

## Laboratorio di Sistemi Operativi – I parte

---

Presentazione del corso - A.A. 2003-2004

Prof. Marco Lapegna

- gruppo 1 - cognomi A → Co
- tel. 081 675623
- studio 155 DMA (VI liv.)
- <http://www.docenti.unina.it/marco.lapegna>

## Avviso importante

---

- corsi di base ICT universita' – regione campania  
**Sistemista LINUX**
- riconosciuti **8 crediti**
- borsa di studio finale **500 Euro**
- **iscriversi elettronicamente sul sito**  
<http://www.unina.it> e leggere bene il bando
- scadenza bando **OGGI ALLE 17.00**

## Obiettivi del corso

---

Fornire **metodologie e strumenti per lo sviluppo di applicazioni in ambiente UNIX**, fornendo i fondamenti delle interfacce di programmazione standard e delle system call a UNIX



Rapporto teoria/pratica = 1:2

## Bibliografia

---

- Kerninghan , Ritchie – Linguaggio C – Jackson libri
- McGilton , Morgan – il sistema operativo UNIX – McGraw Hill
- Stevens – Advanced programming in the Unix environment – Addison Wesley

## Prerequisiti consigliati

---

- Programmazione + lab. A e B
- Architettura degli elab. + lab. A e B
- conoscenza del linguaggio C

Ma soprattutto ...

- seguire i corsi

## Esame

---

Prova di laboratorio

Orale congiunto con il corso di Sistemi Operativi

Voto:

- |                     |   |                     |
|---------------------|---|---------------------|
| – prova pratica LSO | } | 1 voto<br>9 crediti |
| – scritto SO        |   |                     |
| – orale SO + LSO    |   |                     |

## Iscrizione al corso

---

Inviare una e-mail con

- subject: **LSO: ISCRIZIONE**
- testo: nome , cognome , matricola

All'indirizzo **marco.lapegna@dma.unina.it**

## RICHIESTE di cambio di gruppo

---

Inviare una e-mail **entro il 23/3**

- subject: **LSO: CAMBIO GRUPPO**
- testo: nome , cognome , matricola  
dal gruppo x al gruppo y

All'indirizzo **marco.lapegna@dma.unina.it**

Se il saldo tra gli studenti uscenti e quelli entranti e' minore del 10% del numero di iscritti, i cambi saranno accettati e comunicati via e-mail

## Cosa e' un sistema operativo

Chi ha mai usato un computer **senza sistema operativo?**



Un **sistema operativo e' un software** che agisce da intermediario tra utente e hardware

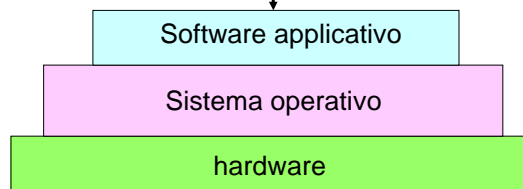
## Compiti di un sistema operativo

- permettere l'esecuzione dei programmi degli utenti
- rendere semplice l'uso del calcolatore
- permettere un uso efficiente dell'hardware



**Strumento indispensabile** nella risoluzione di problemi reali con il calcolatore

## Ruolo del sistema operativo



## Cosa fa un sistema operativo

Tutti i **sistemi operativi** forniscono **servizi**

Esempi:

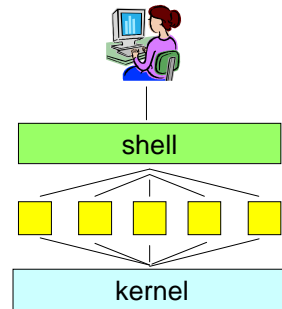
- **Eeguire** un programma,
- **Aprire** un file,
- **Allocare** la memoria

...

## Fase di login

Dopo la fase di login si interagisce con un **interprete dei comandi** (*shell*)

Una *shell* è un **interprete** che accetta l'input dell'utente ed esegue i **comandi**.



## Shell

Tra le shell più utilizzate ci sono:

- La **Bourne shell**, /bin/sh
- La **C shell**, /bin/csh
- La **Korn shell**, /bin/ksh
- La **Basic shell**, /bin/bash

Ciascun utente ha una **shell** che viene attivata **di default**.

## Classi di comandi di Unix

- Gestione dei file e delle directory
- Sviluppo software
- Elaborazione testi
- Amministrazione del sistema
- Comunicazione
- ...

