



netstudent

# Corso GNU/Linux

17 maggio 2011



<puria@netstudent.polito.it>

<http://netstudent.polito.it>



netstudent

# Cercare e archiviare

Demoni e runlevel

Pacchetti e sorgenti





## Il comando locate

- Il comando locate di fatto non effettua una ricerca effettiva sul disco, ma all'interno di un suo elenco di file presenti sul filesystem, pertanto risulta molto veloce, ma ha il difetto di dover essere aggiornato.
- Il suo aggiornamento di norma avviene in modo automatico all'atto di creazione di nuovi file ed in maniera periodica nel caso non vi siano modifiche, ma è sempre possibile forzare l'aggiornamento con l'opzione **-u**.
- E' possibile poi effettuare ricerche *case insensitive* in quanto di default le ricerche con locate sono case sensitive, tramite l'opzione **-i**.
- La sua sintassi di base è: **locate** <nome\_file>, ed il suo database è situato in **slocate.db** in /var/lib/slocate con una dimensione di circa 4 MB, variabile con il numero di file, che viene aggiornato in automatico alle 4.20 am ogni giorno.



## Il comando find

- Sintassi: **find** <directory> **-name** <nome\_file> **-print**
- Con la sintassi sopra descritta il comando **find** permette di visualizzare a video tutte le occorrenze del <nome\_file> all'interno della <directory>.
- Il comando **find** è molto più efficiente di **locate**, ma paga questa potenza con una complessità di utilizzo maggiore: le ricerche possono essere sul nome, ma anche sui permessi, sui timestamp e può essere usato anche per eseguire alcune operazioni sui file trovati.
  - **-amin** determina data ed ora dell'ultimo accesso: -60 modificati entro 60 minuti, +60 modificato da almeno 60 minuti.
  - **-atime**
  - **-type** riduce il campo di ricerca del comando (d directory ed f file).
  - **-or (-and)** operatori logici OR ed AND per combinare più criteri di ricerca



## Ricerche all'interno dei file di testo

- I comandi **find** e **locate** agiscono sul filesystem, ma non possono eseguire ricerche all'interno dei singoli file di testo, per questo esiste un altro comando: **grep**.

**grep** <stringa> <nome\_file1> <nome\_file2> <...>

- Il risultato riporta il file in cui è stata trovata la stringa più la riga in cui è stata trovata. Con l'opzione **-i** è possibile effettuare ricerche case insensitive.
- E' possibile poi combinare le ricerche di file con le ricerche al loro interno, sia con la redirectione che con la pipe, ma soprattutto con il comando **xargs** (che crea dinamicamente gli argomenti per un altro comando ricevendoli dallo standard input).
- E poi possibile utilizzare altri comandi per effettuare ricerche ancora più evolute, quali **awk**, **sed** la cui trattazione però richiederebbe troppo tempo per lo scopo di questo corso



## Il comando tar

- Il comando **Tar** è l'utility standard per l'archiviazione dei file sotto Unix, un po' come Winzip lo è per la compressione sotto Windows. Tar (**Tape Archive**) è stato progettato inizialmente per gestire gli **archivi su drive a nastro** e da questa linea progettuale deriva ancora oggi molte delle sue caratteristiche peculiari. **GNU/Linux** ovviamente utilizza il comando tar del progetto GNU.
- E' possibile poi **combinare Tar con gzip**, per creare archivi di file compressi, in questo caso le estensioni tipiche (ricordiamo che GNU/Linux, come tutti gli Unix non ha necessità di utilizzare estensioni ai nomi di file per svolgere il suo compito) **.tgz** oppure **.tar.gz**.
- Anche il programma di compressione **bzip2** è utilizzabile con Tar, creando file **.tar.bz2**.
- Non esploreremo in dettaglio la sintassi di tar, ma solo le sue opzioni principali, essendo tale comando molto ricco, e rimandando allo studio approfondito della pagina man di tale comando.



