Cos'è Unix?

Unix è un sistema operativo, **nato nel 1969** presso i Bell Labs ad opera di Dennis Ritchie, Ken Thompson, Brian Kernighan ed altri programmatori.

Inizialmente chiunque fosse interessato e possedesse l'hardware occorrente, poteva chiedere ad un costo irrisorio un nastro del software ed i relativi manuali stampati.

Questo accadeva prima dell'avvento dei personal computer, pertanto si trattava in genere di università e centri di ricerca. I singoli centri modificavano il codice sorgente ampliando e personalizzando il sistema in base alle loro necessità.

Una tappa importante è stata raggiunta alla fine degli anni '70 con la realizzazione della versione BSD (Berkley System Distribution) ad opera di alcuni esperti di informatica dell'Università della California di Berkley, che apportarono diverse migliorie, la più importante fu l'adozione del protocollo TCP/IP.

Il codice sorgente venne reso pubblicamente disponibile con una licenza che ne consentiva la distribuzione con o senza codice sorgente, a condizione che queste parti del codice venissero ascritte a Berkley

Evoluzione di Unix

La popolarità di Unix aumentò con il trascorrere degli anni.

La Berkley cedette ad AT&T i diritti sul software e Unix divenne un prodotto commerciale, con costo elevato ed il codice sorgente non era incluso.

Anche acquistando separatamente una copia dei sorgenti, non era più possibile modificarli e condividere le migliorie apportate con altri programmatori.

Altre società commerciali adottarono la modalità di distribuzione del software senza sorgenti, ponendo le basi di un nuovo modello di sviluppo proprietario.

Nel 1984 Richard Stallman, invece di cedere a questa nuova tendenza, decise di dare vita ad un nuovo sistema operativo di tipo Unix il cui codice sorgente potesse essere liberamente copiato e modificato. Nacque il progetto **GNU** (GNU is Not Unix).

Il nuovo modello di sviluppo prese il nome di **Software Libero** (free software).

Venne scritta una licenza specifica **GNU General Public License** (nota come GPL, http://www.gnu.org/copyleft/gpl.txt) che aggirasse i limiti imposti dai diritti d'autore e consentisse a chiunque di copiare e modificare un lavoro, seppur nel rispetto di condizioni e termini rigorosi. È anche nota come **licenza copyleft** in contrapposizione al più noto copyright.

Linux

Agli inizi degli anni '90, Linus Torvald, uno studente finlandese in scienze dell' informazione iniziò ad apportare variazioni a Minix, un sistema operativo di tipo Unix per personal computer allora utilizzato nei corsi universitari sui sistemi operativi.

Torvald decise di migliorare il componente principale del software alla base di Minix, chiamato kernel, e di scriverne uno proprio.

Alla fine del 1991, Torvald pubblicò la prima versione di questo kernel su Internet e la

battezzò “Linux”, un gioco di parole basato sul suo nome e Minix.

La forza di questo progetto fu l'adozione della licenza GNU GPL, in questo modo Linux risultava un software che poteva essere liberamente utilizzato, copiato e modificato da chiunque, a condizione che le stesse libertà fossero estese a tutte le copie e le varianti.

Nel tempo e da tutto il mondo migliaia di programmatori sparsi sull'intero pianeta contribuirono al suo progetto e Linux è diventato un sistema operativo completo, moderno e che può essere utilizzato sia da programmatori che da non addetti ai lavori.

Cos'è il kernel ?

Per kernel si intende il cuore di un sistema operativo.

Il codice che gestisce le risorse presenti sul sistema e le rende disponibili alle applicazioni.

Il kernel si occupa principalmente di gestire:

- le comunicazioni con l'hardware del sistema (device driver)
- i file system e la memoria
- l'accesso alle risorse da parte dei processi (le applicazioni eseguite sul sistema)

Le versioni del kernel Linux sono identificate con numeri dal significato ben preciso.

Per esempio il kernel **2**.**6**.**22** (**Major Number**.**Minor Number**.**Revision**) uno degli ultimi rilasciati ha:

Il **Major Number identifica** il valore più alto della revisione del kernel.

Il Minor Number

- **Se Pari** il kernel viene considerato **stable** e pronto per sistemi in produzione
- **Se Dispari** si considera in **development** da usare con cautela o per sperimentazione.

Le release stable sono sempre figlie delle devel precedenti.

Kernel Monolitico o modulare ?

- **Monolitico**: E' un singolo file binario eseguibile in modalità "kernel" che contiene il gestore del processo, della memoria, del sistema, device driver ecc.. Esempi di tali sistemi sono UNIX, Linux, MS-DOS.
- **Modulare:** si intende un kernel, con la capacità di caricare o scaricare parti di codice (moduli) secondo necessità e richieste.

Vantaggi e svantaggi

- Il kernel monolitico è più veloce, poiché tutto il codice è già stato caricato al bootstrap dell'OS, ma di contro occupa maggiori risorse del sistema. Un altro punto a favore è la maggiore stabilità: non richiede moduli evitando così pericolose dipendenze.
- Il kernel modulare è quello utilizzato da tutte le distribuzioni LINUX.

Principali Distribuzioni

Distribuzione	Note	Web
RedHat	Orientata ad applicazioni di tipo Enterprise	www.redhat.com
Fedora 7	Versione Open Source derivata da RedHat	fedoraproject.org
Mandriva 2007 Spring	ex Mandrake ha dovuto cambiare nome per problemi di copyright	www.mandriva.org
S.U.S.E 10.2	Acquisita da Novell. Tramite Novell Open enterprise Server supporta i servizi Netware.	www.suse.com
Slackware 12	Una delle prime amata dai “puristi”	www.slackware.co
Debian 4.0 Rev 1	Storica	www.debian.com
Ubuntu 7.04	Preinstallata da DELL (Acer e Asus a breve)	www.ubuntu.com
Kubuntu 7.04	KDE come Desktop Environmet	www.kubuntu.com
Linspire	Pubblicizzata come la più facile da usare	www.linspire.com

Directory radice e /bin

Il file system ha origine in **/**, detta **root** o directory radice. Si nota esplicitamente che root è anche la **username** dell'amministratore di sistema, la cui home directory è **/root**.

- **/bin:** è la directory che contiene i programmi principali del sistema, quelli che devono essere disponibili subito all'accensione, per poter avviare il sistema.
- **/home:** questa directory contiene le 'aree locali' dei singoli utenti. Quando un utente si collega facendo login, egli non si trova nella directory principale del sistema (la root /) ma viene posizionato in genere in una directory "privata", diversa per ogni utente. Questa directory, detta home, in genere ha il *nome dell'utente ed è una sottodirectory della directory home*.
- **/usr:** in questa directory risiede la maggior parte del sistema. In usr e nelle sue sottodirectory risiedono tutti i programmi installati, i file del manuale, la documentazione ed altro ancora. Una caratteristica di usr è che i suoi file in genere possono essere solo letti.
- **/sbin** e **/usr/sbin** sono analoghe a /bin e /usr/bin, la **s** staper **Superuser** ed è riferita al root. Queste directory, infatti, contengono soprattutto comandi usati per l'amministrazione di sistema.
- **/etc**: è una delle più importanti del sistema perché contiene la quasi totalità dei file di configurazione del sistema, compresi quelli che servono per la fase di accensione.
- **/lib:** Questa directory contiene le librerie di sistema, cioè quegli archivi di funzioni utilizzati da tutti. Troveremo in questa directory le librerie di funzioni utilizzate dal linguaggio C, richiamate da tutti i programmi compilati in questo linguaggio, ma anche varie librerie utilizzate da altri linguaggi o sistemi.
- **/tmp**: in questa directory vengono immagazzinati i file temporanei durante le elaborazioni. Il contenuto di questa directory viene cancellato ad ogni accensione.
- **/var** contiene, nelle sue sottodirectory, i file che registrano gli **eventi** del sistema (i cosiddetti **log**). Oltre ai log in questa directory troviamo le code di stampa ed in genere tutti i file di sistema che vengono modificati.

