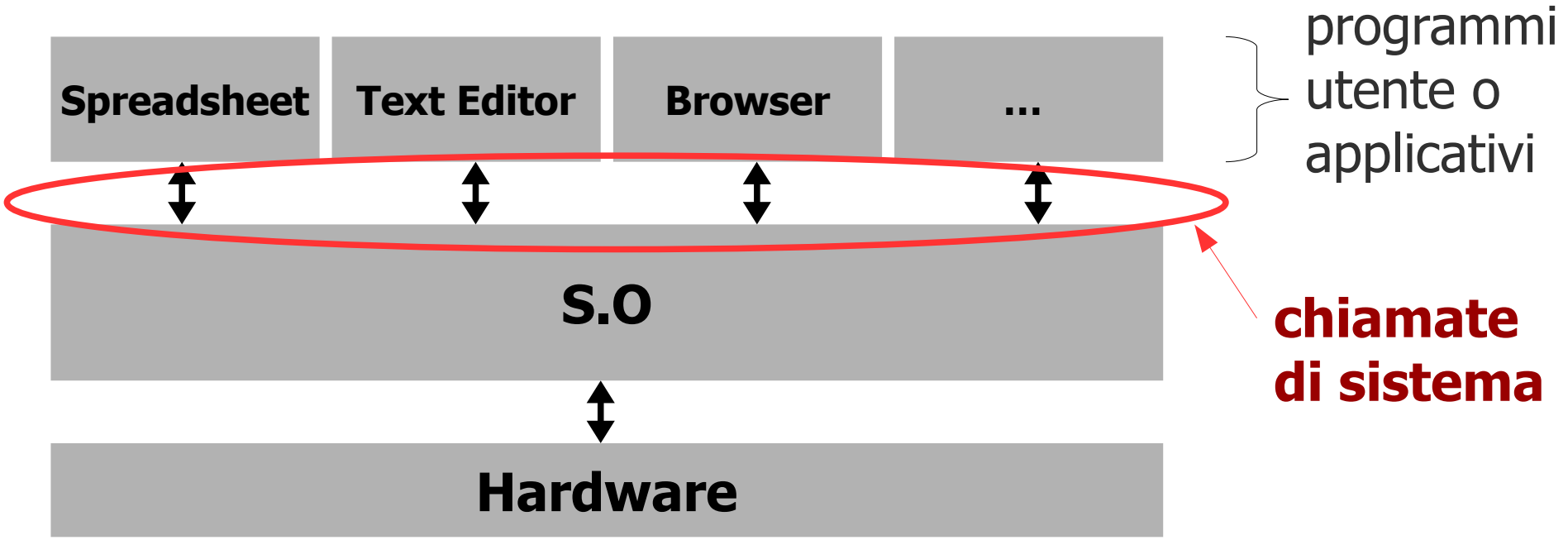
- Esempi di sistemi operativi:

  UNIX, Linux, Windows, Android, Mac OS X, iOS

- Software complesso:
  - milioni di linee di codice
  - migliaia di programmatori
- S.O open source = codice sorgente pubblicamente disponibile

Cos'è un sistema operativo?

Il **Sistema Operativo** è uno strato software che agisce da **intermediario** tra i programmi applicativi (programmi utente) e l'hardware del computer



Chiamate di sistema

- L'utilizzatore del computer interagisce con i programmi applicativi, poichè essi sono quelli che effettuano operazioni utili per l'utente (un computer con solo il S.O sarebbe inutile!)
- I programmi applicativi NON possono interagire direttamente con l'hardware
- Per utilizzare l'hardware, essi devono effettuare richieste al sistema operativo, dette **chiamate di sistema** (**system call**)
- In pratica, una chiamata di sistema è una particolare chiamata di funzione che porta all'esecuzione di codice del sistema operativo, anzichè codice del programma utente
- Ad esempio, è necessaria una chiamata di sistema ogni volta che un programma vuole scrivere o leggere un dato da file

Perchè serve il S.O ?

Due motivi:

1) **Semplificare l'utilizzo dell'hardware (fornire astrazioni dell'hardware)**
   - Esempio: il concetto di file

2) **Gestire/controllare le risorse del sistema**
   - Esempio: gestione della stampante
   - Esempio: gestione della memoria RAM

Semplificare l'uso dell'HW

- il S.O rende disponibile agli utenti e ai programmatori di applicazioni utente un insieme di **astrazioni** che nascondono la reale complessità dell'hardware
- Un esempio di astrazione fornita dal sistema operativo è il concetto di **file** (un insieme di informazioni correlate salvate in modo permanente, a cui assegnamo un **nome** e un **percorso**)
- Il controllore di un hard disk non ha alcuna nozione di file. Gli unici comandi che possono essere inviati al controllore del disco sono comandi di basso livello per leggere o scrivere singole "aree" del disco in base a un **indirizzo numerico**
- Se i programmatori dovessero conoscere tutti i dettagli dell'hardware per poter programmare le applicazioni, impiegherebbero anni per scrivere anche le applicazioni
- il S.O realizza una **piattaforma di esecuzione** per i programmi applicativi