## Il comando tar

- E' buona norma analizzare il contenuto di un file compresso prima di espanderlo, sia per evitare di disperdere file potenzialmente pericolosi sul sistema, sia per verificare la disponibilità di spazio sul sistema:
  - **tar tvfz archivio.tgz** (se compresso con gzip)
  - **tar tvfl archivio.tar.bz2** (se compresso con bzip2)
  - **tar tvf archivio.tar** (per archivi non compressi)
- Verificata l'assenza di problemi o pericoli, è possibile procedere con l'estrazione/decompressione vera e propria:
  - **tar xvfz archivio.tgz** (se compresso con gzip)
  - **tar xvfl archivio.tar.bz2** (se compresso con bzip2)
  - **tar xvf archivio.tar** (per archivi non compressi)
- Per creare un archivio, infine, il comando è
  - **tar cvfz archivio.tgz directory\_bersaglio/**





# Cercare e archiviare

## I comandi gzip – gunzip – zcat

- **gzip** è un comando creato per comprimere e decomprimere i file utilizzando la codifica di **Lempel-Ziv** (LZ77). Il file originario viene sostituito da un file con estensione **.gz**, che mantiene inalterati i timestamp ed i permessi.
- Nel caso in cui i nomi dei file compressi non siano utilizzabili sul sistema una volta decompressi, sarà lo stesso **gzip** a modificarli in modo da renderli compatibili con il filesystem bersaglio.
- **gunzip** riconosce anche le estensioni speciali **.tgz** e **.taz** come abbreviazioni per **.tar.gz** e **.tar.Z**.
- **zcat** ha un comportamento simile a **gunzip** con l'opzione **-c**
- Opzioni principali del comando **gzip**:

**gzip/gunzip/zcat [opzioni] <nome\_file>**





## Opzioni comando gzip

- **-d** - Decomprime.
- **-f** - Forza la compressione o la decompressione anche se il file esiste, altrimenti gzip richiederà la conferma per ogni file che dovrà sovrascrivere.
- **-l** - elenca alcuni campi per ogni file compresso: dimensione file compresso, dimensione file non compresso, rapporto di compressione, nome del file.
- **-n** – In compressione non salva il nome ed timestamp originali. In decompressione non ripristina il nome del file originale se presente (rimuove solo il suffisso *gzip*). Questa opzione è standard per la decompressione.
- **-N** – Si comporta esattamente all'opposto dell'opzione precedente.
- **-q** - Sopprime tutti gli avvertimenti ("warning").



## Opzioni comando gzip

- **-r – Comportamento ricorsivo.**
- **-t** – Permette di verificare l'integrità del file compresso.
- **-v** – Comportamento verboso: in pratica mostra stato di avanzamento e rapporto di compressione per ogni file
- **-# --fast --best** – Regola la velocità di compressione e la sua accuratezza, con una cifra variabile da 1 (fast) a 9 (best). Il livello predefinito è 6, quindi con una predilezione per la compressione a scapito della velocità.

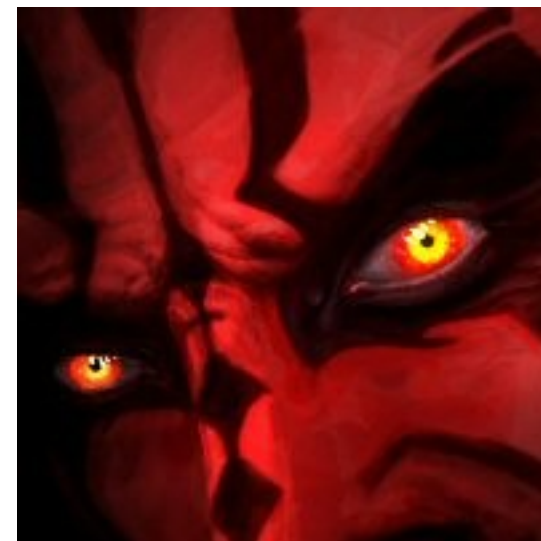


netstudent

Cercare e archiviare

# **Demoni e runlevel**

Pacchetti e sorgenti





# Demoni e runlevel

## Schedulazione

- La schedulazione di particolari comandi è un tipico compito dell'amministratore di sistema, che può in questo modo far eseguire particolari compiti al sistema in momenti non critici.
- Esempio tipico di attività pianificata sono il backup, di solito eseguito in orario notturno o comunque di basso carico del sistema, e la periodica pulizia delle directory temporanee del sistema per recuperare spazio ed eliminare file ormai obsoleti.
- In GNU/Linux è possibile schedulare i comandi utilizzando due comandi: **cron** ed **at**: il primo pianifica l'esecuzione di comandi secondo una cadenza fissa, mentre il secondo programma l'esecuzione ritardata del comando.
- Per eseguire un comando utilizzando **cron**, è necessario editare il file **crontab**, inserendo una nuova linea con il comando da attivare. Per eseguire tale operazione è utilizzato di norma il comando **crontab**.