PRINCIPALI COMANDI LINUX

Un comando può avere una serie di opzioni che solitamente sono indicate con il segno "-"

<**comando**><-opzione1><-opzione2> ...
Ogni spazio bianco separe il comando da ogni opzione

Esempio

```
$ ls -l
```

CD: CAMBIO DIRECTORY (CARTELLA)

cd <**directory**> **consente di cambiare la directory corrente in quella indicata in <directory> (change directory)**

$ cd Desktop	se la directory corrente è home porta nella directory Desktop
$ cd ..	porta della directory superiore (o padre)
$ cd /var	in qualunque directory ci si trovi porta nella directory /var
$ cd ~	porta della directory home dell'utente *(/homs/[nome utente])*

PWD: MOSTRA LA DIRECTORY CORRENTE

pwd: questo comando ci permette di sapere in che directory ci troviamo

Input e output redirect in Linux

Come effettuare il redirect dei flussi stdin, stdout e stderror in Linux: strumenti, descrittori di file, sintassi ed esempi.
I sistemi operativi UNIX based forniscono tre diversi canali input e output:

1. stdin
2. stdout
3. stderr

Essi rendono possibile la comunicazione tra un programma e l'ambiente in cui esso viene eseguito.

Nella shell Bash questi canali vengono numerati rispettivamente con 0 (stdin), 1 (stdout) e 2 (stderr), e vengono chiamati descrittori di file. È possibile manipolarli, copiarli, leggerli o scrivere su di essi, esattamente come per qualunque altro file memorizzato nel sistema.

Quando parte un ambiente Bash, tutti e tre i file descriptor di default puntano al terminale nel quale è stata inizializzata la sessione:

1. l'input (stdin – 0) corrisponde a ciò che viene digitato nel terminale;
2. l'output standard (stdout – 1) per i messaggi tradizionali viene inviato al terminale
3. l'output di errore (stderr – 2) per i messaggi di errore viene inviato al terminale.

Il redirect (redirezione) Input/Output dei comandi Linux consente di convogliare l'output di un comando in un file, oppure il contenuto di un file in un comando, oppure ancora l'output di un comando verso un altro comando, consentendo un notevole risparmio di tempo.

Redirect dello stdout

Nel sistema operativo Linux lo standard output di default è il terminale, che viene mostrato sullo schermo. Viene indicato con l'abbreviazione stdout o con il descrittore di file numero 1.

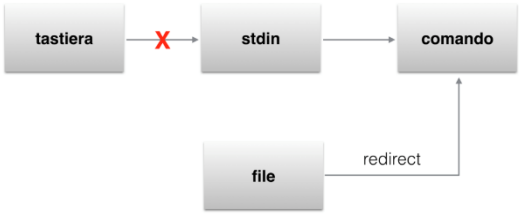
È possibile reindirizzare l'output verso altri dispositivi, ad esempio, in un file.
Per reindirizzare l'output si utilizza il simbolo **>** oppure **>>.**
Ad esempio, piuttosto che mostrare sullo schermo l'output del comando **ls -la**, lo reindirizzo sul file elenco.txt:

```
ls -l > elenco.txt
```

Redirect dello stdin

Lo standard input di default su Linux è la tastiera. Viene indicato con l'abbreviazione stdin o con il descrittore di file numero zero (0).

Per reindirizzare l'input si utilizza il simbolo minore (<). . Il simbolo < equivale alla forma estesa 0<. Nella forma estesa l'operatore di redirezione specifica il numero del descrittore di file da prendere in considerazione (0=standard input).




```
sort < tecnologie.txt
```

LS: LISTA IL CONTENUTO DI UNA DIRECTORY (CARTELLA)

ls <**opzioni**> [<**directory**>] visualizza il contenuto di una directory (list)

-a lista anche i file nascosti
-l mostra tutte le informazioni per ogni file (formato lungo)
-k dimensione dei file è in Kb (*normalmente è in byte*).
-f classifica i file a seconda del tipo.

$ ls -l	contenuto della directory corrente in formato lungo
$ ls -a /	contenuto della directory iniziale (root) compresi i file nascosti
$ ls -lah	contenuto della directory corrente in formato lungo

CP: COPIA FILE E DIRECTORY

cp: questo comando ci permette di copiare un file o una directory (copy)

-r copia le directory e le sottodirectory ricorsivamente

$ cp file1 cart1	copia file1 dentro la directory cart1
$ cp -r cart1 cart2	copia tutta la directory cart1 dento a cart2
$ sudo cp -a cart1 cart2	copia tutta la directory cart1 dento a cart2 mantenendo gli stessi permessi e le informazioni su data e ora di creazione

MV: MUOVE O RINOMINA UN FILE O UNA DIRECTORY

mv: con questo comando è possibile spostare file o directory (move). Può essere anche utilizzato per rinominare un file o una directory.

-i chiede la conferma

$ mv vecchio nuovo	rinomina il file vecchio in nuovo
$ mv file1 cart1	sposta il file file1 dentro la directory cart1

RM: RIMUOVE UN FILE O UNA DIRECTORY

rm con questo comando è possibile cancellare uno o più file (remove)

-rf cancella ricorsivamente sia le cartelle che il loro contenuto

$ rm file1 file2	cancella file1 e file2
$ rm *.*	**cancella TUTTO il contenuto della directory corrente**. Le eventuali directory presenti non saranno rimosse.
$ rm -rf cart1	cancella tutto il contenuto della directory cart1

RMDIR: RIMUOVE UNA DIRECTORY VUOTA

rmdir: con questo comando è possibile cancellare una o più ***directory vuote*** (remove directoy)

$ rmdir prova	cancella la directory vuota prova ***directory vuote***

MKDIR: CREA UNA DIRECTORY

mkdir: con questo comando è possibile creare una directory (make directory)

-p Crea anche eventuali directory intermedie esplicitate nei parametri dir.

$ mkdir prova	crea la directory prova
$ mkdir -p /Classi/4INF/Allievi	creare, saranno create anche le directory intermedie se esse non esistono già.

PWD: MOSTRA LA DIRECTORY CORRENTE

pwd: con questo comando è possibile sapere in quale directory ci troviamo (printworking directory)

$ pwd	Se l'utente ECOmmunication si trova nella cartella principale visualizza /home/ecommunication

Ricerca di stringhe di caratteri con grep

grep stringa file

- **-i**: ignora le differenze tra maiuscole e minuscole
- **-l**: fornisce la lista dei file che contengono il pattern
- **-n**: le linee dell'output sono precedute dal numero di linea
- **-v**: stampa solo le linee che non contengono il pattern
- **-w**: vengono restituite solo le linee che contengono il pattern/stringa come parola completa
-

Attraverso le espressioni regolari è possibile specificare dei pattern più complessi della semplice stringa contenuta.

