



Introduzione a Linux

Testo di riferimento:
Vincenzo Manca
“Metodi Informazionali”
Bollati Boringhieri

GNU: un po' di storia...

- **GNU** è un acronimo “ricorsivo” e significa **GNU's Not UNIX**. Proprio perché nasce con l'idea di sviluppare un S.O. stile UNIX ma libero, che permettesse di sviluppare liberamente favorendo così la collaborazione tra programmatori.
- Il Progetto GNU (1983, **Richard Stallman**), si basa sulla gestione dei diritti d'autore secondo la definizione di **software libero** (contrapposta a software proprietario).
- Fulcro del Progetto GNU è la licenza GNU General Public License (**GNU GPL**), che sancisce e protegge le libertà fondamentali che, secondo Stallman, permettono l'uso e lo sviluppo collettivo e naturale del software.
- Nel 1984 inizia lo sviluppo del Sistema GNU. In realtà il kernel di tale sistema Hurd è tuttora in lavorazione. Ma nel 1991 **Linus Torvalds** scrisse il kernel Linux e lo distribuì sotto licenza GNU GPL. I sistemi GNU con kernel Linux vengono ufficialmente chiamati **GNU/Linux**.
- Per poter gestire alcuni casi il Progetto GNU ha creato anche la GNU Lesser General Public License (**GNU LGPL**), che permette di integrare software libero all'interno di software proprietario.

Il SO Linux: comandi principali

- In generale la sintassi di un comando Linux è:

```
comando [opzioni] [argomenti]
```

- Un manuale dei comandi descrive l'utilizzo e le caratteristiche di ogni comando. Le pagine del manuale si invocano con **man argomento** e hanno tutte la seguente struttura comune:

NAME: riporta il nome del comando e una breve descrizione delle sue funzioni

SYNOPSIS: descrive la sintassi del comando

DESCRIPTION: descrive lo scopo e il funzionamento del comando

OPTIONS: riporta il funzionamento di tutte le opzioni

ENVIRONMENT: descrive eventuali variabili d'ambiente che interagiscono con il comando

AUTHOR: note sull'autore del comando

COPYRIGHT: note su copyright

BUGS: eventuali errori o malfunzionamenti noti

SEE ALSO: eventuali altre pagine del manuale a cui fare riferimento

Il SO Linux: comandi principali (2)

- Elencare il contenuto di una cartella

- Sintassi:

```
ls [opzioni...] [cartella...]
```

- Opzioni:

- -l (informazioni estese)
- -a (visualizza file nascosti, cioè iniziati con il .)
- -R (visualizza sottocartelle)

- Esempio:

```
$ ls -laR
```

Il SO Linux: comandi principali (3)

- Cambiare la cartella corrente
- Sintassi:

```
cd path_nuova_directory
```

- Opzioni:
 - cartella corrente: .
 - cartella padre: ..
 - home directory: ~

- Esempio:

```
$ cd ..
```

```
$ cd ./home/mialogin/miacartella (se esiste)
```

Il SO Linux: comandi principali (5)

- Creare nuove cartelle
- Sintassi:

```
mkdir nome_cartella
```

- Esempio:

```
$ mkdir nuovacartella1 nuovacartella2
```

Il SO Linux: comandi principali (6)

- Copiare file e cartelle

```
cp [opzioni...] sorgente... destinazione
```

- Spostare o rinominare file e cartelle

```
mv [opzioni...] sorgente... destinazione
```

- Visualizzare path assoluto cartella corrente

```
pwd
```

- Eliminare file

```
rm [opzioni...] file
```

- Eliminare una cartella

```
rmdir cartella
```

Caratteri jolly o *metacaratteri*

- * Sostituisce un insieme di zero o più caratteri qualsiasi

Esempio:

```
$ ls c*
```

- ? Sostituisce un carattere qualsiasi

Esempio:

```
$ ls co??o.txt
```

- [] Permettono di specificare una lista e/o un intervallo di caratteri possibili

Esempio:

```
$ ls [a-c]*.txt
```


Il SO Linux: i processi

- Linux è un sistema operativo **multitasking**: può eseguire contemporaneamente più programmi
- Un programma in esecuzione è definito **processo**
- Ad ogni processo viene assegnato un identificativo univoco: **PID**
- Un processo può essere **attivo** o **sospeso** ed eseguito in **foreground (fg)** o in **background (bg)**
- Un programma può essere eseguito in bg usando il carattere **&** (`$ ls c* &`) e può essere sospeso con la combinazione di tasti **CTRL+Z**
- Il sistema operativo fornisce comandi per visualizzare informazioni sui processi e per gestirne l'esecuzione.