# Demoni e runlevel

## I comandi cron ed at

- Il file crontab prevede cinque campi che indicano l'ora di pianificazione secondo lo schema (minuto-ora-data-mese-giorno) dove giorno indica il giorno della settimana. E' possibile inserire più valori per ogni campo, separandolo con una virgola, mentre il metacarattere \* indica ALL.

**10 10 \* \* \* /bin/comando\_qualsiasi**

- Il comando at, invece, prevede una singola esecuzione del comando all'orario prefissato. Per immettere un comando in tal modo si deve inserire l'orario con **at hh:mm (gg.mm:aa) <invio>** seguito dal comando, o comandi da eseguire, uscendo poi con CTRL+D. Con **atq** si verifica l'esecuzione di un comando e con **atrm** si rimuovono eventuali comandi che non devono essere più eseguiti.



# Demoni e runlevel

## Init

- **Init** potrebbe essere definito come il “padre di tutti di processi”, in quanto è a lui che il **kernel** cede il controllo dopo l’avvio, ed è lui il solo responsabile del lancio di tutti i programmi, tramite il suo file di configurazione **/etc/inittab**.
- In questo file sono definite le directory in cui sono contenuti gli script di avvio per tutti i runlevel, il runlevel predefinito per l’avvio ed altri comandi e script che devono essere avviati al boot.
- Il primo script ad essere avviato è `/etc/rc.d/rc.sysinit` che esegue tutta una serie di operazioni per avviare il sistema tra cui l’avvio della memoria di swap, il montaggio del filesystem di root, check di tutti i filesystem previsti, mount di tutti i filesystem presenti come auto in `etc/fstab`, gestione dell’orologio dei sistema e caricamento dei moduli del kernel.



# Demoni e runlevel

## Init e runlevel

- Tutte queste procedure, tranne il check dei filesystem, avvengono automaticamente, ma se qualche filesystem non viene ripristinato in automatico, init chiede la password di root, ed avvia una shell per permettere all'amministratore di gestire la situazione.
- I runleveles previsti sono:
  0. - halt
  1. - single user
  2. - multi-user privo dei servizi di rete
  3. - multi-user
  4. - non usato
  5. - multi-user con interfaccia grafica (server X) attivata
  6. - reboot





# Demoni e runlevel

## Runlevel

- Nel mondo Unix ci sono due sistemi di gestione del **processo di boot e startup** del sistema, quello di **System V**, molto complesso e flessibile, utilizzato anche da **GNU/Linux** e quello di **BSD**, meno efficace ma molto meno complesso.
- Il sistema usato è quindi quello dei **runlevels**, gestiti tramite gli script contenuti nelle directory **/etc/rc.d/rc#.d** in cui # è il runlevel prescelto: al loro interno sono presenti dei link simbolici che puntano ad altri script.
- Gli script presentano dei numeri e delle lettere, che assumono un significato ben preciso: la lettera **K** sta per **Kill** e fa stoppare un determinato servizio se attivo, la lettera **S** sta per **Start** ed avvia il servizio, mentre il numero indica la priorità di quel servizio nella procedura di avvio.
- In realtà, quindi, gli script risiedono in **/etc/rc.d/init.d/**.



# Demoni e runlevel

## Runlevel

- Quindi modificando ad hoc le lettere degli script è possibile attivare o stoppare particolari servizi in un determinato **runlevel**.
- In realtà, esistono sistemi più user-friendly, come **chkconfig**, **ntsys** su shell e **serviceconf** su interfaccia grafica.
- Per ragioni di sicurezza e praticità, **se un servizio non è utilizzato dovrebbe essere perlomeno stoppato**, se non addirittura rimosso dal sistema, per eliminare potenziali punti critici agli attacchi esterni.
- Tramite **inittab** è possibile determinare il **runlevel** predefinito di avvio del sistema: la parola chiave **initdefault** indica in quale **runlevel** sarà avviato il sistema. Attenzione a non immettere mai i valori 0 e 6.
- E' possibile, se installato, utilizzare anche il comando **service** per gestire i servizi, anche se non è nulla di più di uno **shell script** per evitare di scrivere **/etc/rc.d/init.d/<nome\_servizio> <azione>**.