- **^** inizio della linea
- **$** fine della linea
- **.** un singolo carattere (qualsiasi)
- **[str]** un qualunque carattere in stringa
- **[bstr]** un qualunque carattere non in stringa
- **[a-z]** un qualunque carattere tra a e z
- **** inibisce l'interpretazione del carattere successivo
- ***** zero o più ripetizioni dell'elemento precedente
- **+** una o più ripetizioni dell'elemento precedente
- **?** zero o una ripetizione dell'elemento precedente

Cerca tutte le righe che contengono questa parola anche come parte di altre parole

grep "stringa1" file

Cerca tutte le righe che contengono OR una di queste parole

grep "stringa1\|stringa2" file

Cerca tutte le righe che iniziano per b

grep '^b' lista

Cerca tutte le righe che finiscono per b

grep 'b$' lista

fornisce in output le linee del file relazione.txt che non contengono stringhe composte dai caratteri a, g, t (ogni linea `e preceduta dal suo numero)

egrep -nv '[agt]+' relazione.txt

fornisce in output le linee di tutti i file con estensione txt che contengono la parola intera ciao

grep -w ciao *.txt

Cerca tutte le righe che contengono OR una di queste parole

grep "stringa1\|stringa2" file

Estrae dal file lista tutti i gruppi di tre caratteri in cui "de" siano i primi due, ad esempio "dei", "del", "scodella" e "code" (anche gli spazi vuoti sono considerati caratteri)

grep 'de.' lista

Estrae dal file lista tutte le righe che contengono 12 122 1222 12-bello perché l'asterisco si riferisce alla presenza di 0 o più occorrenze

grep '122*' lista

Individua tutte le righe che iniziano con un punto

grep ^\.

Amministrazione degli utenti e permessi

CHMOD: MODIFICARE I PERMESSI DI UN FILE

chmod con questo comando è possibile modificare il permessi di un file.
Per un approfondimento dettagliato sui permessi vedi
Il parametro -R consente di modificare ricorsivamente i permessi delle directory indicate e del loro contenuto.

Valore binario (rwx)	Valore decimale	Permessi
100	4	solo lettura
011	3	scrittura ed esecuzione
010	2	solo scrittura
001	1	solo esecuzione
000	0	nessuno

Esempi:

- `chmod 734 nomefile`

assegna tutti i permessi all'utente, scrittura ed esecuzione per il gruppo e solo lettura per tutti gli altri.

- `chmod 777 nomefile`

assegna tutti i permessi all'utente corrente, al suo gruppo ed anche a tutti gli altri.

- `chmod -R 777 nomedirectory`

Rappresentazione simbolica

La LETTERA U rappresenta il livello di permessi per l'utente, G il livello di permessi del gruppo e O il livello di permessi generale `chmod "u=rwx","g=rx","o=x" nomeFile`

- *`chmod "o=+rx" nomefile`*
- `chmod "u=+rwx","go=-" nomefile`
- `chmod "u=+rw","go=-" nomefile`

Come creare i gruppi su Linux

Per creare un nuovo gruppo sul sistema operativo Linux si utilizza il comando groupadd.
Soltanto gli amministratori del sistema (super user) possono creare nuovi gruppi.

groupadd **nomegruppo**

sudo groupadd **INF** **#crea il gruppo INF**

Come aggiungere un utente esistente a un gruppo

Per aggiungere un utente esistente a un gruppo secondario, utilizzare il comando usermod -a -G seguito il nome del gruppo e l'utente:

```
sudo usermod -a -G groupname username
sudo usermod -a -G INF studente
```

Come rimuovere un utente da un gruppo

Per rimuovere un utente da un gruppo, utilizzare il comando **gpasswd** con l'opzione **-d**.

Nel seguente esempio stiamo rimuovendo l'utente username dal gruppo groupname

```
sudo gpasswd -d username groupname
```

Creare, modificare e cancellare gli utenti

Di default Linux dispone di un unico utente, cioè l'utente root che gode dei pieni poteri. Spetta quindi a questo utente creare nuovi utenti nel sistema.
Per aggiungere un utente si ricorre al comando useradd, questa è la sintassi:

```
useradd [opzioni] numeutente
```

Facciamo un esempio: si supponga di voler creare l'utente "pippo" e di volerlo assegnare al gruppo "visitatori"

```
useradd -G visitatori pippo
```

Per modificare il profilo di un utente si utilizza usermod, in questo modo:

```
usermod [opzioni] nomeutente
```

Si supponga di voler modificare il gruppo di appartenenza di "pippo" da "visitatori" a "colleghi":

```
usermod -G colleghi pippo
```

Per cancellare un utente si userà, invece, il comando userdel; supponendo di voler cancellare l'utente "pippo" dovremo scrivere:

```
userdel pippo
```

UID e GID

Esattamente come accade per i processi, anche utenti e gruppi sono contraddistinti da codici univoci che vengono assegnati automaticamente all'atto della loro creazione. Questi prendono il nome, rispettivamente, di **UID** (User Identifier) e **GID** (Group Identifier).

CHOWN: MODIFICARE IL PROPRIETARIO DI UN FILE

chown: con questo comando è possibile modificare il proprietario di un file.

Il parametro **-R** consente di modificare ricorsivamente i permessi delle directory indicate e del loro contenuto.

$ chown nome1 file1	rende il file1 proprietario di nome1
$ chown nome1:gruppo1 file1	rende il file1 proprietario di nome1 e del gruppo1
$ chown -R nome1:gruppo1 dir1	rende la directory dir1 e i file contenuti proprietari di nome1 e del gruppo1
$ sudo chown -R root:rootdrupal	rende la directory drupal e i file contenuti proprietari di root e del root

Pipes – Cosa sono ed Esempi di utilizzo

I sistemi operativi basati su Unix come Linux offrono un approccio unico per unire due comandi sul terminale, con esso si può prendere l'output del primo comando e usarlo come input del secondo comando, questo è il concetto di **pipe** o **|**. Le Pipes permettono a due processi separati di comunicare tra loro anche se non sono stati creati per farlo, quindi questo apre una serie infinita di opportunità.

```
ls -l | grep rwxrwxrwx
```

CAT: VISUALIZZA IL CONTENUTO DI UNO O PIÙ FILE

cat con questo comando è possibile visualizzare il contenuto di uno o più files

$ cat file1	Visualizza il contenuto del file1
$ cat file1 file2 > file3	crea file3 con il contenuto di file1 e file2
$ cat file1 file2 >> file3	aggiunge il contenuto di file1 e file2 al file3

MORE: VISUALIZZA IL CONTENUTO DI UN FILE SU PIÙ PAGINE VIDEO

more:t con questo comando è possibile visualizzare il contenuto di un file in più pagine video. Il tasto di Invio fa avanzare la visualizzazione riga per riga mentre la barra spaziatrice fa avanzare di pagine video. Per interrompere utilizzare CTRL+Z

$ more file1	Visualizza il contenuto del file1 in pagine video
$ ls -l | more	visualizza il contenuto della directory corrente in formato lungo su più pagine video (vedi comando ls)

TOP: MOSTRA I PROCESSI ATTUALMENTE IN ESECUZIONE

top: questo comando visualizza i processi attualmente in esecuzione e le informazioni importanti relative a tali processi.
Si possono utilizzare i seguenti comandi

q - per uscire da top
u - ordina per utente
M - ordina per uso della memoria
P - ordina per l'uso del processore
h - visualizza la guida (help)
k - termina un processo
n - modifica il numero di processi visualizzati

DF: SPAZIO LIBERO SU DISCO

df: con questo comando è possibile visualizzare lo spazio libero su disco

$ df -h	Visualizza il contenuto dello spazio libero su disco utilizzando G,M,K byte

FREE: MOSTRA LO STATO DELLA MEMORIA

free: con questo comando è possibile visualizzare lo stato della memoria.