Comandi per operare sui processi

- Visualizzare informazioni sui processi

```
ps [opzioni...] [PID]
```

- Eliminare un processo

```
kill [opzioni...] PID
```

- Visualizzare i processi sospesi o in background

```
jobs
```

- Riprendere l'esecuzione in foreground di processi sospesi o in background

```
fg job_id
```

Comandi per operare sui processi (2)

- Attivare l'esecuzione in background di processi sospesi

```
bg job_id
```

- Monitorare l'utilizzo delle risorse da parte dei processi

```
top [opzioni]
```

Il File System

Materiale tratto dai lucidi ufficiali a corredo del testo:

D. Sciuto, G. Buonanno e L. Mari

“Introduzione ai sistemi informatici” - 2005 – McGrawHill

e dal testo di riferimento

M. Bertacca, e A. Guidi

“Introduzione a Linux” - McGrawHill



Gli obiettivi del File System di un S.O.

- Gestire in modo efficiente la **memoria di massa**
- Presentare all'utente l'**organizzazione logica** dei dati (ad es. in file e cartelle) e le **operazioni** che è possibile compiere su di essi
- Fornire all'utente e ai programmi applicativi alcuni servizi di base:
 - La **creazione/cancellazione** di file e cartelle
 - La **manipolazione** di file e cartelle esistenti
 - La **copia** e lo **spostamento** di dati su supporti diversi
 - L'associazione tra file e dispositivi di memorizzazione secondaria (memorie di massa)
 - La gestione di **collegamenti** (**link o alias**) tra file e cartelle.
Un collegamento è un riferimento ad un oggetto (file o cartella) presente nel file system.

Il File System

- I dati vengono organizzati in **file**
 - Un file è un contenitore logico di informazioni (dati o istruzioni)
 - Ogni file è identificato da un **Identificatore** o **filename** (nome.estensione), dalla **periferica** (drive) e dal **percorso** (path) sulla periferica, da varie altre informazioni (data di creazione e di ultima modifica, dimensione, diritti di accesso al contenuto del file, ecc...)
 - I file possono essere raggruppati in più contenitori logici, **cartelle** o **directory**, e **sottocartelle** o **sottodirectory**, organizzati secondo una struttura gerarchica ad albero
 - I **collegamenti** (o **link**, alias) permettono di creare riferimenti ad altri oggetti (file e directory) nel file system. Permettono di accedere ad un oggetto da più punti dell'albero.

Il File System di Linux

- Opera su 5 tipi file:

- **normali**

Archivi di dati, testi, comandi, programmi sorgente, eseguibili.

- **directory**

Insiemi di sottodirectory e file normali.

- **device**

Dispositivi hardware collegati, vengono visti come file speciali.