## Runlevel corrente e cambio di runlevel

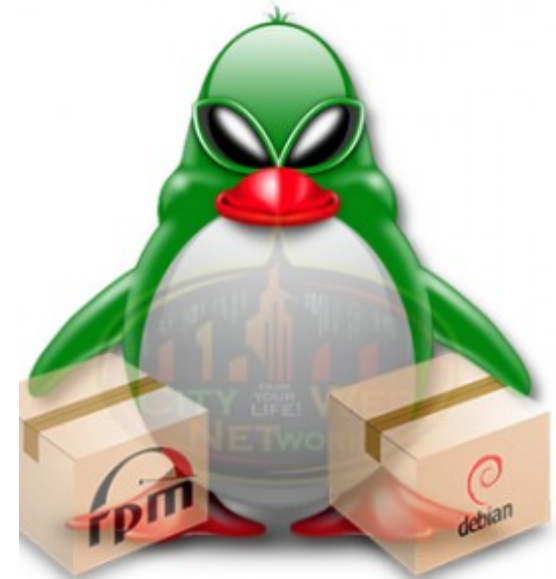
- Per conoscere il runlevel attuale di una macchina, ed anche quello utilizzato in precedenza nella stessa sessione, esiste il comando **runlevel**. Un output del tipo **N 3** significa che il sistema è attualmente in **runlevel 3** ed è stato avviato in questa modalità. Il comando legge il contenuto di **/var/run/umtp**.
- Per cambiare il runlevel è necessario utilizzare il comando **init** seguito dal numero di runlevel desiderato o dal termine **s** per indicare il runlevel single user.
- **/sbin/telinit** è un altro comando per variare il runlevel con opzioni a carattere singolo
  - **0 ,1 ,2 ,3 ,4 ,5, 6** indicano il runlevel desiderato.
  - **a ,b ,c** per processare solo le voci a,b,c del file **/etc/inittab**.
  - **q** riesamina il file **/etc/inittab**.
  - **s** passa in single user mode.
  - **-t** indica ad **init** quanto tempo, in secondi, far passare tra i comandi **TERM** e **KILL**.



Cercare e archiviare

Demoni e runlevel

## Pacchetti e sorgenti





## Pacchettizzazione

- I sistemi GNU/Linux si basano, oltre che sul kernel e la shell, su tutta una serie di programmi ed applicazioni che possono entrare a far parte del sistema sia in fase di installazione che su di un sistema già funzionante: in questo caso si ricorre tipicamente alla compilazione del programma direttamente dal codice sorgente, tramite la nota sequenza di comandi (**./configure** – che controllerà i requisiti del sistema e creerà il file `makefile`, che permette la compilazione del programma tramite i comandi seguenti (eseguiti da root) **make &&** **make install** – `make` compila il file con le indicazioni e `make install` lo installa nelle directory prestabilite (`/usr/local/bin`) oppure, molto più semplicemente, ricorrendo ad un sistema di pacchettizzazione.



# Pacchetti e sorgenti

## RPM

- Questi sistemi in pratica sostituiscono la compilazione con degli script che si occupano di installare sul sistema versioni precompilate (binari) dei programmi in questione.
- I vantaggi della pacchettizzazione sono quelli della praticità, semplicità e velocità dell'installazione, pagati però con una ridotta configurabilità ed un controllo ridotto.
- La tecnologia RPM (Red hat Package Manager) deriva dai primi sistemi di pacchettizzazione per GNU/Linux ed è stata rilasciata già nel 1995, riscritta poi nel 1996, per arrivare fino ai giorni nostri: si tratta quindi di una tecnologia molto consolidata.
- Di fatto RPM, pur essendo stato sviluppato da Red Hat è uno standard a tutti gli effetti.



# Pacchetti e sorgenti

## RPM

- Inoltre RPM ha al suo interno un database di tutti i pacchetti installati, mantenendo anche la versione del software installato e la data in cui tale pacchetto è stato installato sul sistema.
- Per quanto concerne la sicurezza, poi, RPM permette la gestione di chiavi GPG ed MD5 per verificare l'integrità dei pacchetti.
- Ovviamente rpm può essere gestito sia tramite dei frontend grafici, ma anche attraverso il comando rpm direttamente dalla linea di comando, che risulta come la maggior parte dei comandi di questo genere più versatile, configurabile ed efficiente.





# Pacchetti e sorgenti

## Il comando rpm

- Il formato standard del comando rpm è il seguente:

**rpm <opzioni> <nome\_pacchetto>**

- L'utility a riga di comando è la più completa, infatti permette anche di:

- Verificare la posizione dei pacchetti
- Installare pacchetti via rete
- Conoscere dettagli supplementari sui pacchetti RPM

e molto altro ancora.