LSHW: LISTA HARDWARE DEL COMPUTER
lshw: con questo comando è possibile avere una lista di tutti i dispositivi che compongono l'hardware del computer

$ sudo lshw	lista completa di tutti i dispositivi hardware del computer
$ sudo lshw -short	lista abbreviata di tutti i dispositivi hardware del computer

UNAME: INFORMAZIONI SUL SISTEMA

uname: con questo comando è possibile sapere se il processore è a 32 o 64 bit

$ uname -a	informazioni complete
$ sudo lshw -short	lista abbreviata di tutti i dispositivi hardware del computer

touch

Il comando touch serve per aggiornare la data dell'ultimo accesso o quello dell'ultima modifica di un file. Se seguito da un nome di file non ancora presente, ne crea uno vuoto con l'estensione indicata.

La sintassi del comando è la seguente:

```
touch [opzioni] file
```

Alcune opzioni da utilizzare con il comando **touch**:

Opzione	Risultato
-a	cambia solo la data dell'ultimo accesso
-c	non creare il file
-m	cambia solo la data dell'ultima modifica
-t STAMP	specifica la data nel formato «[[CC]YY]MMDDhhmm[.ss]»

Alcuni esempi di uso del comando:
digitando il comando: touch miofile **si crea un file vuoto di nome miofile**

cut prende una fetta verticale di un file

stampa solo le colonne o i campi specificati. Le colonne sono selezionate utilizzando il IFS standard o un delimitatore di campi specificato.

Avendo il file test.txt che contiene

```
12345:Administration:james:smith
67891:Staff:stephan:york
18230:Staff:luke:cutwine
62913:Sales:red:blues
```

Stampare solo i caratteri da 1 a 5 di ogni linea:

cut -c1-5 test.txt

```
12345
67891
18230
62913
```

cut **-c1-5 test.txt**

```
12345
67891
18230
62913
```

Stampare solo la terza parola delimitata da :

cut -c1-5 test.txt

```
12345
67891
18230
62913
```

Stampare solo la terza parola delimitata da :

cut -d: -f3 test.txt

```
james
stephan
luke
red
```

Ottenere informazioni sul sistema

du

Il comando du visualizza lo spazio occupato sul disco da file o directory.
La sintassi è la seguente: du [opzioni] [file...]
Alcune opzioni da utilizzare con il comando du:

Opzione	Risultato
-a	visualizza le informazioni sia sui file che sulle directory
-s	visualizza la dimensione totale complessiva
-x	esclude le sottodirectory che siano parte di un altro filesystem

Alcuni esempi di uso del comando **du**:

- visualizzare la quantità di spazio occupata da miofile:

  du miofile

- Visualizza la quantità di spazio complessiva occupata dalla propria directory home:

  du -s ~

disk free, disco libero

df

Il comando **df** visualizza a schermo lo spazio rimasto sulle partizioni e sui dischi del proprio sistema.

La sintassi del comando è la seguente:

```
df [opzioni] [file...]
```

Alcune opzioni da utilizzare con il comando **df**:

Opzione	Risultato
-a	include nell'elenco anche i filesystem con una dimensione di 0 blocchi, che sono di natura omessi. Normalmente questi filesystem sono pseudo-filesystem con scopi particolari, come le voci per l' *automounter*. Filesystem di tipo «ignore» o «auto», supportati da alcuni sistemi operativi, sono inclusi solo se quest'opzione è specificata
-h	aggiunge a ciascuna dimensione un suffisso, come «M» per megabyte, «G» per gigabyte, ecc
-H	ha lo stesso effetto di **-h**, ma usa le unità ufficiali SI (con potenze di 1000 piuttosto che di 1024, per cui M sta per 1000000 invece di 1048576)
-t tipofs	limita l'elenco a filesystem del tipo specificato
-x tipofs	limita l'elenco a filesystem *non* del tipo specificato

Un esempio di uso del comando **df**:

- mostrare lo spazio occupato solo dai dischi con filesystem **ext3**, utilizzando il suffisso specifico per l'unità di misura:

```
df -Ht etx3
```

free

Il comando **free** mostra informazioni sulla memoria di sistema. Molto utile se si vuole rendersi conto della memoria disponibile sul sistema, della memoria attualmente in uso e di quella libera.

La sintassi del comando è la seguente:

```
free [opzioni]
```

Alcune opzioni da utilizzare con il comando **free**:

Opzione	Risultato
-b	mostra la quantità di memoria in byte
-k	mostra la quantità di memoria in KiB (impostato di default)
-t	mostra una riga contente i totali

top

Il comando **top** visualizza informazioni riguardanti il proprio sistema, processi in esecuzione e risorse di sistema, utilizzo di CPU, RAM e spazio swap utilizzato e il numero di task in esecuzione.

La sintassi del comando è la seguente:

```
top
```

Per uscire dal programma, premere il tasto «**q**».

Eseguire comandi con privilegi di amministrazione sudo

Per eseguire alcuni comandi come amministratori del sistema o per modificare file non all'interno della propria directory **Home**, è necessario anteporre al comando la parola **sudo**. Per maggiori informazioni sui privilegi di amministrazione consultare la relativa guida.

Ottenere maggiore aiuto

Per ottenere maggiore aiuto o informazioni riguardo un determinato comando, esiste il comando **man** che serve per visualizzare il manuale di un determinato comando.

La sintassi del comando **man** è la seguente:

```
man [comando]
```

Ad esempio, per visualizzare la pagina di manuale dello stesso comando **man** è sufficiente digitare il seguente comando:

```
man man
```

Quasi tutti i comandi accettano anche l'opzione **-h** (o **--help**) che fornisce una breve descrizione sull'utilizzo del comando e delle sue opzioni.

Il comando sudo

sudo (***s****uper****u****ser* ***do***) consente di eseguire un comando come se si fosse un altro utente. Effettua una specie di sostituzione, previa autorizzazione, tra l'utente attuale (colui che esegue il comando **sudo**) e l'utente **target** (colui che esegue l'effettivo comando). Mentre con il comando **su** si cambia utente fino al termine della sessione del terminale, **sudo** assegna i privilegi dell'utente **target** al solo processo che viene con esso avviato.

Per eseguire dei comandi con privilegi d'amministrazione è sufficiente digitare **sudo** e successivamente il comando che si desidera eseguire come utente **root**, come nel seguente esempio:

```
sudo nano /etc/modules
```

L'utente **target** non deve essere necessariamente l'amministratore, ma può essere un qualsiasi utente del sistema. Per scegliere l'utente **target**, usare l'opzione **-u**:

```
sudo -u target comando
```

Aggiornamento del sistema

Quello che segue è un breve elenco di comandi utili all'aggiornamento dei pacchetti installati nel sistema.