- **pipe**

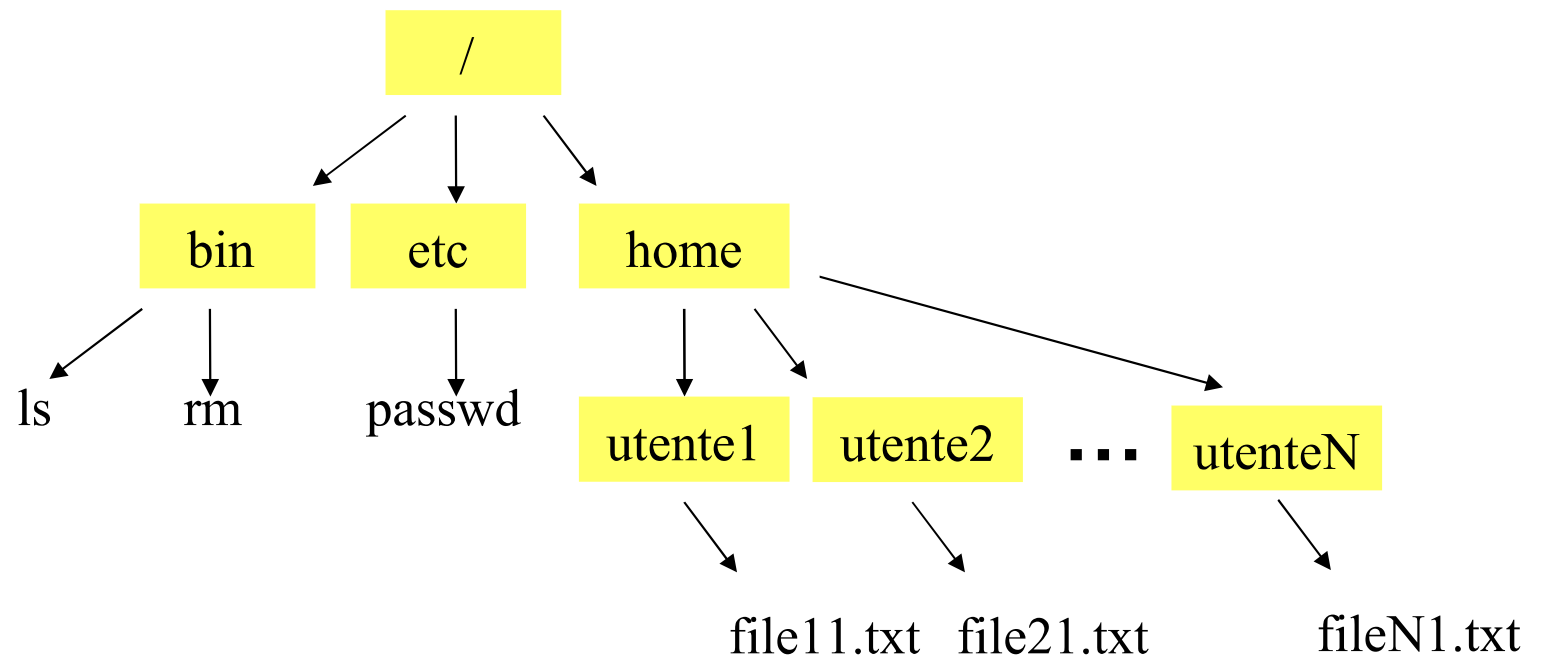
File speciali che permettono lo scambio di dati sincrono tra due processi concorrenti.

- **link**

Riferimento ad un altro file o directory. Le operazioni sul link si riflettono sull'oggetto collegato.

Struttura logica

- **Esempio:** parte di un file system



- / (root): radice dell'albero
- bin, etc, home: directory di sistema
- utente1, utente2, ..., utenteN: directory e file utente
- ls, rm, passwd: eseguibili (comandi)

Struttura logica (2): pathnames

- Un file è individuabile attraverso il **nome** e le **sottodirectory** del percorso dalla root /

Esempio: `/home/utente1/file11.txt`

- I cammini possono essere **relativi** (rispetto a directory di lavoro) o **assoluti**

Esempio: cammino assoluto e cammino relativo rispetto alla directory utente1

```
$ rm /home/utente1/subdir1/file1.txt
```

```
$ rm subdir1/file1.txt
```

Permessi e protezioni

- A file e cartelle sono assegnati dei **permessi** che garantiscono l'integrità e la riservatezza dei dati
- Ciascun file è collegato ad un utente, detto **proprietario**, e ad un **gruppo**
- Affinché un utente possa creare, cancellare o utilizzare un file deve possedere i permessi necessari per quella operazione

Permessi e protezioni (2)

