Sistemi Operativi

UniVR - Dipartimento di Informatica

Fabio Irimie

1° Semestre 2024/2025

Indice

1 Introduzione

Il sistema operativo **UNIX** deriva dal **MULTIX** e da UNIX sono derivati tutti gli altri sistemi operativi odierni, tra cui **Linux** e **MacOS**.

1.1 Algoritmi

Non esiste una definizione rigorosa di algoritmo, ma ne esiste una informale degli algoritmi sequenziali

Definizione 1.1. Chiamiamo algoritmo un insieme finito P di istruzioni per le quali esiste una macchina M capace di interpretare ed eseguire le istruzioni di P t.c.

1. M ha a disposizione una memoria illimitata per i risultati intermedi
2. In un tempo finito M può effettuare solamente un numero finito di passi di calcolo (si dice che M calcola in modo **discreto**)
3. Il calcolo da parte di M avviene in modo deterministico
4. La dimensione dei dati manipolabili da M è finita ma non limitata.

Per esegiure gli algoritmi serve un linguaggio di programmazione e per interpretarli c'è bisogno di un **interprete**. Il compito dell'interprete è eseguire programmi scritti nel linguaggio L; a tal fine deve essere in grado di decodificare e valutare istruzioni scritte nel linguaggio L.

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 ci sono le interfacce utente, che si dividono in:

1. **GUI**
2. **Touch screen**
3. **Command line**

Il sistema operativo fornisce dei servizi per comunicare con l'hardware e questi servizi possono essere usati tramite delle **system calls**. Questi servizi sono ad esempio:

- Esecuzione dei programmi
- Gestione dei file
- Operazioni I/O
- Gestione degli errori

- Comunicazione

L'unico programma che è sempre in esecuzione su un computer è il **kernel**.

2.1 Operazioni

- **Bootstrap program**: è una piccola porzione di codice che inizializza il sistema e carica il kernel
- Viene caricato il kernel
- Vengono caricati i **system daemons**, cioè dei servizi forniti al di fuori del kernel
- Gestione delle chiamate di sistema:
  - Hardware interrupt
  - Software interrupt

Su un computer vengono eseguiti più programmi alla volta salvando la coda dei processi da eseguire in memoria e lo **scheduler** si occupa di gestire l'ordine di esecuzione e di interruzione. Per permettere di eseguire più processi alla volta si utilizza il **time sharing**, cioè la CPU cambia processo così frequentemente che si crea l'illusione che i processi vengano eseguiti in parallelo anche se in realtà non è così.

3 Modalità di esecuzione

Il sistema operativo può eseguire il codice in 2 modalità:

1. **Modalità utente**: il codice viene eseguito in modo normale
2. **Modalità kernel**: il codice viene eseguito con privilegi speciali che permettono di accedere all'hardware

Per capire in che modalità si sta eseguendo il codice si utilizza un bit nel **program status word** (PSW) che indica la modalità di esecuzione chiamato **mode bit**.
Si può entrere in modalità kernel tramite:

1. **Nuovo processo**: Per creare un nuovo processo, il kernel copia il programma nella memoria, setta il program counter alla prima istruzione del processo e setta lo stack pointer alla base dello stack del processo e infine si torna in modalità utente.
2. **Ritorno da un interrupt o system call**: Quando il kernel finisce di gestire la richiesta, riprende l'esecuzione del processo che ha chiamato l'interrupt o la system call e torna in modalità utente.
3. **Cambio di contesto**: Quando il kernel decide di cambiare il processo in esecuzione, salva lo stato del processo corrente e carica lo stato del nuovo processo e torna in modalità utente.

Per prevenire che un processo faccia un ciclo infinito esiste un timer che interrompe il processo se viene eseguito per troppo tempo.

3.1 Protezione e sicurezza

1. **Protezione**: qualsiasi meccanismo per controllare l'accesso dei processi o degli utenti alle risorse del sistema.
2. **Sicurezza**: protezione da accessi esterni non autorizzati.

Ogni utente è identificato da un **user id** e ogni utente può far parte di un gruppo, identificato da un **group id**. Ogni file ha un **owner** e un **group owner** e per ogni file ci sono 3 tipi di permessi:

1. **Read**
2. **Write**
3. **Execute**

4 Linux

4.1 Filesystem

In linux qualsiasi cosa è un file, cioè un contenitore di dati. I principali tipi di file sono:

1. **Regolare**: File classici utente
2. **Directory**: Informazioni relative ad altri file
3. **Pipe**: Comunicazioni tra processi
4. **Link**: Alias
5. **Socket**: Comunicazioni tra processi
6. **Speciale**: Dispositivi

Tutti i file sono organizzati in un albero, chiamato **file system**.

1. **root**: La radice del file system: /
2. **nodo**: File (se ha già figli è una directory)