Comandi	
apt update	Aggiorna la lista dei pacchetti disponibili dai repository. Va lanciato dopo aver apportato delle modifiche a `/etc/apt/sources.list`, `/etc/sources.list` o `/etc/apt/preferences`. Può essere eseguito periodicamente per verificare che la propria lista di pacchetti sia aggiornata.
apt upgrade	Scarica e installa gli aggiornamenti per tutti i pacchetti installati. Non sarà rimosso alcun pacchetto installato: se un aggiornamento di un pacchetto richiede la rimozione di un altro pacchetto, l'aggiornamento non sarà effettuato.

Installazione e rimozione pacchetti

Per quanto riguarda l'installazione e la rimozione dei pacchetti dal sistema, di seguito vengono elencati i comandi più comuni:

Comandi	
apt install *packagename*	Installa un nuovo pacchetto.
apt remove *packagename*	Rimuove un pacchetto, senza rimuovere i file di configurazione.
apt purge *packagename*	Rimuove un pacchetto, compresi tutti i file di configurazione.
apt autoremove	Rimuove tutti i pacchetti che sono stati automaticamente installati per soddisfare le dipendenze di altri pacchetti, ma che ora non sono più richiesti.
apt -f install	Tenta di riparare i pacchetti con delle dipendenze non soddisfatte.
apt --reinstallinstall *packagename*	Reinstalla pacchetti che sono già installati e all'ultima versione.

Editor Nano

Nano è l'editor di testo da riga di comando predefinito e più usato con la distribuzione Linux **Ubuntu**. Evoluzione dell'obsoleto **Pico**, vecchio editor dei sistemi **Unix** che tra l'altro viene distribuito con una licenza che non gli consente di essere considerato un **Software Libero**, Nano si distingue dai più potenti e complessi editor del mondo Linux, come **Vi** e **Emacs**, per immediatezza e semplicità di utilizzo.

Iniziamo quindi immediatamente nel modo più semplice, creando ed editando da terminale un file nuovo chiamato, ad esempio, "documento" con Nano:

```
nano documento
```

Si avvierà immediatamente il programma aprendo il file vuoto appena creato, con un aspetto simile al questo:

Notiamo che la parte bassa dell'editor mostra la combinazione dei tasti per l'esecuzione dei comandi più comuni come salvare il contenuto, tagliare e incollare, editare la pagina della guida, uscire dal programma, ecc..
Ogni comando viene eseguito attraverso la pressione sulla tastiera del tasto **Ctrl**, indicato in Nano con il carattere dell'accento circonflesso "**^**", più la lettera associata.
Ad esempio, premendo **Ctrl+o** e poi il tasto **Invio** per confermare il nome del documento, viene salvato il contenuto del file fino a quel momento inserito; premendo **Ctrl+x** si esce dal programma; il controllo ortografico **Ctrl+t** di default non funziona, vedremo dopo perchè.

Altri comandi vengono eseguiti utilizzando il tasto **Esc**.
Ad esempio, per rimpiazzare un testo con un'altro usiamo la combinazione **Esc+r**; per marcare un testo ci posizioniamo con il cursore all'inizio del testo in questione e premiamo **Esc+a**; per giustificare il contenuto del file, **Esc+j**; ecc...
I comandi mostrati sono solo una piccola parte; per avere una panoramica generale lanciamo la pagina della guida con **Ctrl+g**:

Comandi per gestire una rete

NMAP (Network Mapper) è uno strumento open-source per la network exploration e l'auditing. È stato progettato per scansionare rapidamente reti di grandi dimensioni, ma è indicato anche per l'utilizzo verso singoli host

Scan multiple IP address or subnet (IPv4)

```
nmap 192.168.1.1 192.168.1.2 192.168.1.3
```

Scan range of IP address

```
nmap 192.168.1.1-20
```

Scan all subnet:

```
nmap 192.168.1.0/24
```

Determina O.S and version

```
nmap -v -A 192.168.1.1
```

Creare il file nmaptest.txt

```
localhost
server2.tecmint.com
192.168.0.101
```

Esegue la scan di tutti gli indirizzi IP/URL contenuti

```
nmap -iL nmaptest.txt
```

Eseguire una scansione veloce È possibile eseguire una scansione veloce con l'opzione "-F" per cercare le porte elencate nei file nmap-services e lasciare tutte le altre porte.

```
nmap -F 192.168.0.101
```

Find out if a host/network is protected by a firewall

```
nmap -sA 192.168.1.254
nmap -sP 192.168.1.0/24
```

Scan port 80

```
nmap -p 80 192.168.1.1
```

Scan TCP port 80

```
nmap -p T:80 192.168.1.1
```

Scan UDP port 53

```
nmap -p U:53 192.168.1.1
```

Scan two ports

```
nmap -p 80,443 192.168.1.1
```

 # Scan port ranges #

```
nmap -p 80-200 192.168.1.1
```

How do I detect remote operating system?

```
nmap -v -O 192.168.1.1
```

How do I detect remote services (server / daemon) version numbers?

```
nmap -sV 192.168.1.1
```

Find out the most commonly used TCP ports using TCP SYN Scan

```
# Stealthy scan #
nmap -sS 192.168.1.1
```

 # Find out the most commonly used TCP ports using TCP connect scan (warning: no stealth scan)

```
#  OS Fingerprinting #
nmap -sT 192.168.1.1
```

 # Find out the most commonly used TCP ports using TCP ACK scan

```
nmap -sA 192.168.1.1
```

NETSTAT : IL SUO SCOPO È QUELLO DI RESTITUIRCI INFORMAZIONI COMPLETE SULLE CONNESSIONI **DEL NOSTRO SISTEMA**

Netstat –a	lista di tutte le porte upd e tcp attive
Netstat –at	lista delle sole porte TCP
Netstat –au	solo le connessioni attive riguardanti i processi UDP
Netstat –an	tutte le connessioni ma formato hostname ma numerico.
Netstat -l	restituirà solo la lista delle connessioni in ascolto (listening)
Netstat -ap	mostrato anche il suo PID ed il nome del processo a cui ogni socket appartiene

Visualizzare informazioni su tutti gli indirizzi IP

Per visualizzare un elenco di tutte le interfacce di rete e l'indirizzo IP associato digitare il comando seguente:

```
ip addr show
```

Visualizzare informazioni su una singola interfaccia di rete

Per ottenere informazioni su un'interfaccia di rete specifica, utilizzare `ip addr show dev` seguito dal nome del dispositivo. Ad esempio, per eseguire una query `eth0`, digitare:

```
ip addr show dev eth0
```

Assegnare indirizzi IP a un'interfaccia

Per assegnare un indirizzo IP a un'interfaccia, utilizzare la sintassi seguente:

```
ip addr add ADDRESS dev IFNAME
```

Copy

Dove `IFNAME` è il nome dell'interfaccia e `ADDRESS` è l'indirizzo IP che si desidera assegnare all'interfaccia.