- I permessi si possono visualizzare con il comando **\$ ls -l**

```
Carlo@your-ae2c3fc363 /cygdrive/e/Recycled
$ ls -l
total 2449
-rw-r--r-- 1 Carlo Nessuno 2474233 Jun 29 16:34 De1.zip
-rw-r--r-- 1 Carlo Nessuno 820 Jun 30 13:47 INF02
-rw-r--r-- 1 Carlo Nessuno 65 Jun 30 00:18 desktop.ini
Carlo@your-ae2c3fc363 /cygdrive/e/Recycled
$
```

Permessi e protezioni (2)

- I permessi si possono visualizzare con il comando **\$ ls -l**

```
Carlo@your-ae2c3fc363 /cygdrive/e/Recycled
$ ls -l
total 2449
-rw-r--r-- 1 Carlo Nessuno 2474233 Jun 29 16:34 De1.zip
-rw-r--r-- 1 Carlo Nessuno 820 Jun 30 13:47 INF02
-rw-r--r-- 1 Carlo Nessuno 65 Jun 30 00:18 desktop.ini
Carlo@your-ae2c3fc363 /cygdrive/e/Recycled
$
```

Permessi

Proprietario

Gruppo

Codifica dei permessi

- I *permessi*:

i primi 10 caratteri sono suddivisi in 4 campi secondo la struttura:

```
-rw-r--r-- 2 Carlo Nessuno 0 Jul 2 09:47 prova.txt
```



l u g o

- **l**: specifica il tipo di file (- = file normale; d = directory; c = file di i/o, es terminale o stampante; b = file su blocchi di caratteri, es hd; p = pipe; l = link)
 - **u**: permessi concessi al proprietario del file
 - **g**: permessi concessi ai membri del gruppo
 - **o**: permessi concessi agli altri utenti
-
- I permessi **u**, **g**, ed **o**, sono formati da tre caratteri che specificano i permessi di lettura (r), scrittura (w) ed esecuzione (x).

Codifica dei permessi per i file

- Il primo carattere di ogni insieme indica il permesso relativo alla lettura del file:
 - – la lettura non è permessa
 - **r** la lettura è permessa
- Il secondo carattere di ogni insieme indica il permesso relativo alla scrittura:
 - – la scrittura non è permessa
 - **w** la scrittura è permessa
- Il terzo carattere di ogni insieme indica il permesso relativo alla esecuzione:
 - – la esecuzione non è permessa
 - **x** la esecuzione è permessa

Codifica dei permessi per le dir

- Il significato di **r**, **w**, e **x** per le directory è il seguente:
 - **r** è permesso leggere il contenuto delle directory
 - **w** è permesso modificare il contenuto delle directory
 - **x** è permesso usare pathname che contengono la directory

Cambiare i permessi

- Cambiare il proprietario di un file o una directory

```
chown [-opzioni...] nuovo_utente file ...
```

- Cambiare il gruppo di un file o una directory

```
chgrp [-opzioni...] nuovo_gruppo file ...
```

- Cambiare i permessi di un file o una directory

```
chmod [-opzioni...] modifica_permessi file ...
```


Cambiare i permessi: esempi

Il comando `chmod` permette di cambiare i permessi con

- operatore di assegnazione (=)

Esempio:

```
$ chmod u=rwx miofile
$ chmod go= miofile
$ chmod a=rx miofile
```

NB:
“a” : all (tutti)

- operatori di aggiunta (+) e eliminazione (-)

Esempio:

```
$ chmod go-rx miofile
$ chmod a+rx miofile
```

- codifica numerica: “111” = “001001001” = “--x--x--x”
“321” = “011010001” = “-rx-r---x”

....

Esempio:

```
$ chmod 000 miofile
$ chmod 777 miofile
```

File di tipo pipe: premessa

Redirezione

- Un programma Linux per gestire i dati di un file deve richiedere al sistema di aprire un **flusso** di comunicazione tra il programma e il file.

- **Flusso di input**

Il programma può solo leggere il contenuto del file.

- **Flusso di output**

Il programma può solo scrivere nel file.