- Le opzioni principali sono:

- **-v**: verbose mode: aumenta il livello di informazioni nello standard output durante l'esecuzione del comando



## Opzioni del comando rpm (1)

- **-h:** indicatore di avanzamento: un'indicatore testuale di avanzamento fornirà informazioni sulla percentuale di completamento del processo
- **-i:** installa i pacchetti selezionati, presenta alcune opzioni secondarie:
  - **--excludedocs:** non installa i pacchetti di documentazione, allo scopo di risparmiare spazio. Si tratta di un'opzione pericolosa, in quanto priva l'amministratore di un utile strumento in caso di problemi con il programma.
  - **--replacepkgs:** rimpiazza la copia esistente di un pacchetto con una altra copia: Utile in caso di malfunzionamenti ed errori nel database dei pacchetti installati.



# Pacchetti e sorgenti

## Opzioni del comando rmp (2)

- **--force**: installa i pacchetti indipendentemente dai messaggi di warning presenti
- **--noscripts**: non esegue eventuali script pre e post-installazione presenti nel pacchetto
- **--nodeps**: ignora le eventuali dipendenze del pacchetto.
- **--root *path***: imposta una diversa directory per il pacchetto.
- **-e: elimina i pacchetti selezionati**
  - **--nodeps**: ignora le eventuali dipendenze del pacchetto.
- **-U**: aggiorna i pacchetti selezionati, rimuovendo i vecchi pacchetti ed installando i pacchetti successivi mantenendo inalterati i file di configurazione



## Opzioni del comando rpm (3)

- **--oldpackage**: permette di installare una versione meno recente di quella attualmente in uso sul sistema
- **-q**: interroga il sistema in merito ai pacchetti selezionati.
  - **-p file**: fornisce tutte le informazioni sul pacchetto selezionato
  - **-f file**: risale dal nome del file al pacchetto RPM che lo conteneva
  - **--whatprovides x**: determina quali pacchetti forniscono x.
  - **--whatrequires x**: determina quali pacchetti richiedono x.



## Opzioni del comando rpm (4)

- **-i**: riassume le informazioni sul pacchetto.
  - **-l**: elenca tutti i files contenuti nel pacchetto.
  - **--scripts**: visualizza gli script contenuti nel pacchetto per le fasi di pre-installazione e post-installazione.
  - **--provides**: elenca le funzionalità del pacchetto.
  - **--requires**: elenca ciò che serve al pacchetto.
- **-V**: verifica i pacchetti selezionati. (con la sottopzione **-a**) verifica tutti i pacchetti installati.
- **-K**: utilizza GPG per verificare i pacchetti (con la sottopzione **-nosignature** utilizza MD5 invece di GPG).



# Pacchetti e sorgenti

## Il comando rpm: chiavi GPG

- RPM nelle ultime versioni prevede la verifica della firma elettronica dei pacchetti: tutti i pacchetti originali Fedora, ad esempio, hanno come firma elettronica la chiave pubblica GnuPG di Red Hat Inc.
- Per installare le chiavi GnuPG è necessario usare questo comando:  
**rpm -import /media/cdrom/RPM-GPG-KEY-Fedora-test**
- Utilizzando poi l'opzione **-K** è possibile verificare la firma elettronica dei pacchetti.
- E' poi possibile estrarre un singolo file da un pacchetto rpm senza dover necessariamente installare tutto il pacchetto: è sufficiente trattare il pacchetto rpm come un normale archivio compresso, ad esempio con l'utility Midnight Commander (mc).



# Pacchetti e sorgenti

## yum

- YUM (Yellowdog Updater Modified) è un sistema evoluto di gestione dei pacchetti rpm, che permette una gestione evoluta, compresa la possibilità di aggiornamento automatico del sistema, senza intervento umano.
- Yum si basa su degli archivi condivisi, detti repository, ed è scritto interamente in Python, ma manca ad oggi di un'interfaccia grafica, anche se le sue potenzialità non ne risultano in alcun modo limitate.
- YUM è disponibile sul sito <http://linux.duke.edu/projects/yum>, oppure sul sito di Fedora o su FreshRPMs.net. YUM ha sia funzioni di server che di client, ma per il corso ha rilevanza solo l'aspetto client: il comando **yum** si basa su di un file di configurazione: **yum.conf** diviso in una parte generale ed una specifica per i server (repository).