Per aggiungere l'indirizzo `192.168.111.111` con netmask 24 al dispositivo `eth0`, digitare:

```
sudo ip address add 192.168.111.111/24 dev eth0
```

Rimuovi / Elimina un indirizzo IP dall'interfaccia

La sintassi per rimuovere un indirizzo IP da un'interfaccia è la seguente:

```
ip addr dev ADDRESS dev IFNAME
```

Visualizzare informazioni sulle interfacce di rete

Per visualizzare un elenco di tutte le interfacce di rete, digitare il comando seguente:

```
ip link show
```

Copy

```
ip link show dev eth0
```

Copy

Visualizzare e modificare la tabella di routing

Per assegnare, rimuovere e visualizzare la tabella di routing del kernel utilizzare l'oggetto route. I comandi più comunemente utilizzati quando si lavora con gli oggetti itinerari sono: list, add e del.

Visualizzare la tabella di routing

Per ottenere un elenco delle voci della route del kernel, utilizzare uno dei seguenti comandi:

```
ip route
```

Copy

```
ip route list SELECTOR
```

Copy


```
ip route list SELECTOR
```

Copy

Se usato senza un comando SELECTOR, elencherà tutte le voci della route nel kernel:

```
ip route list
```

Per visualizzare solo il routing per una rete specifica, ad esempio, 172.17.0.0/16 digitare:

```
ip r list 192.168.111.0/24
```

Copy

Aggiungere un nuovo route

Per aggiungere una nuova voce alla tabella di routing, utilizzare il comando route add seguito dalla rete o dal nome del dispositivo.

Aggiungi un percorso a 192.168.111.0/24 tramite il gateway a 192.168.111.1

```
ip route add 192.168.111.0/24 via 192.168.111.1
```

Copy

Aggiungi un percorso a 192.168.111.0/24 che può essere raggiunto sul dispositivo eth0.

```
ip route add 192.168.111.0/24 dev eth0
```

Copy

Per aggiungere un percorso predefinito, utilizzare la parola chiave default. Il comando seguente aggiungerà una route predefinita tramite il gateway locale 192.168.111.1 che può essere raggiunta sul dispositivo eth0.

Eliminare un percorso

Per eliminare una voce dalla tabella di routing, utilizzare il comando route del, La sintassi per eliminare una route è la stessa di quando si aggiunge.

Il seguente comando eliminerà la route predefinita:

```
ip route del default
```

Copy

Elimina un percorso per 192.168.111.0/24 tramite il gateway in 192.168.111.1

```
ip route del 192.168.111.0/24 via 192.168.111.1
```

SYSTEMD COME CONTROLLARE CHE UN SERVIZIO SIA IN ESECUZIONE IN LINUX

Oramai praticamente tutte le nuove release delle distribuzioni Linux hanno abbandonando il denome **init** (SystemV init o SysV init) per passare al demone **systemd** come controllore dei servizi e del sistema.

Uno dei principali vantaggi di systemd è la capacità di avviare i processi in parallelo (*aggressive parallelization capabilities*). Con **init** infatti i processi vengono avviati in modo seriale e un task può essere avviato solo quando il precedente si è concluso correttamente. Come per avviene per init systemd è il primo processo che sia avvia ed è il padre di tutti i processi nati dopo successivamente, tipicamente ha assegnato pid=1.

A differenza di SysV init, **systemd** init è responsabile del mount point, della crittografia, di syslog, etc superando così le funzionalità di un init system di base. La gran parte delle azioni di systemd si svolge agendo su risorse denominate "units". Le units sono definite da files noti come unit files. Il tipo di unit files è in genere riconoscibile a partire dall'estensione del file. Conseguentemente lo unit file di un servizio avrà l'estensione **.service**: il servizio di dbase postgresql avrà come target uno unit files del tipo postgresql.service.

Il comando utilizzato per controllare systemd è **systemctl**.

Conseguentemente la situazione abbastanza frequente di verificare che un servizio o demone (service o daemon) sia in esecuzione ad esempio: apache ,mysqld, postrgresql, samba, ssh, nfs, rsync, cron, syslogd, etc. in **un sistema Linux con systemd init sarà diversa da quella con SysV init**.

Controllare che un servizio sia attivo: comparazione tra SysV init e systemd init

SysV init

per controllare che il servizio postgresql sia in esecuzione con SysV init utilizziamo il comando

```
service nome-del-servizio status

per postgresql

service postgresql status
```

ubuntu 12

```
$ service postgresql status

9.1/main (port 5432): online
```

per dettagli sul controllo dello stato di un servizio in SysV init fare riferimento al post:

Come controllare che un servizio (service) sia in esecuzione in linux

Systemd init

per controllare che un servizio con systemd sia in esecuzione è necessario conoscere lo **unit file di riferimento** e

utilizzare il comando **systemctl**

```
systemctl status nome-unit-file.estensione

per postgresql
systemctl status postgresql@9.6-main.service
```

debian 9

```
# systemctl status postgresql@9.6-main.service

● postgresql@9.6-main.service - PostgreSQL Cluster 9.6-main
   Loaded: loaded (/lib/systemd/system/postgresql@.service; disabled; vendor
preset: enabled)
   Active: active (running) since Sat 2018-04-28 16:23:29 CEST; 4min 36s ago
  Process: 19786 ExecStop=/usr/bin/pg_ctlcluster --skip-systemctl-redirect -m fast
9.6-main stop (code=exited, status=0/SUCCESS)
  Process: 19792 ExecStart=postgresql@9.6-main --skip-systemctl-redirect 9.6-main
start (code=exited, status=0/SUCCESS)
 Main PID: 19799 (postgres)
    Tasks: 6 (limit: 4915)
   CGroup: /system.slice/system-postgresql.slice/postgresql@9.6-main.service
           ├─19799 /usr/lib/postgresql/9.6/bin/postgres -D
/var/lib/postgresql/9.6/main -c
config_file=/etc/postgresql/9.6/main/postgresql.conf
           ├─19801 postgres: 9.6/main: checkpointer process
           ├─19802 postgres: 9.6/main: writer process
           ├─19803 postgres: 9.6/main: wal writer process
           ├─19804 postgres: 9.6/main: autovacuum launcher process
           └─19805 postgres: 9.6/main: stats collector process
```

```
Apr 28 16:23:27 deb9 systemd[1]: Starting PostgreSQL Cluster 9.6-main...
Apr 28 16:23:29 deb9 systemd[1]: Started PostgreSQL Cluster 9.6-main.
```

come si può osservare l'outup è decisamente più articolato di sysV.