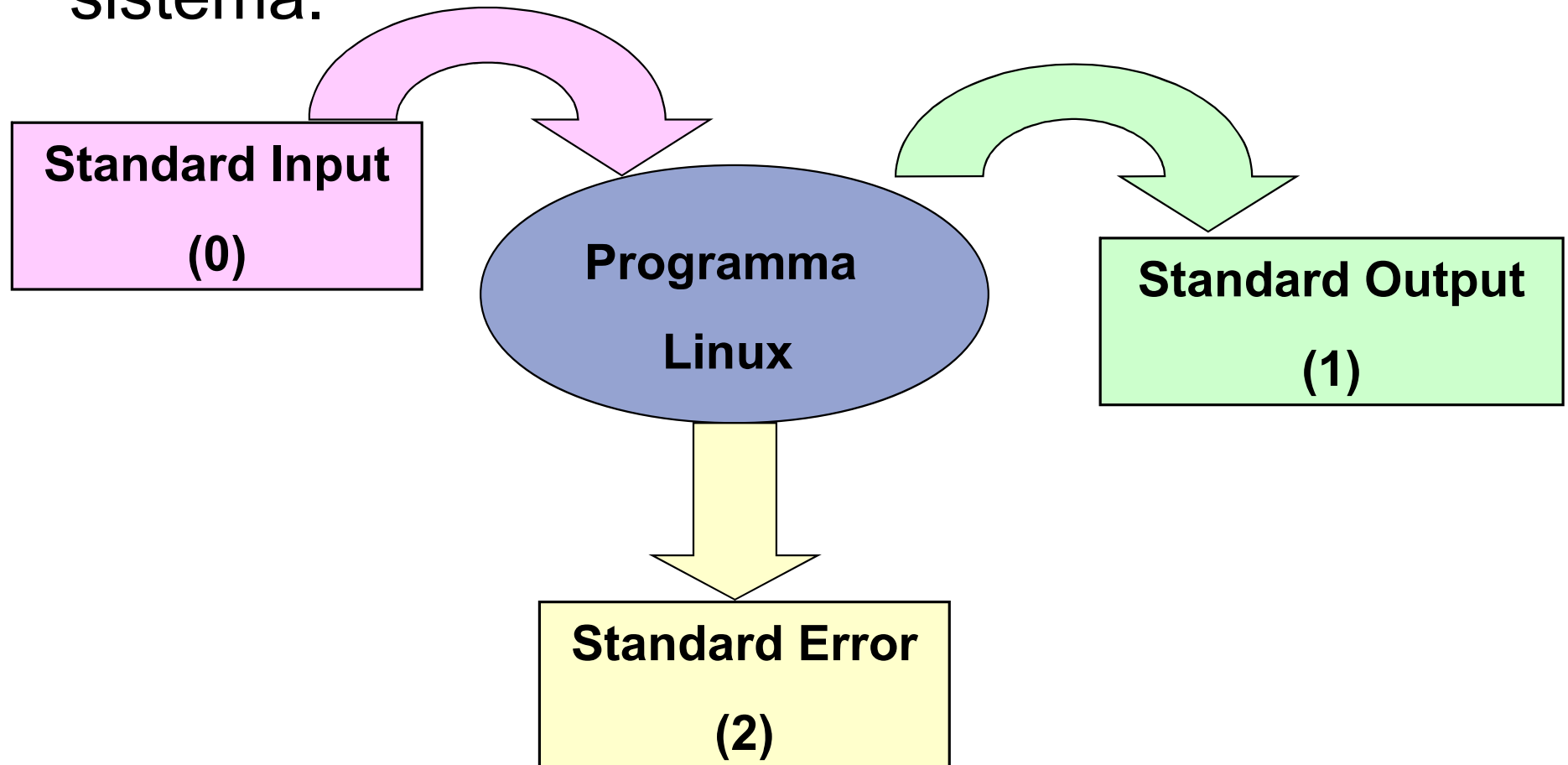
- **Flusso di input/output**

Il programma può sia leggere che scrivere nel file.

File di tipo pipe: premessa

Redirezione (2)

- Un programma Linux quando viene eseguito ha sempre tre flussi preventivamente aperti dal sistema.



