Sistemi Operativi

UniVR - Dipartimento di Informatica

Fabio Irimie

1° Semestre 2024/2025

Indice

1 Introduzione

2 Sistema Operativo

Il **sistema operativo** è il livello del software che si pone tra l'hardware e gli utenti. E quindi il sistema operativo incapsula la macchina fisica. Per mettere in comunicazione l'utente e l'hardware solitamente si usano le **applicazioni**, ma quando si vuole accedere direttamente all'hardware si usano le interfacce utente, ad esempio:

1. **Interfaccia grafica** (GUI)
2. **Command line** (Terminale o Shell)
3. **Touch screen**

Gli obiettivi principali sono:

- Facilitare l'uso del computer
- Rendere efficiente l'utilizzo dell'hardware
- Evitare conflitti nell'allocazione delle risorse hardware e software

Questo rimuove la necessità di conoscere la struttura dell'hardware attraverso l' **astrazione** facilitando la programmazione.

2.1 Compiti del sistema operativo

2.1.1 Gestione delle risorse

Il sistema operativo deve gestire le risorse hardware, come ad esempio i dischi, la memoria, gli input/output e la CPU. Deve anche gestire le risorse software, come ad esempio i file, i programmi e la memoria virtuale.

2.1.2 Programma di controllo

Un altro compito del sistema operativo è quello di controllare l'esecuzione dei programmi e del corretto utilizzo del sistema.

2.2 Interrupt

Un **interrupt** è un segnale hardware che interrompe il normale flusso di esecuzione di un programma. Gli interrupt possono essere generati da:

- **Hardware**: Per esempio quando un dispositivo ha finito un'operazione
- **Software**: Per esempio quando un programma chiama una system call

Questo serve per permettere alla CPU di lavorare per più tempo.

2.2.1 Operazioni input/output

Per gestire gli input/output ad esempio si usano i **device driver** che sono programmi che permettono di programmare la periferica per comunicare con il sistema operativo. Finchè il dispositivo non ha finito l'operazione, la CPU esegue altri processi e quando riceve l'interrupt dal dispositivo che segnala la fine dell'operazione, la CPU interrompe il processo corrente e inizia a gestire l'interrupt.

- **Buffering**: Sovrapposizione di CPU e I/O dello **stesso** processo
- **Spooling**: Sovrapposizione di CPU e I/O di **diversi** processi

Lo spooling serve quando l'elaborazione dei dati letti da un processo è più veloce della lettura stessa, quindi si avrebbero dei tempi morti rimossi con lo spooling che permette di elaborare i dati, già pronti, letti da un altro processo.

2.3 Multiprogrammazione

È la possibilità di tenere caricati in memoria più programmi e di eseguirli in modo alternato. Questo permette di sfruttare al meglio la CPU e di ridurre i tempi morti dovuti all'I/O.

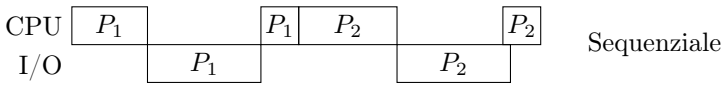

Figura 1: Senza multiprogrammazione

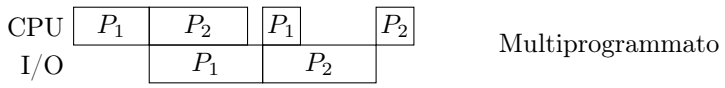

Figura 2: Con multiprogrammazione

Per eseguire più programmi alla volta, non soltanto quando un programma è in attesa di I/O, si utilizza il concetto di **time sharing** che permette di eseguire più programmi in modo alternato ad intervalli di tempo molto brevi chiamati **quanto di tempo** creando l'impressione che i programmi vengano eseguiti in parallelo.

Quando si hanno tanti processi che concorrono per l'utilizzo della CPU bisogna implementare delle regole per decidere quale processo eseguire. Queste regole sono definite dal sistema operativo, nello specifico dallo **scheduler**.