# Pacchetti e sorgenti

## yum.conf (1)

- La sezione **main** contiene tutte le opzioni di configurazione:
  - **cachedir**: definisce dove YUM posizionerà i file di cache ed i database.
  - **debuglevel**: definisce il grado di accuratezza del file di log.
  - **logfile**: definisce la posizione del file di log.
  - **pkgpolicy**: definisce il criterio di scelta dei pacchetti nei repository (quello predefinito è la scelta del più recente).
  - **distroverpkg**: definisce la versione della distribuzione per cui YUM deve rinvenire i pacchetti nel repository.
  - **tolerant**: definisce il livello di tolleranza di YUM durante l'installazione dei pacchetti.



# Pacchetti e sorgenti

## yum.conf (2)

- **exactarch:** definisce se YUM deve aggiornare i pacchetti solo esattamente corrispondenti all'architettura o meno.
- **retries:** definisce il numero di tentativi che YUM deve effettuare in casi di errore.
- **obsoletes:** definisce se YUM deve ignorare o meno i pacchetti obsoleti.
- **gpgcheck:** definisce se YUM deve verificare le chiavi GPG.
- Ecco l'aspetto tipico di un file yum.conf tipico:

```
[main]
cachedir=/var/cache/yum
debuglevel=2
logfile=/var/log/yum.log
[ ]
```



# Pacchetti e sorgenti

## /etc/yum/repos.d e client yum

- Oltre alla sezione **MAIN**, il file **yum.conf** ha le sezioni dedicate ai repository, contenute all'interno di **/etc/yum/repos.d** al cui interno vengono definiti **name**, **baseurl**, **mirrorlist enabled** e **gpgcheck**.
- Il **client YUM** è il comando da riga di comando **yum**, che permette, tramite le sue opzioni di gestire l'aggiornamento manuale del sistema:
  - **yum list**: elenca i pacchetti disponibili nel repository.
  - **yum list installed**: elenca i pacchetti installati.
  - **yum list update**: elenca gli update disponibili.
  - **yum install <nome\_pacchetto>**: installa il pacchetto.
  - **yum update**: aggiorna tutti i pacchetti o il pacchetto selezionato senza rimuovere i pacchetti obsoleti.



# Pacchetti e sorgenti

## client yum

- **yum remove <nome\_pacchetto>**: elimina il pacchetto selezionato e le sue dipendenze.
- **yum upgrade**: aggiorna i pacchetti e rimuove i pacchetti obsoleti.
- Se invece si vuole ottenere un aggiornamento automatico del sistema, bisogna ricorrere allo script **/etc/init.d/yum start/stop/status**.
- Le opzioni del file di configurazione sono molto chiare e ricalcano vagamente come sintassi il sistema adottato da SAMBA: esse definiscono anche la directory provvisoria dove sono salvati i file prima di essere installati (`/var/cache/yum`), il livello di dettaglio del debug, la directory del log (`/var/log/yum.log`), le opzioni di sicurezza (come il controllo della firma digitale) ed altro.



# Pacchetti e sorgenti

## apt (1)

- APT (Advance Package Tool) fu sviluppato originariamente per i pacchetti deb di Debian ma è ora disponibile anche per altre distribuzioni tra cui Fedora.
- La suite APT si compone di:
  - **apt-get**: per il reperimento e l'installazione dei pacchetti.
  - **apt-cdrom**: per la gestione dei cdrom.
  - **apt-config**: per gestire le impostazioni di APT.
  - **apt-cache**: per gestire la cache dei pacchetti APT.
- **apt-get** richiede i permessi di root per essere eseguita, ma non è necessario essere root, in quanto apt-get chiederà autonomamente tale password nel caso sia lanciato da un utente non root.



# Pacchetti e sorgenti

## apt (2)

- Il primo comando da effettuare è sempre **apt-get update**, il quale aggiornerà i file indice dei pacchetti disponibili.
- In questo modo, avendo eseguito `apt-get update`, i pacchetti disponibili saranno quelli più recenti.
- Le fonti di repository si trovano in **/etc/apt/sources.list** ed hanno la forma seguente:

```
deb http://host/debian distribuzione sezione1 sezione2 sezione3
```

```
deb-src http://host/debian distribuzione sezione1 sezione2 sezione3
```

```
rpm http://host/red-hat distribuzione/versione/arch sez1 sez2 sez3
```

```
rpm-src http://host/red-hat distribuzione/versione/arch sez1 sez2 sez3
```

- Nel caso le voci siano precedute dal carattere sharp (#) di fatto sono commentate, e quindi non saranno prese in considerazione.