File di tipo pipe: premessa

Redirezione (3)

- I tre flussi sempre aperti di un programma si possono redirigere da o verso un file.

1) Un programma può leggere le informazioni di cui necessita da un file piuttosto che da tastiera (standard input).

```
$ bc < conto
```

2) Un programma può scrivere le informazioni che produce su un file piuttosto che sul video (standard output).

```
$ ls -l > lista
```

```
$ ls -l >> lista
```

3) Un programma può scrivere i messaggi di errore su un file piuttosto che sul video (standard error).

```
$ ls -l /qkxq 2> lista
```

File di tipo pipe: premessa

Convogliamento (piping)

- Linux possiede un meccanismo di comunicazione tra due processi che permette di convogliare direttamente l'uscita di un file ottenuta da un processo sull'ingresso di un file in un altro processo.
- Risorsa che permette questo tipo di comunicazione: **pipe** (condotto).

```
$ ls -l /bin |more  
$ ls -l /bin |grep a|more
```

- La **pipe** permette solo uno scambio di dati unidirezionale.
- I processi possono avere più **pipe** aperte contemporaneamente.
 - ➔ Si possono realizzare comunicazioni bidirezionali.

File di tipo link

- Lo scopo dei link è potersi riferire a file e directory tramite **due** o **più pathname** (link nella home ad un file usato spesso e con path molto lungo)
- Tipi di link:
 - **hard link**: nell' i-node di un file è memorizzato il n. di riferimenti al file. Quando si aggiunge un link a quel file, il n. di riferimenti viene incrementato, e tutte le operazioni su uno dei due file si riflette anche sull'altro. Non può essere usato per le cartelle.
 - **soft link** (o **link simbolici**): file speciali che contengono un pathname. Quando in un comando si usa un link simbolico per riferirsi a un file, il sistema individua il file sostituendo il pathname nel comando.

Creazione di link

- Sintassi del comando per un hard link

```
ln [-opzioni] nomefile nomelink
```

- Sintassi del comando per un soft link

```
ln -s [-opzioni] nomefile nomelink
```

Visualizzazione dei link

- Con il comando `$ ls -l` vengono visualizzate informazioni sul numero di link per file e directory e sulla natura del file

```
Carlo@your-ae2c3fc363 ~  
$ ls -l  
total 2  
lrwxrwxrwx 1 Carlo Nessuno 5 Jul 2 09:44 link1 -> temp1  
-rwxr-xr-x 2 Carlo Nessuno 0 Jul 2 09:47 linkhardaprova  
lrwxrwxrwx 1 Carlo Nessuno 9 Jul 2 10:03 linksoftaprova -> prova.txt  
-rwxr-xr-x 2 Carlo Nessuno 0 Jul 2 09:47 prova.txt  
drwxr-xr-x 2 Carlo Nessuno 0 Jun 8 10:33 temp1
```


Eliminazione di link

- Con il comando `$ rm nomelink` è possibile cancellare un link
- Nel caso di **hard link**: il comando provoca un **decremento del numero di riferimenti** nell' i-node del file collegato. Quando questo numero assume valore zero, il file è **rimosso dal disco** e l' i-node viene reso disponibile per altro utilizzo
- Nel caso di **soft link**: il comando provoca la cancellazione unicamente del pathname sostitutivo e **mai di file o directory** a cui il link si riferisce

File di testo

- Per file di testo si intende un file che contiene semplicemente caratteri ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange, ovvero *Codice Standard Americano per lo Scambio di Informazioni*).
- Si noti che un file prodotto con un elaboratore di testi “**evoluto**”, cioè con formattazione, non è un semplice file di testo, in quanto contiene svariate informazioni in più (tipi dei caratteri, dimensione dei caratteri, ecc.).
- Spesso identificati dall'estensione “.txt” ma non è un obbligo. Infatti, per esempio, anche i file contenenti il codice sorgente dei programmi sono file di testo, ma assumono estensioni diverse a seconda del linguaggio di programmazione utilizzato (.c, .cpp, ecc.).