## Formato dei comandi

**>> comando [ argomento ...]**

Gli argomenti possono essere:

- **opzioni** o **flag** (-)
- **parametri**

separati da almeno un separatore

### Esempio

**>> ls -l -F file1 file2 file3**

Forme equivalenti:

**>> ls -F -l file1 file2 file3**

**>> ls -lF file1 file2 file3**

**>> ls -lF file1 file2 file3**

## Comando man

Unix ha un **manuale di riferimento**, accessibile "in linea", mediante il comando **man**

Il manuale è organizzato in **sezioni** e **sottosezioni**; ogni sezione è composta di **pagine**

Ogni pagina descrive **un singolo argomento** (es.: un comando)

## Sezioni del manuale

Sezione	Contenuti
1	Commands
2	System Calls
3	Library Functions
4	Administrative Files
5	Miscellaneous Information
6	Games
7	I/O and Special Files
8	Maintenance Commands

## Il comando man

```
>> man [ opzione...] titolo...
```

Visualizza le pagine del manuale specificate mediante i suoi parametri

-s permette di specificare la sezione

## esempio

```
%man who
....
%man -s 2 kill
....
%man -s 2 intro
```

### Note:

- Se il numero di sezione non è specificato, viene selezionata la prima occorrenza
- Ogni sezione o sottosezione inizia con una pagina chiamata **intro**

```
% man man
man(1) man(1)
NAME
man - format and display the on-line manual pages
manpath - determine user's search path for man pages

SYNOPSIS
man [-acdfhkkW] [-m system] [-p string] [-C con-
fig_file] [-M path] [-P pager] [-S section_list] [section]
name ...

DESCRIPTION
man formats and displays the on-line manual pages. This
version knows about the MANPATH and (MAN)PAGER environment
variables, so you can have your own set(s) of personal man
pages and choose whatever program you like to display the
formatted pages.
<omissis>
```

```
OPTIONS
-C config_file
Specify the man.conf file to use; the default is
/etc/man.config. (See man.conf(5).)

<omissis>
OPERANDS
The following operand is supported:

name A keyword or the name of a standard utility.

USAGE
Manual Page Sections
Entries in the reference manuals are organized into sec-
tions. A section name consists of a major section name,
typically a single digit, optionally followed by a subsec-
tion name, typically one or more letters.
<omissis>
```

```
ENVIRONMENT
See environ(5) for descriptions of the following environment
<omissis>
EXIT STATUS
The following exit values are returned:

0 Successful completion.

>0 An error occurred.

FILES
/usr/share/man root of the standard manual
page directory subtree
<omissis>
SEE ALSO
apropos(1), cat(1), col(1), eqn(1), more(1), nroff(1),
refer(1), tbl(1), troff(1), vgrind(1), whatis(1),
catman(1M), attributes(5), environ(5), eqnchar(5), man(5)
```

## Gestione delle directory

<b>pwd</b>	<b>p</b> rint <b>w</b> orking <b>d</b> irectory
<b>cd</b>	<b>c</b> hange <b>d</b> irectory
<b>ls</b>	<b>l</b> ist directory
<b>du</b>	<b>d</b> isk <b>u</b> sage
<b>mkdir</b>	<b>m</b> ake <b>d</b> irectory
<b>rmdir</b>	<b>r</b> emove <b>d</b> irectory
<b>ln</b>	<b>l</b> ink
...	

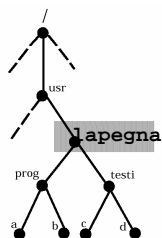
## Il comando pwd

### pwd

"print working directory"  
stampa il path della directory corrente

### Esempio:

```
% pwd
/usr/lapeгна
%
```



## Il comando cd

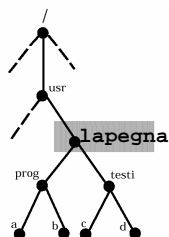
### cd [ directory ]

"change directory"

- la directory specificata diviene la **working directory**
- se nessuna directory è specificata, si "ritorna" alla home directory

## esempio

```
%cd /usr
%pwd
/usr
%cd
%pwd
/usr/lapeгна
%
```



## Il comando ls

### ls [ options ][ directory... ]

"list directory"

- lista (in ordine alfabetico) il contenuto della o delle directory indicate
- se nessuna directory è indicata, lista il contenuto della working directory
- possiede numerose opzioni

## Opzioni del comando ls

- s fornisce la dimensione in blocchi (size)
- t lista nell'ordine di modifica (prima il file modificato per ultimo) (time)
- l un nome per ogni riga
- F aggiunge / al nome delle directory e \* al nome dei file eseguibili
- R si chiama ricorsivamente su ogni sottodirectory
- i fornisce l'i-number del file
- e molte altre!