2.4 Protezione

La possibilità di eseguire più processi contemporaneamente crea dei problemi, come ad esempio la possibilità che un processo possa accedere ad un'altra area di memoria di un altro processo. Per evitare questo si usano delle tecniche di **protezione**:

- **Protezione I/O**: Programmi diversi non devono usare I/O contemporaneamente
- **Protezione della memoria**: Un processo non può leggere o scrivere in un'area di memoria che non gli appartiene
- **Protezione della CPU**: Un processo non può usare la CPU per più tempo di quello che gli è stato assegnato

In generale la protezione è realizzata tramite il meccanismo della **modalità duale** di esecuzione:

- **Modalità utente**: Il programma viene eseguito in modo normale, senza accesso alle risorse hardware
- **Modalità kernel**: Il programma viene eseguito con privilegi speciali che permettono di accedere all'hardware

2.4.1 Protezione I/O

Quando un processo vuole usare un dispositivo I/O, deve passare per il **device driver** che esegue le operazioni di I/O in modalità **supervisor** (o kernel mode). Questo viene fatto attraverso una **system call**, cioè un interrupt software che permette di passare dalla modalità utente alla modalità kernel per esequire operazioni privilegiate. Al termine della system call si ritorna in modalità utente.

2.4.2 Protezione della memoria

Per proteggere la memoria si devono imporre dei limiti di accesso alla memoria ai processi. Il sistema operativo tiene traccia di questi limiti e li controlla ad ogni accesso alla memoria.

2.4.3 Protezione della CPU

Per proteggere la CPU bisogna garantire che il sistema operativo abbia sempre il controllo su di essa per poter interrompere qualsiasi processo che non restituisca il controllo della CPU. Per fare ciò si usano i **timer** che interrompono il processo se viene eseguito per troppo tempo.

2.5 Tipi di sistema operativo

Esistono diversi tipi di sistemi operativi che si differenziano e si classificano sulla base di determinate architetture di calcolo e determinati scopi. I principali tipi sono:

- **S.O. per PC e workstation**: Uso personale dell'elaboratore
- **S.O. di rete**: Separazione logica delle risorse remote e locali
- **S.O. distribuiti**: Non c'è la separazione logica tra risorse remote e locali, quindi l'accesso alle risorse remote viene effettuato nello stesso modo di quello alle risorse locali

- **S.O. real-time**: Vincoli sui tempi di risposta del sistema
- **S.O. embedded**: Per dispositivi con risorse limitate

3 Componenti di un sistema operativo

Le componenti principali di un sistema operativo sono:

- **Gestione dei processi**
- **Gestione della memoria primaria** (RAM): Spazio dei processi
- **Gestione della memoria secondaria** (Memoria di massa): Spazio dei programmi e dei dati
- **Gestione dell'I/O**
- **Gestione dei file**
- **Protezione**
- **Rete**
- **Interprete dei comandi**

3.1 Gestione dei processi

Un processo è l'istanza di un programma in esecuzione. Ogni processo ha bisogno di risorse per poter essere eseguito, come ad esempio la CPU, la memoria, i file e i dispositivi I/O. Il sistema operativo deve gestire i processi e assegnare le risorse per garantire che i processi vengano eseguiti in modo corretto e senza conflitti e deve gestire anche se stesso perchè anche il sistema operativo è un'insieme di processi in modalità kernel.

Il sistema operativo è responsabile della:

- **Creazione e distruzione di processi**: Creare un processo significa allocare le risorse necessarie e inizializzare le strutture dati del processo; distruggere un processo significa rilasciare le risorse e liberare la memoria
- **Sospensione e riesumazione di processi**: Un processo viene sospeso quando deve aspettare un evento, come ad esempio un I/O
- **Sincronizzazione e comunicazione tra processi**: I processi devono poter comunicare tra di loro e sincronizzarsi per evitare conflitti

3.2 Gestione della memoria primaria

Un programma deve essere caricato in memoria per essere eseguito e nella RAM sono anche presenti dati condivisi tra CPU e dispositivi I/O.

