Laboratorio di Sistemi Operativi  
Appunti del corso a cura dello studente Gianni Adamo

A.A. 2024 - 2025

Sommario

[0. Premessa 3](#_Toc179919945)

[0.1 Riferimenti 3](#_Toc179919946)

[0.2 Info e Contatti 3](#_Toc179919947)

[1. Introduzione a Unix 4](#_Toc179919948)

[1.1 Comandi Unix 4](#_Toc179919949)

[1.2 Tipi di file 5](#_Toc179919950)

[2. [LEZIONE 2] [BETA] 6](#_Toc179919951)

# 0. Premessa

Questi appunti sono stati scritti con il solo scopo di approfondire e assistere lo studio degli argomenti trattati nell’omonimo corso *“Laboratorio di Sistemi Operativi”*, tenuto dalla prof. Alessandra Rossi presso l’*Università degli Studi di Napoli Federico II*, per il CDL in Informatica. Tale documento fa riferimento al corso dell’A.A 2024-2025.

Ulteriori informazioni sono contenute in fondo alla pagina corrente.

## 0.1 Riferimenti

Il materiale didattico utilizzato per la creazione di questo documento è così elencato:

* *Slides del corso “Laboratorio di Sistemi Operativi” – Prof. Alessandra Rossi*
* *Appunti degli anni precedenti (riferiti al medesimo corso)*
* *Annotazioni scritte a lezione*
* *Approfondimenti trovati in rete*
* *Esercitazioni e test vari effettuati da terminale*

Tutte le informazioni contenute in questo documento sono state accuratamente selezionate per coprire al meglio gli argomenti trattati durante il corso e fornire allo studente una guida per assimilare i concetti spiegati a lezione. L’autore non si assume alcuna responsabilità riguardo la completezza dell’elaborato e agli esiti che ne susseguono. Eventuali variazioni che il corso potrebbe subire negli anni a venire non saranno indicate all’interno del documento. Per ulteriori informazioni, consultare la sezione *Info e Contatti.*

Tale documento è stato interamente scritto dallo studente Giovanni (Gianni) Adamo, iscritto al CDL in Informatica presso l’*Università degli Studi di Napoli Federico II.*

## 0.2 Info e Contatti

Di seguito, alcuni canali e piattaforme tramite le quali è possibile raggiungermi, per contatti o visione di ulteriori produzioni (in ambito accademico e/o personale).

* **Email:** [gianni.adamo@hotmail.it](mailto:gianni.adamo@hotmail.it)
* **GitHub:** <https://github.com/Gazen27>
* **Social:** Mi trovi su tutti i social come @gazen27

# 1. Introduzione a Unix

I sistemi operativi Unix si basano sullo standard Posix e ne esistono molti (sicuramente tra i più conosciuti ci sono Linux e OSX); le caratteristiche principali di Unix sono molteplici: la possibilità di multi-processi con time-sharing, l’uso di multi-utenti, dove ogni utente può accedere alle proprie risorse o a quelle generali dell’OS grazie a un sistema di protezioni e permessi; tali permessi vengono stabiliti dall’utente amministratore, nei sistemi Unix chiamato ***root***.

La struttura del sistema operativo di base di tutti i sistemi Unix è il ***file system*** gerarchico: rappresentabile come un albero, dove la radice, è proprio *root*.

Il sistema Unix è basato su ***kernel***, ovvero il nucleo dell’OS, che si deve adattare all’architettura dell’hardware (ad esempio sistemi intel o AMD), ed è l’unica porzione dell’OS alla quale un utente generico non può accedere se non ha i permessi di *root* (modalità privilegiata); il *kernel* gestisce tutte le risorse essenziali (CPU, memorie e periferiche, trattate come file), mentre tutto il resto, anche l’interazione con l’utente, è ottenuto tramite programmi eseguiti dal *kernel*, che accedono alle risorse hardware mediante richieste a quest’ultimo.

Il *kernel* comunica con le varie applicazioni e a tutte le varie risorse tramite le *system call*.

I sistemi Unix sono *multi-utente*, dove ogni utente ha un username, una password e un identificativo univoco (user id o semplicemente id). Ogni utente ha un set di permessi associati che vanno a definire cosa può fare o meno; l’unico *superuser* è *root*, il cui *user-id* è 0.

La ***shell*** è un interprete di comandi che si interpone tra utente e sistema operativo; nei sistemi Unix, qualsiasi operazione può essere eseguita da una serie di comandi da *shell*.