## esempio

```
% ls
dir1 file1
% ls -s
total 4 2 dir1 2 file1
% ls -t
file1 dir1
% ls -l
dir1
file1
% ls -F
dir1/ file1
% ls -R
dir1 file1
./dir1:
file1 file2 file3 file4
```

## I file nascosti

I file il cui nome inizia con "." vengono listati solo specificando l'opzione **-a** ("all")

### Esempio

```
% ls -a
. .cshrc .mailrc dir1
.. .login .sh_history file1
%
```

## Il comando du

**du [ options][ name...]**

"disk usage"

- stampa il numero di blocchi contenuti in tutti i file e (ricorsivamente) directory specificate
- se **name** non è specificato, si intende la directory corrente
- **-s**: solo il totale

## esempio

```
% du
2 ./dir1
2 ./dir2
14 .
% du -s ..
198812 ..
%
```

## Il comando mkdir

**mkdir directory...**

"make directory"

- crea la/le directory indicata/e

### Esempio

```
% mkdir dir1 dir2
% ls
dir1 dir2
%
```

## Il comando rmdir

**rmdir directory**

"remove directory"

- rimuove la/le directory indicata/e
- le directory devono essere vuote

### Esempio:

```
% rmdir dir
rmdir: dir: Directory not empty
% ls dir
a
% rm dir/a
% rmdir dir
%
```

## Il comando ln

**ln name1 name2**

"link"

- associa il nuovo nome (link) **name2** al file (esistente) **name1**, che non può essere una directory

### Esempio

```
% ln b d
%
```

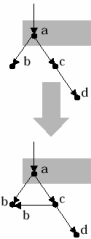


### Il comando ln (cont.)

In name1 name2  
•se name2 è una directory, il nuovo nome è name2/name1

#### Esempio

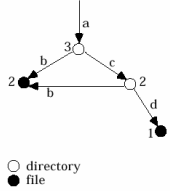
```
% ln b c
%
```



### Numero di link

È uno degli attributi dei file gestiti dal sistema  
E' il numero di volte che si fa riferimento all'i-number

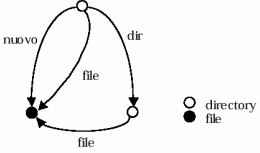
#### Esempio



Per vedere il numero di link: **ls -li**

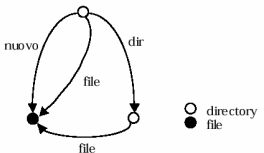
### esempio

```
% mkdir dir
% touch file
% ls -li
total 2
drwxr-sr-x 2 lapegna staff 512 Mar 11 19:40 dir
-rw-r--r-- 1 lapegna staff 0 Mar 11 19:40 file
% ln file nuovo
% ls -li
total 0
-rw-r--r-- 3 mariog staff 0 Mar 11 19:40 file
% ln dir nuovissimo
ln: dir is a directory
%
```



### Esempio

```
% ls -li
total 2
drwxr-sr-x 2 mariog staff 512 Mar 11 19:40 dir
-rw-r--r-- 2 mariog staff 0 Mar 11 19:40 file
-rw-r--r-- 2 mariog staff 0 Mar 11 19:40 nuovo
% ln file dir
% ls -li
total 0
-rw-r--r-- 3 mariog staff 0 Mar 11 19:40 file
% ln dir nuovissimo
ln: dir is a directory
%
```



## Note

- Tutti i **link** allo **stesso file** hanno identico status e caratteristiche
- Non è possibile distinguere la entry originaria dai nuovi link
- I **link** di questo tipo **non** possono essere fatti con file che si trovano su **filesystem diversi**

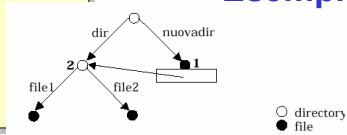
## Link simbolici

### In -s name1 name2

- Permette di creare link a directory e link fra file o directory che stanno su file system diversi
- Viene creato un file **name2** che contiene il link simbolico (path di **name1**)

## Link simbolici

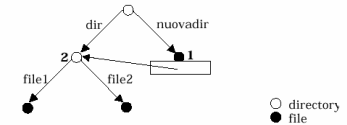
```
% ls
dir
% ls dir
file1 file2
% ln -s dir nuovadir
% ls
dir nuovadir
% ls nuovadir
file1 file2
```



## Link simbolici

```
% ls -l
total 4
drwxr-sr-x 2 mariog staff 512 Mar 11 19:24 dir
lrwxrwxrwx 1 mariog staff 3 Mar 11 19:24 nuovadir -> dir
%
```

### Esempio



## Gestione dei file

<b>mv</b>	<b>move</b> file
<b>cp</b>	<b>copy</b> file
<b>rm</b>	<b>remove</b> file
<b>touch</b>	modifica data e ora dell'ultimo accesso/modifica di un file
<b>find</b>	cerca file con specificati attributi
...	