Unit file

Per visualizzare il contenuto dello unit file responsabile dell'avvio del servizio postgresql si può utilizzare il

comando: **systemctl cat nome-unit**

```
systemctl cat postgresql@9.6-main.service

# /lib/systemd/system/postgresql@.service
# systemd service template for PostgreSQL clusters. The actual instances will
# be called "postgresql@version-cluster", e.g. "postgresql@9.3-main". The
# variable %i expands to "version-cluster", %I expands to "version/cluster".
# (%I breaks for cluster names containing dashes.)

[Unit]
Description=PostgreSQL Cluster %i
ConditionPathExists=/etc/postgresql/%I/postgresql.conf
PartOf=postgresql.service
ReloadPropagatedFrom=postgresql.service
Before=postgresql.service

[Service]
Type=forking
# @: use "postgresql@%i" as process name
ExecStart=@/usr/bin/pg_ctlcluster postgresql@%i --skip-systemctl-redirect %i start
ExecStop=/usr/bin/pg_ctlcluster --skip-systemctl-redirect -m fast %i stop
ExecReload=/usr/bin/pg_ctlcluster --skip-systemctl-redirect %i reload
PIDFile=/var/run/postgresql/%i.pid
SyslogIdentifier=postgresql@%i
# prevent OOM killer from choosing the postmaster (individual backends will
# reset the score to 0)
OOMScoreAdjust=-900
# restarting automatically will prevent "pg_ctlcluster ... stop" from working,
# so we disable it here. Also, the postmaster will restart by itself on most
# problems anyway, so it is questionable if one wants to enable external
# automatic restarts.
#Restart=on-failure
# (This should make pg_ctlcluster stop work, but doesn't:)
#RestartPreventExitStatus=SIGINT SIGTERM

[Install]
WantedBy=multi-user.target
```

Visualizzare le dipendenze di una unit

In systemd init è possibile visualizzare anche tutte le dipendenze di un servizio quale postgresql con il comando systemctl

list-dependecies postgresql@96-main.service.

```
systemctl cat systemctl list-dependencies  postgresql@9.6-main.service
postgresql@9.6-main.service
● ├─system-postgresql.slice
● └─sysinit.target
●   ├─apparmor.service
●   ├─dev-hugepages.mount
●   ├─dev-mqueue.mount
●   ├─keyboard-setup.service
●   ├─kmod-static-nodes.service
●   ├─lvm2-lvmetad.socket
●   ├─lvm2-lvmpolld.socket
●   ├─lvm2-monitor.service
●   ├─proc-sys-fs-binfmt_misc.automount
●   ├─sys-fs-fuse-connections.mount
●   ├─sys-kernel-config.mount
●   ├─sys-kernel-debug.mount
●   ├─systemd-ask-password-console.path
●   ├─systemd-binfmt.service
●   ├─systemd-hwdb-update.service
●   ├─systemd-journal-flush.service
●   ├─systemd-journald.service
●   ├─systemd-machine-id-commit.service
●   ├─systemd-modules-load.service
●   ├─systemd-random-seed.service
●   ├─systemd-sysctl.service
●   ├─systemd-timesyncd.service
```

```
● ├─systemd-tmpfiles-setup-dev.service
● ├─systemd-tmpfiles-setup.service
● ├─systemd-udev-trigger.service
● ├─systemd-udevd.service
● ├─systemd-update-utmp.service
● ├─cryptsetup.target
● ├─local-fs.target
● │ ├─-.mount
● │ ├─media-ArchivioR.mount
● │ ├─media-Home_Dati.mount
● │ ├─systemd-fsck-root.service
● │ └─systemd-remount-fs.service
● └─swap.target
●   └─dev-sda15.swap
```

Verificare lo stato di tutti i servizi in systemd init – linux

SysV init

Per verificare lo stato di tutti i servizi presenti un sistema sysV init basta usare l'opzione –status-all

```
$ service --status-all
```

In pratica –status-all consente di vedere i servizi installati.

systemd

Come già affermato systemd gestisce un numero più elevato di funzionalità rispetto a sysV quindi la visualizzazione di tutte le unit diventa una schermata piuttosto estesa.

In questo caso conviene filtrare le units chiedendo di visualizzare sono le unit di tipo servizio. Per visualizzare tutte le units che systemd può gestire si usa il comando

```
# systemctl list-units
```

per restringere il campo a quelle di tipo service aggiungere l'opzione **–type=service**

 # systemctl list-units --type=service

```
systemctl list-units --type=service
  UNIT                                                                    LOAD   ACTIVE SUB
DESCRIPTION
  alsa-restore.service                                                    loaded active exited
Save/Restore Sound Card State
  avahi-daemon.service                                                    loaded active running
Avahi mDNS/DNS-SD Stack
  binfmt-support.service                                                  loaded active exited
Enable support for additional executable binary formats
  colord.service                                                          loaded active running
Manage, Install and Generate Color Profiles
  console-setup.service                                                   loaded active exited
Set console font and keymap
  cron.service                                                            loaded active running
Regular background program processing daemon
  cups-browsed.service                                                    loaded active running
Make remote CUPS printers available locally
  cups.service                                                            loaded active running
CUPS Scheduler
  dbus.service                                                            loaded active running
D-Bus System Message Bus
  ebtables.service                                                        loaded
...                                                 loaded active exited  Set the console keyboard
layout
  kmod-static-nodes.service                                               loaded active exited
Create list of required static device nodes for the current kernel
  libvirt-guests.service                                                  loaded active exited
Suspend/Resume Running libvirt Guests
  libvirtd.service                                                        loaded active running
Virtualization daemon
  lightdm.service                                                         loaded active running
Light Display Manager
```

```
...                                                                    loaded active exited  Raise
● nginx.service                                                                    loaded failed failed
A high performance web server and a reverse proxy server
  polkit.service                                                                   loaded active running
Authorization Manager
  postgresql.service                                                               loaded active exited
PostgreSQL RDBMS
  postgresql@9.6-main.service                                                      loaded active running
PostgreSQL Cluster 9.6-main
  rpcbind.service
.....
LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.
```

runlevel e System state

SysV init runlevel

In SysV init il modo più semplice per visualizzare i servizi attivi in un dato runlevel è digitare un comando del tipo:

```
sudo ls -l /etc/rc(num-run-level).d/
```

systemd init System state (runlevel)

In systemd il concetto di runlevel è sostituito dal concetto di System State o punto di sincronizzazione. Per definire un determinato System State vengono utilizzati degli unit file particolari con estensione .target. Uno unit.target garantirà di avere disponibile tutti gli units che definiscono uno specifico System State senza doversi preoccupare dei singoli elementi. Se si utilizza lo unit sound.target questo starà ad indicare che servizi per l'audio sono pronti all'audio.

In systemd i System State sono più numerosi dei runlevel quelli comparabili con i runlevel possono essere elencati con il comando:

```
systemctl list-units --type=target
```

```
systemctl list-units --type=target
UNIT                   LOAD   ACTIVE SUB    DESCRIPTION
basic.target           loaded active active Basic System
cryptsetup.target      loaded active active Encrypted Volumes
getty.target           loaded active active Login Prompts
graphical.target       loaded active active Graphical Interface
local-fs-pre.target    loaded active active Local File Systems (Pre)
local-fs.target        loaded active active Local File Systems
multi-user.target      loaded active active Multi-User System
network-online.target  loaded active active Network is Online
network-pre.target     loaded active active Network (Pre)
network.target         loaded active active Network
paths.target           loaded active active Paths
remote-fs-pre.target   loaded active active Remote File Systems (Pre)
remote-fs.target       loaded active active Remote File Systems
rpcbind.target         loaded active active RPC Port Mapper
slices.target          loaded active active Slices
sockets.target         loaded active active Sockets
sound.target           loaded active active Sound Card
swap.target            loaded active active Swap
sysinit.target         loaded active active System Initialization
time-sync.target       loaded active active System Time Synchronized
timers.target          loaded active active Timers
virt-guest-shutdown.target loaded active active Libvirt guests shutdown

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.
```

```
22 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
```