Gestire le risorse del sistema

- S.O agisce da **gestore delle risorse**, cioè garantisce che le risorse del sistema (processori, memoria, periferiche) siano utilizzate dai programmi in modo efficiente e corretto
- Esempio: se i programmi potessero controllare direttamente la stampante, e più programmi inviassero contemporaneamente righe di testo alla stampante, la stampante alternerebbe alcune righe provenienti dal primo programma e alcune righe provenieneti dal secondo
- Soluzione: la risorsa "stampante" è gestita dal S.O e non può essere controllata direttamente dai programmi. Se un programma desidera stampare, deve richiedere il servizio al S.O, che invierà i dati alla stampante solo dopo che questa avrà terminato le stampe già iniziate o in attesa

Il kernel

Il **kernel** (nucleo o "nocciolo") è la parte centrale del codice di un sistema operativo

- ha il **controllo completo dell'hardware**
- viene caricato in memoria principale al termine del **bootstrap**
- risiede **stabilmente in memoria** principale
- risiede in un'area di **memoria protetta**

Il kernel

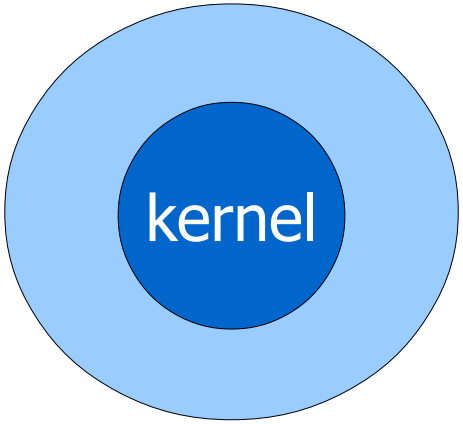
- **Controllo completo dell'hardware** = il kernel può eseguire qualsiasi istruzione macchina, al contrario dei programmi utente che possono eseguire solo un sottoinsieme delle istruzioni macchina della CPU

- Il kernel è posto in memoria al termine della fase di avvio del computer (fase di **bootstrap**) e da quel momento rimane sempre in esecuzione

- Se un programma utente va in crash, il controllo ritorna al kernel

- **Area di memoria protetta** = area di memoria non sovrascrivibile dai programmi utente (un programma utente non può eliminare o modificare il kernel)

Il kernel



- Il kernel esegue le funzioni "critiche" del S.O come: gestione della CPU, gestione della memoria, e gestione dell'I/O
- Si distingue da altre porzioni di software che, pur essendo considerate parte del sistema operativo, sono eseguite come normali programmi utente
- Ad esempio: in Linux la parte di S.O che si occupa dell'interfaccia grafica non è parte del kernel, ma è realizzata come un programma utente. Ciò consente agli utenti Linux di installare diversi "desktop environments" (i più noti sono **GNOME** e **KDE**)
- Altri esempi: programmi di utilità del sistema operativo ...

Modalità della CPU

Come può il kernel eseguire istruzioni vietate ai programmi utente?

- La maggior parte delle CPU prevede almeno due **modalità operative**: **kernel-mode** e **user-mode**
- Si tratta di caratteristica hardware introdotta nelle CPU proprio per supportare la realizzazione dei S.O, senza la quale non sarebbe possibile realizzare sistemi operativi sicuri
- Particolari registri tengono traccia della modalità operativa corrente della CPU (con due modalità basta 1 bit)
- Quando la CPU si trova nella modalità kernel, tutte le istruzioni macchina possono essere eseguite
- Se la CPU si trova in modalità utente ed incontra un'istruzione non consentita (esempio istruzioni di I/O), viene generato un errore e il controllo ritorna al S.O

Modalità della CPU

- La modalità **kernel** è anche detta modalità **privilegiata** o modalità **supervisore**
- Tipiche istruzioni non eseguibili in modalità utente sono le istruzioni di I/O
- All'avvio del computer, la CPU è in kernel-mode: il codice necessario all'avvio del computer è eseguito in questa modalità
- Quando il S.O avvia un programma utente, esso modifica opportunamente i registri hardware in modo che il programma sia eseguito in user-mode
- Ovviamente un programma utente non può cambiare la modalità e continuare la propria esecuzione
- ... però ....