# Pacchetti e sorgenti

## apt-get (1)

- Per aggiornare indistintamente tutta la distribuzione il comando da eseguire è **apt-get upgrade**, che può utilizzare l'opzione **-y** per automatizzare la procedura, e l'opzione **-u** per visualizzare i pacchetti aggiornati.
- E' poi anche possibile installare singoli pacchetti, utilizzando la sintassi:  
**apt-get install <nome\_pacchetto>**
- E' ovviamente possibile anche rimuovere un pacchetto installato con apt-get, tramite l'opzione **remove <nome\_pacchetto>**. Non è possibile rimuovere un pacchetto senza toccare le sue dipendenze, mentre i file di configurazione rimangono, a meno di non usare l'opzione **--purge remove <nome\_pacchetto>**.





# Pacchetti e sorgenti

## apt-get (2)

- Per usare un CD-ROM/DVD come fonte, è necessario aggiungerlo nel file **sources.list**, utilizzando il programma **apt-cdrom**:
  - **apt-cdrom add**: inserisce tutti i pacchetti del CD-ROM di Debian nel lettore.
  - **apt-cdrom -d /home/iso add**: aggiunge i pacchetti anche se non in formato standard Debian.
- E' possibile anche aggiornare direttamente una distribuzione, con il comando **apt-get -u dist-upgrade**. Vi sono poi molte altre funzionalità di APT non trattate in questo corso.
- **apt-get --reinstall install <nome\_pacchetto>**: reinstalla un pacchetto già presente nel sistema



# Pacchetti e sorgenti

## apt-get (3)

- Nel caso in cui invece di avere un repository abbiamo direttamente un pacchetto .deb sul sistema il comando da eseguire sarà: **dpkg -i file.deb**
- Per la rimozione il comando è invece **apt-get remove <nome\_pacchetto>** a cui dobbiamo aggiungere **-purge** per rimuovere anche i file di configurazione.
- Eventuali dipendenze di pacchetti rimossi sono gestibili con il comando **apt-get autoremove**.
- Infine, con il comando **apt-get clean** vengono eliminati tutti i file, tranne i file lock, dalle directory `/var/cache/apt/archives/` e `/var/cache/apt/archives/partial/`. Di conseguenza, se si volesse reinstallare un pacchetto APT dovrà nuovamente scaricarlo.



# Pacchetti e sorgenti

## dpkg

- Nel caso di pacchetti .deb si utilizzerà dpkg, con le seguenti opzioni:
  - `dpkg --remove <nome_pacchetto>`: rimozione
  - `dpkg --purge <nome_pacchetto>`: pulizia dipendenze
  - `dpkg -I \*`: lista pacchetti installati
  - `dpkg -I <nome_pacchetto>`: informazioni sul pacchetto
  - `dpkg -I vim\*`: informazioni sul pattern
  - `dpkg -L nome_pacchetto`: lista dei files installati con un pacchetto
  - `dpkg -S nome_file`: da quale pacchetto dipende quel file
  - `dpkg -c nome_file.deb`: lista il contenuto del pacchetto
  - `dpkg-reconfigure <nome_pacchetto>`: riconfigura un pacchetto già installato
  - `dpkg -x file.deb`: estrae localmente i file del pacchetto senza installarlo



# Pacchetti e sorgenti

## apt-cache

- apt-get deve essere eseguito come root, o con i permessi di root (sudo) mentre esiste un comando utilizzabile da tutti gli utenti per avere informazioni sui pacchetti: apt-cache (gestisce la cache di supporto).
- con apt-cache possiamo invece cercare anche tra i pacchetti non ancora installati sul sistema:
  - apt-cache search "vim.\*"
  - apt-cache showpkg <nome\_pacchetto>: fornisce informazioni sul pacchetto
  - apt-cache showsrc <nome\_pacchetto>: fornisce i sources record relativi
  - apt-cache stats: fornisce statistiche sulla cache di supporto
  - apt-cache dump: mostra il contenuto della cache
  - apt-cache show <nome\_pacchetto>: mostra tutto il contenuto
  - apt-cache depends <nome\_pacchetto>: mostra le dipendenze del pacchetto



# Pacchetti e sorgenti

## Tips x i pacchetti .deb

- Con dpkg è possibile realizzare un file per automatizzare l'installazione su altri sistemi:
- Creo un file con tutte le configurazioni ed i settaggi, tramite il comando

```
dpkg --get-selections \* > dpkg_settings
```