Benchè il concetto di System State conseguito con gli unit.target sia decisamente più flessibile del concetto di runlevel è possibile evidenziare alcuni target che sono in relazione proprio i principali runlevel di SysV. A questo proposito si può precisare sono diponibili anche unit dal nome evocativo:

```
systemctl list-units-files --type=target
UNIT FILE              STATE
...
runlevel0.target       disabled
runlevel1.target       disabled
runlevel2.target       static
runlevel3.target       static
runlevel4.target       static
runlevel5.target       static
runlevel6.target       disabled
...
59 unit files listed.
```

Elencare i servizi abilitati in Systemctl

Per elencare tutti i servizi abilitati, **enabled,** si può far ricorso al solito greo

```
systemctl list-unit-files | grep enabled
```

es:

```
# systemctl list-unit-files | grep enabled
acpid.path                                 enabled
cups.path                                  enabled
apache2.service                            enabled
atd.service                                enabled
autovt@.service                            enabled
avahi-daemon.service                       enabled
clamav-freshclam.service                   enabled
console-setup.service                      enabled
cron.service                               enabled
...
...
```

per elencare i servizi in esecuzioni utilizzare il filtro **running**

```
systemctl  | grep running
```

es:

```
# systemctl | grep running
  proc-sys-fs-binfmt_misc.automount             loaded active running   Arbitrary Executable File
Formats File System Automount Point
  acpid.path                                    loaded active running   ACPI Events Check
  cups.path                                     loaded active running   CUPS
Scheduler
  init.scope                                    loaded active running   System and Service Manager
  session-877.scope                             loaded active running   Session 877 of user maurizio
...
...
```

Comparazione tra runlevel e target (System state) :

- Run level 3 è emulato dal target **multi-user.target;**

- Run level 5 è emulato da target **graphycal.target;**
- **runlevel3.target** è un link simbolico a **multi-user.target;**
- runlevel5.target è un link simbolico a graphycal.target.

Comparazione tra alcuni comandi SysV init (service) e systemd init (systemctl)

systemctl start sshd.service	service sshd start	start
systemctl status sshd.service	service sshd status	status
systemctl stop sshd.service	service sshd stop	stop
systemctl reload sshd.service	service sshd reload	reload
ls /lib/systemd/system/*.service /etc/systemd/system/*.service	ls /etc/rc.d/init.d/	lista tutti i servizi avviabili
systemctl enabled sshd.service	service sshd enabled	attivare un servizio al prossimo boot

---

Avviare e abilitare un servizio al prossimo boot

Per avviare un servizio come il webserver apache2 e come abilitarlo al riavvio al successvio boot

```
systemctl start apache2



systemctl enable apache2
```

Inauguriamo con **systemctl** un ciclo di articoli tecnici su **SystemD**.

systemctl é il tool con cui gestiamo gran parte degli aspetti di SystemD e va a sostituire gran parte delle funzioni dei comandi *service* e *chkconfig*. Vedremo qui come analizzare lo stato del sistema, gestire lo stato della macchina e – *aspetto forse piú importante* – gestire le units.

TIP aggiungendo -H user@host possiamo eseguire il comando su una macchina remota via ssh.

```
NAME
       systemctl - Control the systemd system and service manager

SYNOPSIS
       systemctl [OPTIONS...] COMMAND [NAME...]

DESCRIPTION
       systemctl may be used to introspect and control the state of the systemd(1) system and

       For Unit Commands the NAME represents full name of unit.

           systemctl start foo.service

       For Unit File Commands the NAME represents full name of the unit file, or absolute path

           systemctl start /path/to/foo.service

       While working with services/service files, systemctl is able to append .service suffix

           systemctl start foo
```

Analizzare lo stato del sistema

Se volevamo vedere lo stato dei servizi attivi sulla macchina con avremmo usato il comando:

```
service --status-all
```

Con systemctl invece possiamo dare il comando senza alcun parametro, oppure per visualizzare solo i servizi che hanno dato errore diamo il comando con il parametro –*failed*.

```
systemctl
# gdm.service
loaded active running   GNOME Display Manager
# gpm.service
loaded active running   Console Mouse manager
# irqbalance.service
loaded active running   irqbalance daemon

systemctl --failed
# UNIT           LOAD   ACTIVE SUB    DESCRIPTION
# mcelog.service loaded failed failed Machine Check Exception Logging Daemon
# rngd.service   loaded failed failed Hardware RNG Entropy Gatherer Daemon
#
# LOAD   = Reflects whether the unit definition was properly loaded.
# ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
# SUB    = The low-level unit activation state, values depend on unit type.
#
# 2 loaded units listed. Pass --all to see loaded but inactive units, too.
# To show all installed unit files use 'systemctl list-unit-files'.
```

Dalle estensioni del nome, vedrete che ci saranno diversi tipi di units: .automount, .mount, .device, .service etc etc. Vedremo piú avanti di cosa esattamente si tratta, vi basti sapere che i daemon sono quelli con estensione **.service**.

Gestione dei servizi

Vediamo ora come sostituire i vecchi comandi

```
service myservice [start|stop|restart|status]
chkconfig myservice [on|off]
```

Abbiamo detto che i servizi hanno estensione *.service*, quindi myservice verrá ora chiamato *myservice.service*.

Iniziamo con l'abilitare il servizio:

```
systemctl enable httpd.service
# ln -s '/usr/lib/systemd/system/httpd.service' '/etc/systemd/system/multi-
user.target.wants/httpd.service'
```

Adesso il servizio Apache verrá avviato all'avvio. Allo stesso modo usiamo il comando disable per disabilitarlo.

Ora vedremo come studiarne lo stato, avviare e fermare il servizio.

```
systemctl status httpd.service
# httpd.service - The Apache HTTP Server
#    Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled)
#    Active: inactive (dead)
systemctl start httpd.service
systemctl status httpd.service
# httpd.service - The Apache HTTP Server
#    Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled)
#    Active: active (running) since Tue 2014-02-25 08:39:56 CET; 16s ago
#  Main PID: 18285 (httpd)
#    Status: "Total requests: 0; Current requests/sec: 0; Current # traffic:
0 B/sec"
#    CGroup: /system.slice/httpd.service
#            ├─18285 /usr/sbin/httpd -DFOREGROUND
#            ├─18286 /usr/sbin/httpd -DFOREGROUND
#            ├─18287 /usr/sbin/httpd -DFOREGROUND
#            ├─18288 /usr/sbin/httpd -DFOREGROUND
#            ├─18289 /usr/sbin/httpd -DFOREGROUND
#            └─18290 /usr/sbin/httpd -DFOREGROUND
#
# Feb 25 08:39:56 alorenzi-vaio.local systemd[1]: Started The Apache HTTP
Server.
systemctl stop httpd.service
```

Notate quanti dettagli restituisce *systemctl service* quando il servizio é attivato: non solo mostra se il servizio é acceso o spento, ma anche il PID, il cgroup, e anche le ultime righe di log del servizio.

Gestione energetica

Possiamo anche utilizzare systemd per riavviare, spegnere, sospendere e ibernare la macchina, rispettivamente:

```
systemctl reboot
systemctl poweroff
systemctl suspend
systemctl hibernate
```

Se utilizziamo il tool *polkit* anche gli utenti saranno in grado di utilizzare il power management di systemctl senza dover inserire alcuna password e senza configurare sudo. Su Fedora gli utenti amministratori sono giá configurati per poter utilizzare queste funzioni