Il sistema operativo è responsabile di:

- **Gestire lo spazio di memoria**: Decide quali parti della memoria allocare e a quali processi assegnarla

- **Decidere quale processo caricare in memoria quando esiste spazio disponibile**
- **Gestire l'allocazione e il rilascio dello spazio di memoria**

3.3 Gestione della memoria secondaria

La memoria secondaria è usata per memorizzare programmi e dati che non possono stare nella memoria primaria e per mantenere grandi quantità di dati in modo permanente. Il sistema operativo deve gestire l'ottimizzazione dell'uso della memoria secondaria.

Il sistema operativo è responsabile di:

- **Gestire lo spazio sulla memoria di massa**
- **Allocare spazio su memoria di massa**
- **Ordinare gli accessi ai dispositivi**

3.4 Gestione dell'I/O

Il sistema operativo deve creare un livello software che permetta all'utente di usare le periferiche senza doverne capire le caratteristiche fisiche.

Il sistema di I/O consiste di:

- **Sistema per accumulare gli accessi ai dispositivi**: Buffering, cioè l'accumulo di dati in memoria prima di essere scritti o letti dal dispositivo
- **Generica interfaccia verso i device driver**
- **Device driver specifici per alcuni dispositivi**

3.5 Gestione dei file

I file servono per memorizzare informazioni su supporti fisici diversi controllati da driver con caratteristiche diverse. Un file è l'**astrazione logica** per rendere conveniente l'uso della memoria non volatile e permette di raccogliere informazioni correlate, come ad esempio dati o programmi.

Il sistema operativo è responsabile di:

- **Creare e cancellare file e directory**
- **Fornire primitive per la gestione di file e directory**: Ad esempio aprire, leggere, scrivere, chiudere, rinominare, cancellare ecc...
- **Gestire la corrispondnza tra file e spazio fisico su memoria di massa**: Bisogna tenere traccia di dove sono memorizzati i file
- **Salvare informazioni a scopo di backup**

3.6 Protezione

È il meccanismo per controllare l'accesso alle risorse da parte di utenti e processi.

Il sistema operativo è responsabile di:

- **Definire accessi autorizzati e non**
- **Definire controlli per gli accessi**
- **Fornire strumenti per verificare le politiche di accesso**

3.7 Rete

Il sistema operativo deve gestire tutte le risorse di calcolo connesse tramite una rete.

Il sistema operativo è responsabile della gestione di:

- **Processi distribuiti**
- **Memoria distribuita**
- **File system distribuito**

3.8 Interprete dei comandi

L'interprete dei comandi è un programma che permette all'utente di comunicare con il sistema operativo. L'interprete dei comandi permette di:

- **Creare e gestire processi**
- **Gestire l'I/O**
- **Gestire il disco, la memoria, il file system**
- **Gestire le protezioni**
- **Gestire la rete**

Un interprete dei comandi è la **shell**, che è un programma che legge e interpreta i comandi.

3.9 System call

Sono un'interfaccia tra processi e sistema operativo e permettono ai processi di comunicare con il sistema operativo. Le system call sono chiamate tramite un'istruzione software e permettono di eseguire operazioni privilegiate. Le possibili opzioni per comunicare tra sistema operativo e processo sono:

1. Passare i parametri della system call tramite registri
2. Passare i parametri tramite lo stack del programma

```
// Programma utente
void main() {
  ...
  A(x);
  ...
}

A(int x) {
  ...
  push x;
  _A(); // Vera e propria system call
  ...
}

_A() {
  scrivi 13
  TRAP // Interruzione software
  ...
}
```

```
// Sistema operativo
Leggi 13
Salta al gestore 13

handler_13() {
  ...
}
```

3. Memorizzare i parametri in una tabella in memoria
    - L'indirizzo della tabella è passato in un registro o nello stack