- Trasferisco questo file sulla macchina da configurare e lancio il comando

```
dpkg --set-selections < dpkg_settings
```

- Infine utilizzo apt-get:

```
apt-get -u dselect-upgrade
```

- Come nota di colore: anche i programmatori GNU/Linux si divertono ad



# Pacchetti e sorgenti

## Installazione dai sorgenti

- Se possibile l'uso dei pacchetti è sempre consigliabile, a meno di particolari esigenze, sia per la facilità che per la rapidità. Inoltre, non vi è modo di tenere traccia dei pacchetti compilati da sorgente, rendendo di fatto il sistema meno lineare e più difficile da gestire.
- Nel caso si debba ricorrere alla compilazione dei sorgenti è necessario, ovviamente, avere tutti i pacchetti necessari per la compilazione ed ovviamente il codice sorgente del pacchetto che si vuole installare.
- Normalmente tale codice sorgente è distribuito sotto forma di pacchetti tar.gz, per cui come prima operazione è necessario scompattarli con il comando:

*tar -xvzf nomefile.tar.gz*



# Pacchetti e sorgenti

## Installazione dai sorgenti

- Una volta scompattato il codice sorgente, viene creata una directory con i file avente lo stesso nome dell'archivio, all'interno vi saranno dei file .c ed un file chiamato configure, eseguibile.
- Lo avviamo con il comando: ***./configure***
- Questo script verifica la presenza di tutto il necessario per procedere alla compilazione del programma, ed in caso positivo crea un file Makefile, che permette di compilare il programma in base al sistema destinatario. A questo punto è possibile dare il secondo comando:

***make***

- Ed infine il comando conclusivo:

***make install***



# Pacchetti e sorgenti

## Installazione dai sorgenti

- E' ovviamente possibile combinare i due comandi precedenti: **make && make install**.
- Nel caso il programma non sia di nostro interesse è possibile, avendo ancora i sorgenti, eseguire il comando **make uninstall** o **make deinstall** per rimuovere i file binari appena creati.
- Spesso purtroppo il comando **make uninstall** non funziona, perché l'autore del programma non lo ha implementato. In questi casi non resta che cancellare manualmente tutte le tracce del programma.





# Copyleft



Quest'opera, per volontà degli autori, è rilasciata sotto la disciplina della seguente licenza

## **Creative Commons Public License**



### **Attribuzione-Condividi allo stesso modo 2.5 Italia**



Tu sei libero:

-  di riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire e recitare quest'opera
-  di modificare quest'opera

Alle seguenti condizioni:

-  **Attribuzione.** Devi attribuire la paternità dell'opera nei modi indicati dall'autore o da chi ti ha dato l'opera in licenza e in modo tale da non suggerire che essi avallino te o il modo in cui tu usi l'opera.
-  **Condividi allo stesso modo.** Se alteri o trasformi quest'opera, o se la usi per crearne un'altra, puoi distribuire l'opera risultante solo con una licenza identica o equivalente a questa.

Ogni volta che usi o distribuischi quest'opera, devi farlo secondo i termini di questa licenza, che va comunicata con chiarezza. In ogni caso, puoi concordare col titolare dei diritti utilizzi di quest'opera non consentiti da questa licenza. Questa licenza lascia impregiudicati i diritti morali. Le utilizzazioni consentite dalla legge sul diritto d'autore e gli altri diritti non sono in alcun modo limitati da quanto sopra.

Questo è un riassunto in linguaggio accessibile a tutti del codice legale (la licenza integrale) che è disponibile alla pagina web:

<http://creativecommons.org/licenses/by-sa/2.5/it/legalcode>



# Copyleft

Quest'opera, è stata realizzata grazie al contributo di molte persone. La prima versione è stata realizzata a partire dalle slide realizzate da Silvio Colloca distribuite con licenza Creative Commons sul sito <http://linuxhelp.it>. Successivamente sono state modificate dai molti docenti che hanno prestato il loro servizio gratuito nelle lezioni dei corsi Netstudent. In ordine sparso (e sperando di non dimenticare nessuno): Giovanni Berton Giachetti, Daniele Lussana, Alessandro Ugo, Emmanuel Richiardone, Andrea Garzena, Stefano Cotta Ramusino, Roberto Preziosi, Marco Papa Manzillo, Puria Nafisi Azizi, Luca Necchi, Luca Barbato, David Putzer, Alberto Grimaldi, Nicola Tuveri, Stefano Colazzo, Laura De Martini, Massimiliano Bessone, ecc...