Operazioni su file di testo

- Visualizzare file corti o parte finale

```
cat [opzioni...] [file ...]
```

- Visualizzare file lunghi con comando di avanzamento

```
more [opzioni...] [file ...]
```

- Visualizzare file lunghi con comandi di scorrimento avanti/indietro

```
less [opzioni...] [file ...]
```

- Ricerca di parole, frasi o espressioni regolari in uno o più file

```
grep [opzioni...] expr [file ...]
```



Ambiente shell

Testo di riferimento:

V. Manca

“Metodi Informazionali”

Bollati Boringhieri

La shell

- Mezzo principale tramite il quale l'utente può interagire con il computer.
- Offre un insieme di funzionalità che costituiscono un ambiente operativo che permette all'utente di lavorare.
- Linux ha diversi tipi di shell
 - Shell di riferimento: **bash**

Istruzioni e variabili della shell

- La **bash** accetta, oltre ai comandi come quelli visti in precedenza, un certo numero di istruzioni.
- Ogni istruzione:
 - inizia con una parola chiave
 - può avere uno o più argomenti
 - viene chiusa da un ritorno a capo o da ;

(Eccezione: istruzione di assegnamento)

Esempi:

```
$ echo esempio di echo  
esempio di echo  
$
```

```
$ echo esempio1; echo esempio2  
esempio1  
esempio2  
$
```

Istruzioni e variabili della shell

- La **bash** accetta, oltre ai comandi come quelli visti in precedenza, un certo numero di istruzioni.
- Ogni istruzione:
 - inizia con una parola chiave
 - può avere uno o più argomenti
 - viene chiusa da un ritorno a capo o da ;(Eccezione: istruzione di assegnamento)

Esempi:

```
$ echo esempio di echo  
esempio di echo  
$
```

```
$ echo esempio1; echo esempio2  
esempio1  
esempio2  
$
```

Istruzioni e variabili della shell

- La **bash** accetta, oltre ai comandi come quelli visti in precedenza, un certo numero di istruzioni.
- Ogni istruzione:
 - inizia con una parola chiave
 - può avere uno o più argomenti
 - viene chiusa da un ritorno a capo o da ;

(Eccezione: istruzione di assegnamento)

Esempi:

```
$ echo esempio di echo  
esempio di echo  
$
```

```
$ echo esempio1; echo esempio2  
esempio1  
esempio2  
$
```


Istruzioni e variabili della shell

- La **bash** accetta, oltre ai comandi come quelli visti in precedenza, un certo numero di istruzioni.
- Ogni istruzione:
 - inizia con una parola chiave
 - può avere uno o più argomenti
 - viene chiusa da un ritorno a capo o da ;(Eccezione: istruzione di assegnamento)

Esempi:

```
$ echo esempio di echo  
esempio di echo  
$
```

```
$ echo esempio1; echo esempio2  
esempio1  
esempio2  
$
```

Istruzioni e variabili della shell

- La **bash** accetta, oltre ai comandi come quelli visti in precedenza, un certo numero di istruzioni.
- Ogni istruzione:
 - inizia con una parola chiave
 - può avere uno o più argomenti
 - viene chiusa da un ritorno a capo o da ;

(Eccezione: istruzione di assegnamento)

Esempi:

```
$ echo esempio di echo  
esempio di echo  
$
```

```
$ echo esempio1; echo esempio2  
esempio1  
esempio2  
$
```

Modalità di funzionamento shell

- La shell ha tre modalità di funzionamento:

- **Interattiva:**

- La shell attende i comandi digitati dall'utente.

- **Di configurazione:**

- La shell viene utilizzata per definire variabili e parametri d'utente e di sistema.

- **Di programmazione:**

- La shell viene adoperata per realizzare **procedure**, dette **script**, contenenti costrutti di comandi/istruzioni di GNU/Linux.

Esempio di script bash (1)

#!/bin/bash

Questo è un commento dello script. La riga soprastante, invece, indica che
questo file contiene uno script bash.

La riga seguente stampa a video una descrizione dello script

echo "Script bash che elenca il contenuto della cartella corrente"

echo " " **#** questa istruzione inserisce semplicemente una riga vuota

ls -l