## Il comando mv

**mv [ options] name...target**

"move"

- **muove** il file o directory **name** sotto la directory **target**
- se **name** e **target** non sono directory, il **contenuto** di **target** viene **sostituito** dal contenuto di **name**

## Il comando mv

```
% ls
file1 file2 targetdir
% mv file1 file2 targetdir
% ls
targetdir
% ls targetdir
file1 file2
% mv targetdir/file1 targetdir/file2 .
% ls
file1 file2 targetdir
```

*targetdir è una directory!*

**Esempio**

## Il comando mv

```
% ls
file1 file2 file3 targetfile
% mv file1 targetfile
% ls
file2 file3 targetfile
% mv file2 file3 targetfile
mv: Target targetfile must be a directory
Usage: mv [-f] [-i] f1 f2
mv [-f] [-i] f1 ... fn d1
mv [-f] [-i] d1 d2
```

*targetdir è un file!*

**Esempio**

## Il comando mv

```
% ls
file1 file2
% mv file1 file2 target
mv: target not found
% mv file1 target
% cat target
contenuto di file1
%
```

*target  
non esiste!*

**Esempio**

## Il comando cp

**cp [ options][ name...] target**  
"copy"  
come **mv**, ma **name** viene  
copiato

```
% ls
file1 file2 targetdir
% cp file1 file2 targetdir
% ls . targetdir
.:
file1 file2 targetdir
targetdir:
file1 file2
% cd targetdir
% ls
file1 file2
% cp file1 targetfile
% ls
file1 file2 targetfile
%
```

**Esempio**

## Il comando rm

**rm [-r] name**

"remove"

- **rimuove** i file indicati
- se un file indicato è una directory:

messaggio di errore, a meno che non sia specificata l'opzione **-r**

- ... nel qual caso, rimuove ricorsivamente il contenuto della directory
- <altri parametri>

## Il comando touch

**touch [ options][ time] filename...**

- aggiorna la data e l'ora dell'ultimo accesso (opzione **-a**) o dell'ultima modifica (opzione **-m**) di filename (default: **-am**)
- se **time** non è specificato, usa la data e l'ora corrente
- se il file non esiste, lo crea

**Esempio:**

```
% touch 01281738 file1
% ls -l
total 0
----r--r-- 1 mariog staff 0 Jan 28 17:38 file1
```

## Metacaratteri

La shell fornisce un meccanismo che permette di specificare una lista di nomi di file mediante una singola espressione sintetica

```
% ls
gianni giorgio laura mario
% rm g*
% ls
laura mario
% rm *
%
```

**Esempio:**

*Rimuove tutti i file presenti nella directory corrente che iniziano con g*

## Metacaratteri

<b>?</b>	un carattere qualsiasi
Es. <b>rm file?</b>	
<b>*</b>	una stringa di Ø o più caratteri qualsiasi
Es. <b>rm file*</b>	
<b>[...]</b>	uno qualsiasi dei (singoli) caratteri racchiusi fra <b>[ ]</b> (alternativa)
Es. <b>rm file[123]</b>	
<b>rm file[a-z]</b>	(subrange)
<b>N.B.</b>	
• il "." iniziale deve essere sempre specificato	
• per usare il carattere del metacaratteri, occorre anteporre \	

## Il comando find

**find pathname ... [ expression ]**

discende ricorsivamente le directory specificate (**pathname...**), cercando tutti i file che rendono vera **expression**.

Molto flessibile:

- **ricerca** file di specificati **attributi** (nome, tipo, permessi, proprietario, gruppo, numero di link, dimensione, data di ultima modifica/accesso ...)
- **and, or, not** di attributi
- può **eseguire** automaticamente, o previa conferma, uno o più **comandi** sui file individuati

## Esempio

Rimuovi tutti i file nella working directory (o più sotto) di nome **a.out** il cui ultimo accesso è anteriore a 7 giorni:

```
find . -name a.out -atime +7 -exec rm {} \;
```

Come si interpreta:

<b>.</b>	directory da cui iniziare la ricerca
<b>-name a.out</b>	nome del file da cercare
<b>-atime +7</b>	data di accesso superiore a 7 giorni
<b>-exec rm {} \;</b>	esegui il comando <b>rm</b> sul file trovato (\; carattere terminazione)