Chiamate di sistema e modalità della CPU

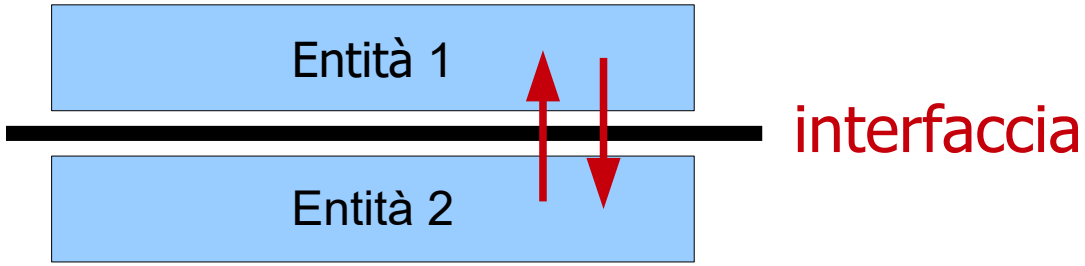
- Un programma utente determina il passaggio alla modaltà kernel ogni volta che effettua una **system call**
- Per eseguire la system call il programma utente deve utilizzare una speciale istruzione che ha l'effetto di:
  - porre il sistema nella modalità kernel
  - saltare a eseguire la porzione di codice del kernel che realizza la funzione richiesta
- Prima di restituire il controllo al programma utente, il kernel pone nuovamente il sistema in user-mode

API del sistema operativo

- L'insieme delle system call disponibili in un dato sistema operativo prende il nome di **API** del sistema
- **API** = **A**pplication **P**rogramming **I**nterface (interfaccia per la programmazione delle applicazioni)
- Sistemi operativi diversi (es. Windows/Linux) espongono API diverse
- Ad esempio, sia Windows che Linux dispongono di una chiamata di sistema per l'apertura di un file, ma esse differiscono per numero e significato dei parametri
- Per questo motivo, un programma compilato per Windows non è eseguibile su Linux, e viceversa, nonostante entrambi i sistemi operativi possano essere eseguiti dalla stessa CPU

Concetto di interfaccia

- In termini molto generali, in elettronica e in informatica, la parola **interfaccia** è usata per indicare il mezzo (fisico o astratto) o la modalità attraverso cui due entità comunicano



- In informatica, l'interfaccia di una funzione è costituita da tutte le informazioni che occorre conoscere per poter richiamare la funzione (numero, tipo e significato dei parametri, tipo e significato del valore di ritorno)
- Per richiamare una funzione occorre conoscerne l'**interfaccia**, mentre non occorre conoscerne l'**implementazione**

Funzioni del sistema operativo

In ogni sistema operativo si possono individuare le seguenti aree (o moduli):

1) Gestione della CPU e dei processi

2) Gestione della memoria principale

3) Gestione dei dispositivi di I/O

4) Gestione dei file (File System)

5) Gestione dell'interfaccia con l'utente

6) Gestione della sicurezza

Funzioni del sistema operativo

Gestione della CPU e dei processi: il SO fornisce agli utenti l'illusione che più programmi siano in esecuzione contemporaneamente, anche in presenza di una sola CPU, condendo ai programmi l'uso della CPU per brevi intervalli di tempo, a rotazione

Gestione della memoria principale: il SO tiene traccia delle aree di memoria libere e di quelle occupate; decide dove posizionare in memoria i programmi da eseguire; fa in modo che un programma non possa accedere alle aree di memoria di un altro programma (o del SO stesso!)

Gestione dei dispositivi di I/O: Il SO è l'unico a poter accedere ai registri dei disposivi di I/O (tramite i driver, eseguiti in modalità kernel). Le routine di risposta agli interrupt sono gestite dal SO.

Gestione dei file (File System): il SO tiene traccia delle posizioni libere e occupate sui dischi, introduce i concetti di file e cartella, decide dove posizionare i nuovi file sul disco, ...

Gestione dell'interfaccia con l'utente: il SO fornisce un'interfaccia (grafica o a caratteri) attraverso cui l'utente può interagire con il SO

Gestione della sicurezza: il SO garantisce che solo un certo insieme di utenti autorizzati possa accedere al sistema, e che gli utenti non possano accedere a file privati di altri utenti (sistema di permessi sui file)

Struttura interna del kernel

In base all'organizzazione interna del kernel, esso può essere classificato come:

- **Kernel monolitico**: nei sistemi operativi con kernel monolitico (es. UNIX e Linux) il kernel è realizzato come un unico enorme programma, eseguito in kernel mode, che si occupa di quasi tutte le funzioni del S.O
- **Microkernel**: nei S.O basati su microkernel, il kernel implementa solo un minimo insieme di funzioni essenziali (esempio: gestione della CPU e della memoria) mentre una parte consistente del S.O è eseguita in modalità utente (ad esempio il file system, gestione dei driver, protocolli di rete.)

- I kernel monolitici tendono ad essere più efficienti, ma meno "comprensibili"
- Esistono anche soluzioni intermedie (Kernel ibridi -> Windows)

Interfaccia con l'utente

Il mezzo (la modalità) attraverso cui l'utente interagisce con il S.O è detto **interfaccia utente.** L'interfaccia può essere:

- **Testuale**
  - detta a anche "**a riga di comando**" (**CLI** = command line interface)
  - l'utente comunica con il S.O digitando comandi tramite un programma chiamato **shell**
  - era l'unica interfaccia disponibile sui S.O del passato (es. DOS) ma è ancora utilizzata per interagire ad es. con il S.O di un server web in rete, poichè più veloce
- **Grafica**
  - in questo caso viene detta **GUI** (graphical user interface)
  - è basata su finestre (window)
  - uno dei primi computer con GUI: Apple Lisa (1983)