La *shell* può eseguire uno script (file bash) oppure interagire in modalità interattiva a linea di comando; l’utente può infatti fornire al prompt un comando tra:

* Nome di un comando built-in
* Nome di un file eseguibile
* Nome di uno script (dotato di permesso di esecuzione)

## 1.1 Comandi Unix

Tutti i comandi Unix hanno una struttura molto simile; il nome del comando è generalmente una sorta di abbreviazione dell’azione che si vuole effettuare e una serie di argomenti, che possono essere delle opzioni o flag (iniziano con “-”), dei parametri oppure dei file. L’ordine delle opzioni solitamente è irrilevante, mentre l’ordine dei parametri è molto importante.   
Inoltre, Unix è case sensitive.

Alcuni comandi Unix ricorrenti:

|  |  |
| --- | --- |
| *ls* | Visualizza la lista di file nella directory corrente |
| *ln* | Crea link per la directory corrente |
| *cd* | Seguito dal nome di una directory, serve a spostarsi nella directory scelta |
| *mkdir* | Seguito dal nome di una directory inesistente, ne crea una con quel nome |
| *rmdir* | Seguito dal nome di una directory già esistente, la elimina |
| *pwd* | Visualizza qual è la directory corrente |
| *exit o logout* | Lascia la sessione |
| *file* | Seguito dal nome del file, consente di vederne le informazioni |
| *cat* | Seguito dal nome del file, consente di visualizzarne il contenuto senza aprirlo |
| *man command* | Legge le pagine manuale su command |
| *info command* | Legge le pagine manuale su command |
| *passwd* | Cambia la password |

Ad ***ls*** è possibile aggiungere delle estensioni, ovvero:

***-a*** mostra anche il “contenuto nascosto” nella directory.  
***-l*** visualizza il contenuto listato nella directory con le info associate.  
***-i*** per mostrare l’i-number  
***-t*** lista in ordine di ultima modifica  
***-R*** visita ricorsivamente le sottodirectory

## 1.2 Tipi di file

Per il file system in Unix, tutto è un file, altrimenti è un processo; i tipi principali di file sono:

* **Ordinario:** Tutti i classici file esistenti (txt, exe, …).
* **Directory:** intesa come le cartelle e i loro percorsi, nei sistemi Unix.
* **Speciale:** tutto ciò che è identificato come una periferica I/O è visto come un file speciale, ovvero un link che viene aggiunto al file system.

Il sistema assegna biunivocamente a ciascun file un identificatore numerico detto *i-number* (“index-number”) per rintracciarli nel file system (all’indice 0 corrisponde *root*).

Quando viene effettuata una richiesta alle informazioni di un file, Unix fornisce:  
Tipo (ordinario, directory, speciale), Posizione, Dimensione, Numero di links (i vari filename che i file possono avere, ma un solo i-number), Proprietario, Permessi, Creazione (data), Modifica (data modifica più recente), Accesso (accesso più recente).

Quando parliamo di posizione, ci riferiamo al ***path*** del file, ovvero il percorso per arrivare a quel determinato file; esistono due tipi di percorso:

* *Path assoluto*: percorso per arrivare al file partendo dalla *root.*
* *Path relativo*: percorso per arrivare al file partendo da un certo punto (solitamente dalla posizione corrente). Questo è il path scelto di “default” nello sviluppo di software.

Se un *path* inizia per “ / ” allora si tratta di un *path* *assoluto*, altrimenti sarà relativo (e inizierà con la directory di partenza del percorso). Ad esempio “*/dir1/dir2/dir3/file1”* è un *path assoluto*.

All’interno di una directory, anche se vuota, avremo sempre almeno due elementi: “ . “ (sé stessa) e “ .. “ (la directory padre). Il contenuto delle directory sono delle coppie (*filename* e *i-number*).

La *shell* opera, ad ogni istante, nella directory corrente, detta ***working directory***, se non cambiata (cd); di default, la *working directory* è la *home directory* dell’utente, che ha come nome lo username del proprietario: è possibile aggiungere files o subdirectory.

# 2. [LEZIONE 2] [BETA]

Un *File System* Unix è sempre unico, ma può avere parti residenti su device rimuovibili:

* “montate” prima di potervi accedere (comando *mount*)
* “smontate” prima di rimuovere il supporto (comando *unmount*)

Chown cambia owner, chgrp cambia gruppo, chmod cambia i permessi.

Possono esistere gruppi vuoti ma mai utenti che non hanno un gruppo di appartenenza.